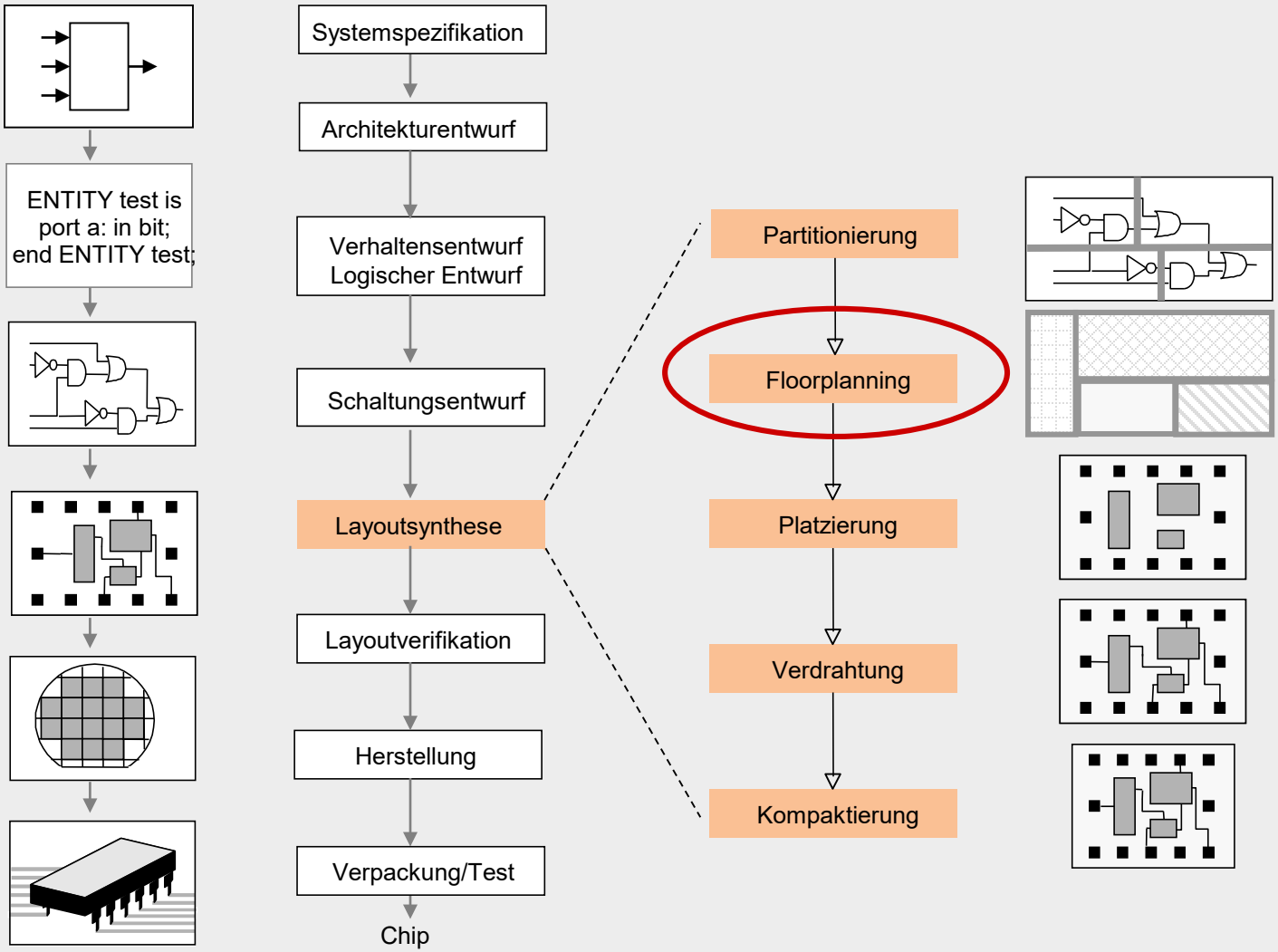
Gliederung Kapitel 3 – Floorplanning

3.1 Einführung



3.1 Einführung

Die Aufgabe des Floorplanning besteht darin, das Ergebnis der Schaltungspartitionierung so aufzubereiten, dass jeder dabei erstellte Block intern platziert und verdrahtet werden kann.

Dazu sind

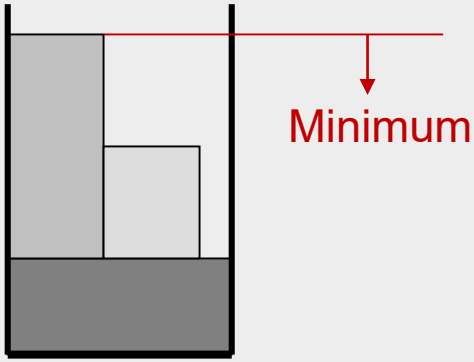
- die Abmessungen bzw. Seitenverhältnisse der einzelnen Blöcke, und evtl. auch der Topzelle, festzulegen,
- die Positionen dieser Blöcke innerhalb der Topzelle zu definieren und
- die Positionen der Außenanschlüsse in den einzelnen Blöcken zu bestimmen (Pinzuordnung).

3.2 Optimierungsziele

- Fläche und Form des umschließenden Rechtecks
- Gesamtverbindungslänge
- Fläche und Gesamtverbindungslänge
- Signalverzögerungen (Performance-driven, Timing-driven)

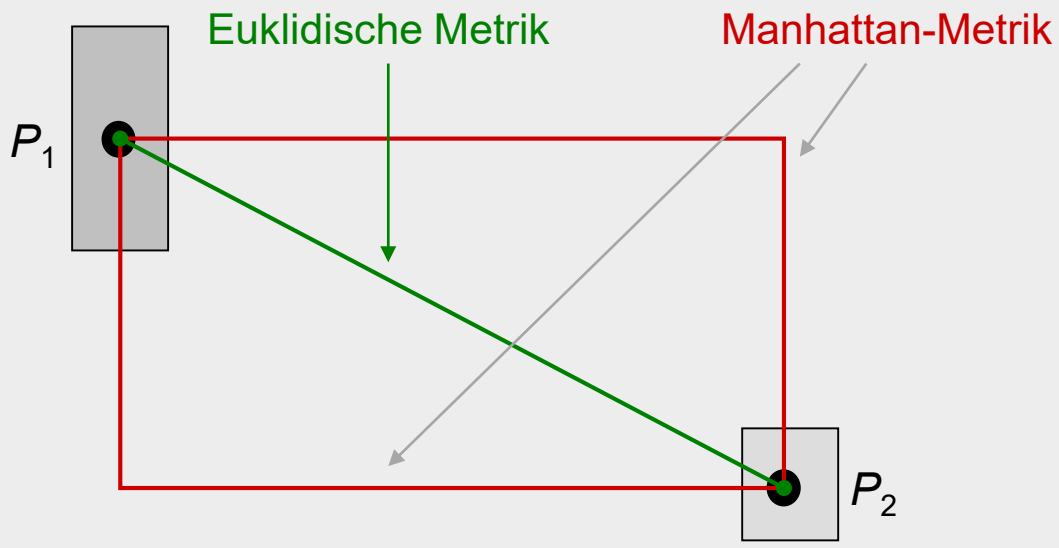
3.2 Optimierungsziel: Fläche und Form des umschließenden Rechtecks

- Flächenminimale Topzelle angestrebt
  - Soft Blocks: Flächenminimierung der Topzelle erfolgt unter Ausnutzung der Flexibilität in den Formen der einzelnen Blöcke, denn diese lassen sich dann flächenminimal anordnen, wenn man ihre Formen zueinander passend gestaltet
  - Hard Blocks: Keine Flexibilität der Blockformen („Bin Packing Problem“)



- Evtl. Einhalten von Formvorgaben an Topzelle

- Minimierung der Toplevel-Verdrahtung angestrebt
  - Längenbestimmung über Manhattan-Entfernung (Manhattan-Metrik)
  - Längenbestimmung über minimalen Spannbaum (euklidische Metrik)



$$\text{Abstand}\ (P_1, P_2) = \sqrt[n]{|x_2 - x_1|^n + |y_2 - y_1|^n}$$

n=1 für Manhattan-Metrik
n=2 für euklidische Metrik

3.2 Optimierungsziel: Fläche und Gesamtverbindungslänge

- Minimierung sowohl der Fläche des die Topzelle umschließenden Rechtecks *A* als auch die Gesamtverbindungslänge *L*
- Zielfunktion: $Z = w_1 * A + w_2 * L$

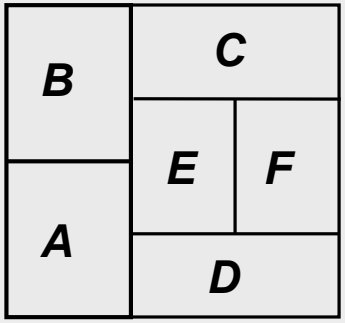
Kapitel 3 – Floorplanning

Flexible Blöcke (Soft blocks) , feste Blöcke (Hard blocks)

- Blöcke mit variablen Formen, bei denen nur die Flächen vorgegeben sind, werden als flexible Blöcke bezeichnet
- Bei festen Blöcken ist die äußere Form festgelegt

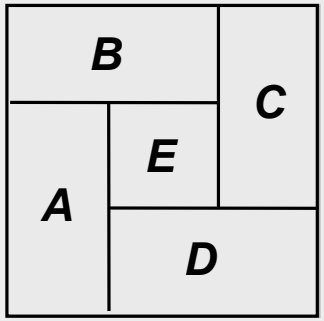
Geschnittener Floorplan (Slicing floorplan)

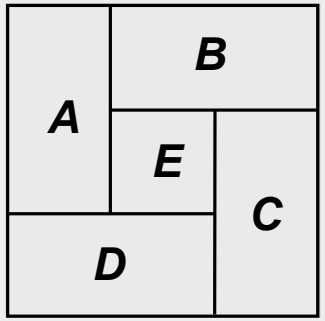
Basiszellen sind durch (wiederholte) vertikale oder horizontale Zweiteilung entstanden



Ungeschnittener Floorplan (Non-slicing floorplan)

Basiszellen sind nicht ausschließlich durch vertikale oder horizontale Zweiteilung entstanden





Mindestens fünf Basiszellen, sog. Rad (Wheel)

Datenstrukturen beim Floorplanning

- Abbildung der Lage der Blöcke zueinander, deren konkrete Formen/Abmessungen kann als Attribut beigefügt sein

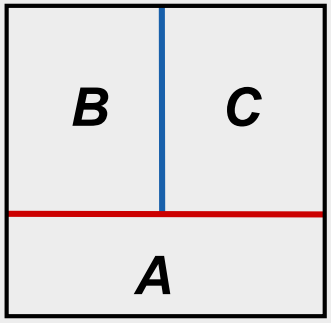
- Datenstrukturen liefern begrenzten (finiten) Lösungsraum, welcher optimale Lösung einschließen sollte

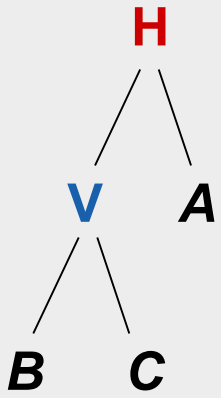
⇒ Geschnittener Floorplan:
  - Schnittbaum
  - umgekehrte polnische Notation

⇒ Ungeschnittener bzw. allgemeiner Floorplan:
  - Floorplanbaum
  - Polargraph
  - Sequence Pair

Schnittbaum (Slicing tree)

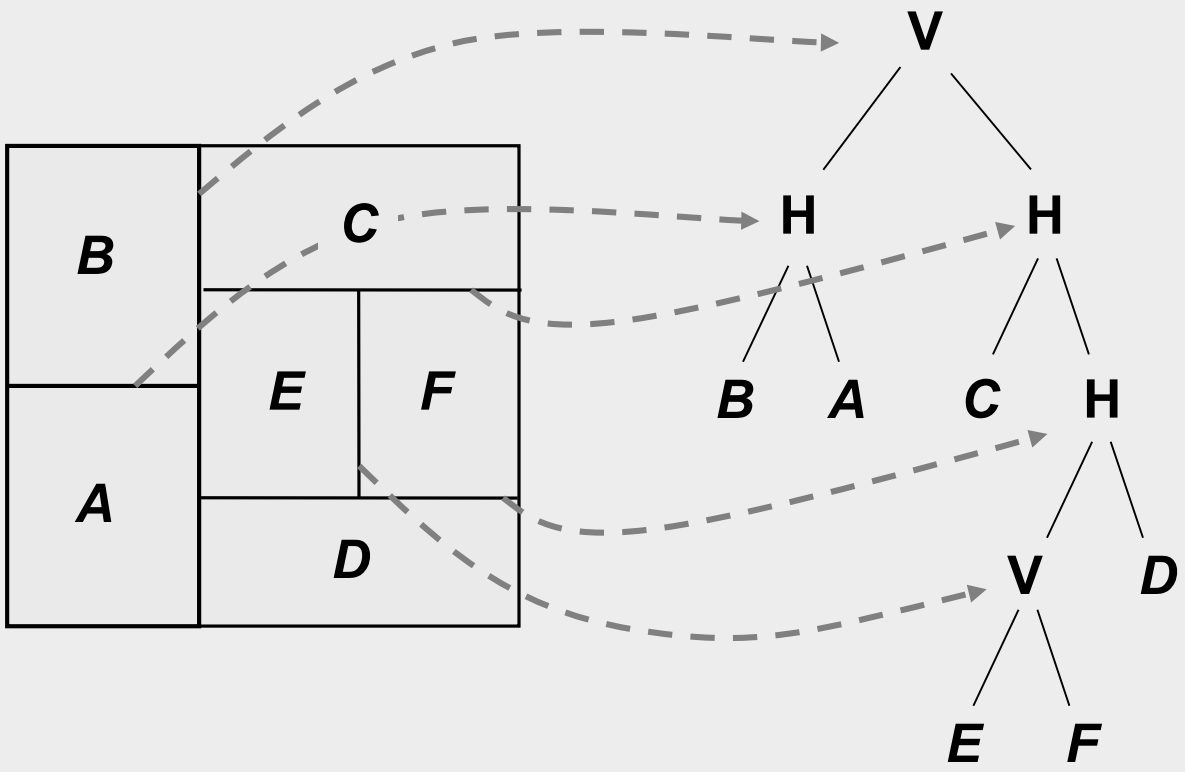
- Ein Schnittbaum ist die Modellierung eines geschnittenen Floorplans durch einen geordneten Binärbaum mit *k* Blättern und *k*-1 Knoten (*k* … Anzahl der Blöcke)



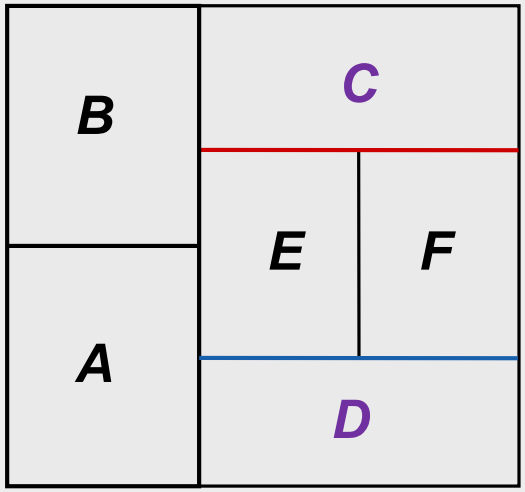


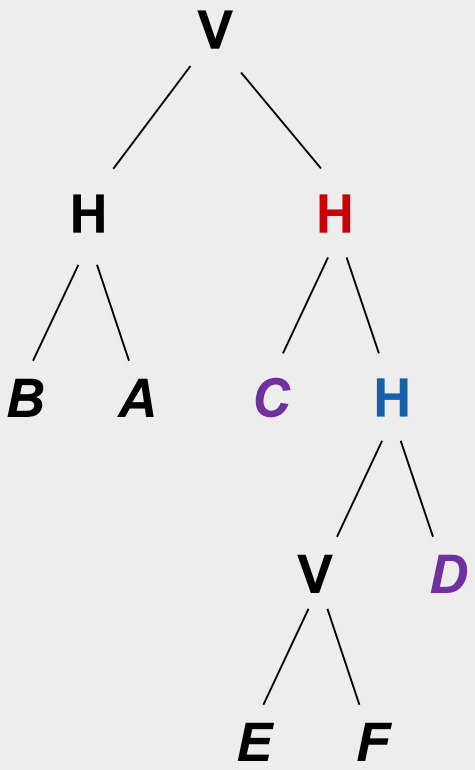
- Jeder Knoten repräsentiert dabei eine Schnittlinie und jedes Blatt einen Block
- Die Reihenfolge der Kindknoten (li./re.) entspricht der relativen Lage der Blöcke, d.h. H: oben/unten, V: links/rechts
- Wesentliches Merkmal eines Schnittbaums ist seine Binärstruktur, d.h. jeder Knoten hat genau zwei Kinder

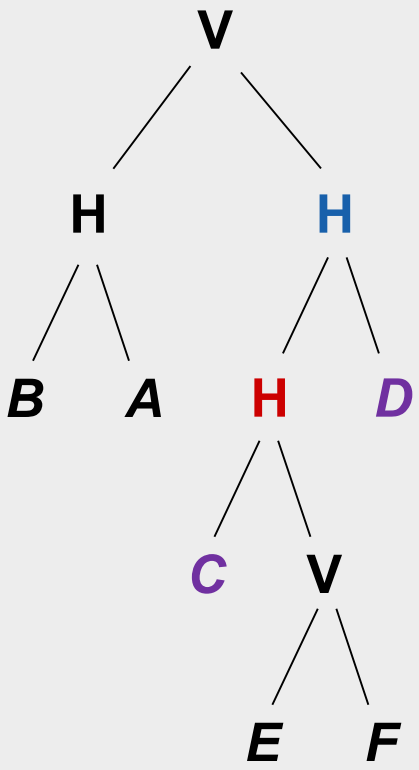
Schnittbaum (Slicing tree)



Schnittbaum (Slicing tree)



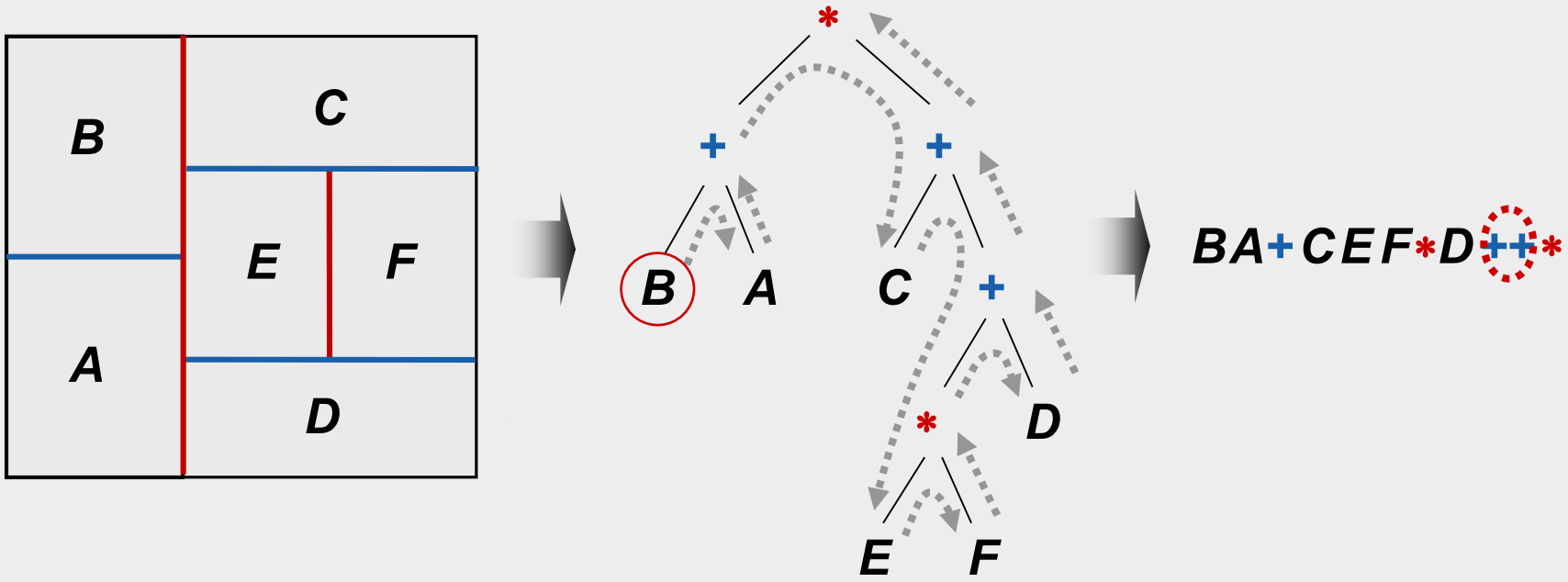




⇒ Zwei aufeinander folgende identische Operatoren: Redundanz in Floorplan-Abbildung

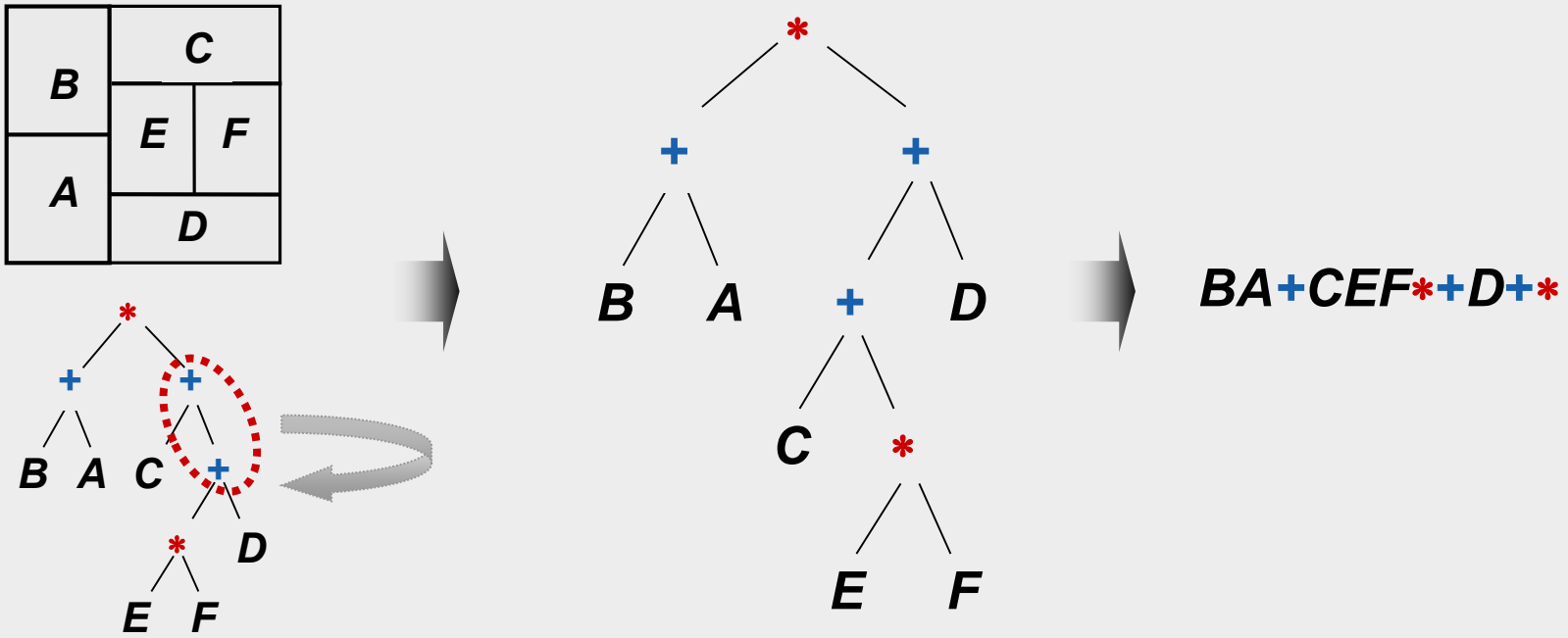
Umgekehrte polnische Notation (Polish expression)

