Layout-Synthese
- Labyrinth-Algorithmen -
(engl. *maze running algorithms*)

Peter Marwedel
Universität Dortmund, Informatik 12

Der Lee-Algorithmus

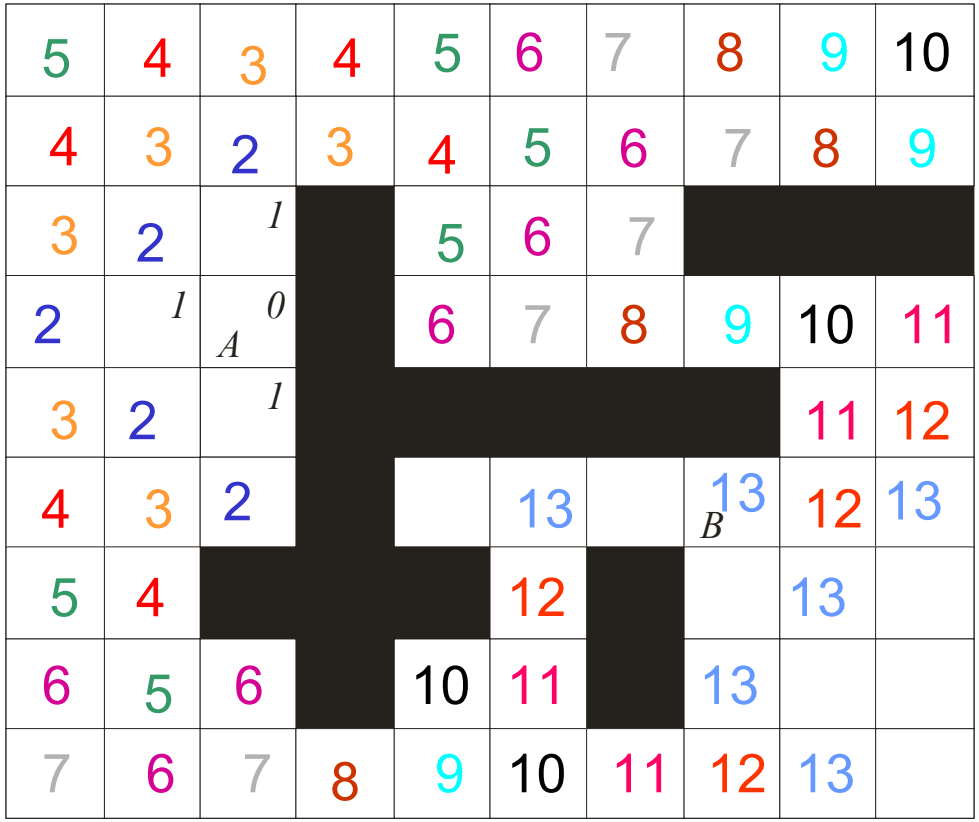
Lee, 1961: erster Algorithmus, der speziell zur Verdrahtung elektronischer Schaltungen entwickelt wurde.

- Ziel des Algorithmus ist das Finden eines kürzesten Weges in einem **Labyrinth** (engl. *maze*) zwischen zwei Punkten A und B.
- Verdrahtung innerhalb **einer** Ebene.
- **Rasterung** der Ebene (☞ungeeignet für große Flächen)

Phasen 1 & 2 des Lee-Algorithmus

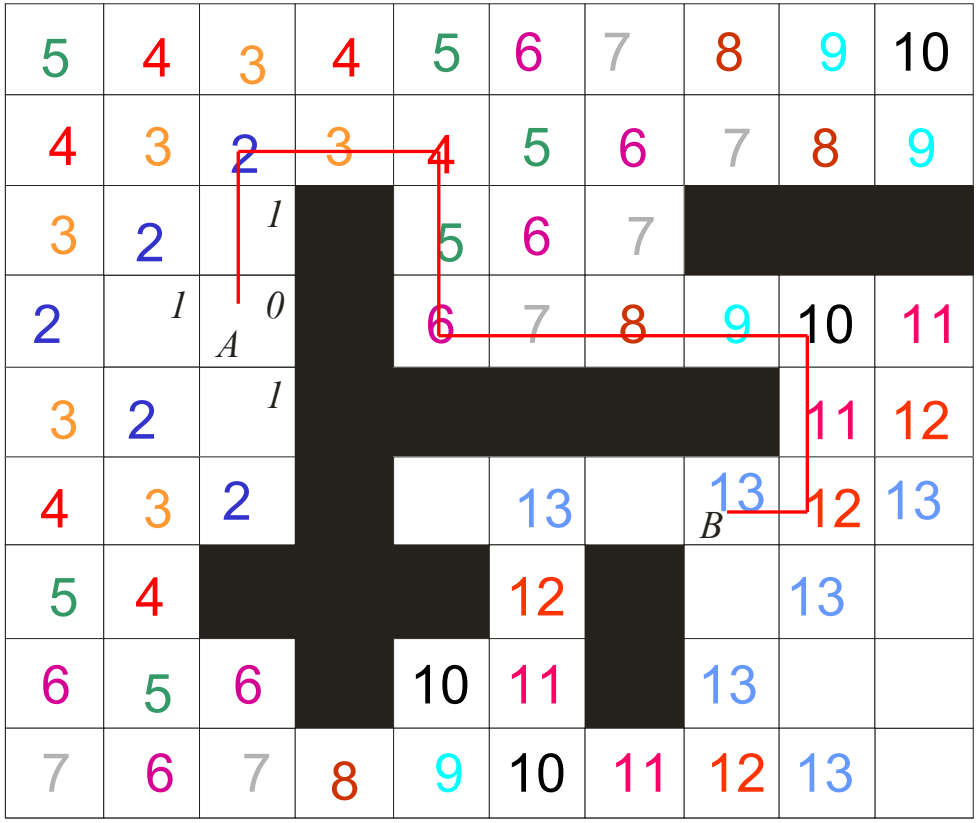
1: Wähle einen der beiden Punkte A bzw. B als Startpunkt aus. Der andere Punkt werde als Zielpunkt bezeichnet. Markiere den Startpunkt mit 0. Setze i:=0;

2: repeat
Unmarkierte Nachbarn von Feldern, die mit dem aktuellen Wert von i markiert sind, werden mit i+1 markiert. Anschließend wird i um 1 erhöht.
until (Zielpunkt erreicht) oder (alle Felder sind markiert).



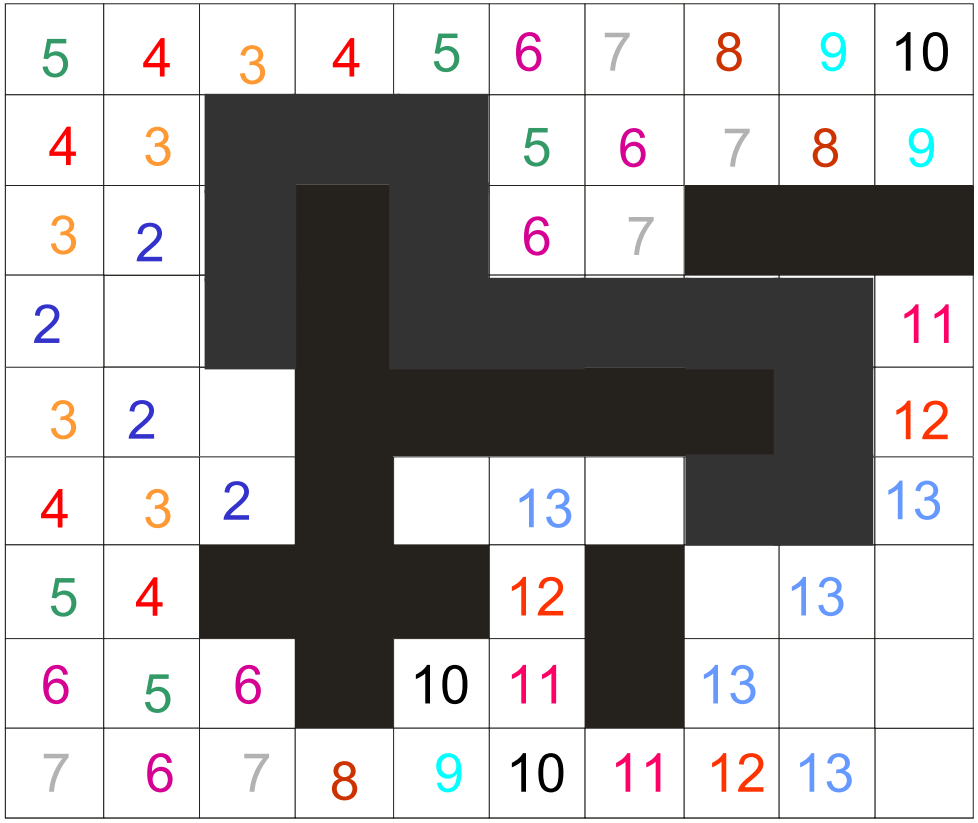
Phase 3 des Lee-Algorithmus

3: Kehre über streng absteigende Markierungen vom Ziel zum Start zurück. Falls verschiedene Wege möglich sind, versuche die aktuelle Richtung beizubehalten, um die Zahl der Knicke klein zu halten. Dieser Vorgang heißt *backtrace*.



