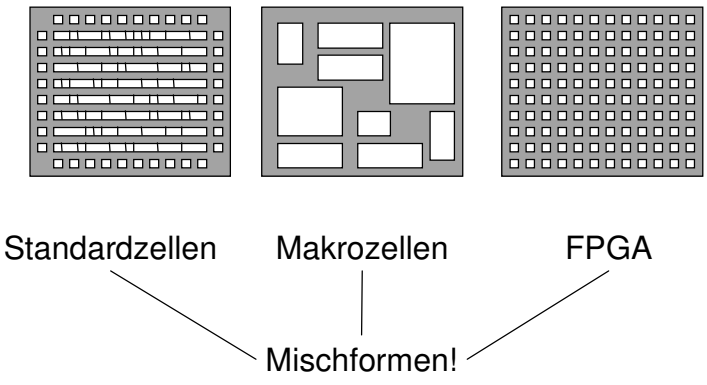
1 Hardwareentwurf

- **1.1** Überblick, Hardwareentwurfsschritte
- **1.2** Hardwarebeschreibungssprachen
- **1.3** Hardwaresimulation-/Verifikation
- **1.4** Hardwaresynthese
- **1.5** Plazierung und Verdrahtung

Realisierungsformen



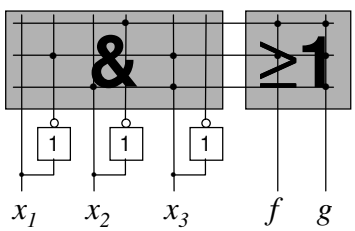
Makrozellen

- Standardisierte Funktionale Einheiten
  - Arithmetische Funktionsblöcke
  - I/O-Interfaces, ...
- PLA
  - programmable logic arrays
  - 2-stufige Logik
- RAM, ROM

Realisierung als PLA

- 2-Stufige Logik läßt sich auf einem ASIC als programmable logic array realisieren

$$f = \overline{x_2} \vee \overline{x_1} x_3 \qquad g = \overline{x_2} \vee \overline{x_1} x_3 \vee x_2 x_3$$



Standardzellen

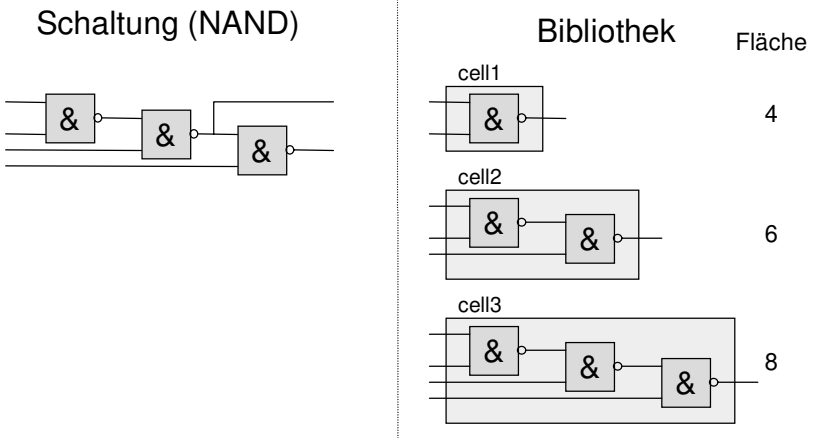
- Der ASIC-Hersteller kann nicht beliebige Logische Gatter produzieren, deshalb wird vom Hersteller eine Zellbibliothek zur Verfügung gestellt, die alle möglichen Gatter enthält.

→ Abbildung der synthetisierten Gatternetzliste auf die Zellbibliothek

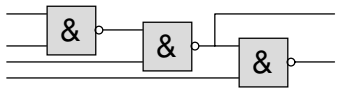
→Transformation der Schaltung und der Bibliothek in eine einheitliche Darstellung (z.B. NAND-Gatter)

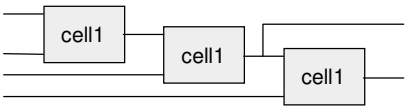
→Finden einer Überdeckung (NP-vollständig)