- Operatoren **V** → * und **H** → +
- Durchlaufen des Schnittbaumes in „umgekehrter Richtung“ („Bottom-up-Bauplan“)
  → vor jedem Schnitt wird rekursiv erst der linke und dann der rechte Teilbaum durchlaufen
  → entspricht einer Tiefensuche in Nebenreihenfolge (post-order depth-first search)



Normalized Polish Expression (NPE)

- Vermeidung von Redundanzen:
  mehrere Schnittbäume bzw. Notationen bilden einen Floorplan ab
- NPE: Keine zwei aufeinander folgende Operatoren sind identisch
- Verdrehter Schnittbaum (Skewed slicing tree): Schnittrichtung des rechten Teilbaums immer verschieden von Vorgängerknoten



Datenstrukturen beim Floorplanning

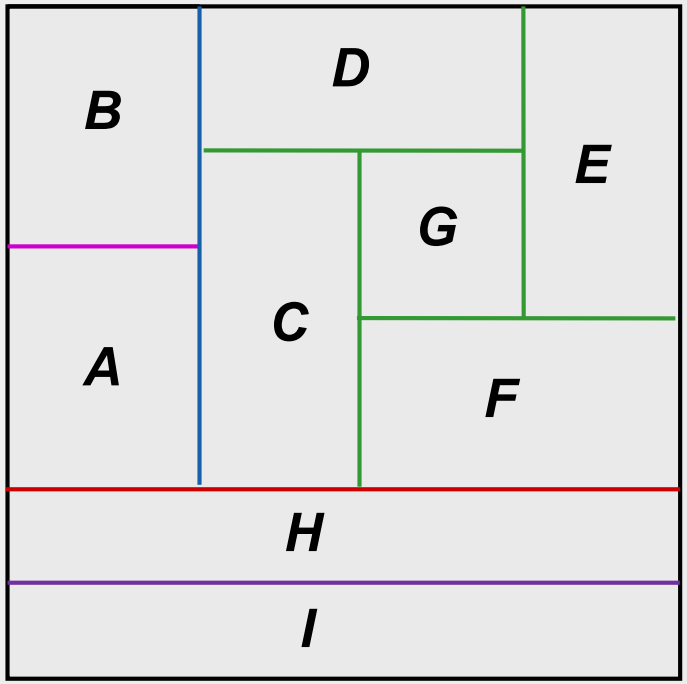
⇒ Geschnittener Floorplan:
- Schnittbaum ✓
- Umgekehrte (normalisierte) polnische Notation ✓

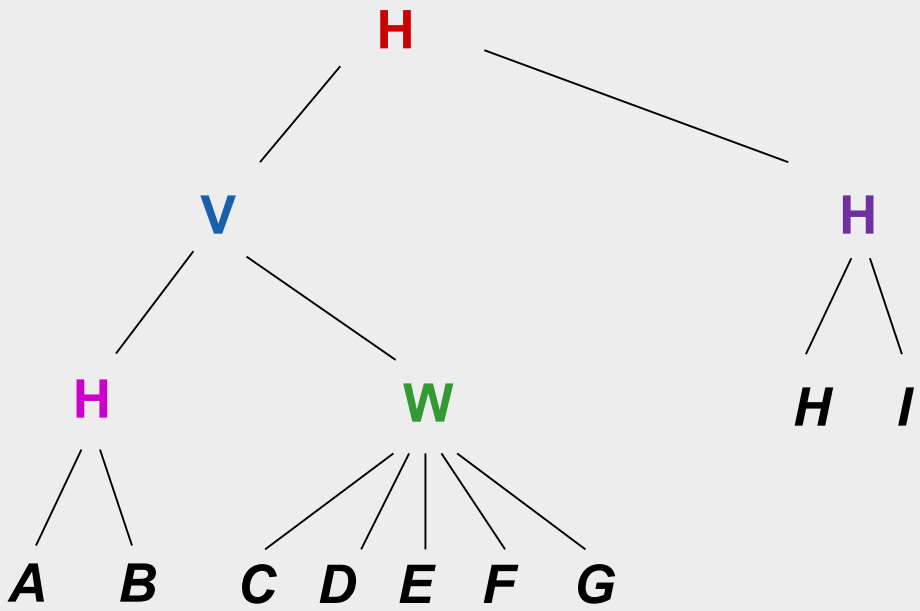
⇒ Ungeschnittener bzw. allgemeiner Floorplan:
- Floorplanbaum
- Polargraph
- Sequence Pair

Floorplanbaum (Floorplan tree)

- Jeder hierarchische Floorplan (geschnitten und ungeschnitten) kann durch einen Floorplanbaum repräsentiert werden
- Jedes Blatt verkörpert dabei einen Block, jeder Knoten entweder einen vertikalen bzw. horizontalen Schnittoperator oder ein Rad
- Binäre Schnittbäume (Slicing trees) sind damit Untermengen der Floorplanbäume

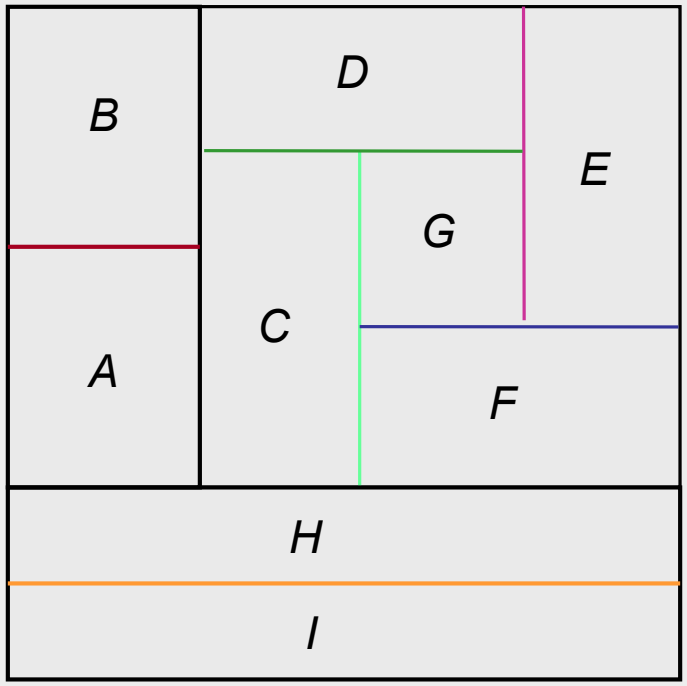
Floorplanbaum (Floorplan tree)

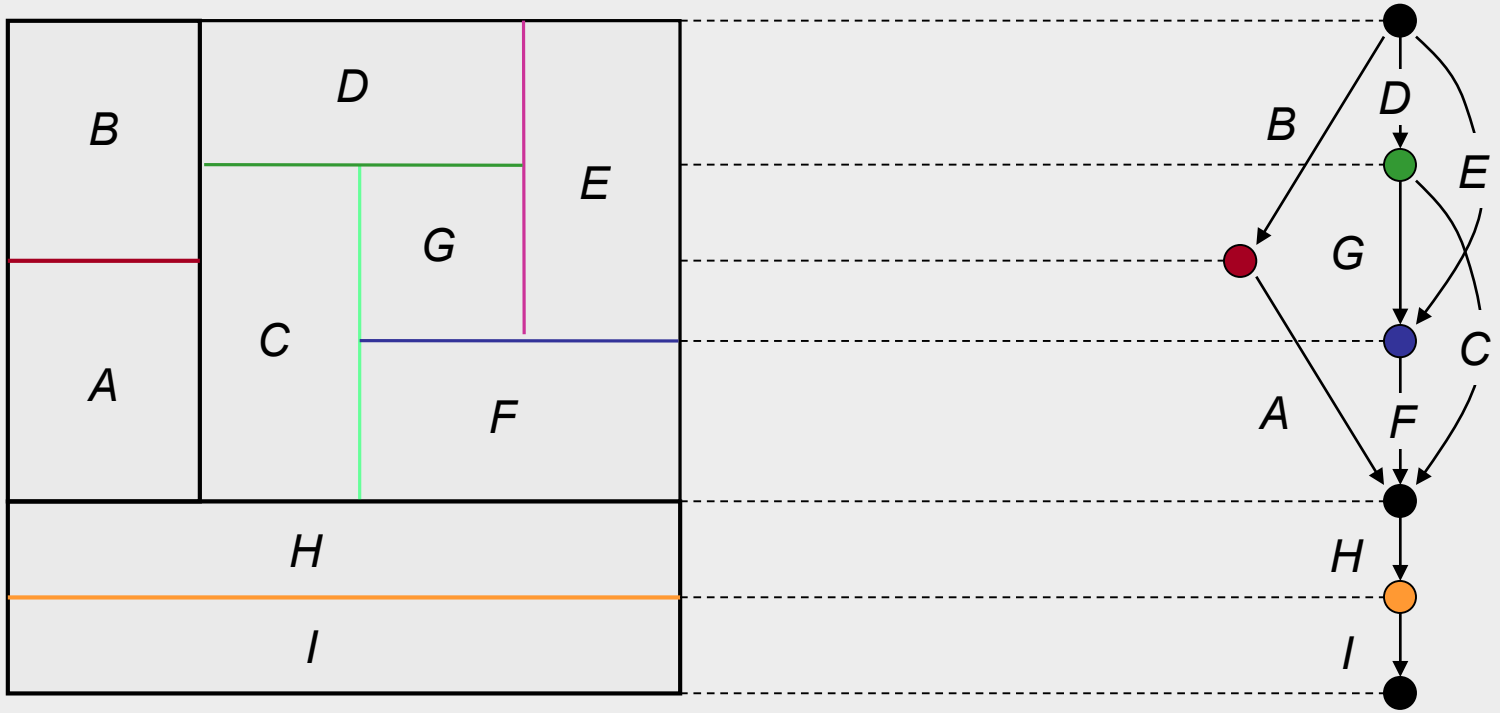




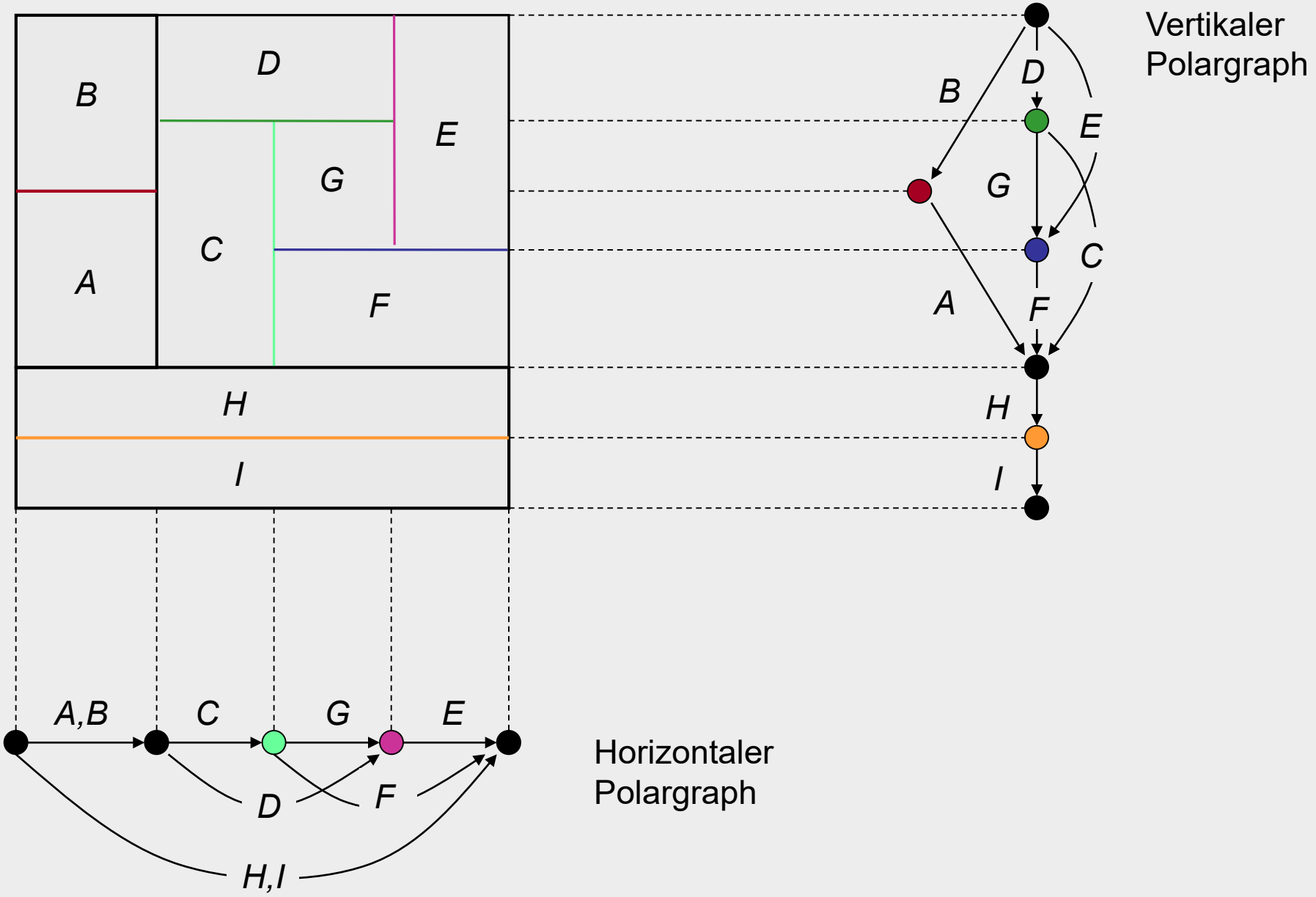
Polargraph (Polar graph)

- Blöcke als Kanten abgebildet, Schnittkanten als Knoten
- Besteht aus zwei gerichteten Graphen
  - Vertikaler Polargraph:
    Knoten sind horizontale Schnittkanten, Kanten dazwischen liegende Blöcke
  - Horizontaler Polargraph:
    Knoten sind vertikale Schnittkanten, Kanten dazwischen liegende Blöcke


B
D
E
G
C
A
F
H
I



Vertikaler Polargraph

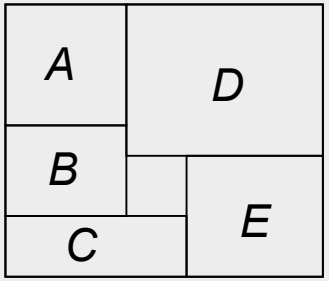

B
D
E
G
C
A
F
H
I
Vertikaler
Polargraph
A,B
C
G
E
D
F
H,I
Horizontaler
Polargraph

Polargraph (Polar graph)

- Längster Pfad im vertikalen Polargraphen: minimal benötigte Layouthöhe
- Längster Pfad im horizontalen Polargraphen: minimal benötigte Layoutbreite
- Nutzbar zur Bestimmung der Fläche des kleinsten umschließenden Rechtecks und damit der Abmessungen der Topzelle

Sequence Pair

- Zwei Folgen von Blocknamen, aus denen sich Blockanordnung eindeutig ableiten lässt
- Beispiel: (*ABDCE*, *CBAED*)



- Angabe der horizontalen bzw. vertikalen Lagebeziehung:

(… *A* … *B* … , … *A* … *B* …) → *A* ist links von *B* platziert
(… *A* … *B* … , … *B* … *A* …) → *A* ist oberhalb von *B* platziert
(… *B* … *A* … , … *A* … *B* …) → *A* ist unterhalb von *B* platziert
(… *B* … *A* … , … *B* … *A* …) → *A* ist rechts von *B* platziert

Datenstrukturen beim Floorplanning

⇒ Geschnittener Floorplan:
- Schnittbaum ✓
- umgekehrte (normalisierte) polnische Notation ✓

⇒ Ungeschnittener bzw. allgemeiner Floorplan:
- Floorplanbaum ✓
- Polargraph ✓
- Sequence Pair ✓

Kapitel 3 – Floorplanning

- Floorplan-Sizing-Algorithmus (flexible Blöcke, soft blocks)
- Cluster-Wachstums-Algorithmus (feste Blöcke, hard blocks)

3.4.1 Floorplan-Sizing-Algorithmus

- „Floorplan-Sizing“: Festlegung einer Außenform der Topzelle und, daraus abgeleitet, die Festlegung der einzelnen Blockformen bzw. -abmessungen

- Prinzipielle Vorgehensweise:

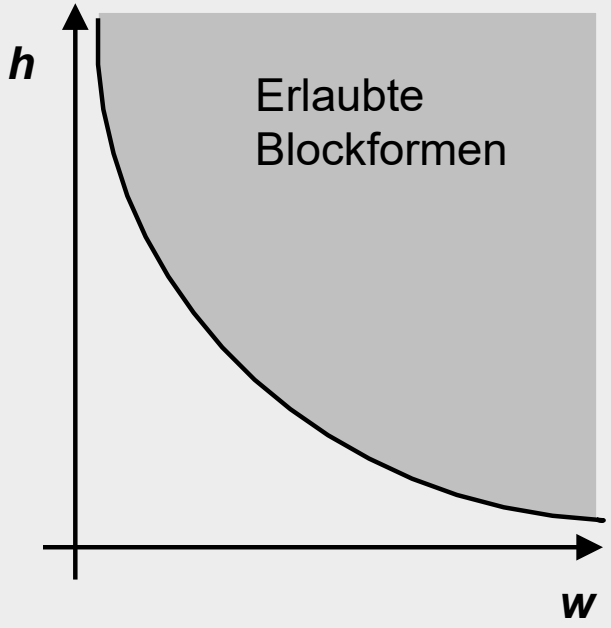
  1. Aus den möglichen Formen der einzelnen Blöcke und ihrer Anordnungsvarianten werden die verschiedenen Form- und Größenvarianten der Topzelle ermittelt (Bottom up)
  2. Von einer dann festgelegten (optimierten) Topzellen-Form ausgehend, legt man die dazu notwendigen Formen und Anordnungen der Blöcke fest (Top down)

3.4.1 Floorplan-Sizing-Algorithmus: Begriffe

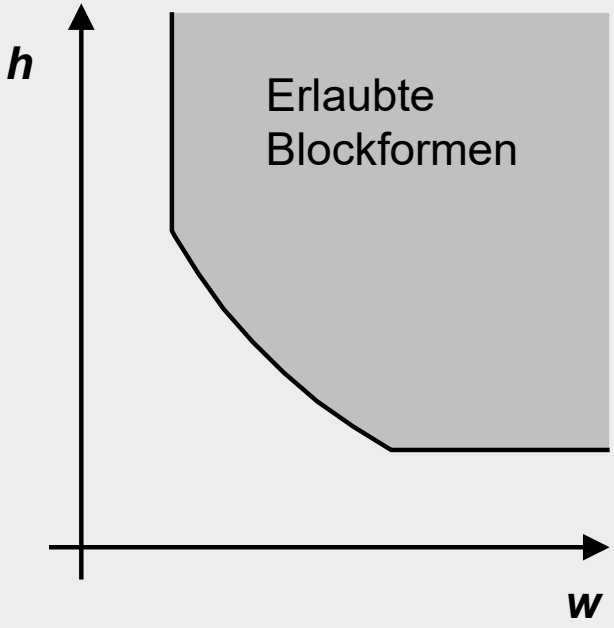
Formfunktionen

- Blöcke mit flexiblen Abmessungen sind durch Flächenvorgabe A bestimmt
- Damit haben die Höhe h und die Breite w der Randbedingung $h * w \geq A$ zu genügen
- Abhängigkeit zwischen Höhe und Breite eines Blocks (z.B. Höhe als Funktion der Breite) wird als **Formfunktion** (Shape function) des Blocks bezeichnet

Formfunktionen



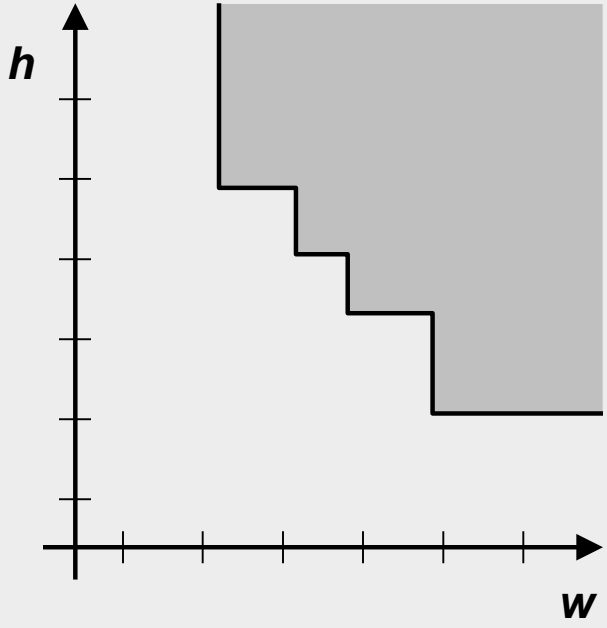
$h * w \geq A$



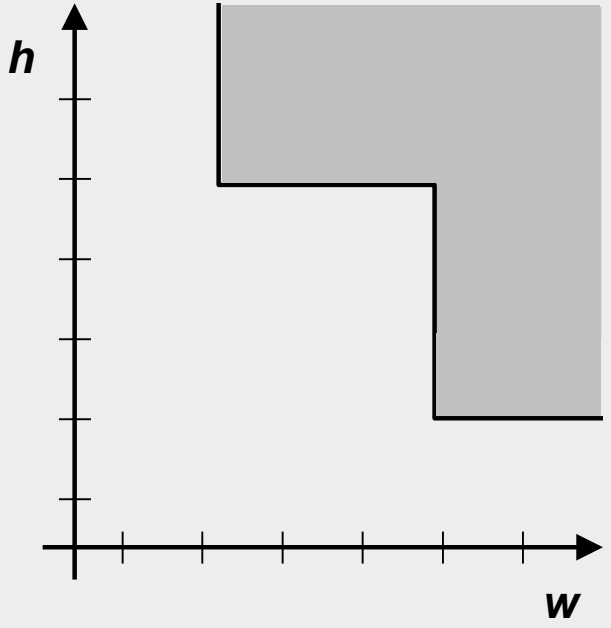
Mit minimalen Höhen- und Breitenbeschränkungen