Phase 4 des Lee-Algorithmus

Im letzten Schritt wird der gefundene Weg für weitere Leitungen blockiert und die Marken werden gelöscht.
Dieser Vorgang heißt *clearance.*



Siehe http://foghorn.cadlab.lafayette.edu/cadapplets/MazeRouter.html

Komplexität

Sei n die Kantenlänge.

- Speicherplatz: O(n^2) für die Matrix und Buchführung über die Wellenfront.
- Laufzeit: Im schlimmsten Fall
  - Für die Wellenausbreitung: O(n^2)
  - Für die Backtrace-Phase: O(ℓ), mit ℓ = Abstand(A,B)

Vorteil: Lässt sich leicht an unterschiedliche Randbedingungen anpassen.

Einfache Methoden der Beschleunigung

- Als Startpunkt wird der Punkt benutzt, der dem Rand näher liegt. Damit brauchen weniger Felder markiert zu werden.
- Man beginnt an beiden Punkten gleichzeitig. Auch damit werden in der Regel weniger Felder markiert.
- Die Markierung wird auf ein Rechteck begrenzt, das A und B einschließt und nur wenig (10-20 %) größer als das minimale umschließende Rechteck ist. Wird hierbei eine Lösung nicht gefunden, so wird in einem zweiten Versuch diese Begrenzung aufgehoben.

Erweiterungen (1)

- **Mehrpunktnetze:**
  Für 2-Punkt-Netze Weg mit kürzestem Abstand, sofern dieser existiert.
  Bei *n*-Punkt-Netzen Konstruktion eines Steiner-Baums.
- **Mehrlagenverdrahtung, 3-dimensional:**
  - ✂☞ Abstände innerhalb von Quadern. Größere Komplexität.
  - ✂☞ Vernachlässigung der Entfernung zwischen den Lagen; Annahme, dass zu verbindende Punkte in allen Lagen erreichbar sind (teuer bei ICs). Für jede zu verdrahtende Lage 1 Tableau für die Blockierungsinformation sowie 1 Tableau für die Abstände.

Siehe http://foghorn.cadlab.lafayette.edu/cadapplets/MultiMaze.html

Erweiterungen (2)

- **Vermeidung der Blockierung für folgende Wege**
  Bislang keinerlei Rücksicht, wie schwer Verdrahtung der folgenden Netze sein wird. Günstig, Wege möglichst eng parallel zu existierenden Wegen anzuordnen. Durch Einführung von gewichteten Rasterpunkten können gewisse Wege bevorzugt werden. Während der Wellenausbreitung Erhöhung um das Gewicht des jeweiligen Rasterpunktes.

Liniensuchalgorithmen

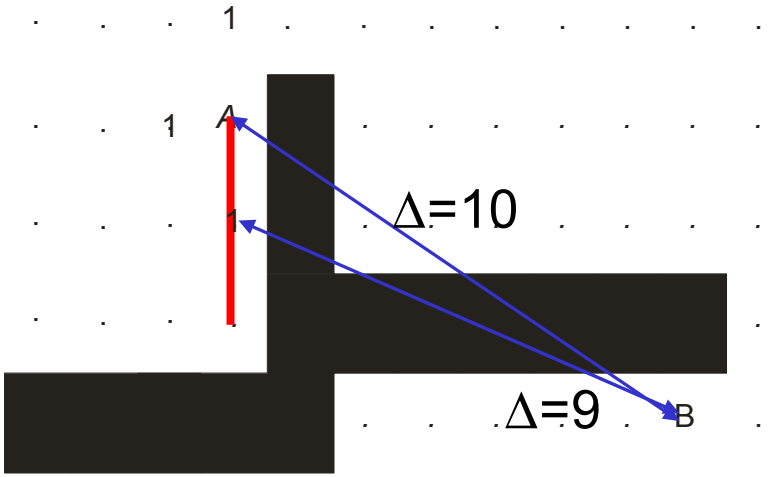
Verdrahtung über relativ große Distanzen ohne Knicke?

Soukup‘s schneller Labyrinth-Algorithmus

- Algorithmus generiert zunächst eine Linie vom Ausgangspunkt in Richtung des Zielpunktes.
- Sofern der Zielpunkt nicht erreicht wird, wird um diese Linie herum mittels des Lee-Algorithmus ein Rasterpunkt gesucht, der den Abstand zum Zielpunkt gegenüber dem bisherigen Abstand von der Linie zum Zielpunkt verkürzt.
- Die Menge der Linien wird sodann um Linien durch diesen Rasterpunkt erweitert.

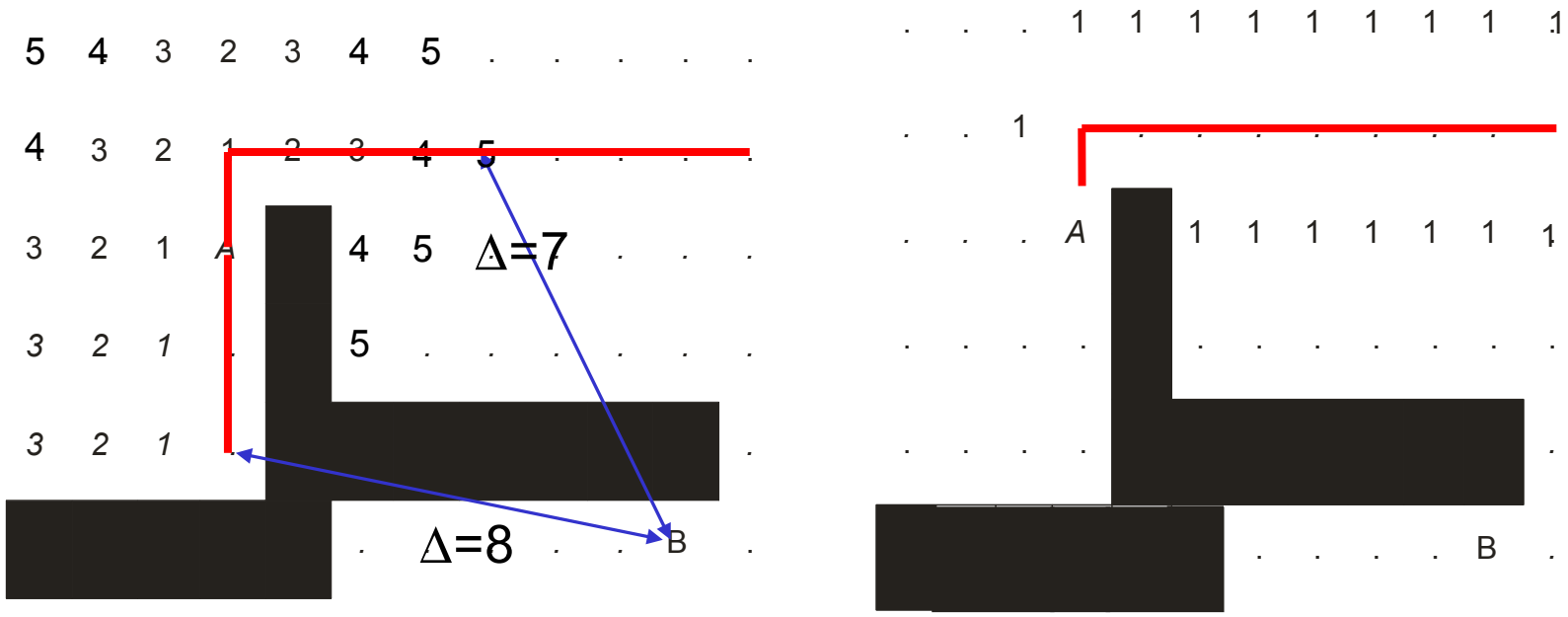
Soukup (1978): Kombination von Lee-Algorithmus und den eigentlichen *line search* Verfahren.