Bibliotheksabbildung

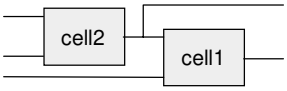


Bibliotheksabbildung

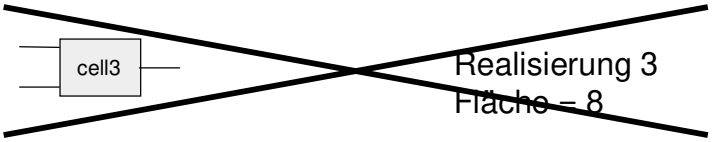




Realisierung 1
Fläche = 12



Realisierung 2
Fläche = 10

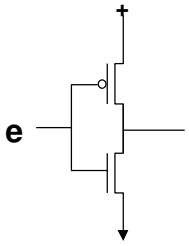


Realisierung 3
Fläche = 8

Abbildung auf Technologie

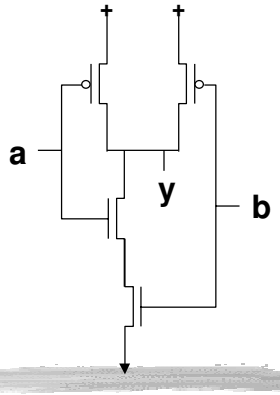
- CMOS

Inverter



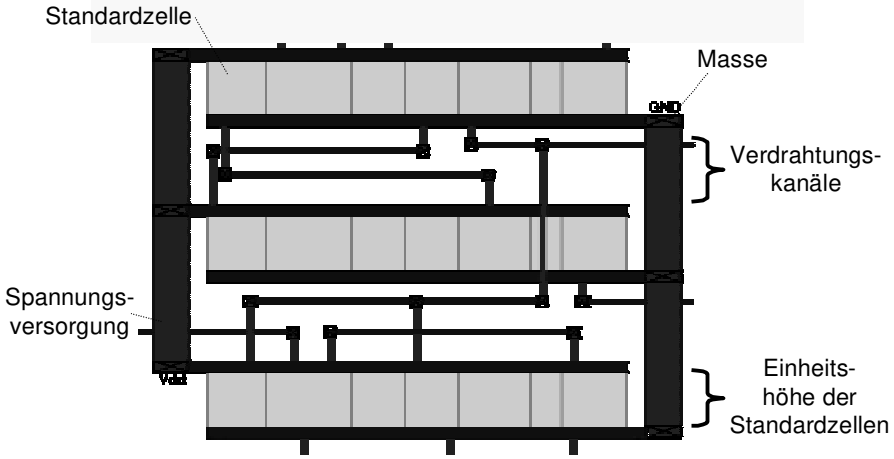
e	y
0	1
1	0

NAND

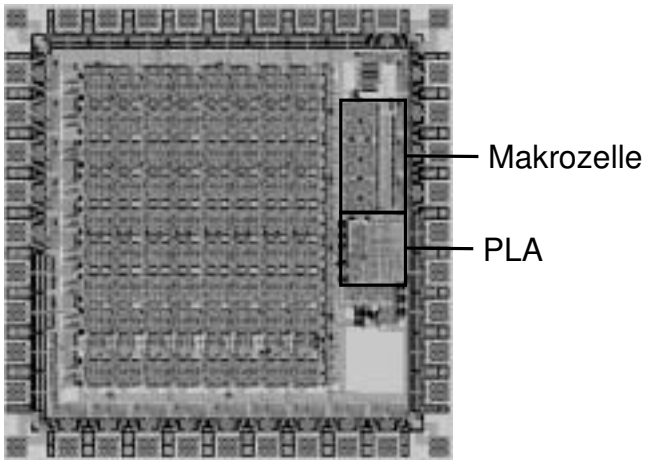


a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

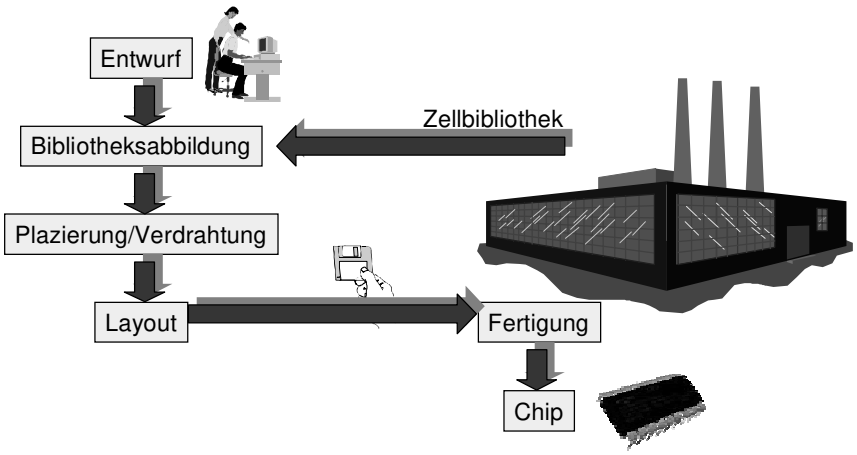
Standardzellen Layout



Standardzellen Layout