Nach Gerez, S. H.: Algorithms for VLSI Design Automation, Wiley, 2000

Formfunktionen



Mit diskreten Höhen- und Breitenwerten

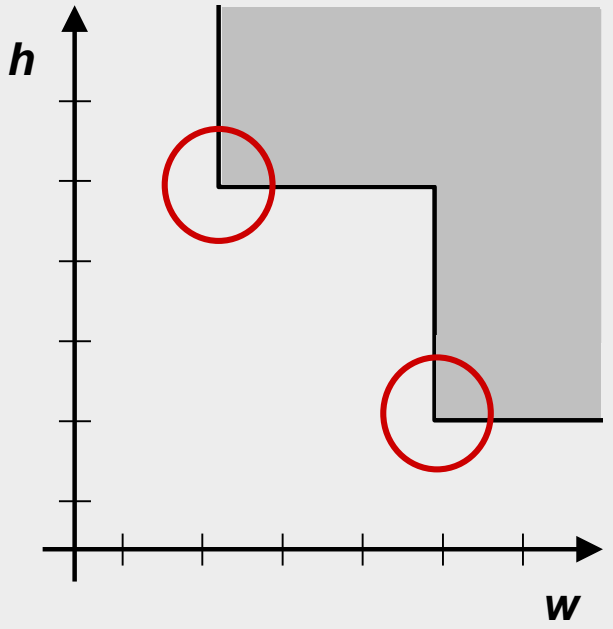


Bibliotheksblock mit zwei Anordnungsmöglichkeiten

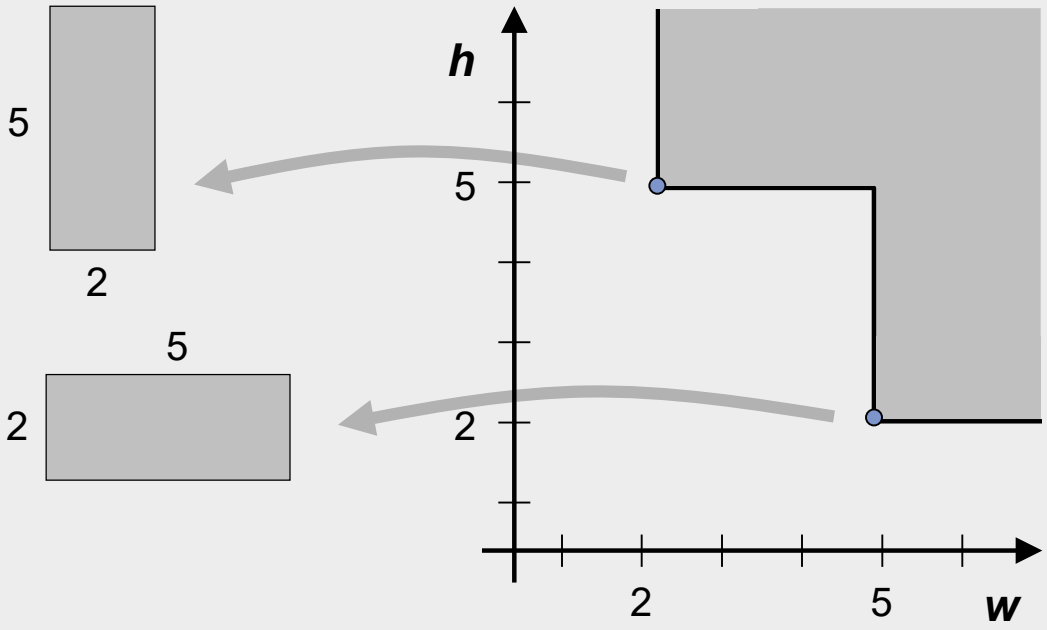
Nach Gerez, S. H.: Algorithms for VLSI Design Automation, Wiley, 2000

Eckwerte

- Die einzelnen diskreten Form-Möglichkeiten eines Blocks, die sich als „äußere“ Eckpunkte in der Formfunktion widerspiegeln, sind **Eckwerte** (Break points) der Formfunktion



Eckwerte

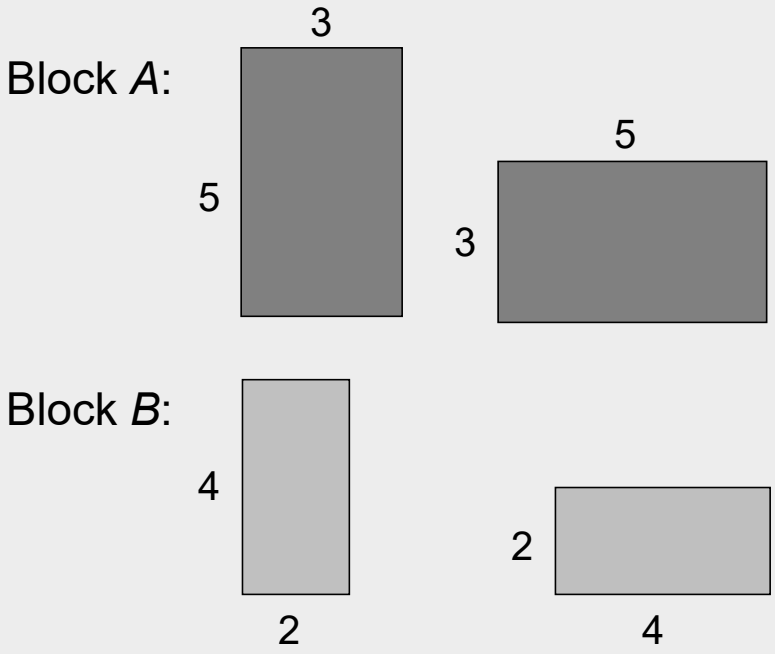


Algorithmus

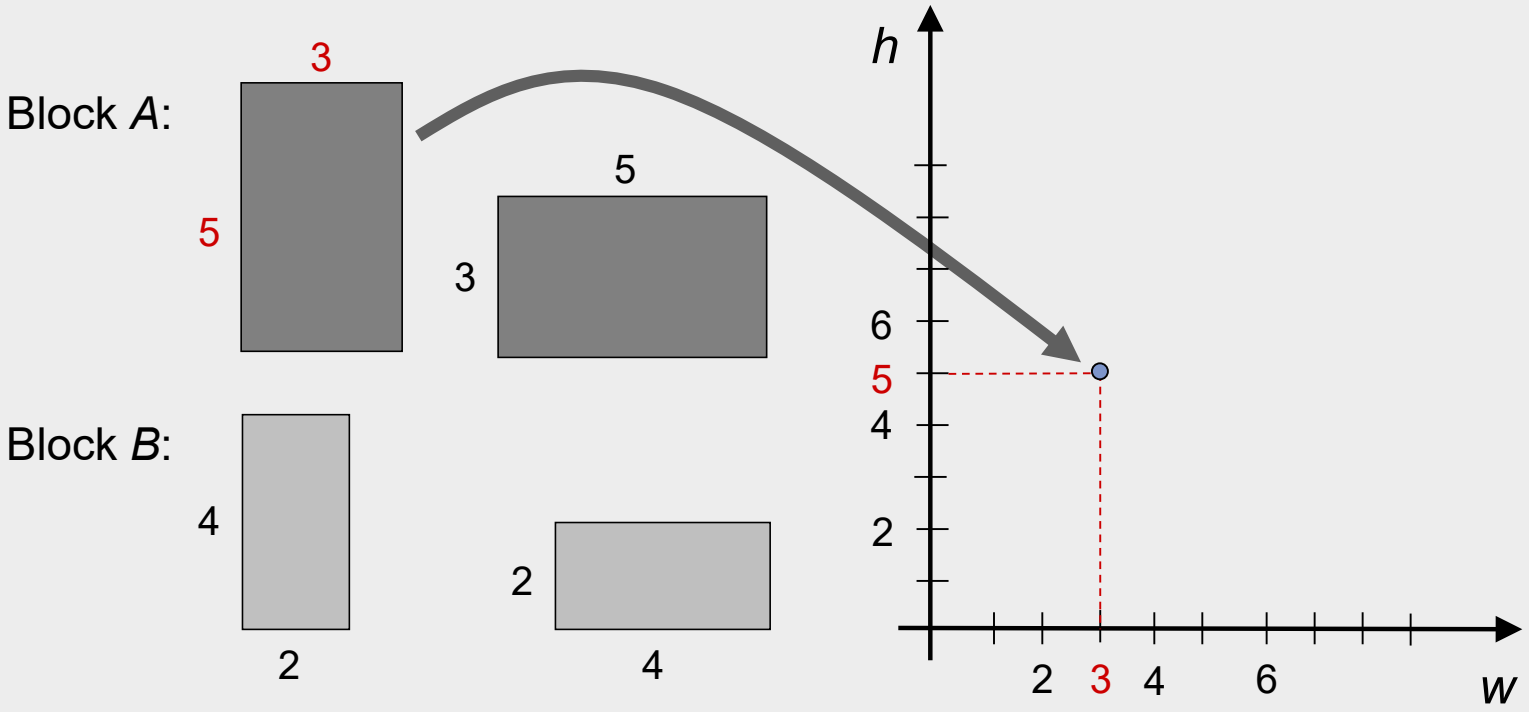
1. Ermittlung der Formfunktionen aller in der Topzelle anzuordnenden Blöcke.
2. Ermittlung der Formfunktion der Topzelle mit einer Bottom-up-Strategie, bei der vom niedrigsten Blockniveau ausgehend, die Formfunktionen der Blöcke kombiniert werden; Ermittlung der optimalen Größe bzw. Form der Topzelle.
3. Von der Formfestlegung der Topzelle ausgehend, wird der Schnittbaum nach „unten“ (Top down) abgearbeitet und dabei die jeweilige zusammengesetzte Blockform ermittelt, bis sämtliche (Basis-) Blöcke erreicht sind.

3.4.1 Floorplan-Sizing-Algorithmus: Beispiel

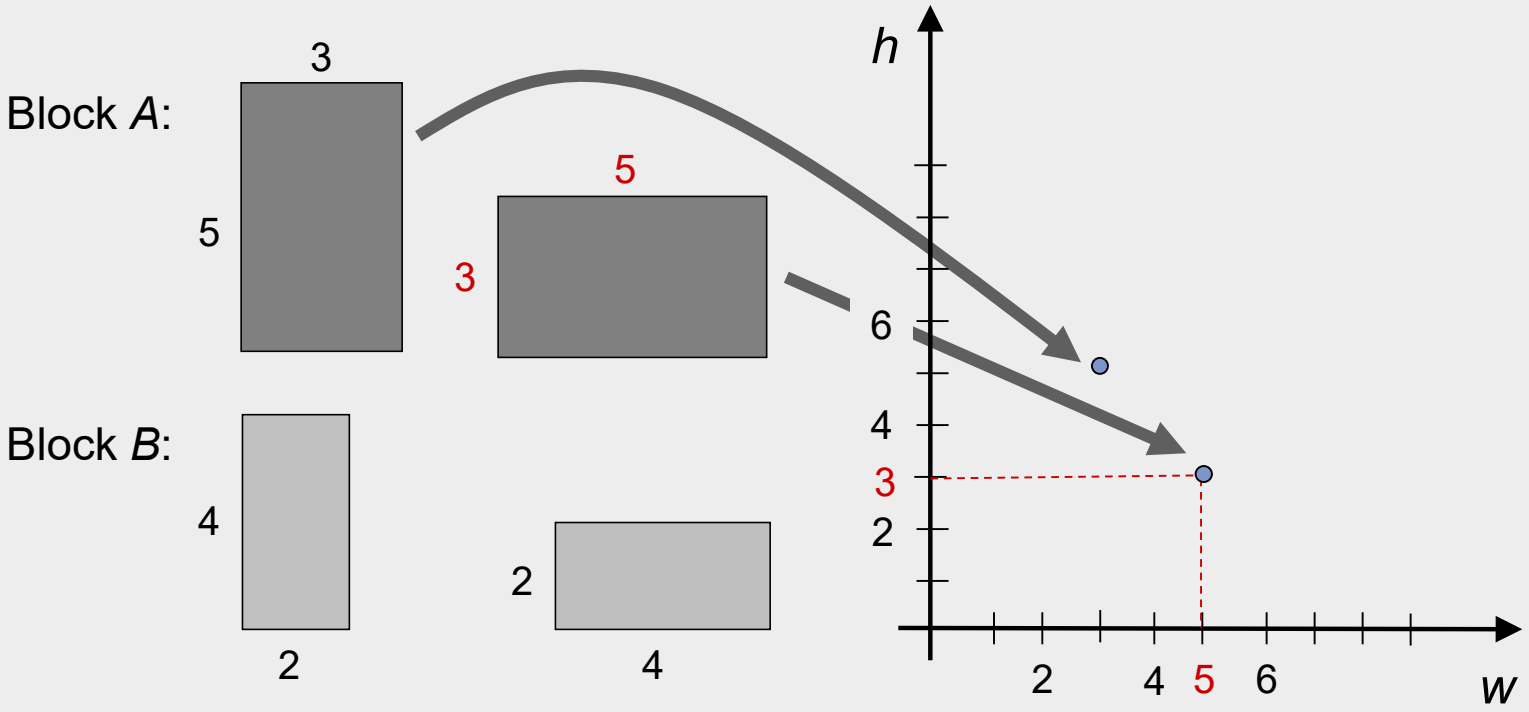
1. Ermitteln der Formfunktionen der Blöcke



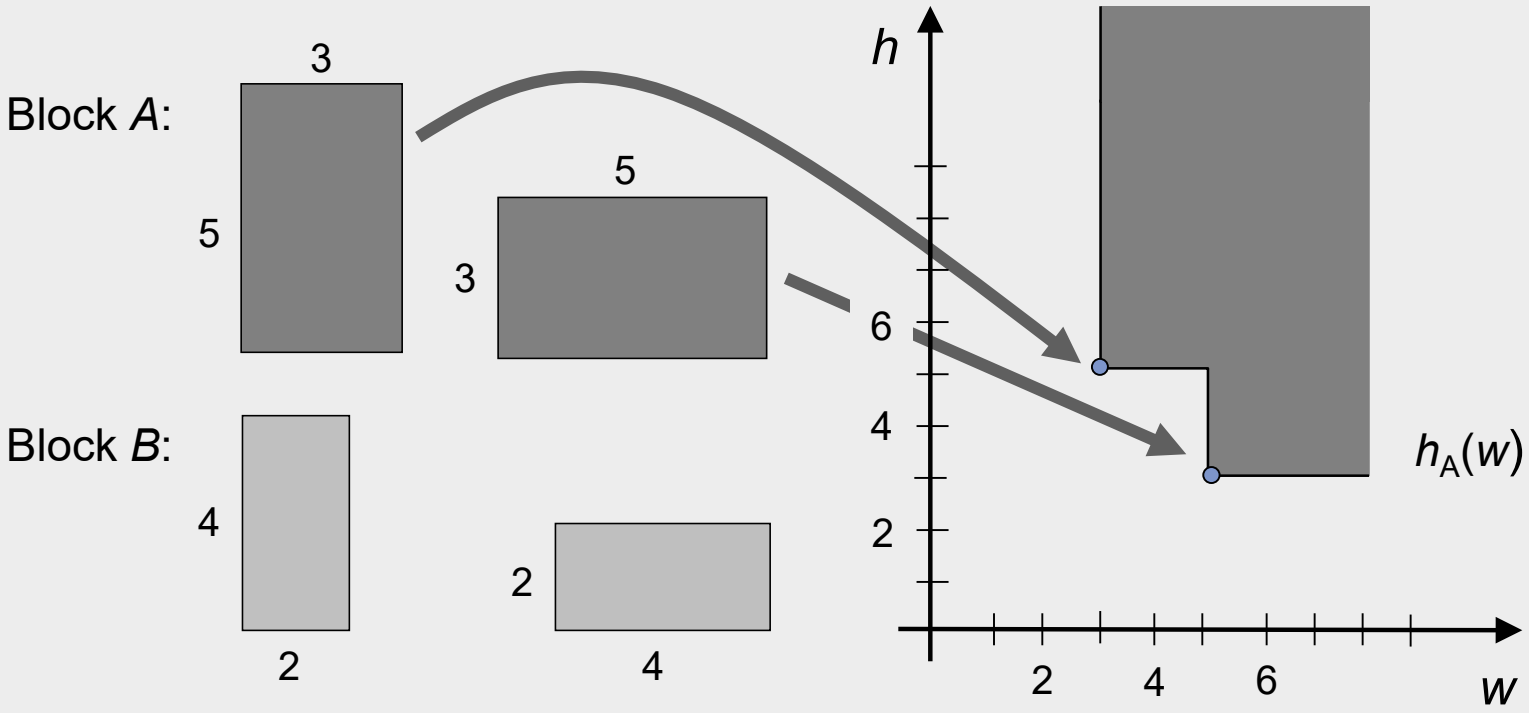
1. Ermitteln der Formfunktionen der Blöcke



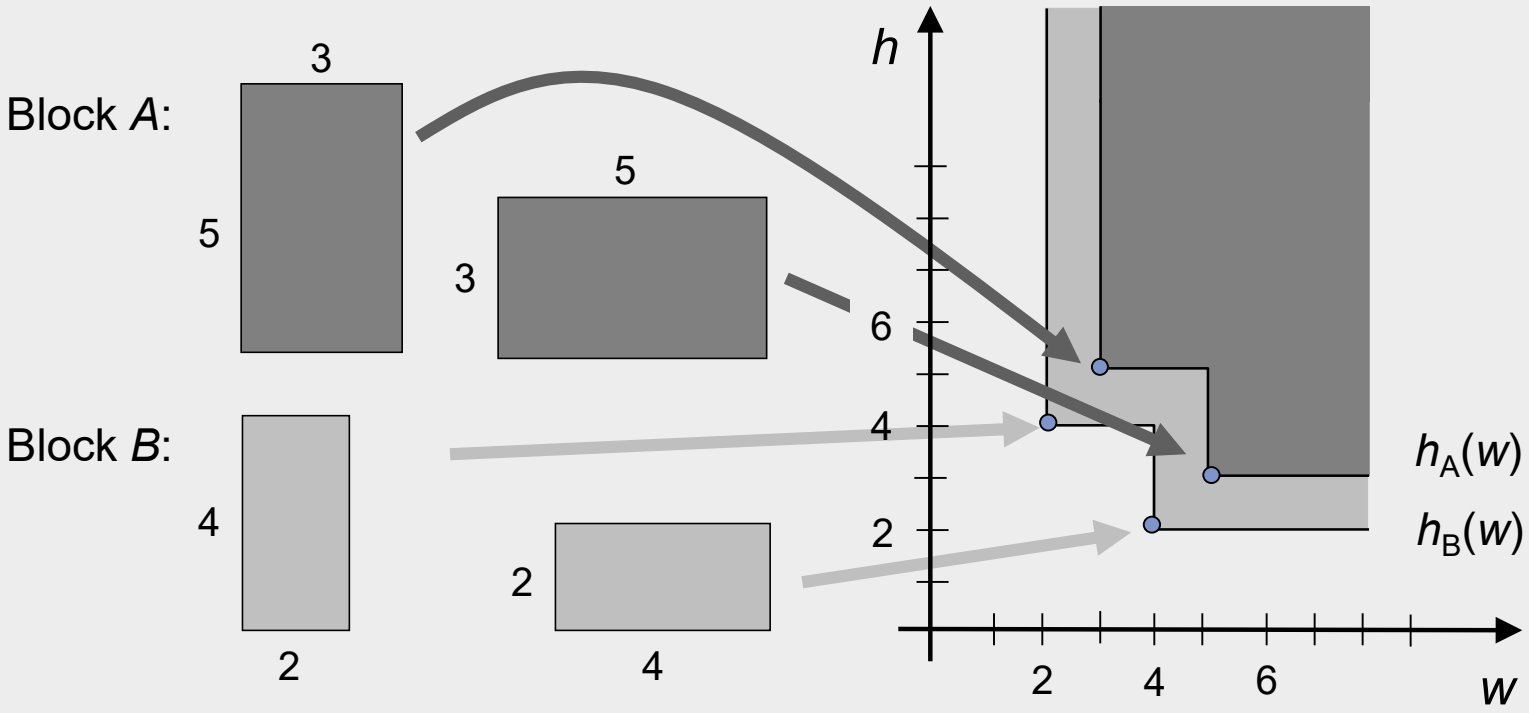
1. Ermitteln der Formfunktionen der Blöcke



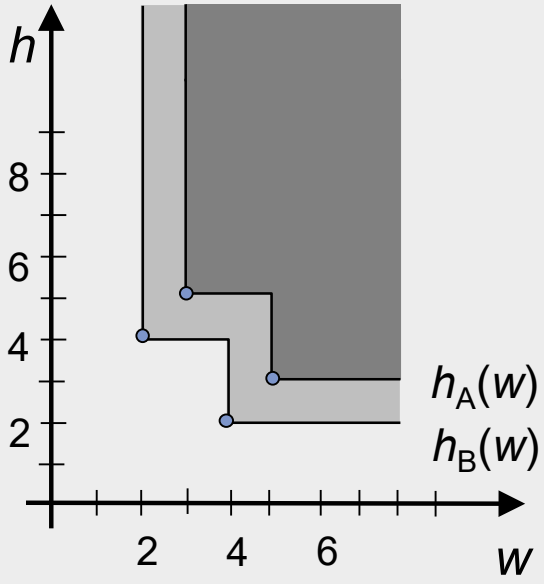
1. Ermitteln der Formfunktionen der Blöcke



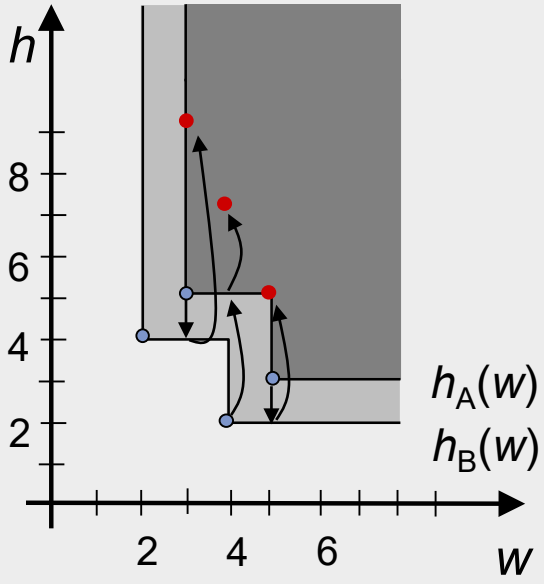
1. Ermitteln der Formfunktionen der Blöcke