Beispiel:



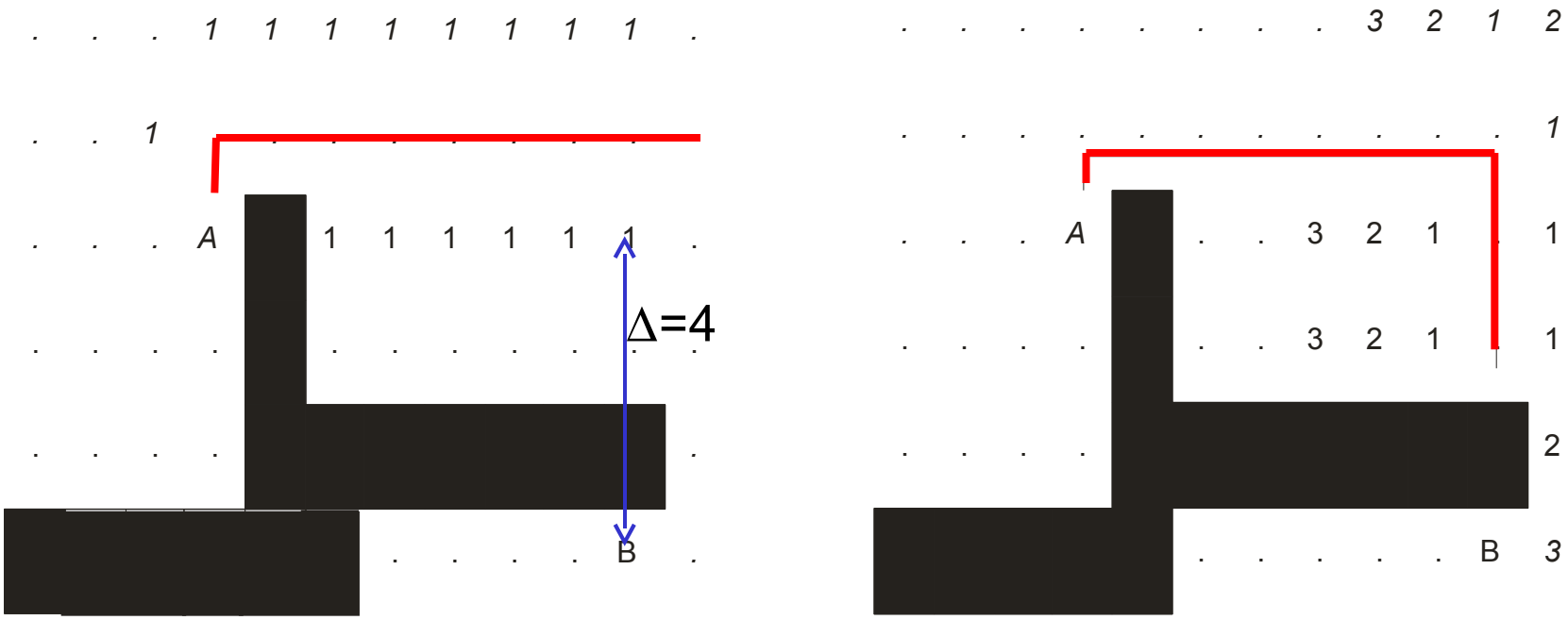
Soukup‘s schneller Labyrinth-Algorithmus (2)

Mit der jeweils zuletzt gefundenen Linie als Menge neuer Startpunkte wird dieser Prozess wiederholt, bis der Endpunkt erreicht ist.



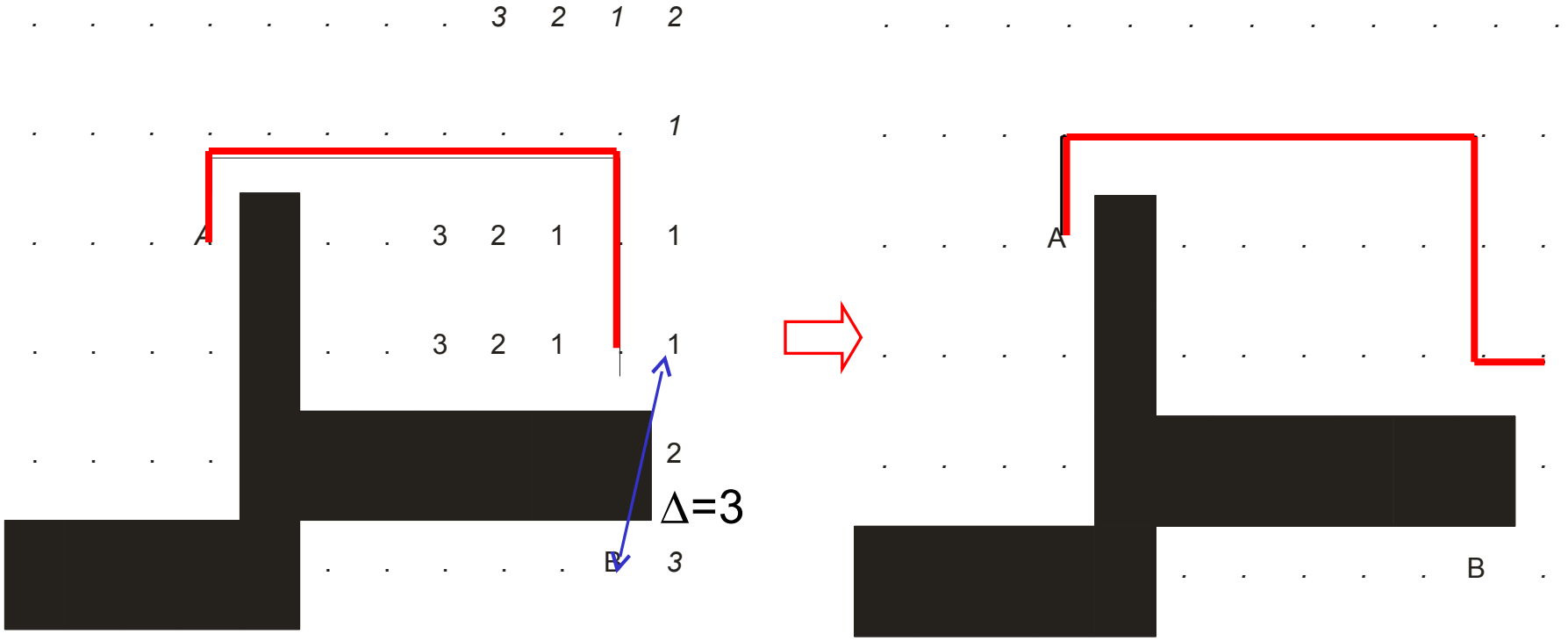
Soukup‘s schneller Labyrinth-Algorithmus (3)

Mit der jeweils zuletzt gefundenen Linie als Menge neuer Startpunkte wird dieser Prozess wiederholt, bis der Endpunkt erreicht ist.