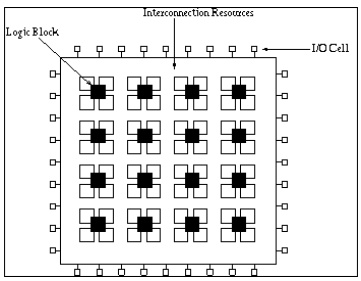


Standardzellen Flow



Realisierung als FPGA

- FPGA (field programmable gate arrays)

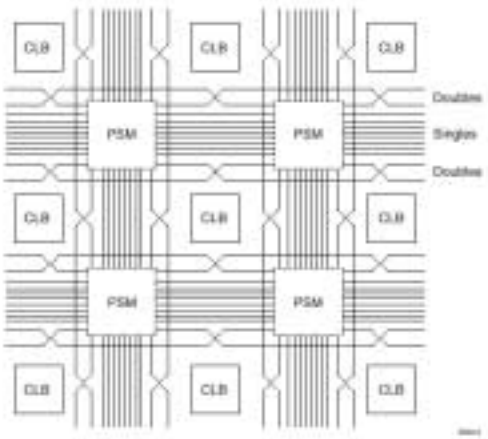


- Verbindungen und Logikblöcke sind programmierbar

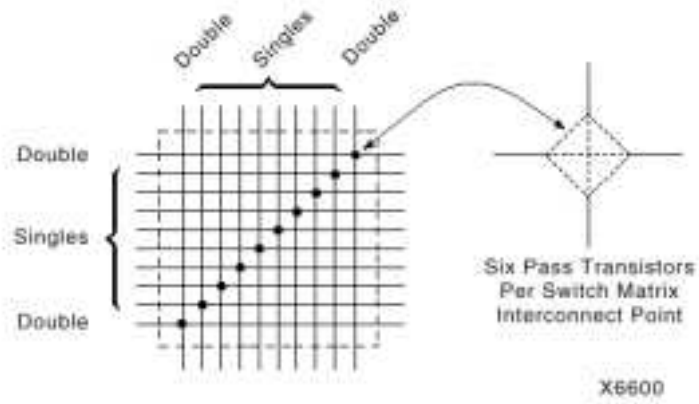
Beispiel: Xilinx 4000E

- CLB = Configurable Logic Block
  - zwei 4-input LUTs (LookUp Tables) und eine 3-input LUT
  - zwei SR D-FlipFlops
  - Bypass-Pfade und carry/cascade-Logic
- PSM = Programmable Switch Matrix
  - 10 Verbindungspunkte pro Matrix
  - Jeder Verbindungspunkt enthält 6 Passtransistoren
    - ➔ jede Verbindung zwischen vier Richtungen möglich