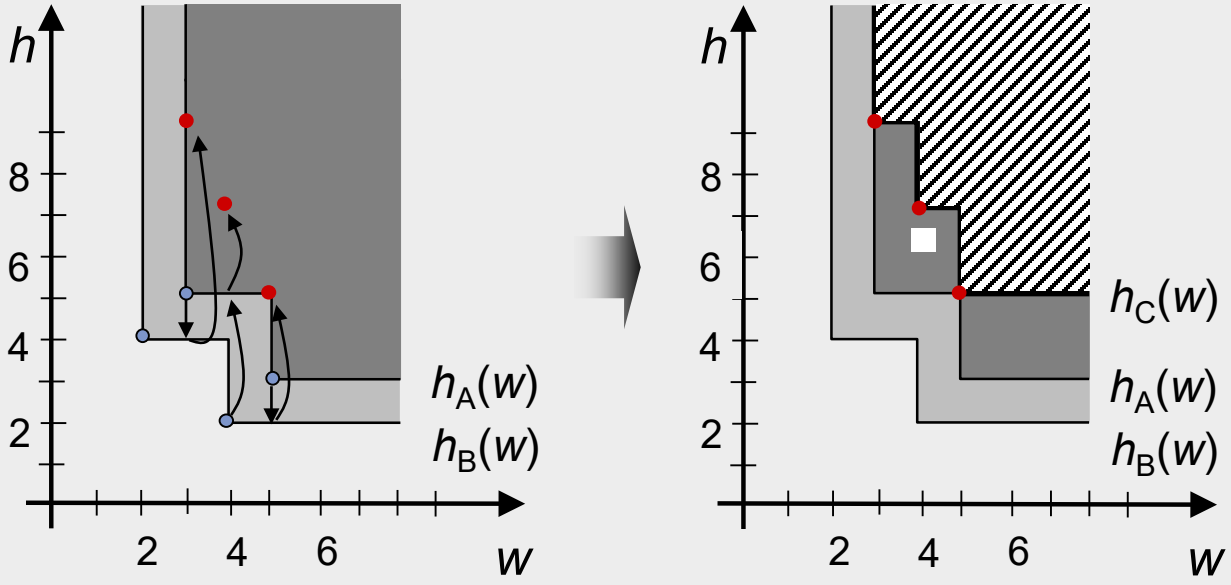
2. Ermitteln der Formfunktion der Topzelle
(vertikale Zusammensetzung)



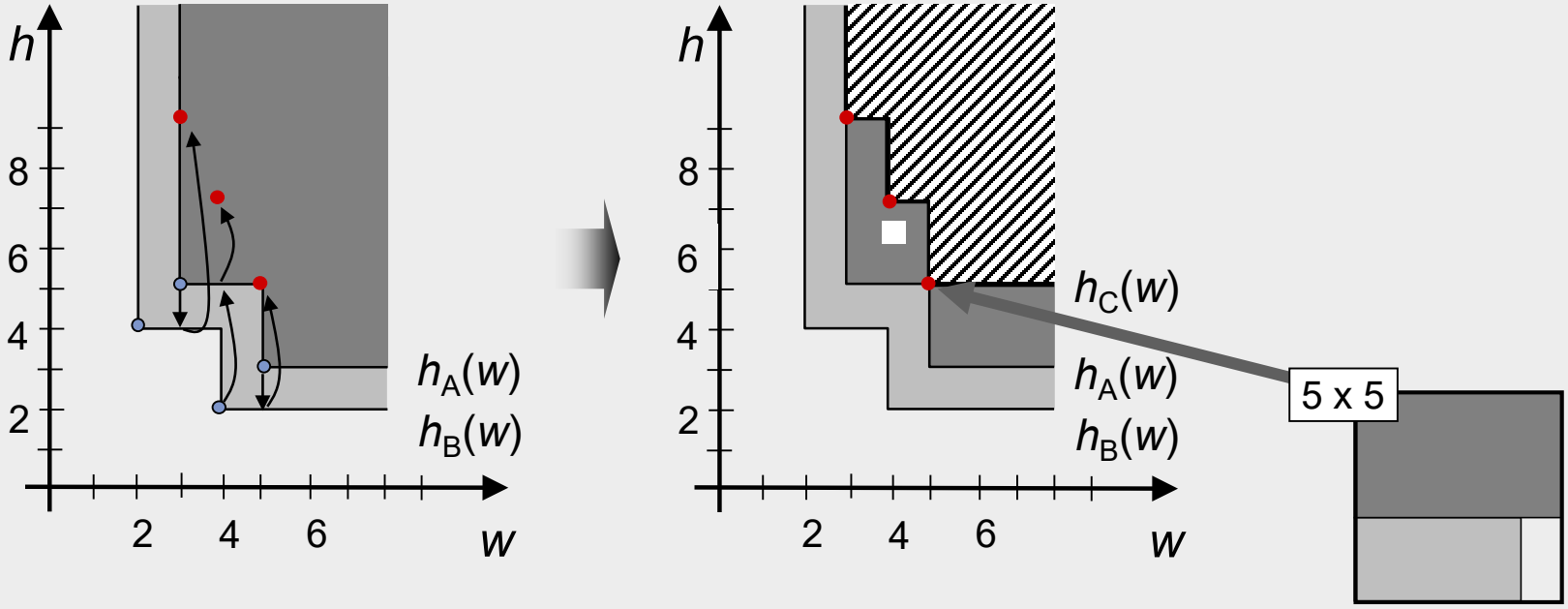
2. Ermitteln der Formfunktion der Topzelle
(vertikale Zusammensetzung)



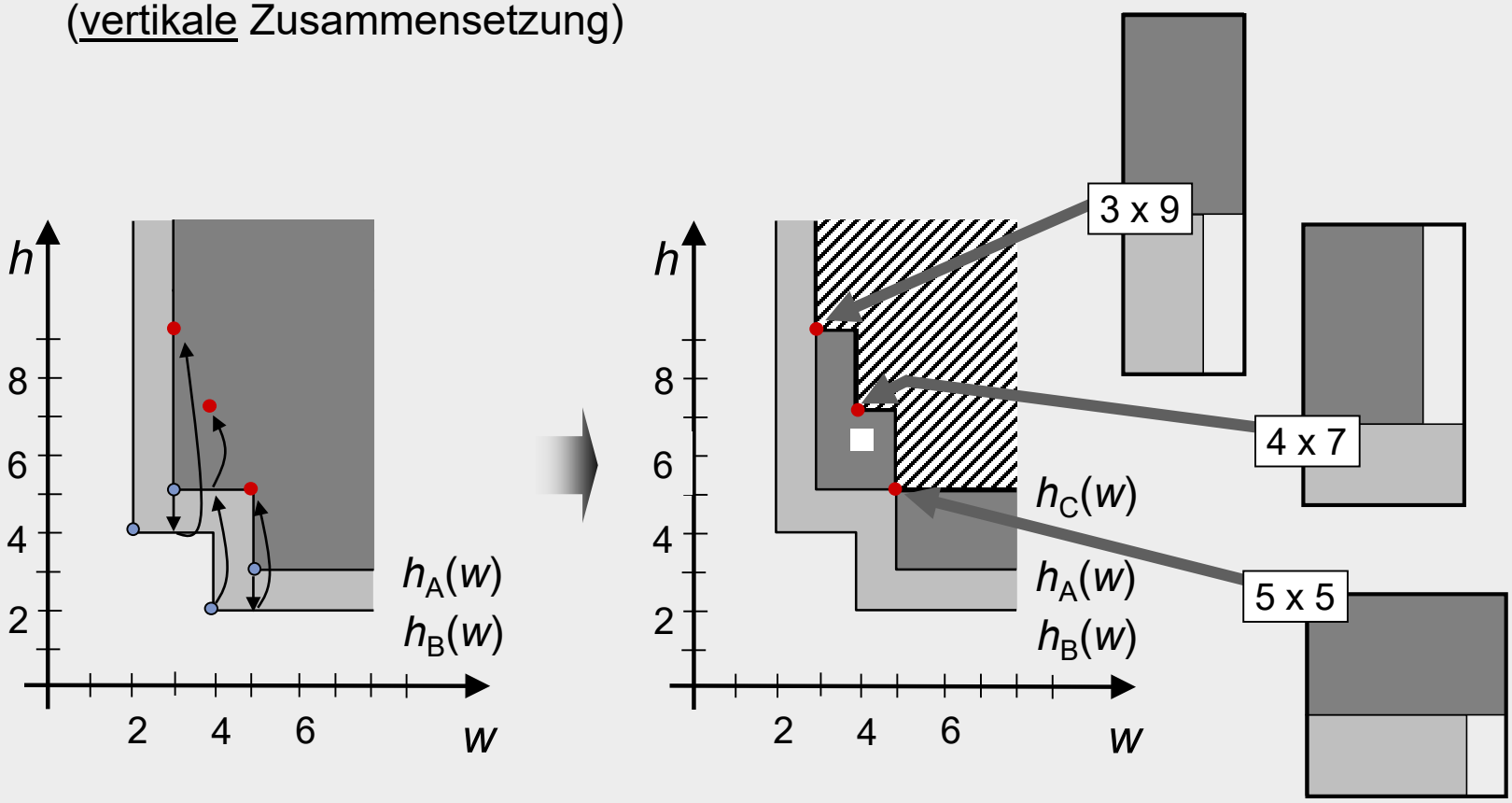
2. Ermitteln der Formfunktion der Topzelle
(vertikale Zusammensetzung)



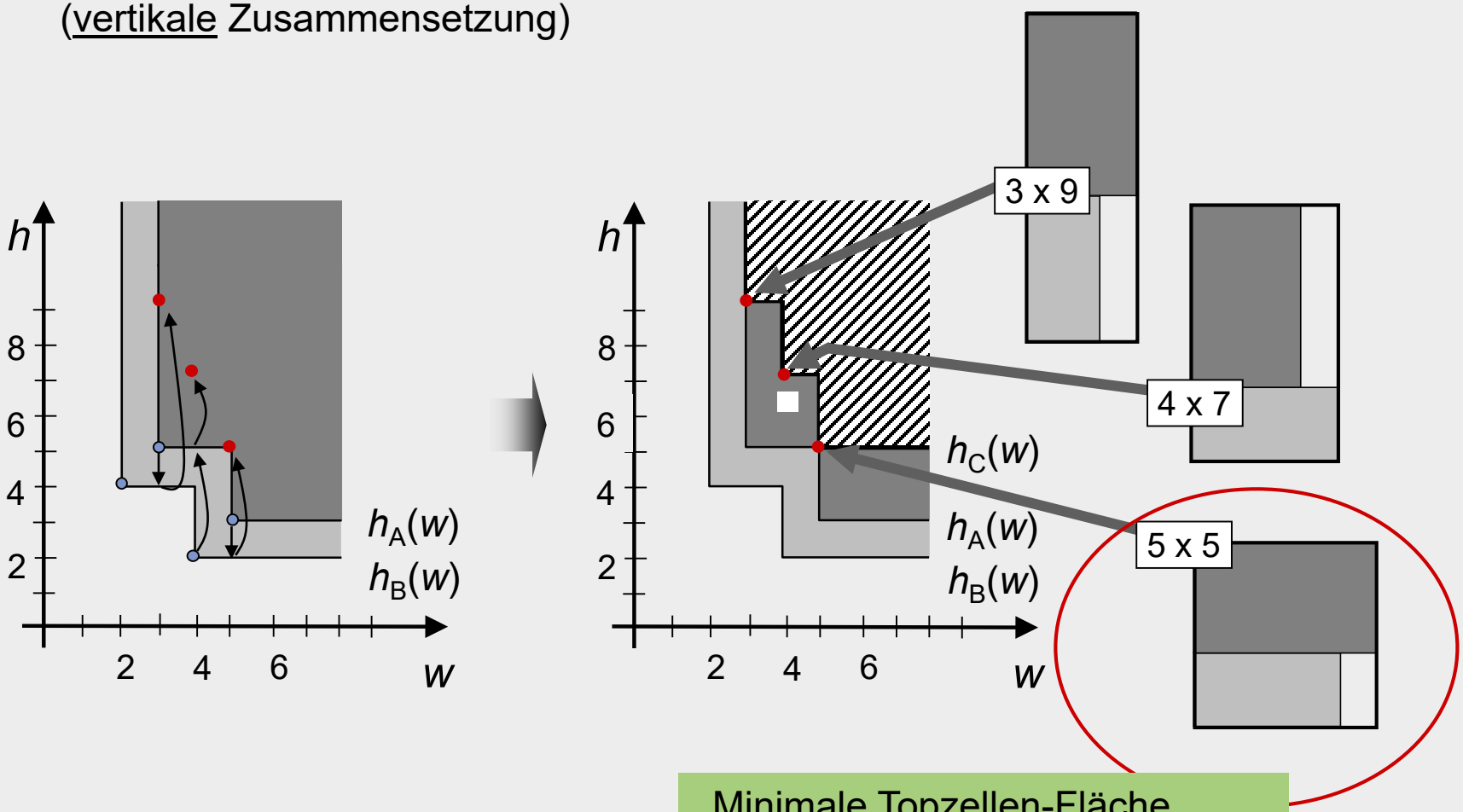
2. Ermitteln der Formfunktion der Topzelle
(vertikale Zusammensetzung)



2. Ermitteln der Formfunktion der Topzelle
(vertikale Zusammensetzung)

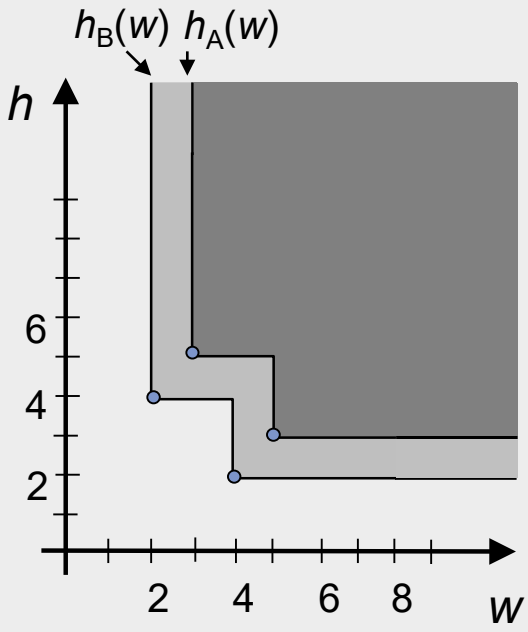


2. Ermitteln der Formfunktion der Topzelle
(<u>vertikale</u> Zusammensetzung)

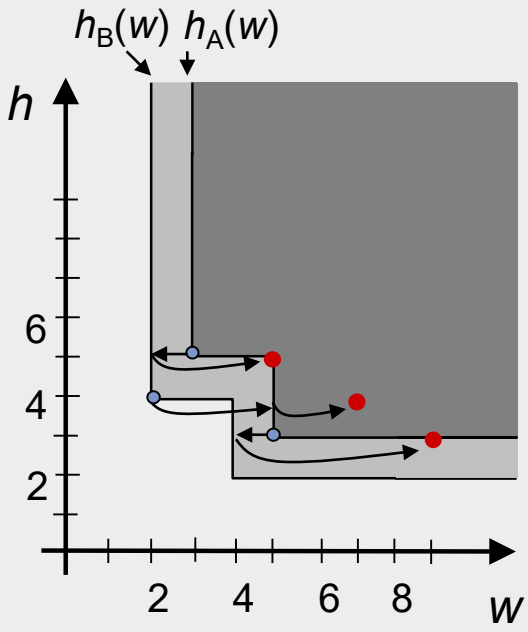


Minimale Topzellen-Fläche
bei vertikaler Zusammensetzung

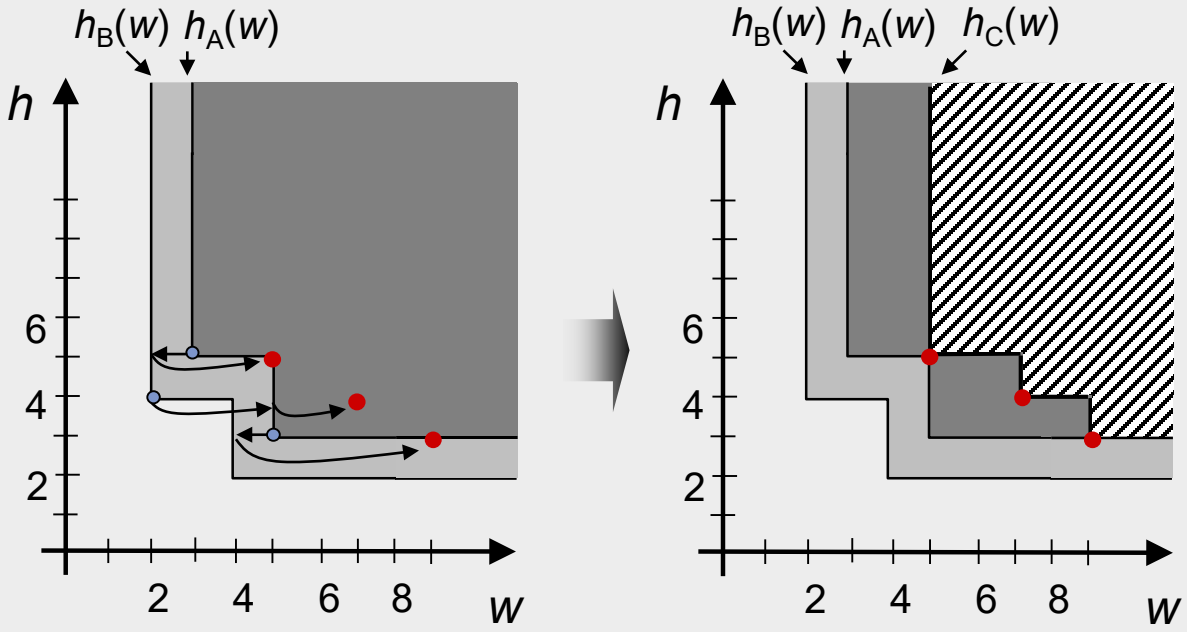
2. Ermitteln der Formfunktion der Topzelle
(horizontale Zusammensetzung)



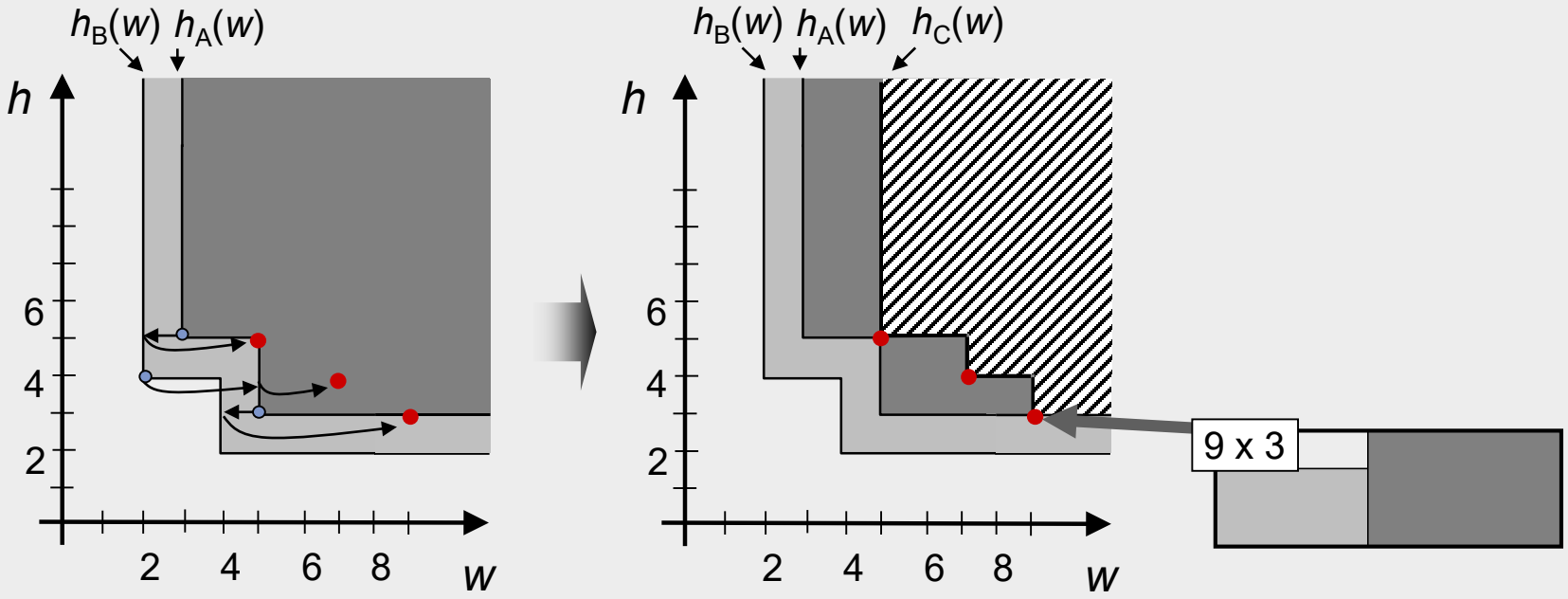
2. Ermitteln der Formfunktion der Topzelle
(horizontale Zusammensetzung)



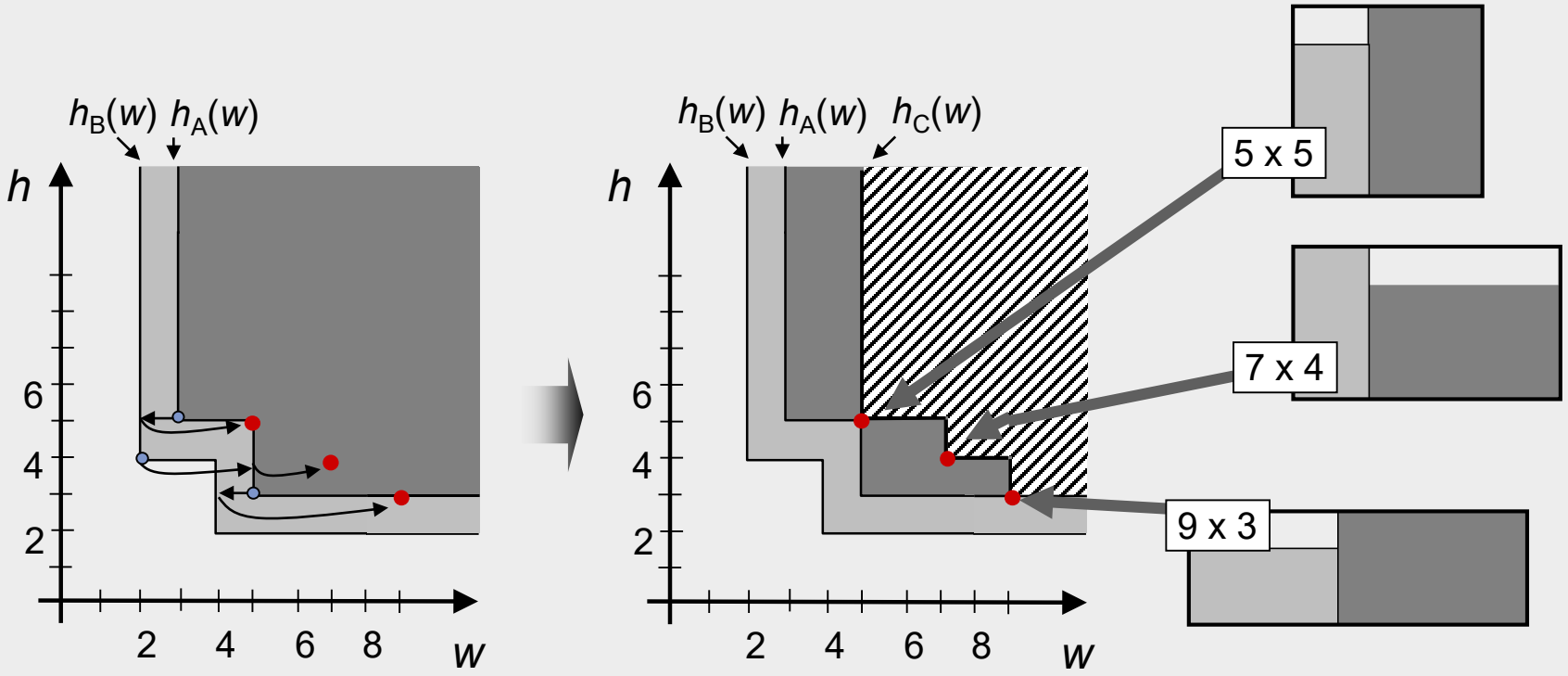
2. Ermitteln der Formfunktion der Topzelle
(horizontale Zusammensetzung)