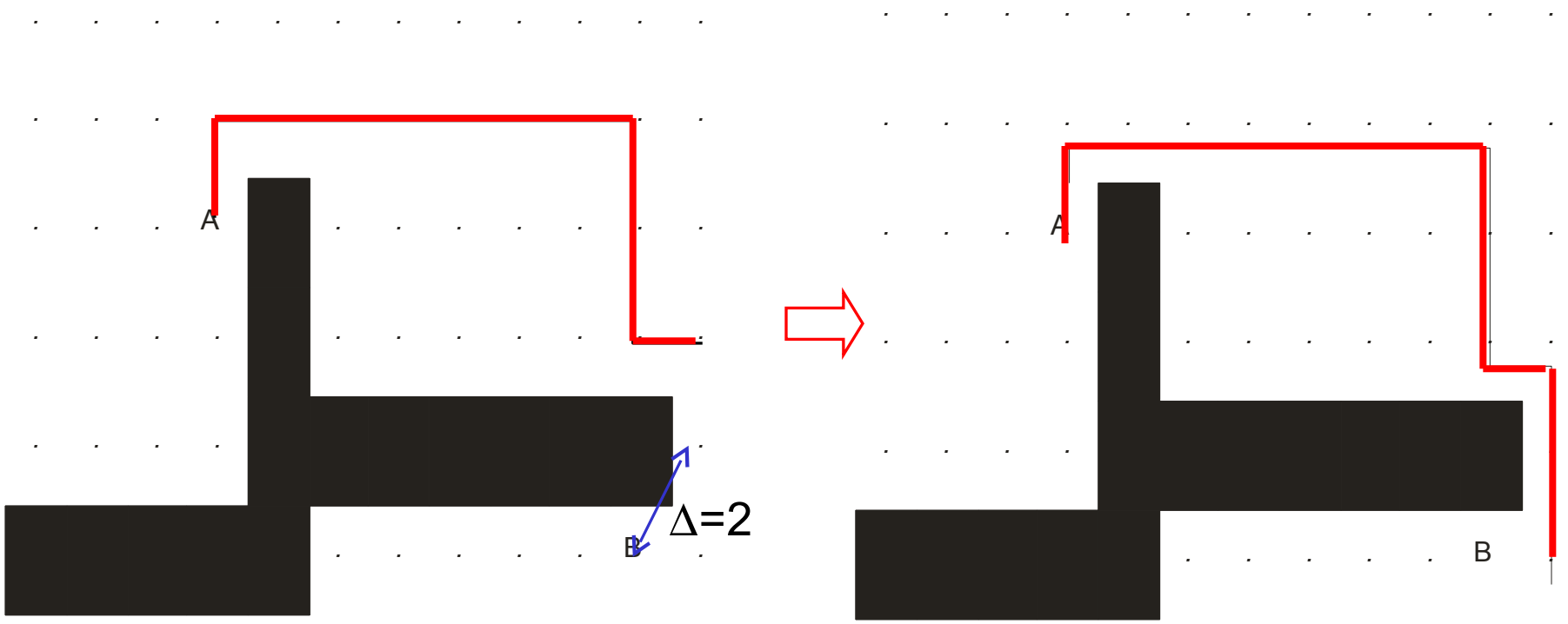
Soukup‘s schneller Labyrinth-Algorithmus (4)

Mit der jeweils zuletzt gefundenen Linie als Menge neuer Startpunkte wird dieser Prozess wiederholt, bis der Endpunkt erreicht ist.



Soukup‘s schneller Labyrinth-Algorithmus (5)

Mit der jeweils zuletzt gefundenen Linie als Menge neuer Startpunkte wird dieser Prozess wiederholt, bis der Endpunkt erreicht ist.



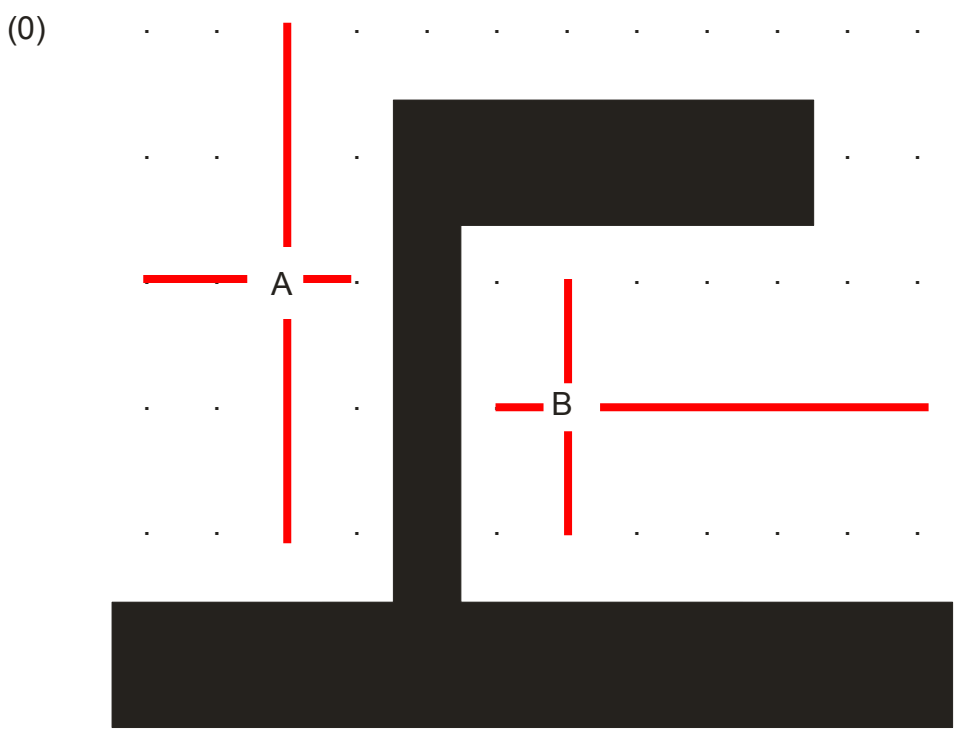
Bewertung

- Falls eine Lösung existiert, wird von diesem Algorithmus auch immer eine Lösung gefunden, da notfalls immer mittels des Lee-Algorithmus gesucht wird.
- Die gefundene Lösung ist aber nicht notwendig optimal.
- Laut Soukup soll der Algorithmus für typische Beispiele 10-50-fach schneller als der Lee-Algorithmus sein. Bei sehr engen Platzverhältnissen reduziert sich dieser Geschwindigkeitsvorteil.

Line-search Verfahren von Mikami und Tabuchi

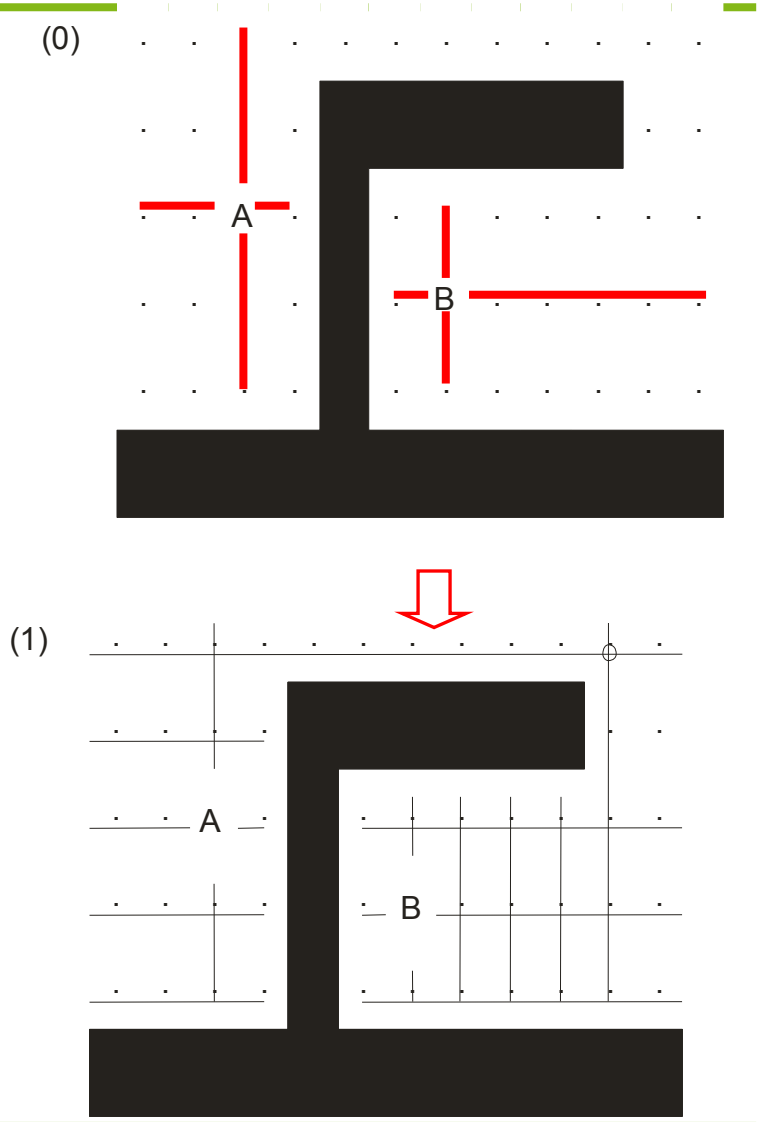
Mikami, Tabuchi (1968): Kein Lee-Algorithmus nötig.