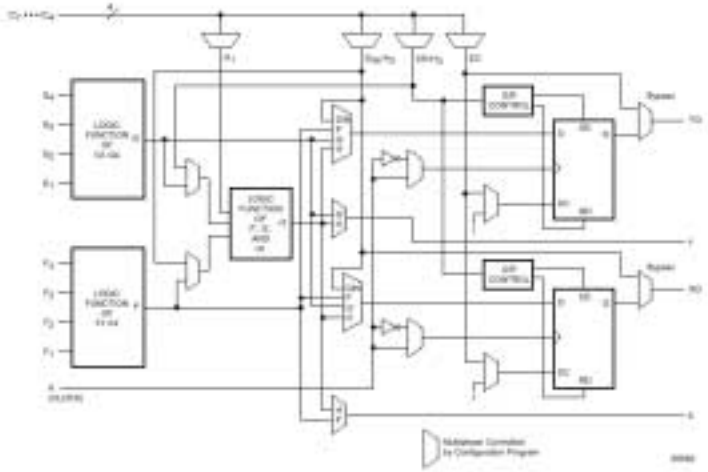
Xilinx 4000E Architektur



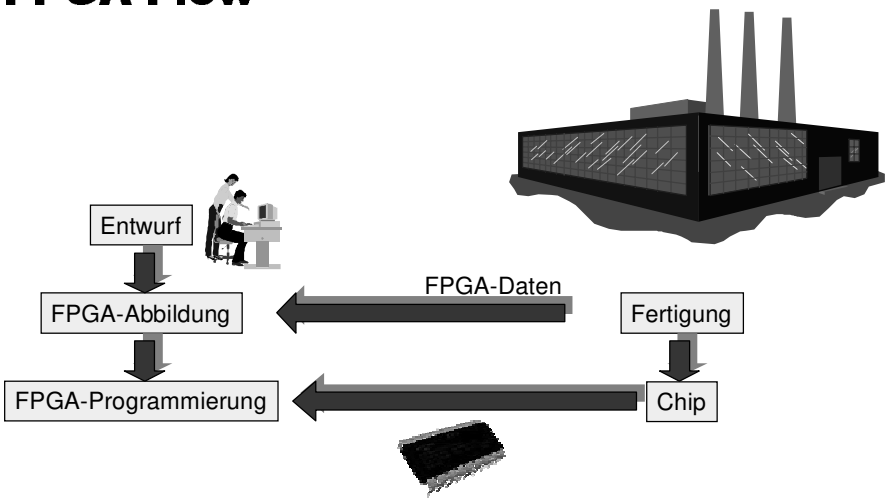
Xilinx 4000E Verbindungsmatrix



Xilinx 4000E CLB



FPGA-Flow



Partitionierung

- umfangreiche Schaltungen sind zu komplex um als „Ganzes" plaziert werden zu können

→ Partitionierung des Problems

- Zuordnung von n Objekten $O=\{o_1, \ldots, o_n\}$ zu m Partitionen $P=\{p_1, \ldots, p_m\}$, so daß

  $p_1 \cup \ldots \cup p_m = O$

  $p_1 \cap \ldots \cap p_m = \varnothing$

  die Kosten $c(P)$ minimal sind

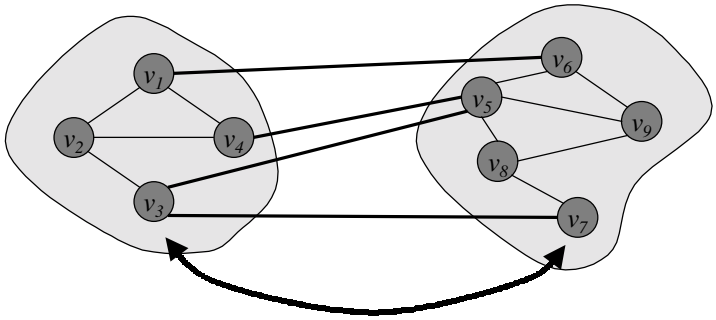
➔ das allgemeine Partitionierungsproblem ist NP vollständig

Allgemeine Partitionierungsverfahren

- **exakte Lösungsverfahren**
  - Enumeration der Lösungen
  - Integer Linear Programs (ILP)
- **heuristische Lösungsverfahren**
  - konstruktive Verfahren
    - random mapping
    - hierarchical clustering
  - iterative Verfahren
    - Kernighan-Lin Algorithmus
    - Simulated Annealing
    - evolutionäre Algorithmen

Kernighan-Lin

- Erzeugung von Bipartitionen:
  vertausche diejenigen Objekt in die jeweils andere Gruppe, die den größten Kostengewinn verursachen



Kernighan-Lin - Erweiterung

- Vertausche diejenigen Objekt, die den größten Kostengewinn oder den kleinsten Kostenzuwachs verursachen
- solange eine bessere Partition gefunden wird:
  - vertausche versuchsweise jede Paarung
  - nimm von diesen *(Versuchs-)*Partitionen diejenige mit dem besten Kostenverhältnis und führe die entsprechenden Umgruppierungen durch
  - einmal vertauschte Objekte werden im weiteren Verlauf nicht wieder vertauscht

Kernighan-Lin

- entkommt aus lokalen Minima
- Zeitkomplexität $O(n^3)$
- Partitionierung in m Blöcke: $O(m \cdot n^3)$

Simulated Anealing

- simuliertes Ausglühen
  - Metalle und Glas nehmen beim Abkühlen unter bestimmten Bedingungen einen Zustand minimaler Energie ein:
    - bei jeder Temperatur wird ein thermodynamisches Gleichgewicht erreicht
    - die Temperatur wird beliebig langsam erniedrigt
  - Wahrscheinlichkeit, dass ein Teilchen in einen Zustand höherer Energie springt

$$P(e_i,e_j,T) = e^{\frac{e_i - e_j}{kT}}$$

Simulated Anealing

Anwendung auf kombinatorische Optimierung

- Energie = Kosten der Lösung
- Verringerung der Kosten mit simulierter Temperatur, aber manchmal auch akzeptieren von Kostenerhöhungen