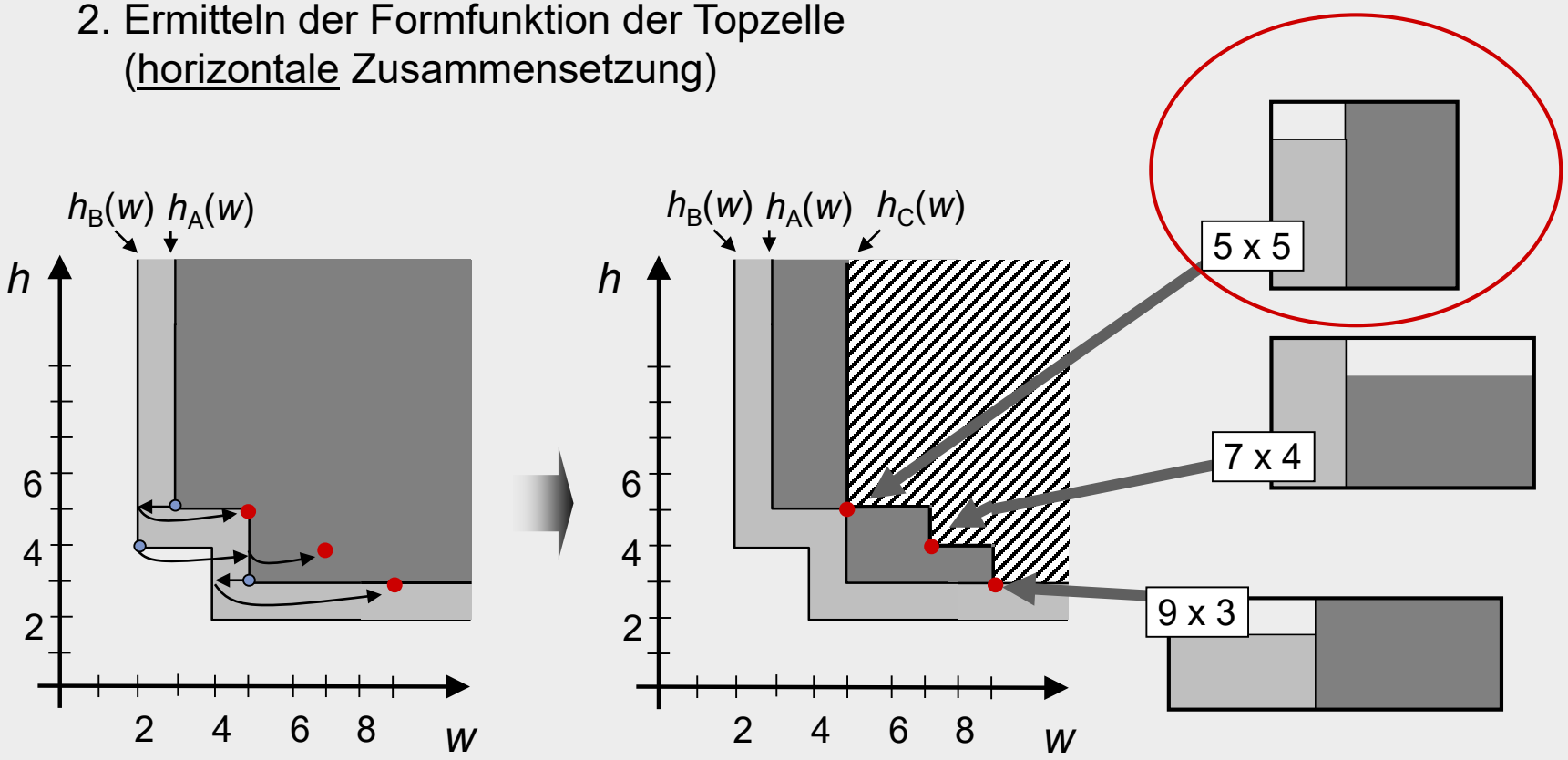
2. Ermitteln der Formfunktion der Topzelle
(horizontale Zusammensetzung)



2. Ermitteln der Formfunktion der Topzelle
(horizontale Zusammensetzung)

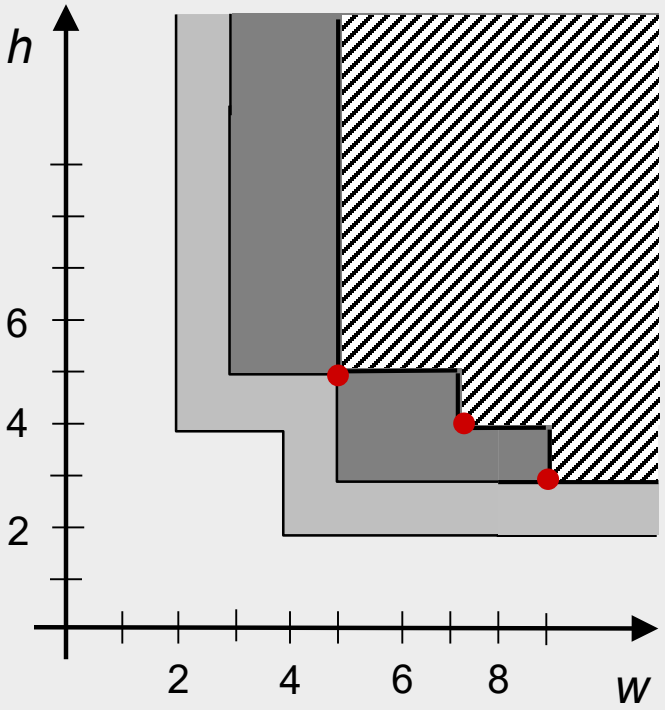


2. Ermitteln der Formfunktion der Topzelle
(horizontale Zusammensetzung)

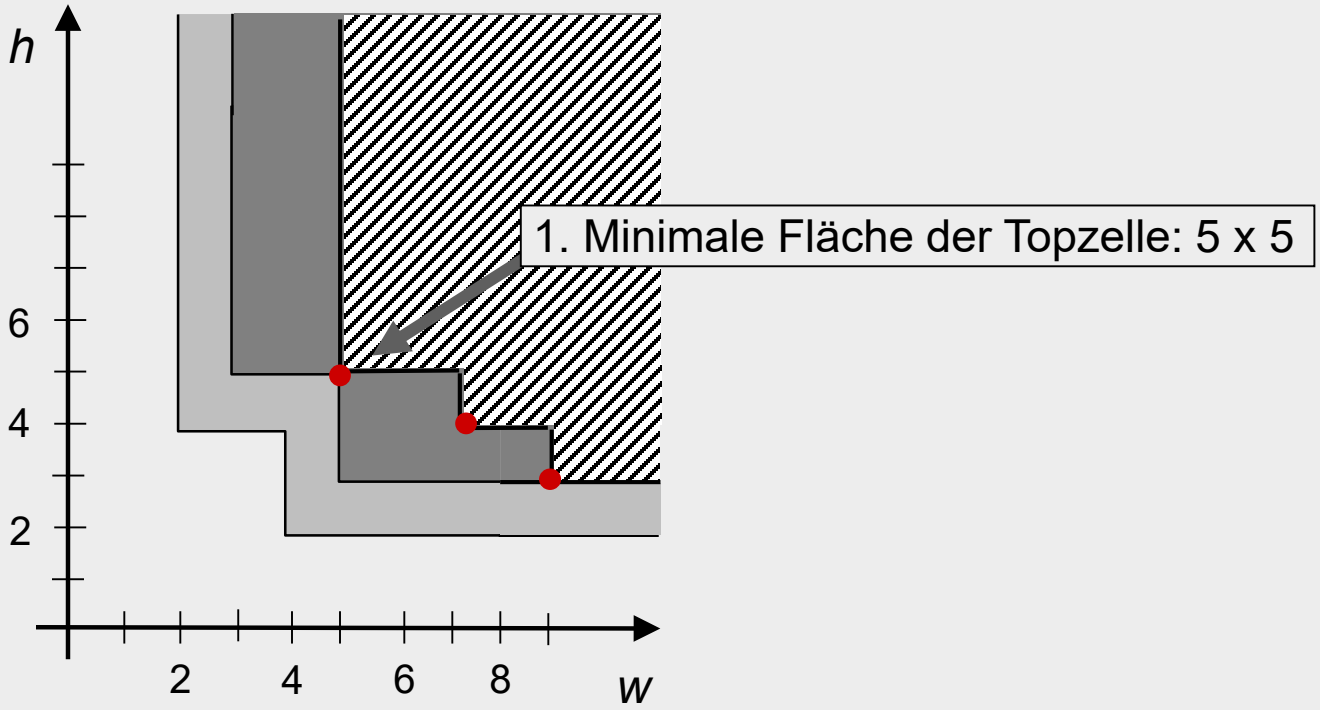


Minimale Topzellen-Fläche
bei horizontaler Zusammensetzung

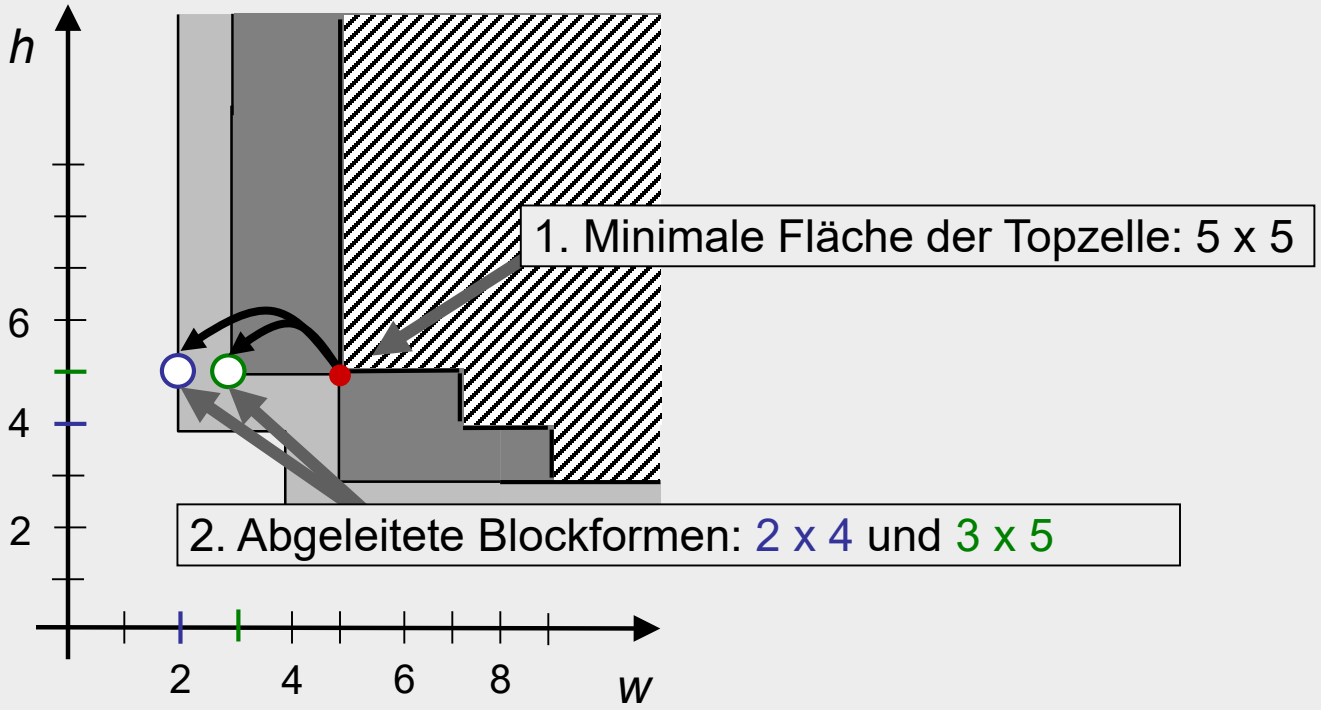
3. Formfestlegung von Topzelle und Blöcken
(horizontale Zusammensetzung)



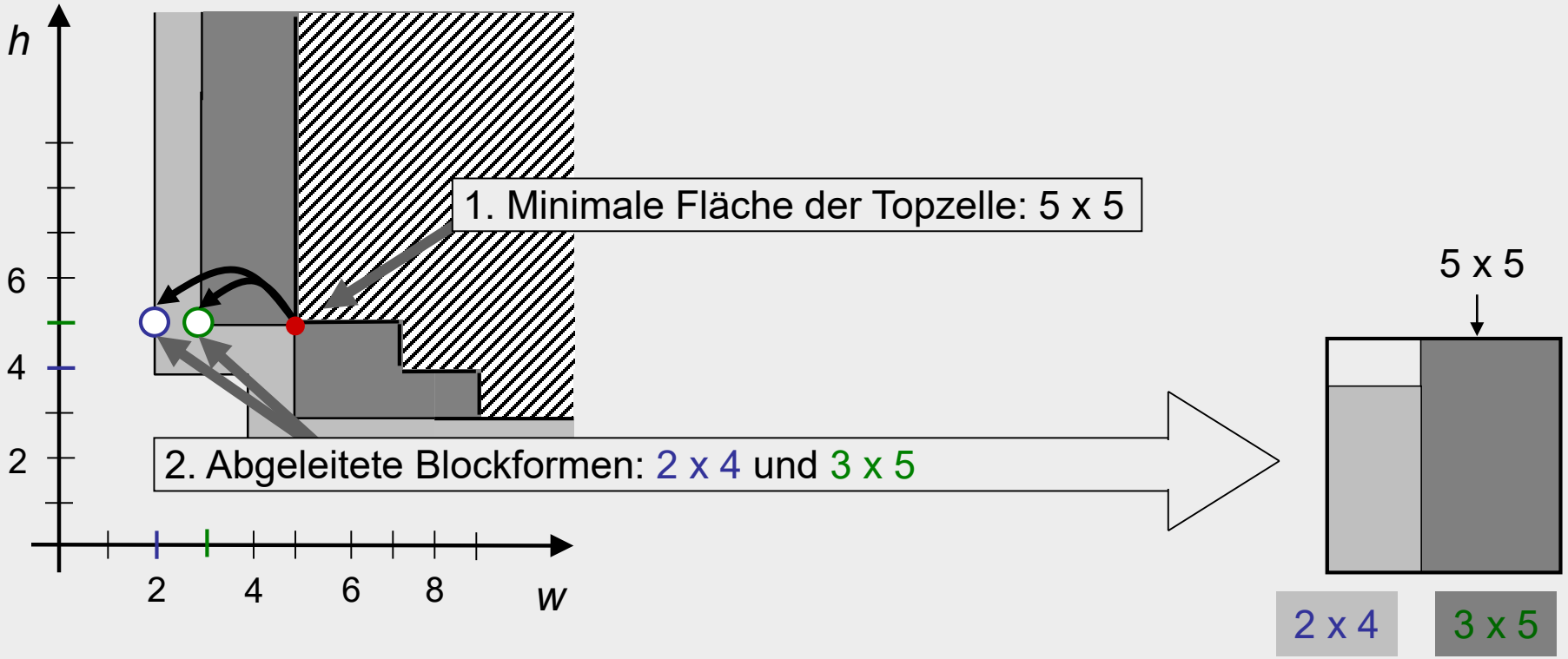
3. Formfestlegung von Topzelle und Blöcken
(horizontale Zusammensetzung)

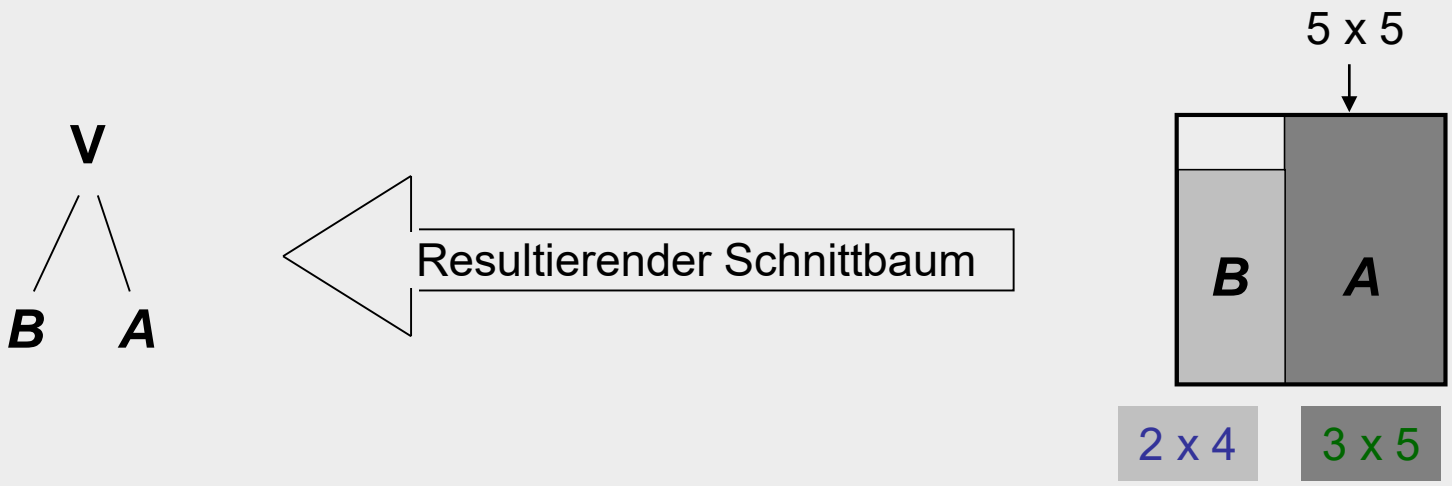


3. Formfestlegung von Topzelle und Blöcken
(horizontale Zusammensetzung)



3. Formfestlegung von Topzelle und Blöcken
(horizontale Zusammensetzung)



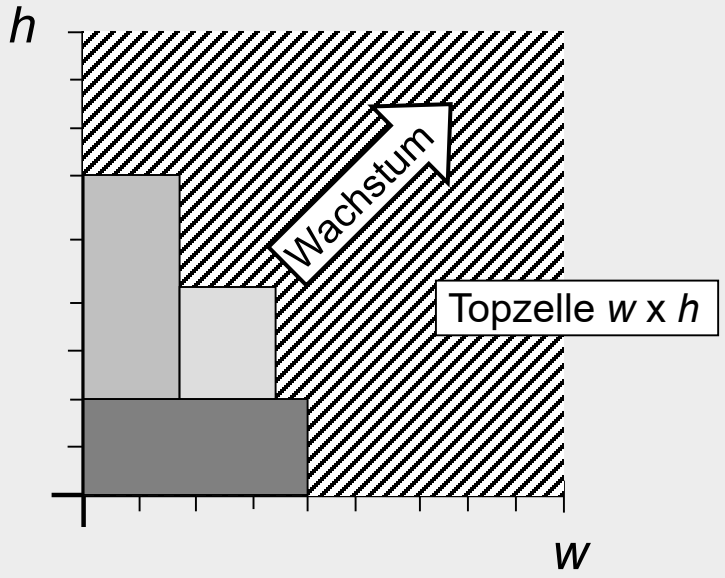

V
B
A
Resultierender Schnittbaum
5 x 5
B
A
2 x 4
3 x 5

Wiederholung:

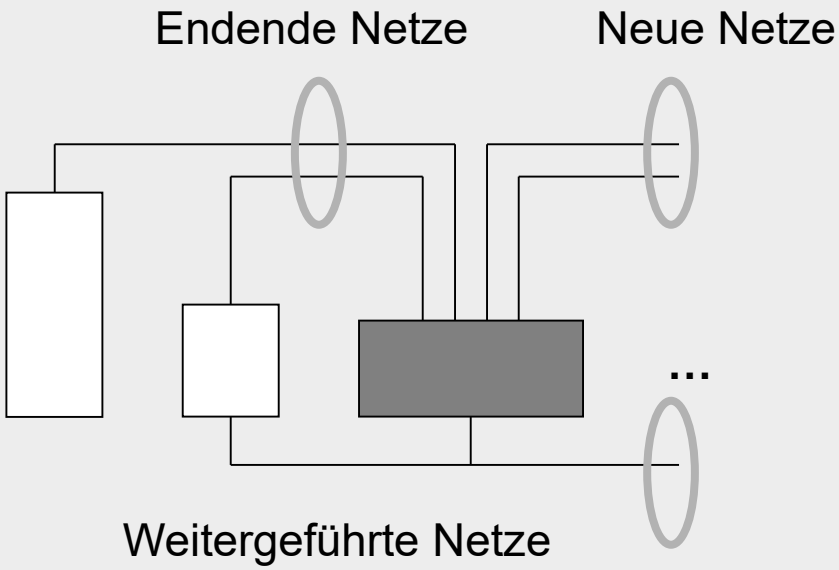
1. Ermittlung der Formfunktionen aller in der Topzelle anzuordnenden Blöcke.
2. Bottom-up: Ermittlung der Formfunktion der Topzelle und deren optimale Größe bzw. Form.
3. Top-down: Von der Formfestlegung der Topzelle ausgehend, Form und Anordnung der Einzelblöcke bestimmen.