- Zunächst werden die senkrechten und waagerechten Linien durch Ausgangs- und Zielpunkte gebildet. Diese 4 Linien heißen Versuchslinien der Ebene 0. Sofern sich diese schneiden, ist ein Weg gefunden.



Line-search Verfahren von Mikami und Tabuchi (2)

- Zunächst werden die senkrechten und waagerechten Linien durch Ausgangs- und Zielpunkte gebildet.
  Diese 4 Linien heißen Versuchslinien der Ebene 0. Sofern sich diese schneiden, ist ein Weg gefunden.
- Sonst: Errichten aller Senkrechten auf den Versuchlinien der Ebene 0. Sofern die vom Startpunkt ausgehenden Versuchlinien die vom Zielpunkt ausgehenden Versuchslinien schneiden, ist ein Weg gefunden.
  Diese heißen Versuchslinien der Ebene 1.



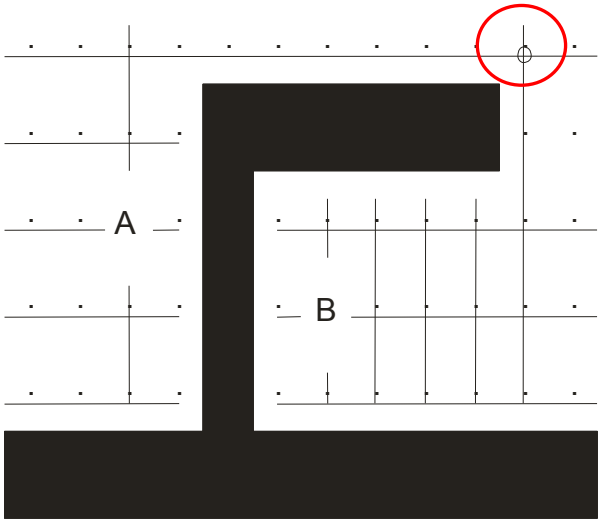
Line-search Verfahren von Mikami und Tabuchi (3)

- Sonst: fahre mit weiteren Ebenen fort.

Eigenschaften (1)

- Ein Schnitt von Versuchlinien der Ebene i mit solchen der Ebene j enthält i+j+1 Knicke. Durch passende Reihenfolge der Linienbildung kann stets mit wachsendem (i+j) auf Schnitt testen. So wird immer ein Weg mit minimaler Zahl von Knicken gefunden.
- Für eine bestimmte Ebene werden die Linien mit dem kleinsten Abstand zu den Ausgangspunkten zuerst betrachtet, um so möglichst auch die Verdrahtungslänge klein zu halten.

(1)

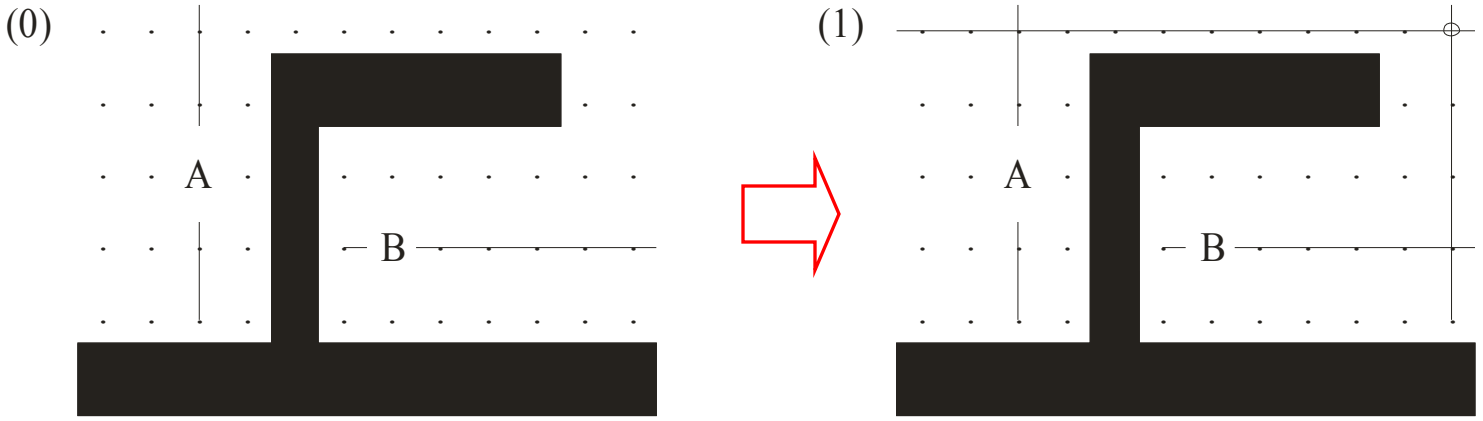


Eigenschaften (2)

- Falls eine Lösung existiert, wird sie von diesem Algorithmus auch immer gefunden, sofern wirklich alle Versuchslinien gespeichert werden.

Line search Verfahren von Hightower

Bei Liniensuch-Algorithmus von Hightower wird statt aller Senkrechten lediglich eine einzelne Linie gespeichert.

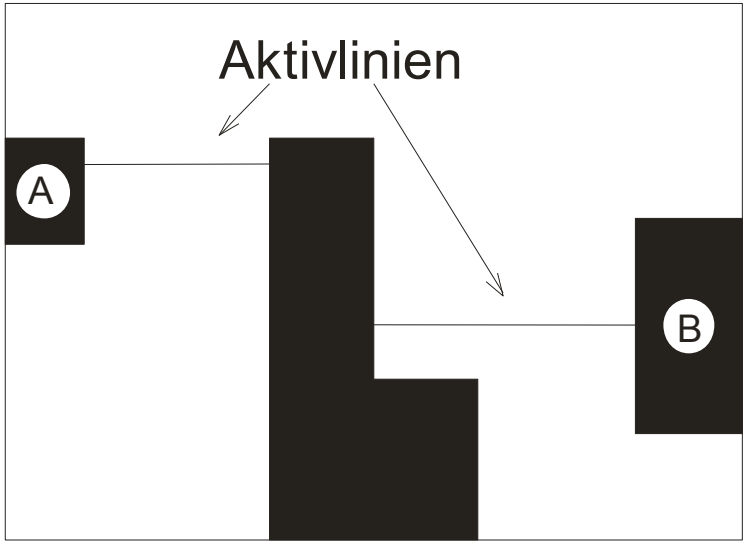


Heuristische Wahl der Linie nach vielen Kriterien. Schneller als Soukup, wenn das Verfahren eine Lösung findet.