Simulated Anealing

```
temp = temp_start
cost = c(P)
WHILE (Frozen() == FALSE) {
  WHILE (Equilibrium() == FALSE) {
      P' = RandomMove(P)
      cost' = c(P')
      deltacost = cost' - cost
      IF (Accept(deltacost,temp) > random[0,1)) {
            P = P'
            cost = cost'
      }
  }
  temp = DecreaseTemp(temp)
}
```

$$min(1, e^{\frac{deltacost}{k*temp}})$$

Simulated Anealing

- Abkühlung: DecreaseTemp(), Frozen()
  - temp_start = 1.0
  - temp = α • temp (typisch: 0.8 = α =0.99)
  - Abbruch bei temp < temp_min
    oder wenn sich keine Verbesserung mehr ergibt
- Gleichgewicht: Equilibrium()
  - nach bestimmter Anzahl von Iterationen
  - oder wenn sich keine Verbesserung mehr ergibt

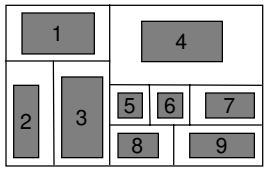
Simulated Anealing

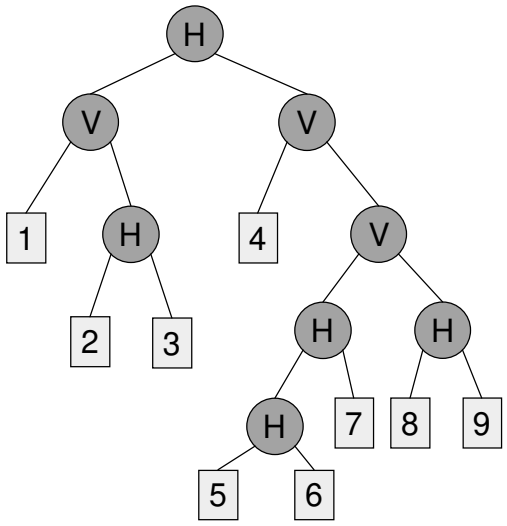
- Zeitkomplexität
  - von exponentiell bis konstant, je nach Implementierung der Funktionen Equilibrium, DecreaseTemp, Frozen
  - je länger die Laufzeit, desto besser die Ergebnisse
  - üblich: Funktionen so konstruiert, dass polynomielle Laufzeit erreicht wird

Plazierung durch Slicing

- Unterteile die Chipfläche in zwei Hälften
- Partitioniere die Objekte so in zwei Partitionen, dass
  - beide Partitionen etwa die gleiche Fläche benötigen
  - die Zahl der Verbindungen zwischen beiden Partitionen minimal ist
- wiederhole diese Schritte, bis Partitionen klein genug sind, um sie zu platzieren

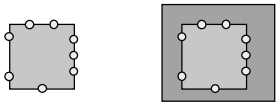
Slicing





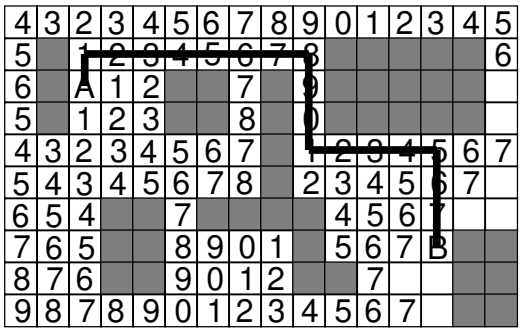
Slicing

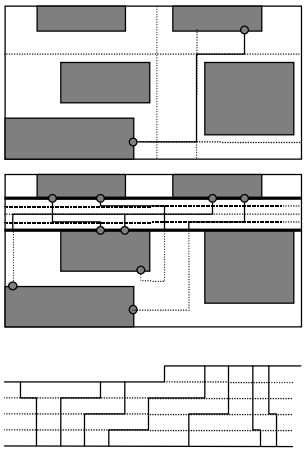
- Abschließend werden
  - genaue Position
  - Orientierung
  - Seitenverhältnis festgelegt
- Beim Positionieren muß auch Platz für die Verdrahtung eingerechnte werden, z.B. „halo“



Verdrahtung

- Verfahren nach Lee



Verdrahtung

- High-Tower-Algorithmus
- Channel-Routing zweilagig
- River-Routing einlagig