3.4.2 Cluster-Wachstum (Cluster Growth)

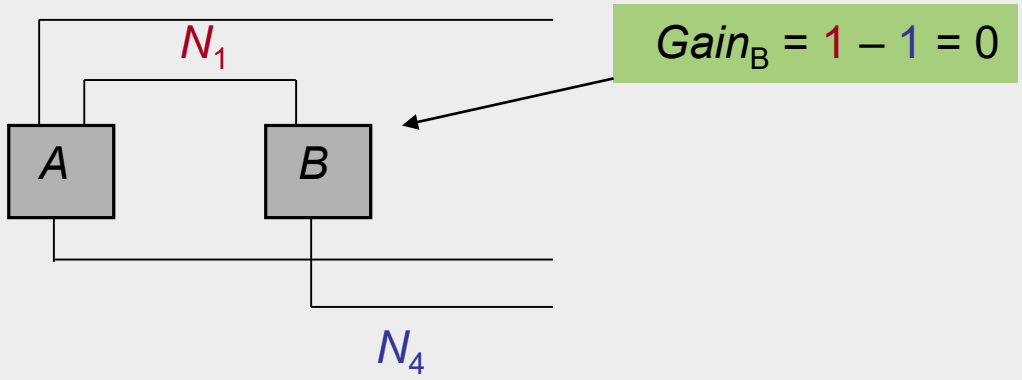
- Konstruktiver Algorithmus:
  Sequentielle Anordnung von festen Blöcken (Hard blocks)
- Beginnend mit einem Initialblock werden die Blöcke einzeln und unter Berücksichtigung einer vertikalen, diagonalen und horizontalen Wachstumsrichtung angefügt
- Wachstumsrichtung ist durch die angestrebte Außenform der Topzelle vorgegeben
- Blockreihenfolge:
  Algorithmus zur linearen Anordnung



- Lineare Blockanordnung mit dem Ziel der Minimierung der Netze, die bei einem beliebigen Schnitt durch diese Blockfolge aufzutrennen sind
- Drei Netzklassen für jeden Block:



- Reihenfolge der Blockanordnung ergibt sich aus der jeweiligen Netzanzahl innerhalb der mit einem Block verbundenen Netzklassen
- Gewinn (Gain) eines Blockes *m*:
  $Gain_m$ = (Anzahl der in *m* endenden Netze) – (von *m* ausgehende neue Netze)



- Es wird immer der Block mit höchstem *Gain*, der also mehr Netze abschließt als jeder andere Block, als nächster platziert

3.4.2 Cluster-Wachstum: Lineare Anordnung

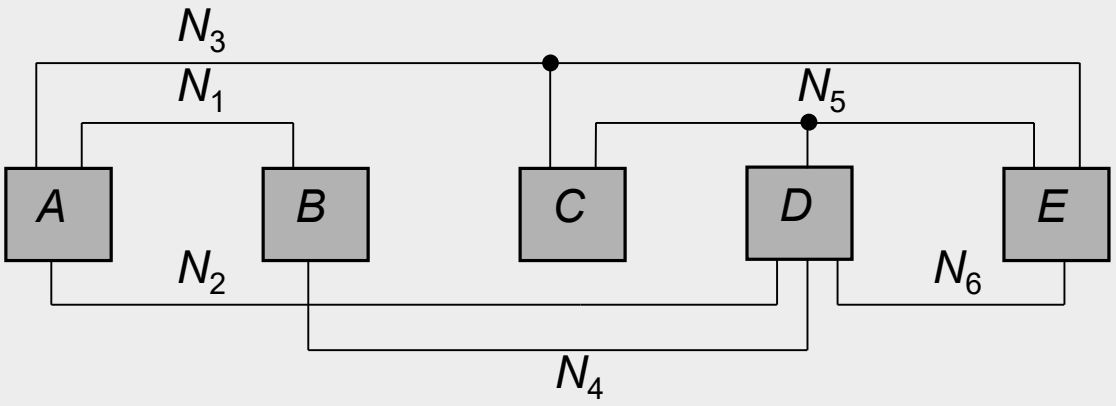
Algorithmus zur linearen Anordnung

S : Menge aller Blöcke
Order: Reihenfolge der Blöcke /* anfangs leer */
Begin
  Seed = Auswahl eines Anfangsblocks
  Order = [*Seed*]
  S = *S* – { *Seed* }
  Repeat
    ForEach Block *m* ∈ *S* **Do**
      Ermitteln des Gewinns (Gain) des aktuellen Blocks *m*
      $Gain_m$ =(Anzahl der in *m* endenden Netze)–(von *m* ausgehende neue Netze)
    End ForEach
    Auswahl des Blocks *m** mit maximalem Gewinn $Gain_m$
    If Gleichheit **Then**
      Auswahl des Blocks, der die meisten Netze beendet
    ElseIf Gleichheit **Then**
      Auswahl des Blocks mit der größten Anzahl weitergeführter Netze
    ElseIf Gleicheit **Then**
      Auswahl des Blocks mit der geringsten Anzahl von Verbindungen
    Else
      Willkürliche Auswahl eines Blocks
    Order = [!*Order*, *m**] /* Hinzufügen von *m** zur existierenden Folge */
    S = *S* – { *m** }
  Until *S* = ∅
End.

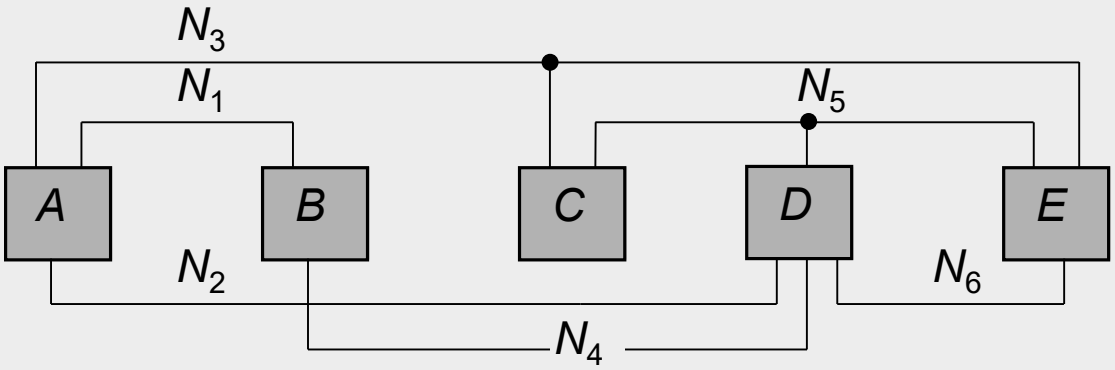
Nach Sait, S. M., Youssef, H.: VLSI Physical Design Automation

Gegeben:

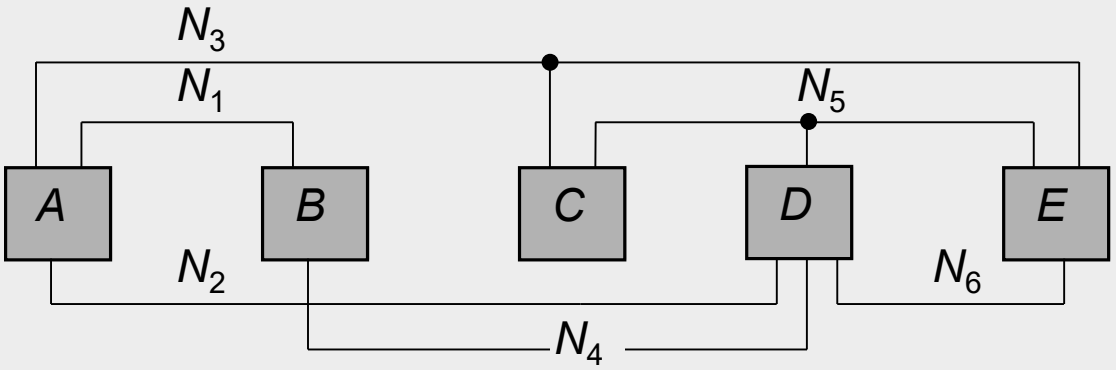
- Netzliste mit fünf Blöcken *A, B, C, D, E* und sechs Netzen
  $N_1 = \{A, B\}$
  $N_2 = \{A, D\}$
  $N_3 = \{A, C, E\}$
  $N_4 = \{B, D\}$
  $N_5 = \{C, D, E\}$
  $N_6 = \{D, E\}$
- Anfangsblock: *A*



Gesucht: Lineare Anordnung mit minimalen Netzkosten



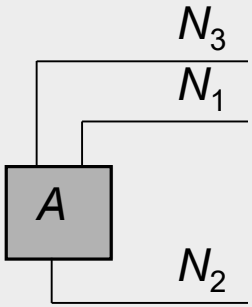
Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze

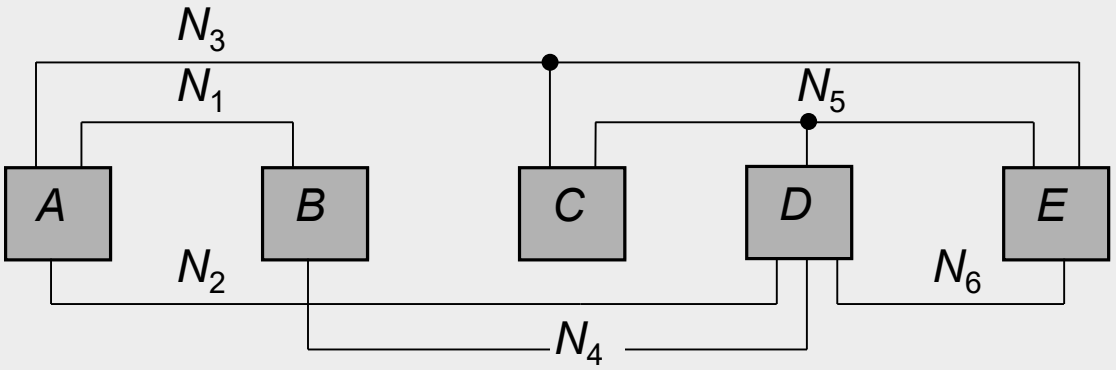


Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-

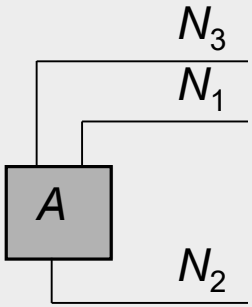
Geg.: Anfangsblock A

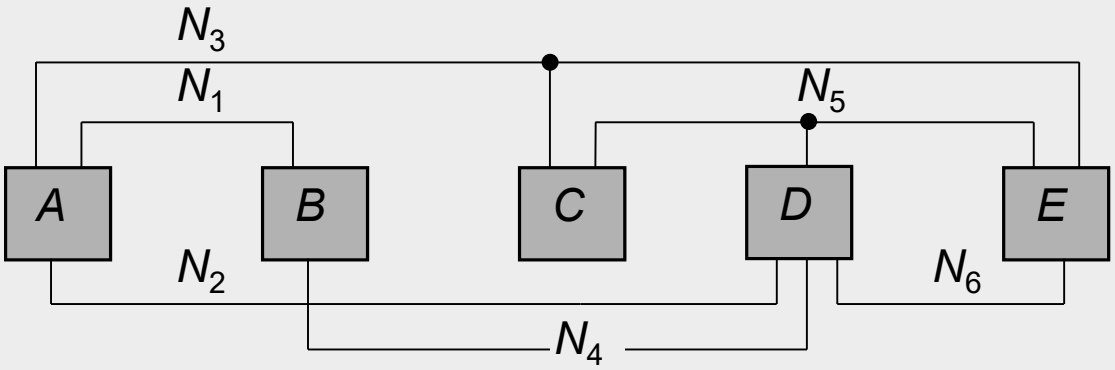
$Gain_A$ = (Anzahl der in *A* endenden Netze) – (von *A* ausgehende neue Netze)



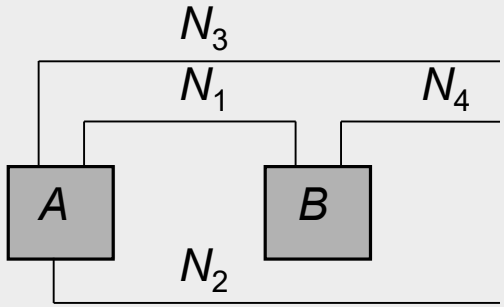


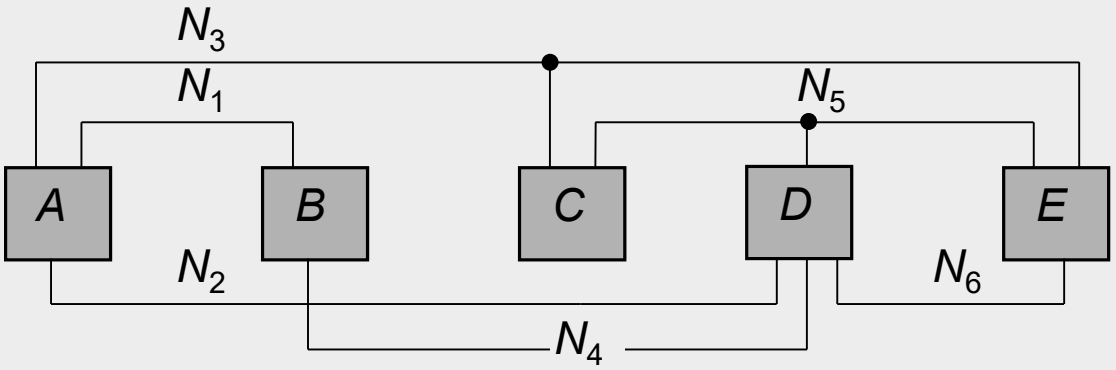
Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-
1	*B*	N_1	N_4	0	-
	C	-	N_5	-1	N_3
	D	N_2	N_4, N_5, N_6	-2	-
	E	-	N_5, N_6	-2	N_3



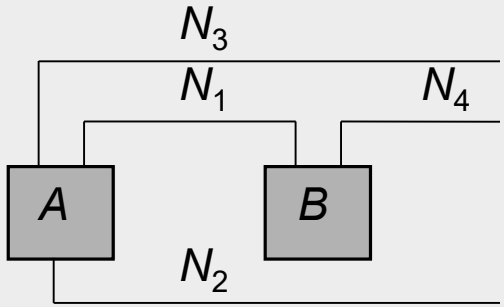


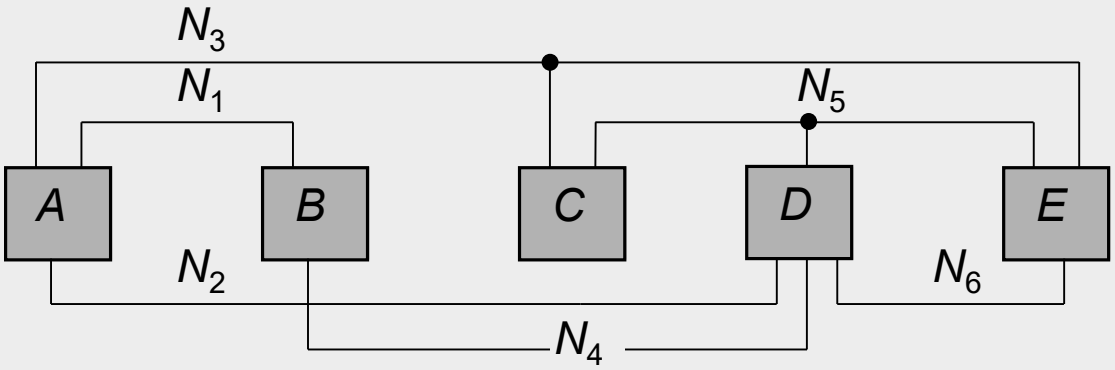
Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-
1	*B*	N_1	N_4	0	-
	C	-	N_5	-1	N_3
	D	N_2	N_4, N_5, N_6	-2	-
	E	-	N_5, N_6	-2	N_3



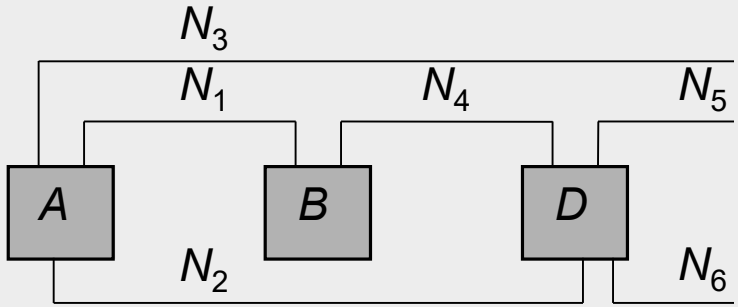


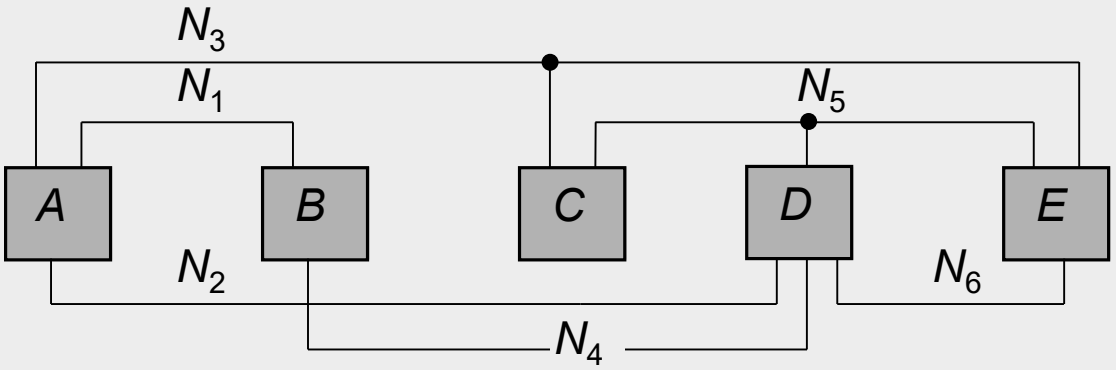
Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-
1	*B*	N_1	N_4	0	-
	C	-	N_5	-1	N_3
	D	N_2	N_4, N_5, N_6	-2	-
	E	-	N_5, N_6	-2	N_3
2	*C*	-	N_5	-1	N_3
	D	N_2, N_4	N_5, N_6	0	-
	E	-	N_5, N_6	-2	N_3



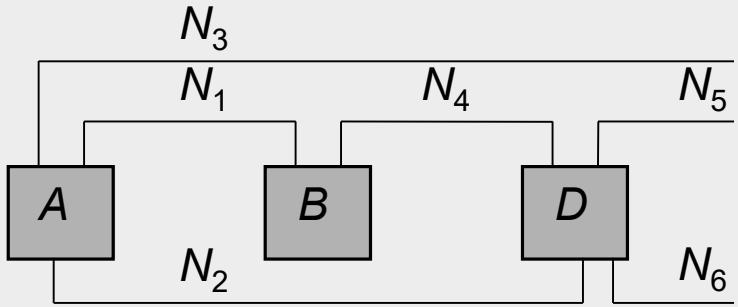


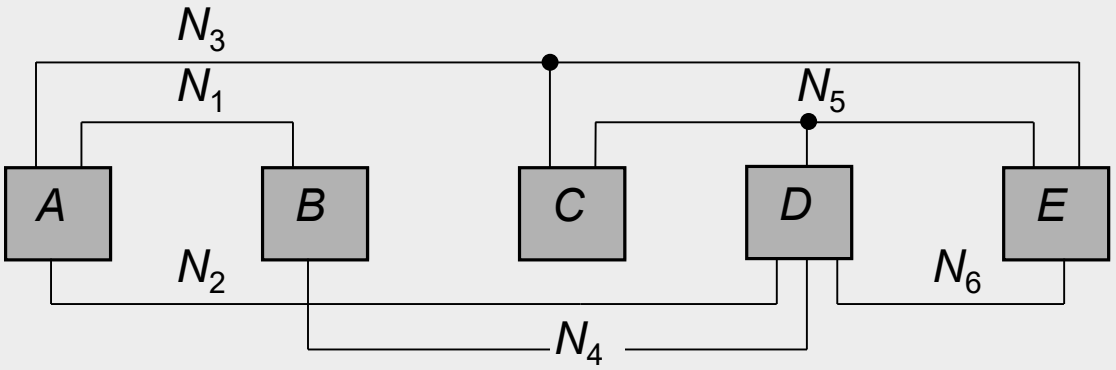
Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-
1	*B*	N_1	N_4	0	-
	C	-	N_5	-1	N_3
	D	N_2	N_4, N_5, N_6	-2	-
	E	-	N_5, N_6	-2	N_3
2	*C*	-	N_5	-1	N_3
	D	N_2, N_4	N_5, N_6	0	-
	E	-	N_5, N_6	-2	N_3



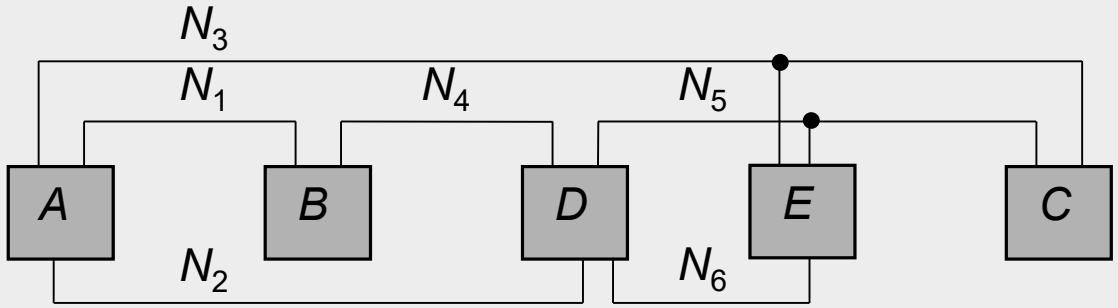


Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-
1	*B*	N_1	N_4	0	-
	C	-	N_5	-1	N_3
	D	N_2	N_4, N_5, N_6	-2	-
	E	-	N_5, N_6	-2	N_3
2	*C*	-	N_5	-1	N_3
	D	N_2, N_4	N_5, N_6	0	-
	E	-	N_5, N_6	-2	N_3
3	*C*	-	-	0	N_3, N_5
	E	N_6	-	+1	N_3, N_5

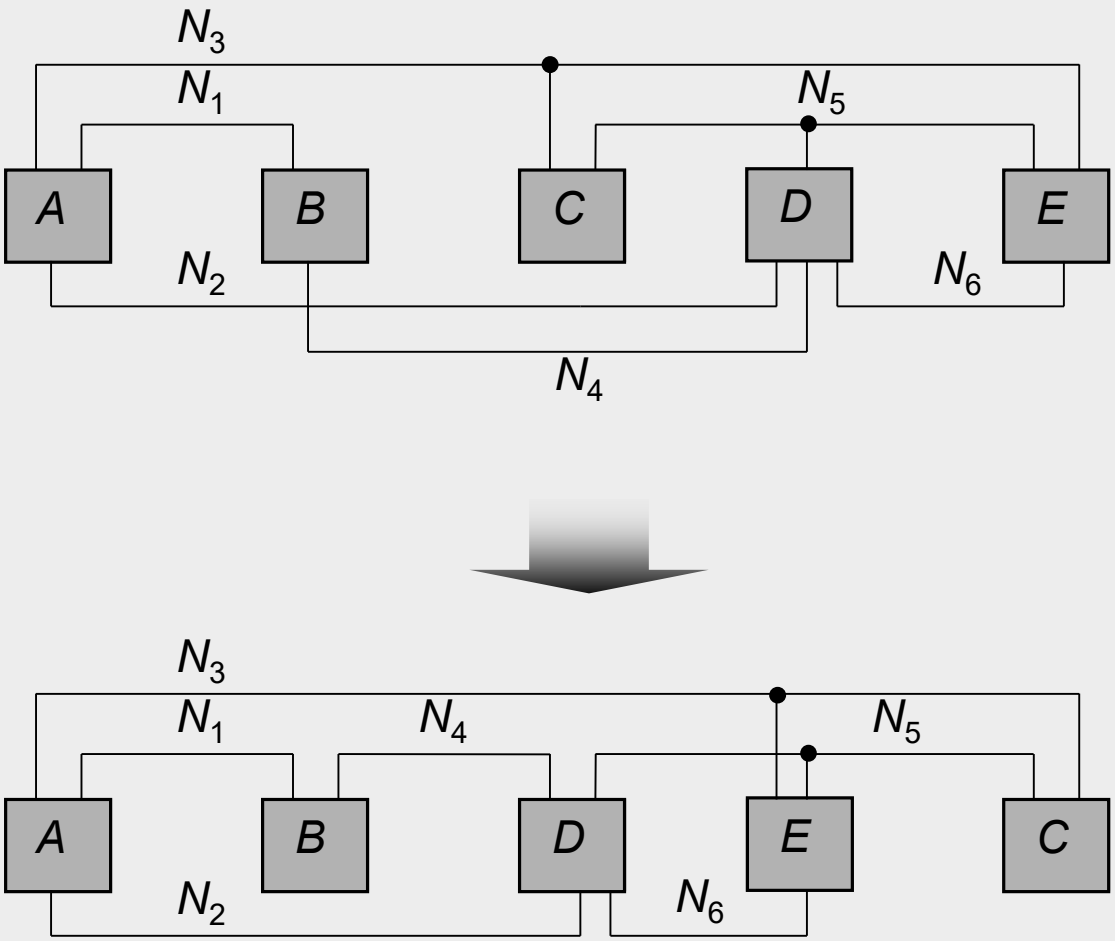




Schritt #	Blöcke	Endende Netze	Neue Netze	Gewinn (*Gain*)	Weitergeführte Netze
0	*A*	-	N_1, N_2, N_3	-3	-
1	*B*	N_1	N_4	0	-
	C	-	N_5	-1	N_3
	D	N_2	N_4, N_5, N_6	-2	-
	E	-	N_5, N_6	-2	N_3
2	*C*	-	N_5	-1	N_3
	D	N_2, N_4	N_5, N_6	0	-
	E	-	N_5, N_6	-2	N_3
3	*C*	-	-	0	N_3, N_5
	E	N_6	-	+1	N_3, N_5
4	*C*	$N_3, N_5,$	-	+2	-



3.4.2 Cluster-Wachstum: Lineare Anordnung – Beispiel



Cluster-Wachstums-Algorithmus

S : Menge aller Blöcke
Rest: Menge aller noch nicht platzierten Blöcke
Begin
    Order = Algorithmus zur linearen Anordnung (*S*)
    Repeat
        nextBlock = *b* aus *Order* = [*b*, !*Rest*]
        Order = *Rest*
        Platzierung und Orientierung von *b* mit minimaler Erhöhung der Kosten
        /* Kosten ergeben sich aus Größe bzw. Form der Topzelle, Verbindungslänge usw. */
    Until *Order* = ∅
End.