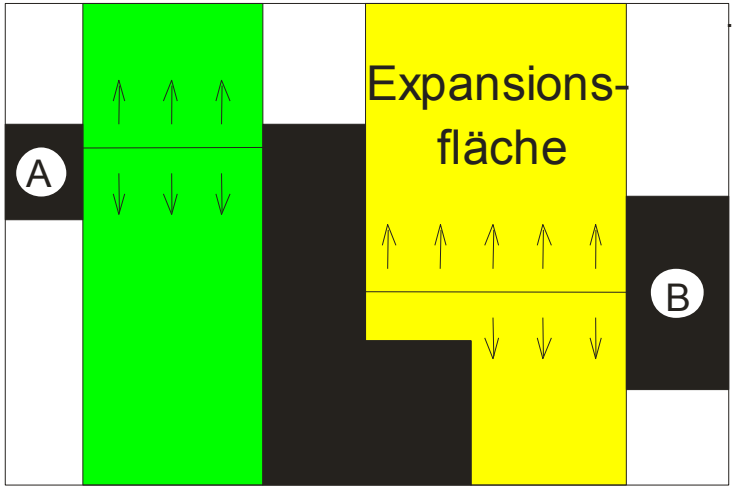
Linienerweiterungs-Algorithmus

Heynes, Beke (1980)

- Errichte Senkrechte („**Aktivlinien**“) auf den Anschlüssen am Rand der Zellen. STOP falls Linienschnitt.

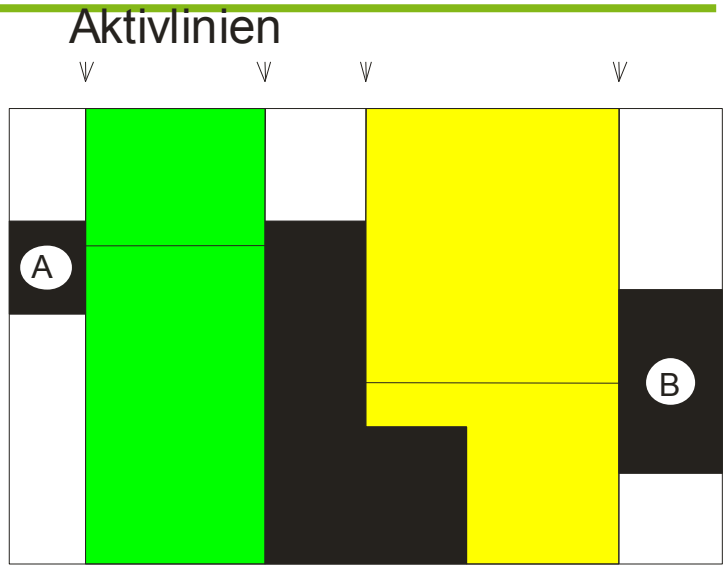


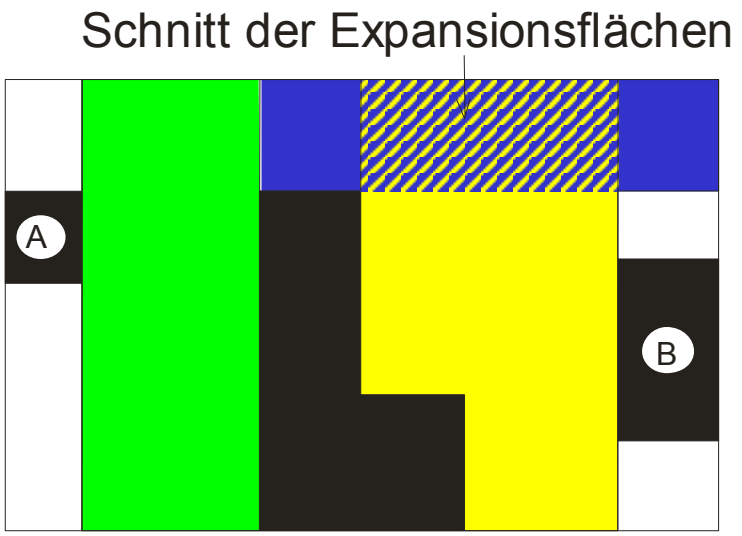
- Analyse, ob Expansion seitlich der Aktivlinien möglich. Flächen, die über Senkrechte auf Aktivlinien zu erreichen sind, heißen **Expansions-flächen**. Senkrechte, welche die Expansionsflächen begrenzen, sind die nächsten Aktivlinien. Falls sich diese schneiden STOP, sonst Wiederholung



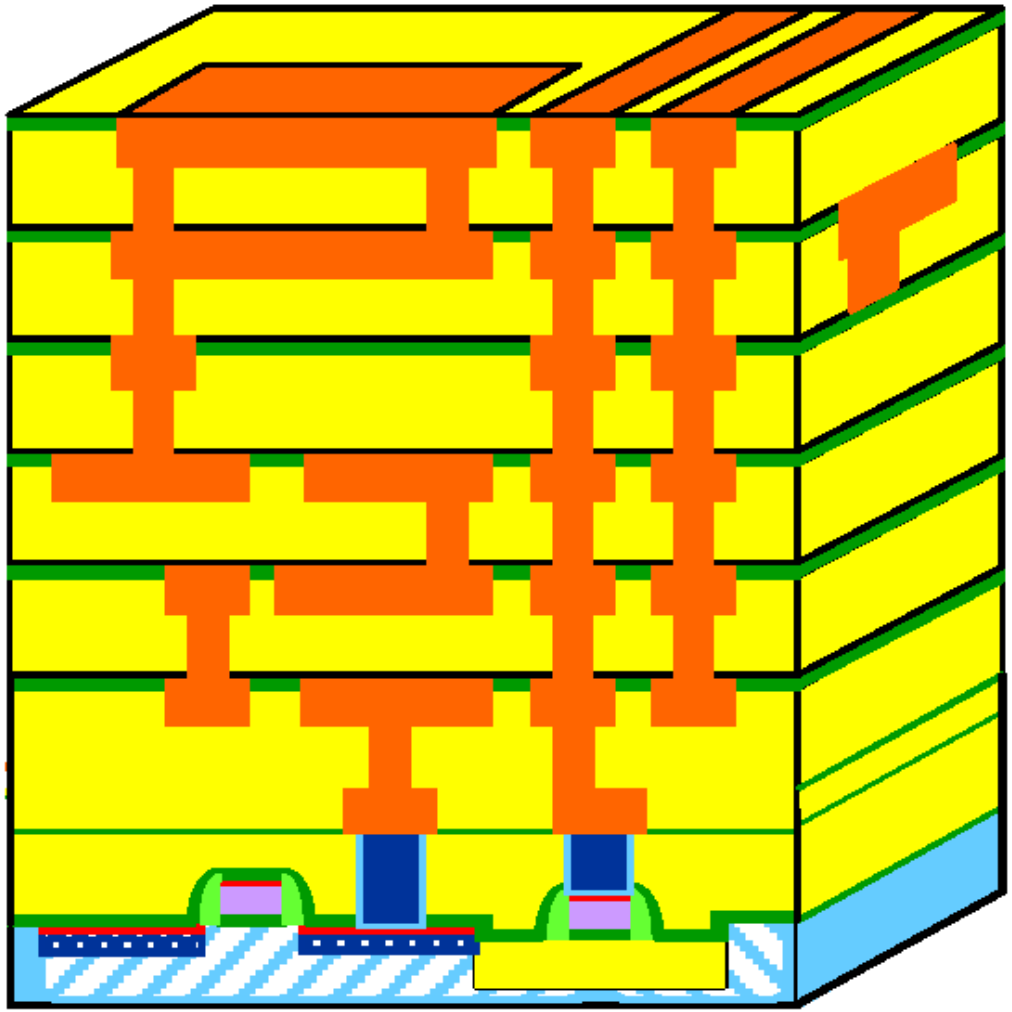
Linienerweiterungs-Algorithmus (2)

- Analyse, ob Expansion seitlich der Aktivlinien möglich. Flächen, die über Senkrechte auf Aktivlinien zu erreichen sind, heißen **Expansions-flächen**. Senkrechte, welche die Expansionsflächen begrenzen, sind die nächsten Aktivlinien. Falls sich diese schneiden STOP, sonst Wiederholung



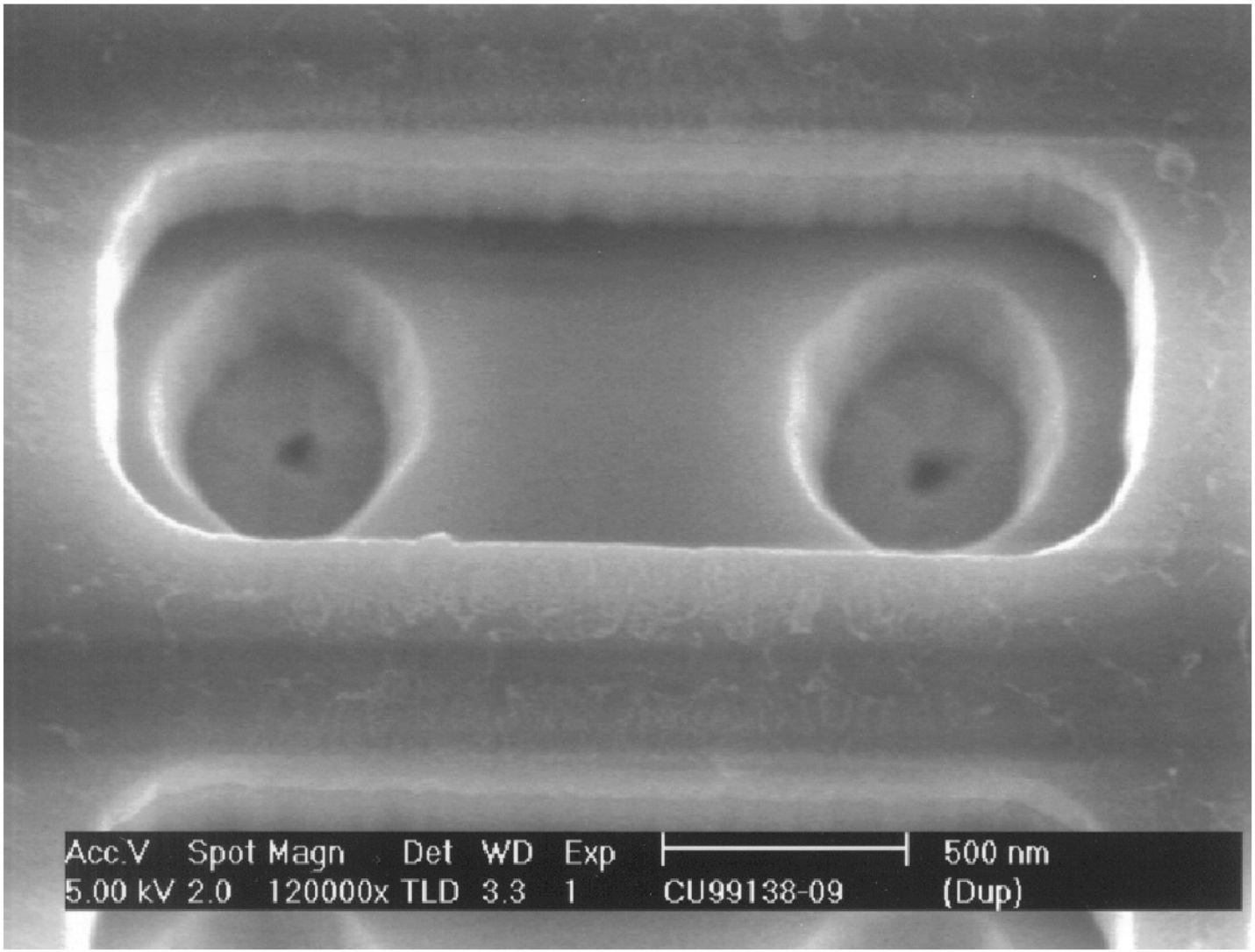


Moderne Mehrebenenverdrahtung



http://www.imec.be/mtc/mtcpdf/
Tutorial-on-chip-interconnect.pdf

Ungefüllte Löcher einer Mehrebenenverdrahtung



http://www.imec.be/mtc/mtcpdf/Tutorial-on-chip-interconnect.pdf

Spezielle Probleme moderner Technologien (1)

1. **Leitungslaufzeiten**
   Bestimmung der Laufzeiten bislang:
   1. Extraktion der Kapazitäten aus dem Layout
   2. Annotation der Netzlisten („*back annotation*“)

   ☞ Simulatoren können einwandfreie Funktion bei einer bestimmten Taktfrequenz überprüfen.

   ☞ Besonders problematische Netze werden als **kritische Netze** gekennzeichnet.

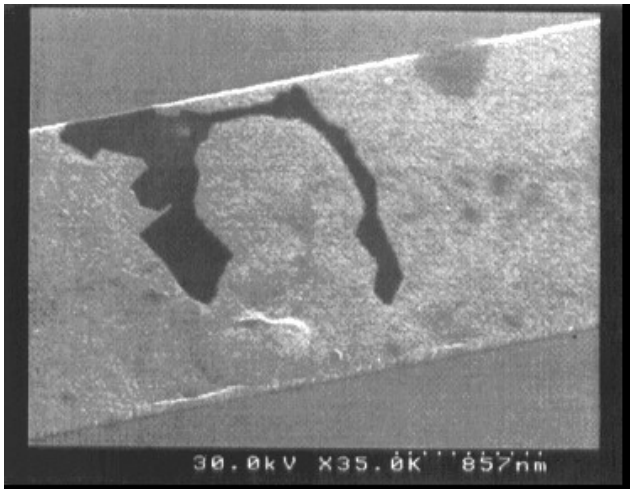
   ☞ In einer **Iteration** werden diese bevorzugt berücksichtigt. Bei älteren Technologien einige wenige Male durchzuführen.

   ☞ Bei neueren Technologien problematisch, Konvergenz („***timing closure***“) zu erreichen.

See http://www.techonline.com/community/tech_topic/timing_closure/14016?csp_id=37

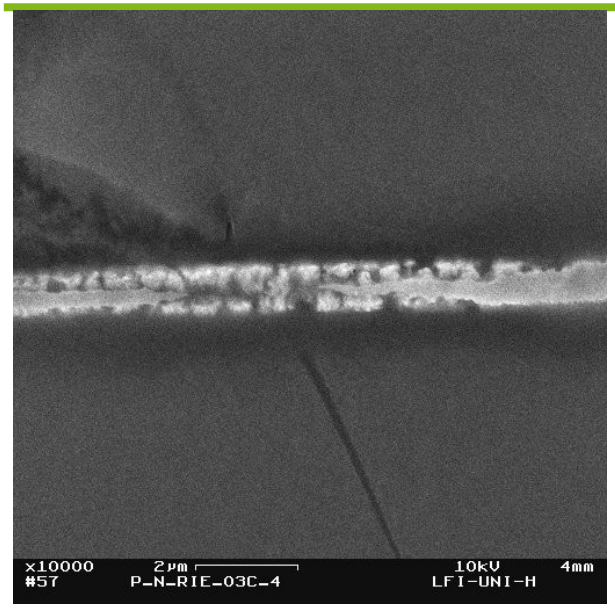
Spezielle Probleme moderner Technologien (2)

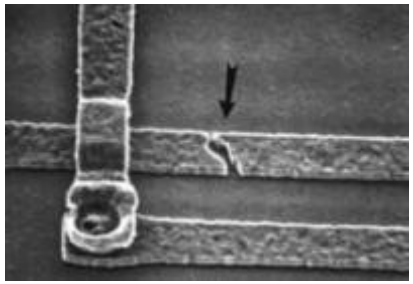
1. **Signalverflachung**
   Spannungspegel können stark reduziert werden. Mögliche Probleme erkennen und korrigieren.
2. **Betrachtung der *electro migration***:
   Bei hoher Stromdichte: Metallwanderung.
   ☞ stark verkürzte Lebensdauer der Schaltung. Problemfälle erkennen und beseitigen.