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

3.4.2 Cluster-Wachstum: Beispiel

Gegeben: Blöcke *A*, *B*, *C*, *D*, *E* mit freier Orientierung und der linearen Anordnung [*A, B, D, E, C*] sowie festen Abmessungen.

Gesucht: Topzelle mit minimaler Fläche

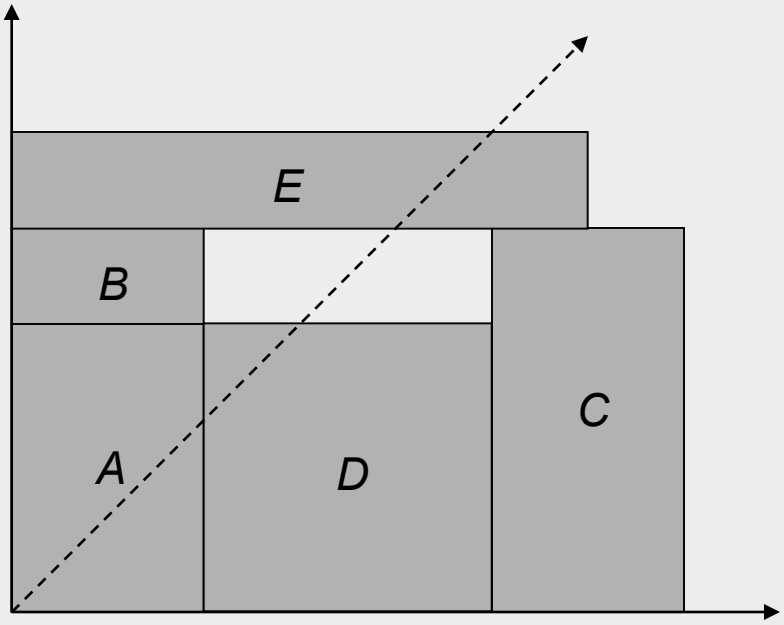
Block	Breite *w*	Höhe *h*
A	2	3
B	2	1
C	2	4
D	3	3
E	6	1

Gegeben: Blöcke *A*, *B*, *C*, *D*, *E* mit freier Orientierung und der linearen Anordnung [*A, B, D, E, C*] sowie festen Abmessungen.

Gesucht: Topzelle mit minimaler Fläche

Block	Breite *w*	Höhe *h*
A	2	3
B	2	1
C	2	4
D	3	3
E	6	1

Lösung:



3.4.3 Weitere Algorithmen für das Floorplanning

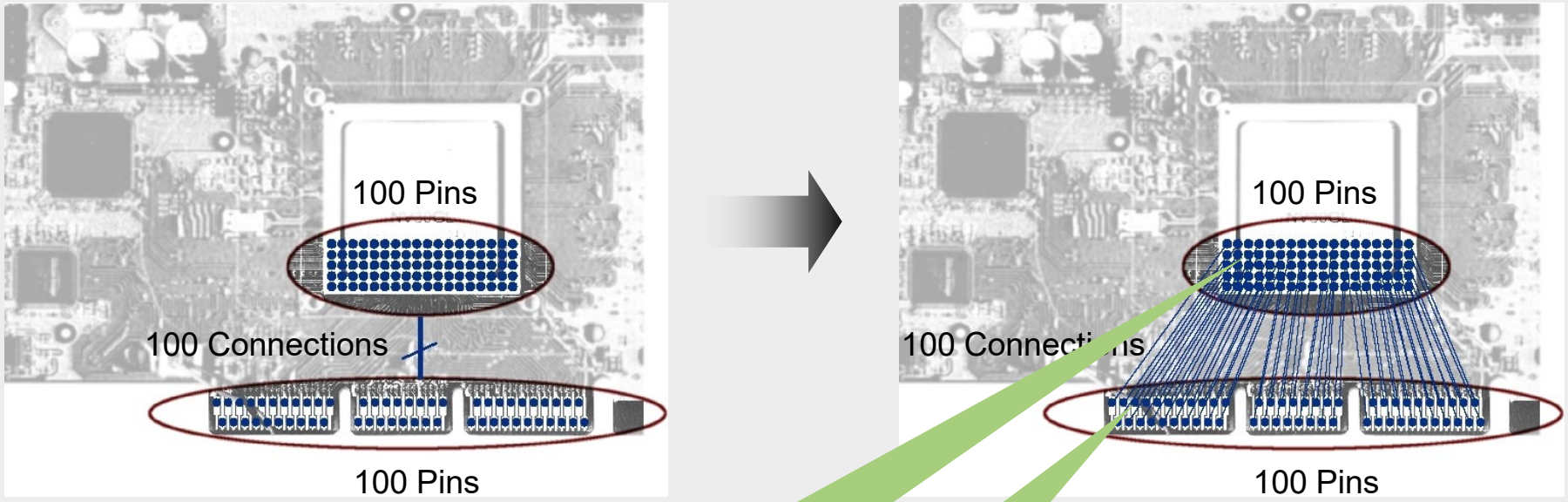
- Simulated-Annealing-Algorithmen
- Floorplanning mittels Gleichungssystemen („Integer programming“)

Kapitel 3 – Floorplanning

3.5 Pinzuordnung

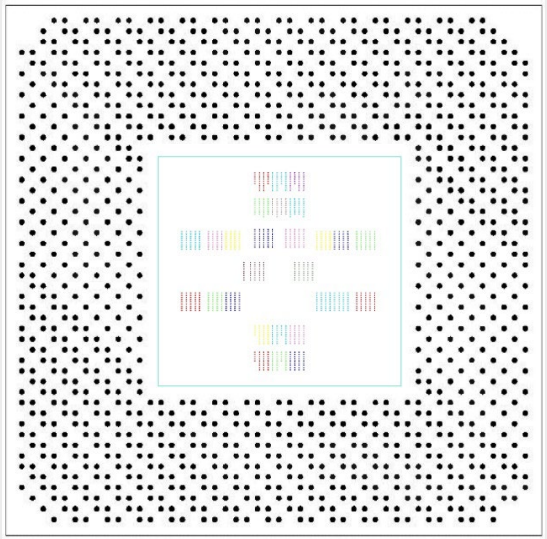
- Bei der Partitionierung anfallende Teilschaltungen (Blöcke) besitzen eine Menge von externen Netzen, mit denen sie untereinander in Verbindung stehen
- Außenanschlüsse: Pinanschlüsse am Rand der Teilschaltungen, zwischen denen die externe Verdrahtung zu realisieren ist
- Die Lage der Außenanschlüsse ist in den meisten Fällen vorgegeben, jedoch nicht ihre Zuordnung zu den einzelnen Netzen

Aufgabe der Pinzuordnung bei Blöcken ist es, jedem Außenanschluss eines Blocks ein Netz so zuzuordnen, dass die anschließende Verdrahtung sowohl innerhalb des Blocks als auch zwischen den Blöcken vereinfacht wird.

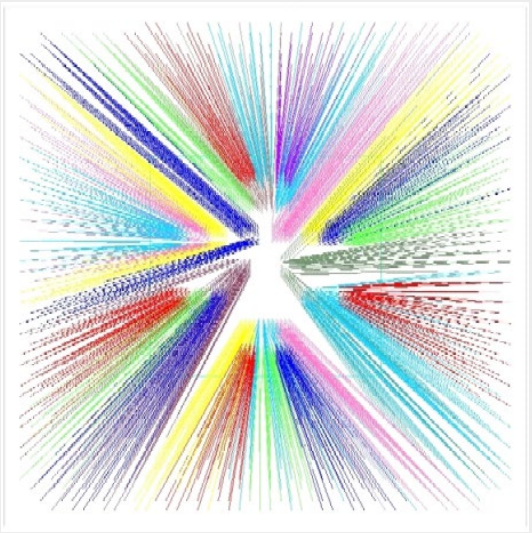


Beispiel: Netz 23 an Pin A_44

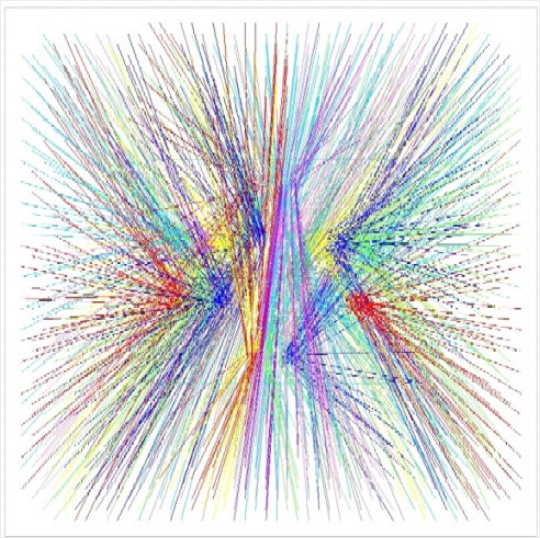
Netz 23 an Pin B_3



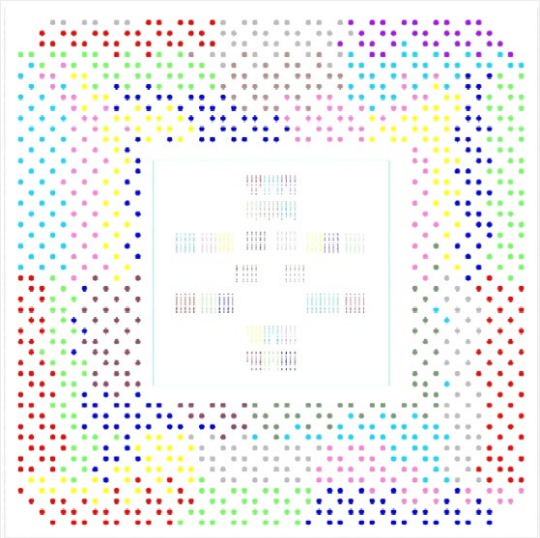
Zuordnungsaufgabe (Nacktchip auf MCM)



Flylines bei optimierter Zuordnung (kreuzungsfrei)



Flylines bei willkürlicher Zuordnung



Optimierte Pinzuordnung

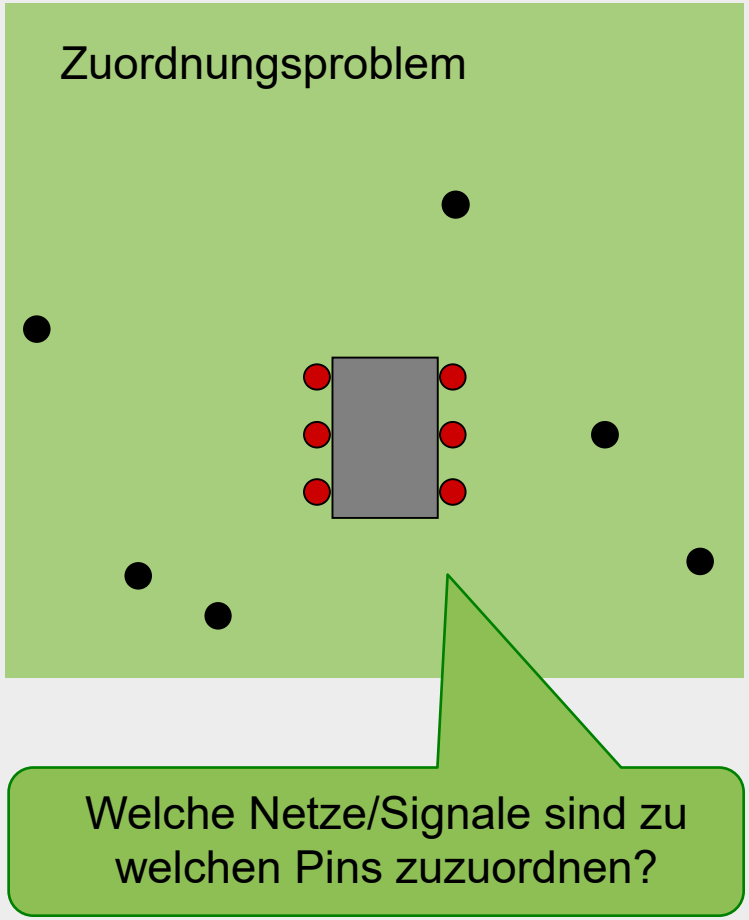
3.5.2 Pinzuordnung mittels konzentrischer Kreise

- Ziel: Planarisierung der Verbindungen zwischen einem Block und mit ihm verbundenen Anschlüssen (Minimierung von Verbindungsüberschneidungen)

- Wesentliches Merkmal dieser Vorgehensweise ist die Nutzung zweier konzentrischer Kreise:
  - Ein innerer Kreis zur Darstellung der (flexiblen) Pins des z.Zt. betrachteten Blocks und
  - ein äußerer Kreis zur Darstellung der (bereits zugeordneten) Verbindungsanschlüsse der anderen Blöcke.

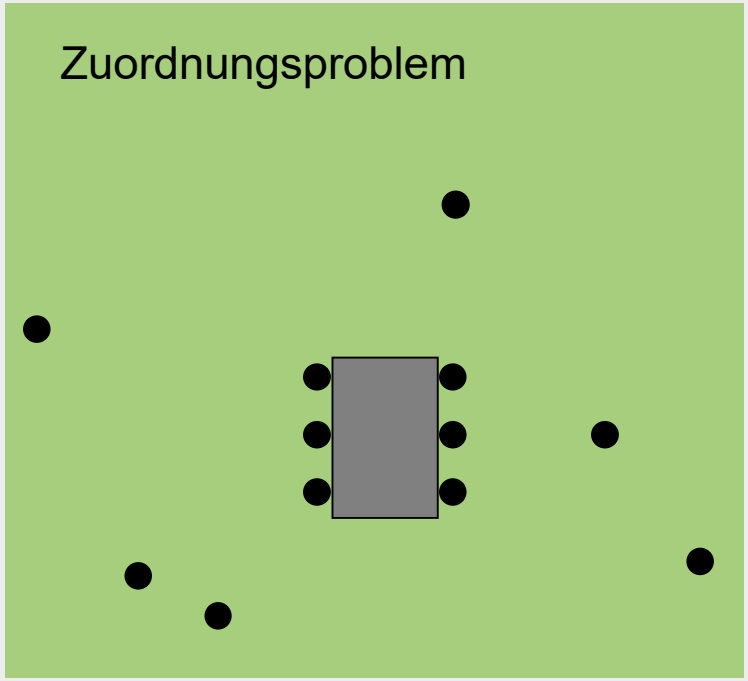
- Mittels einer Kreiszuordnung der inneren und äußeren Pins wird angestrebt, dass sich zu jedem äußeren Pin ein inneres Pin derart finden lässt, dass die resultierenden Verbindungen überschneidungsfrei sind.

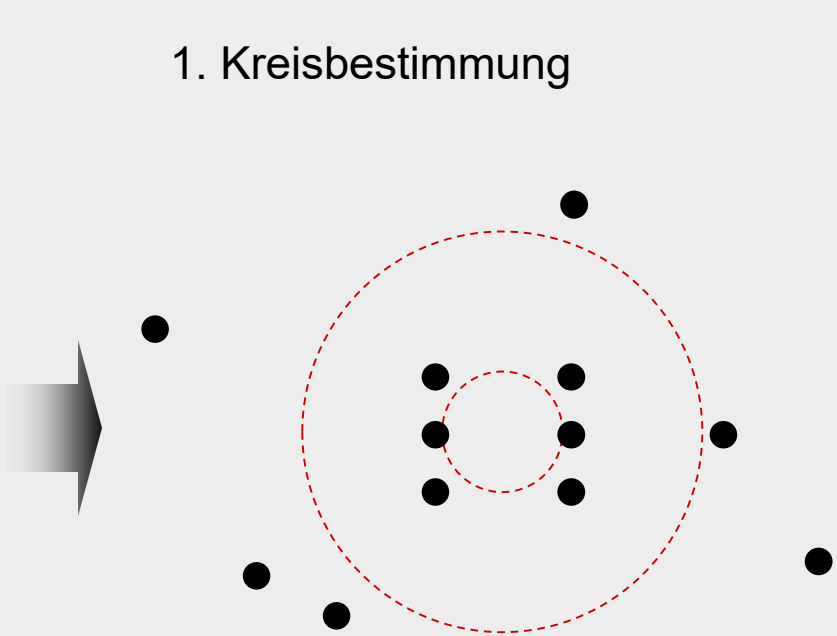
3.5.2 Pinzuordnung mittels konzentrischer Kreise: Beispiel



Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

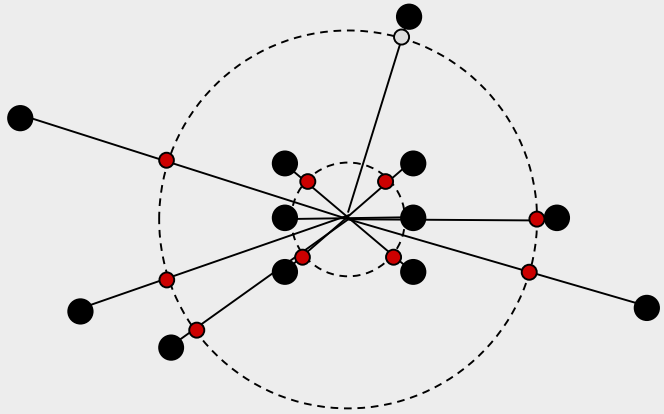
3.5.2 Pinzuordnung mittels konzentrischer Kreise: Beispiel





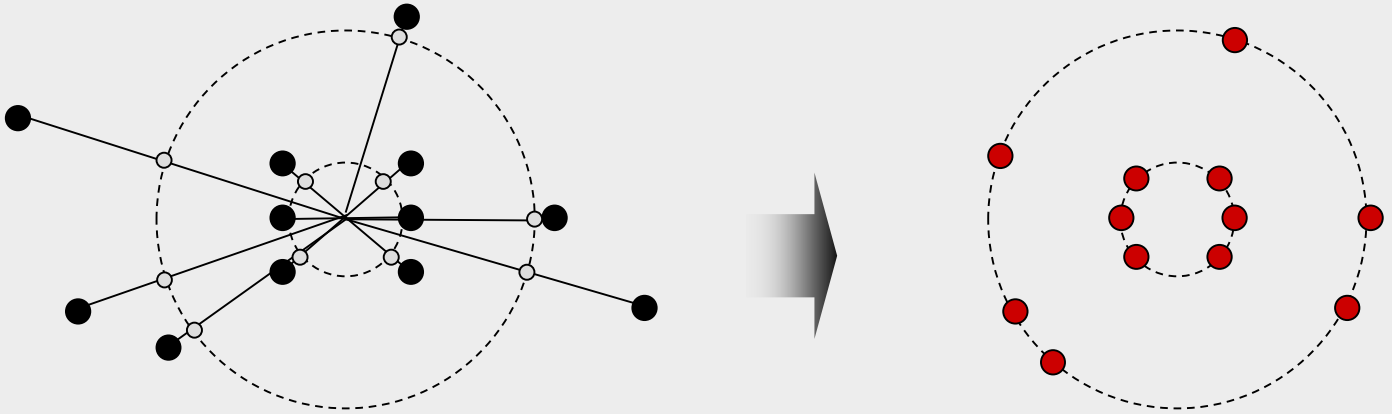
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

2. Punktbestimmung



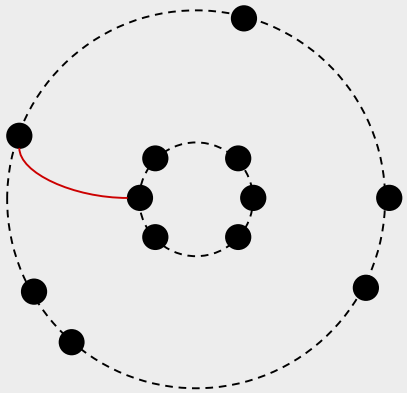
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

2. Punktbestimmung



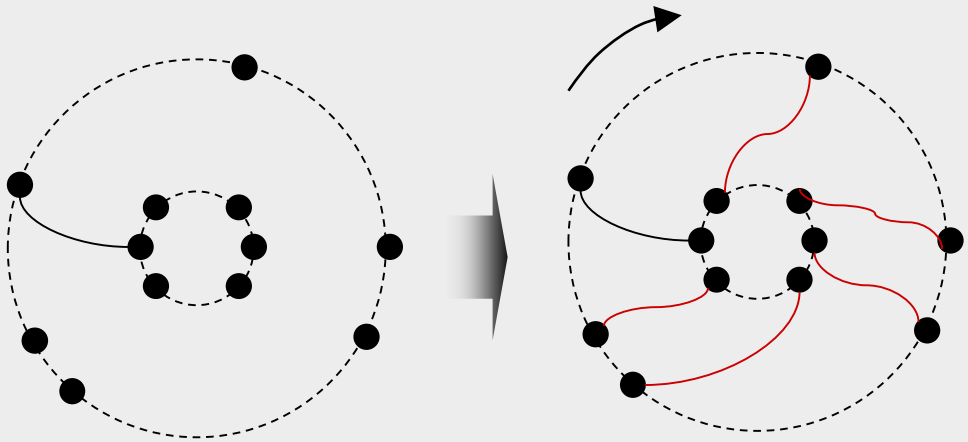
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