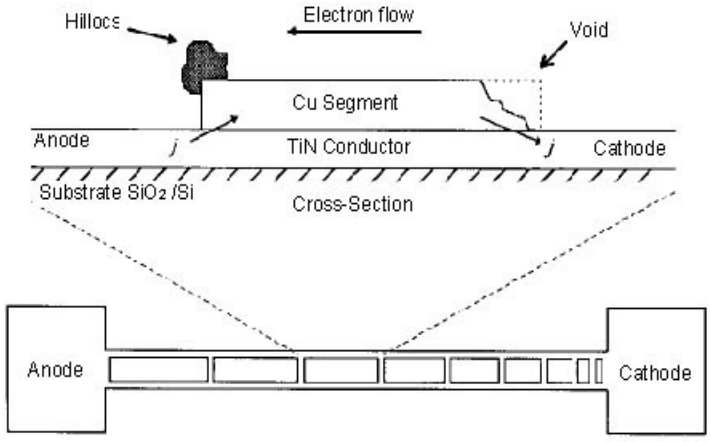


www.nd.edu/~micro/fig20.html

Electro migration







Empirisches Black'sches Gesetz

$$MTTF = AJ^{-n}e^{\frac{E_a}{kT}}$$

A: abhängig vom Querschnitt
J: Stromdichte
$n \sim 2$
E_a: materialabhängig
T: Temperatur

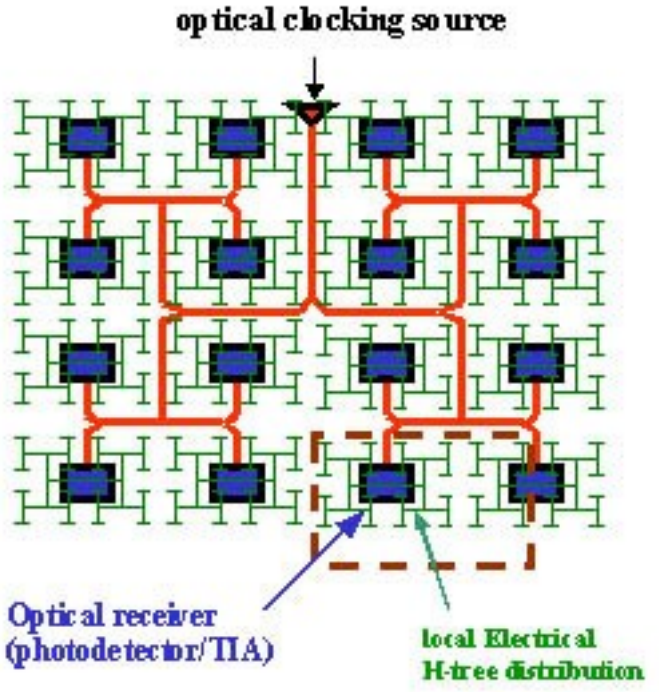
http://www.dwpg.com/content.php?contid=2&artid=68

Spezielle Probleme moderner Technologien (3)

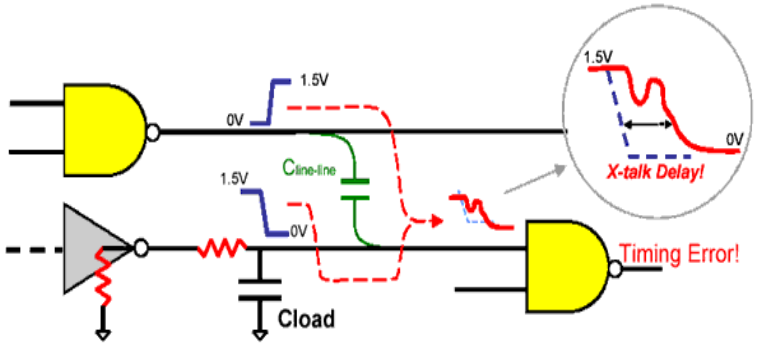
1. ***power and ground routing***
   Metallwanderung muss v.a. auch in Bezug auf die Spannungs-versorgung beachtet werden. Erwarteter und der zulässiger Spannungsabfall müssen miteinander in Beziehung gebracht werden.

Spezielle Probleme moderner Technologien (4)

1. **Taktverdrahtung**
   Der zeitliche Unterschied des Eintreffens der Taktflanken an den verschiednen Teilen der Schaltung muss möglichst klein sein. Spezielle Taktverdrahtung, starke Leitungstreiber, gleich lange Leitungsführung (z.B. H-Baum).
2. **Übersprechen**
   Mögliches Übersprechen muss erkannt und durch Änderung der Geometrie beseitigt werden.



photonics.mit.edu/research/ 2003/opt_clock.html

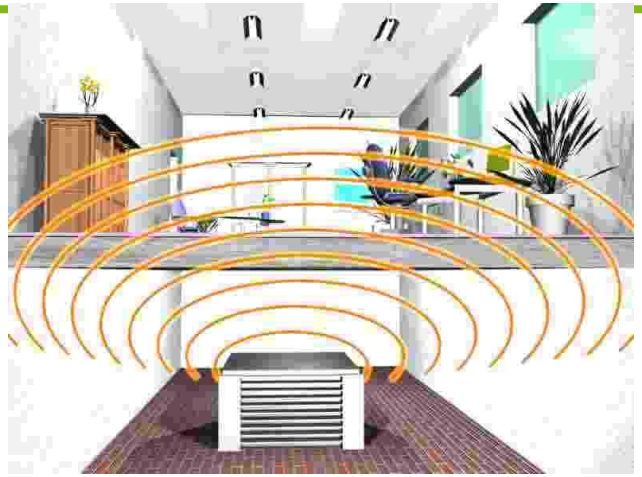


http://www.magma-da.com/c/@q7x5pnxv_yCyo/Pages/Sloverview.html

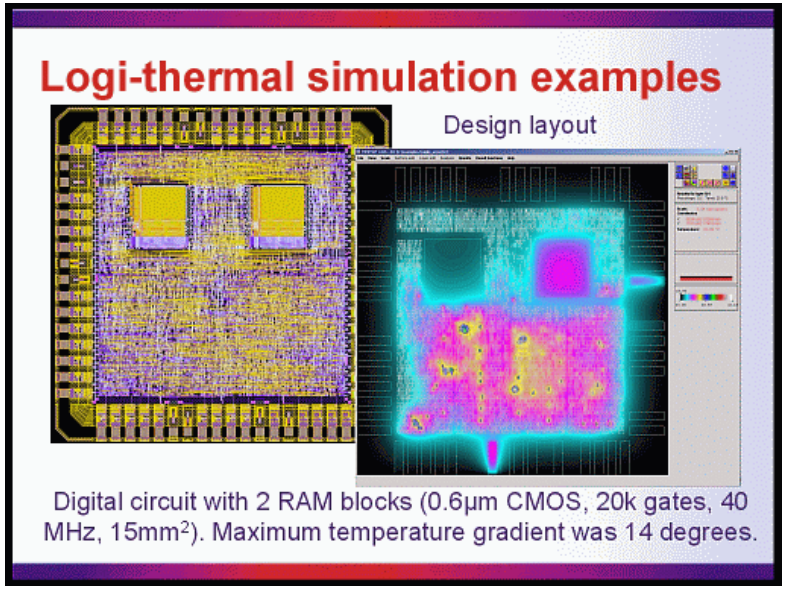
Spezielle Probleme moderner Technologien (5)

1. **Elektromagnetische Verträglichkeit**
   Maßnahmen zur Vermeidung einer starken Empfindlichkeit gegenüber externer elektromagnetischer Strahlung.
2. **Erwärmung der Schaltungen**
   Zu vermeiden, dass einzelne Teile einer Schaltung zu heiß werden.
   Temperaturverteilung muss vorhergesagt werden.

Spezielle Probleme werden in dem Buch von Sherwani angesprochen.



www.emv-tech.de/Magnetfeldabschirmung.htm



http://www.micred.com/events/semitherm2003/elth/ea10.html

Zusammenfassung

Labyrinth-Verdrahtung

- Lee-Algorithmus
- Soukup‘s Algorithmus: Mischung von Raster- und Linien-basierten Verfahren
- Mikami und Tabuchi: Linienbasiertes Verfahren
- Hightower: Linienbasiertes Verfahren
- Heynes, Beke: Linienerweiterungsalgorithmus

Probleme moderner Technologien

Das war‘s (☺)