3. Anfangszuordnung



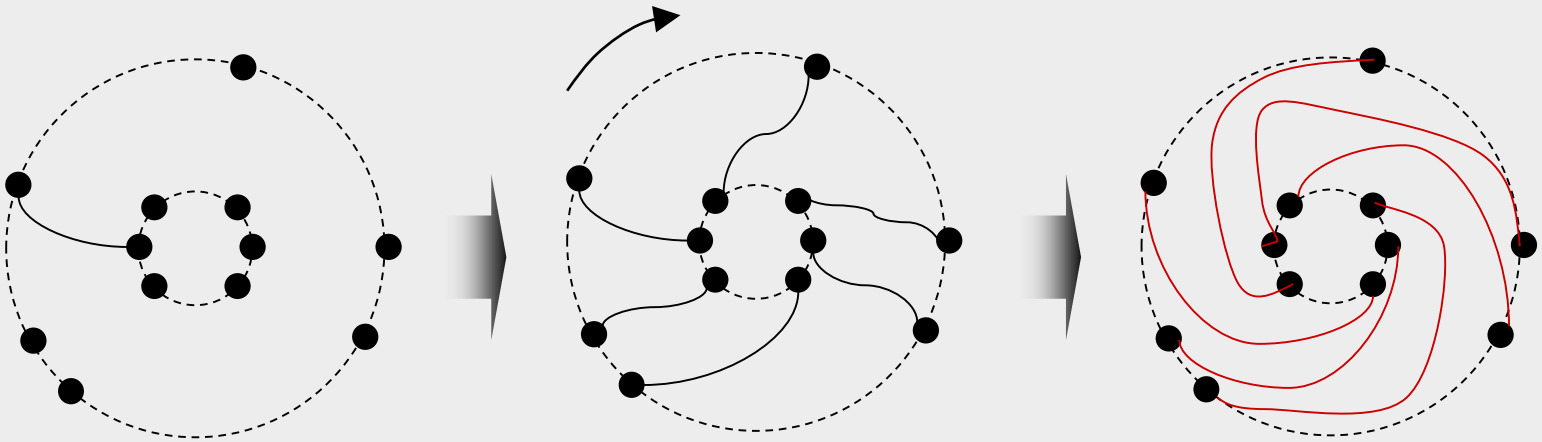
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

3. Anfangszuordnung und 4. Zuordnungsoptimierung (komplette Rotation)



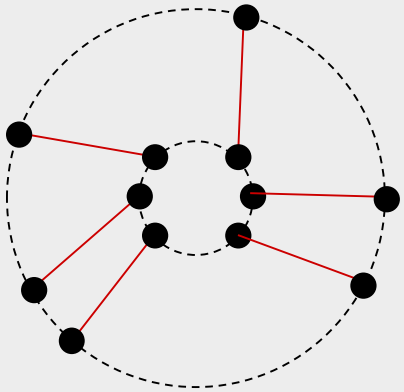
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

3. Anfangszuordnung und 4. Zuordnungsoptimierung (komplette Rotation)



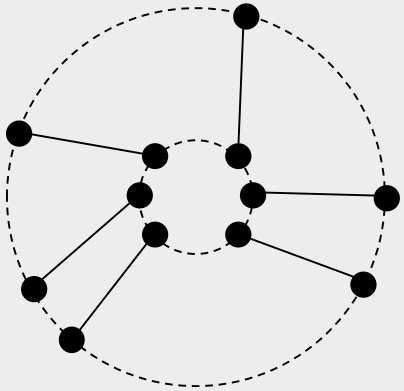
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

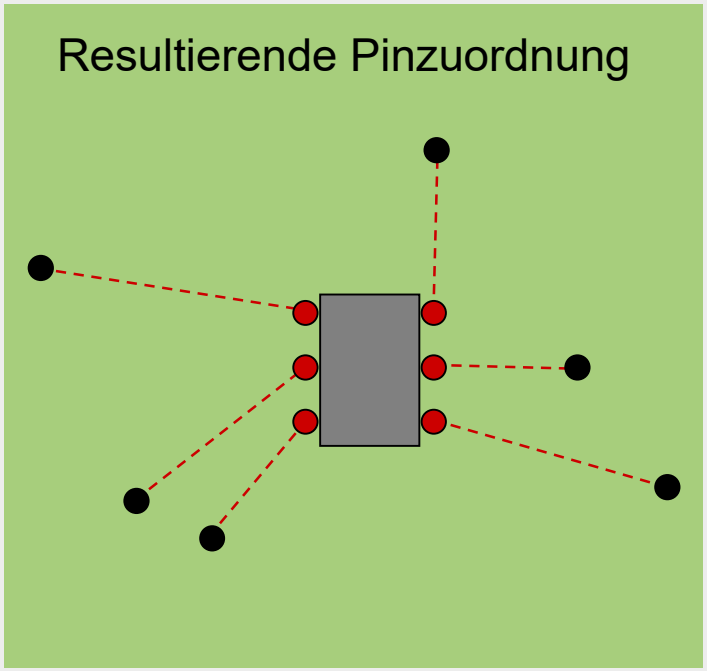
4. Ergebnis der Zuordnungsoptimierung



Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

4. Ergebnis der Zuordnungsoptimierung





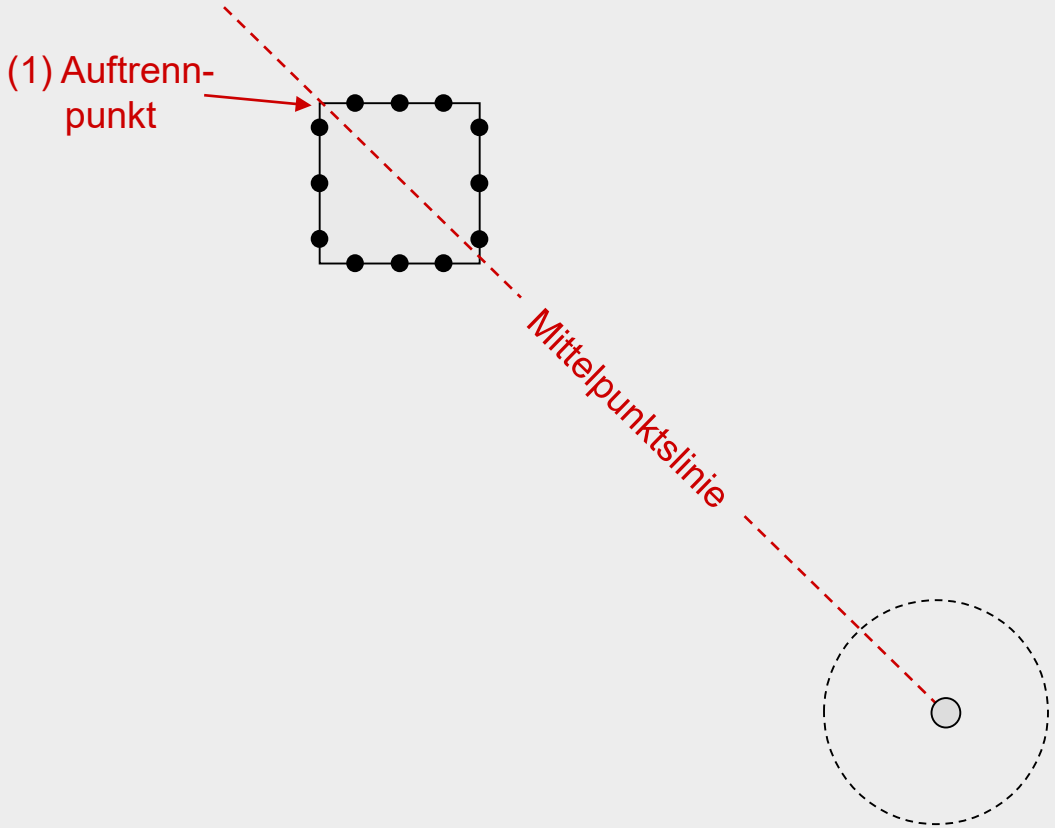
Nach Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

3.5.3 Topologische Pinzuordnung

Pinzuordnung in folgenden Fällen:

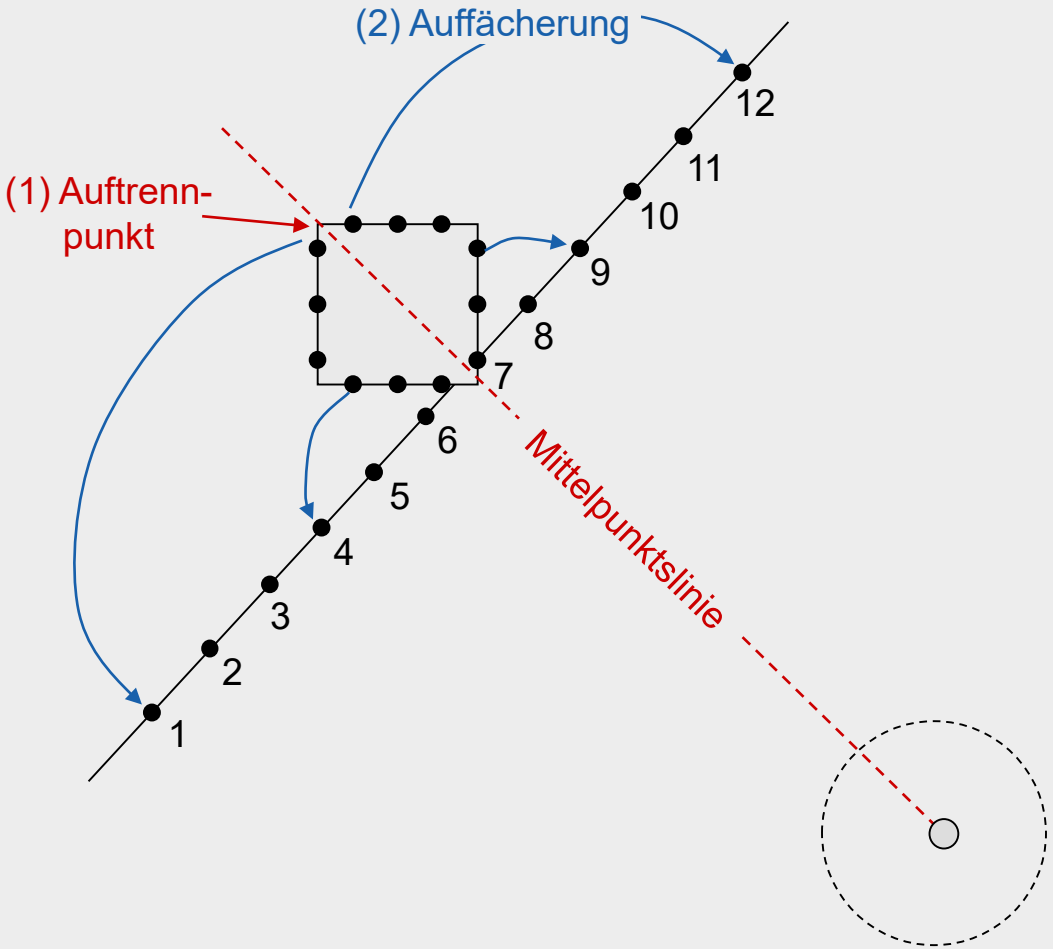
- anzuschließende Pins gehören zu einem Block und liegen so z.B. auf dessen abgewandter Seite oder
- Pins befinden sich hinter ebenfalls anzuschließenden Blöcken oder sonstigen Hindernissen

3.5.3 Topologische Pinzuordnung



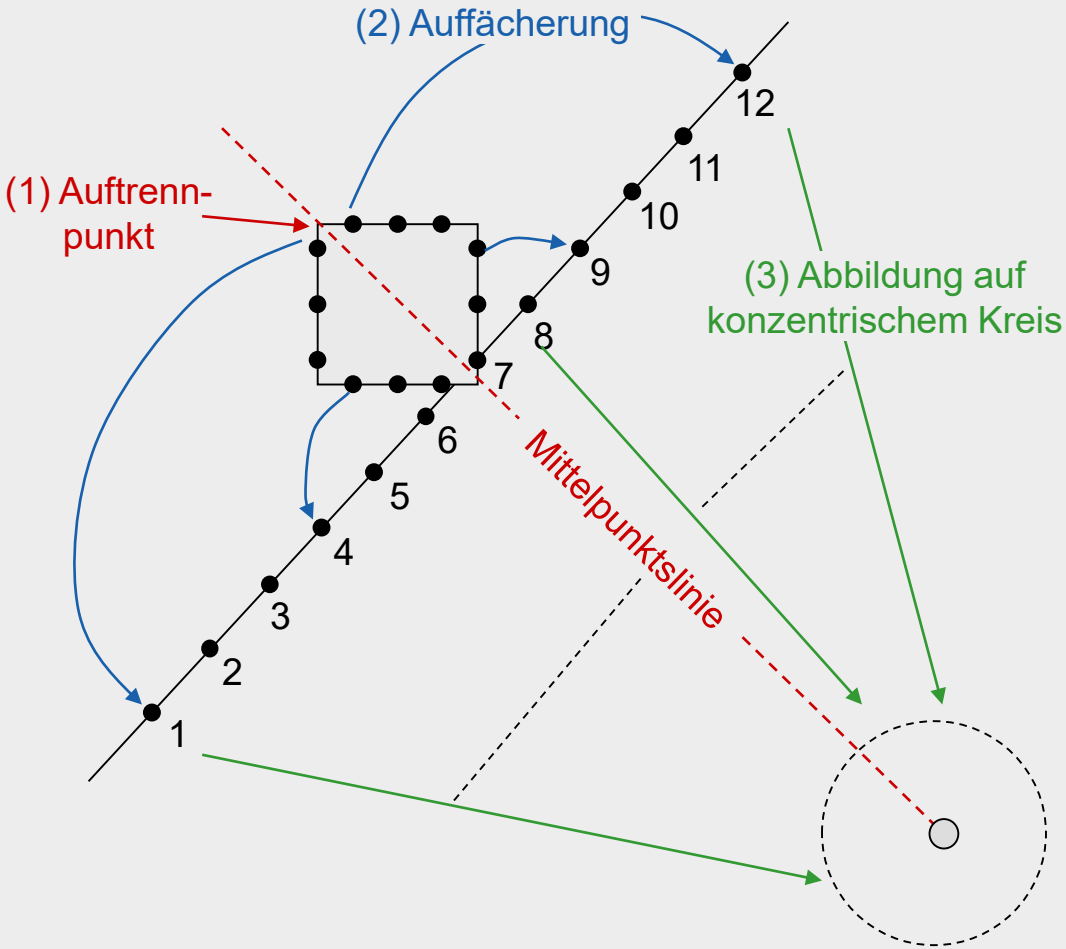
Nach Brady, H. N.: An Approach to Topological Pin Assignment

3.5.3 Topologische Pinzuordnung



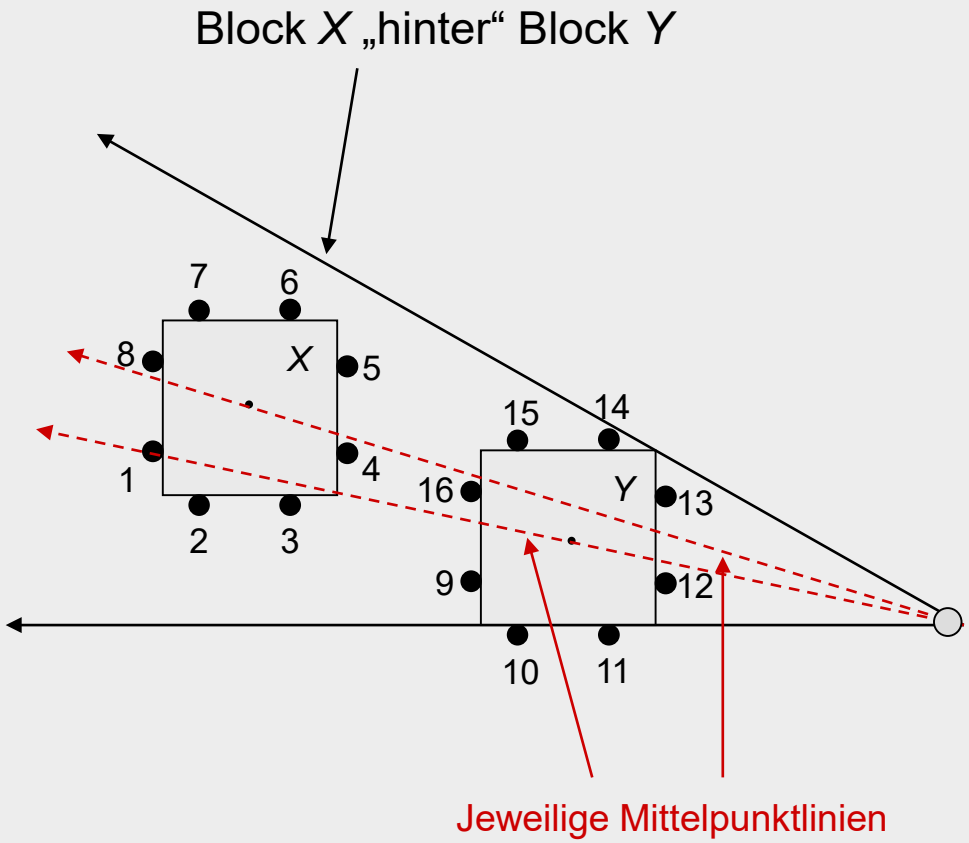
Nach Brady, H. N.: An Approach to Topological Pin Assignment

3.5.3 Topologische Pinzuordnung



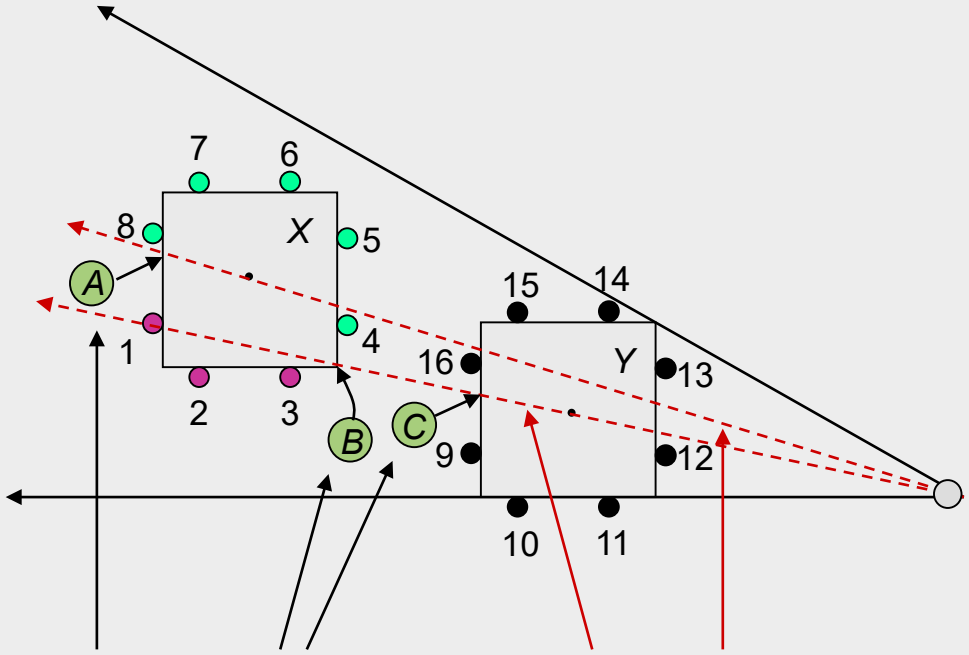
Nach Brady, H. N.: An Approach to Topological Pin Assignment

3.5.3 Topologische Pinzuordnung



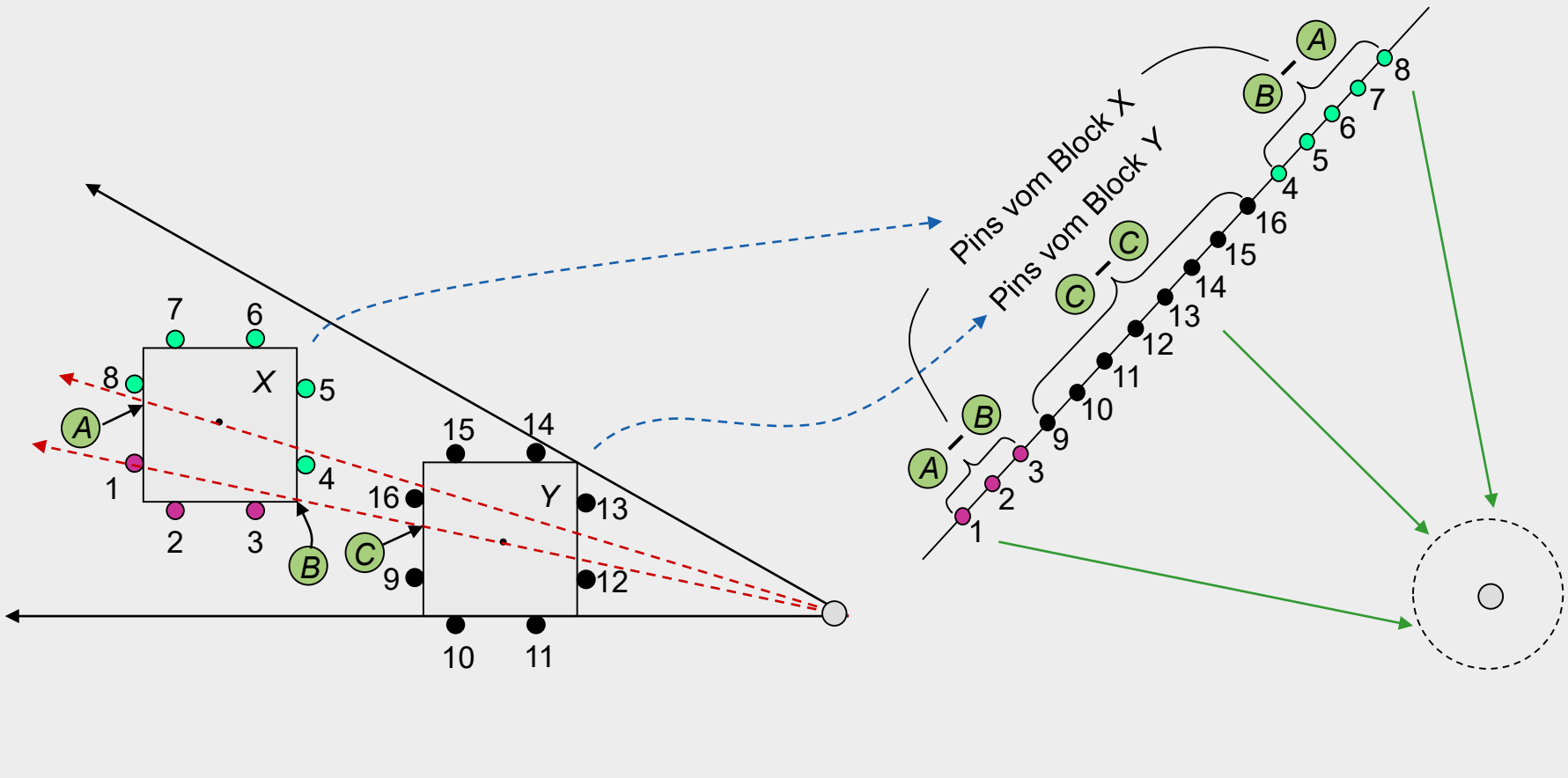
Nach Brady, H. N.: An Approach to Topological Pin Assignment

3.5.3 Topologische Pinzuordnung



Nach Brady, H. N.: An Approach to Topological Pin Assignment

3.5.3 Topologische Pinzuordnung



Nach Brady, H. N.: An Approach to Topological Pin Assignment

Zusammenfassung Kapitel 3 – Floorplanning