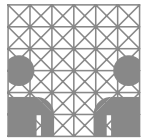


– VLSI- und Systementwurf – Methoden und Werkzeuge

https://tams.informatik.uni-hamburg.de

Andreas Mäder

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

November 2020

Die folgenden Folien sind ein überarbeiteter Auszug aus der bis 2011 angebotenen Vorlesung 64-613 Rechnerarchitekturen und Mikrosystemtechnik

Das komplette Material findet sich unter
https://tams.informatik.uni-hamburg.de/lectures/2011ws/vorlesung/ram

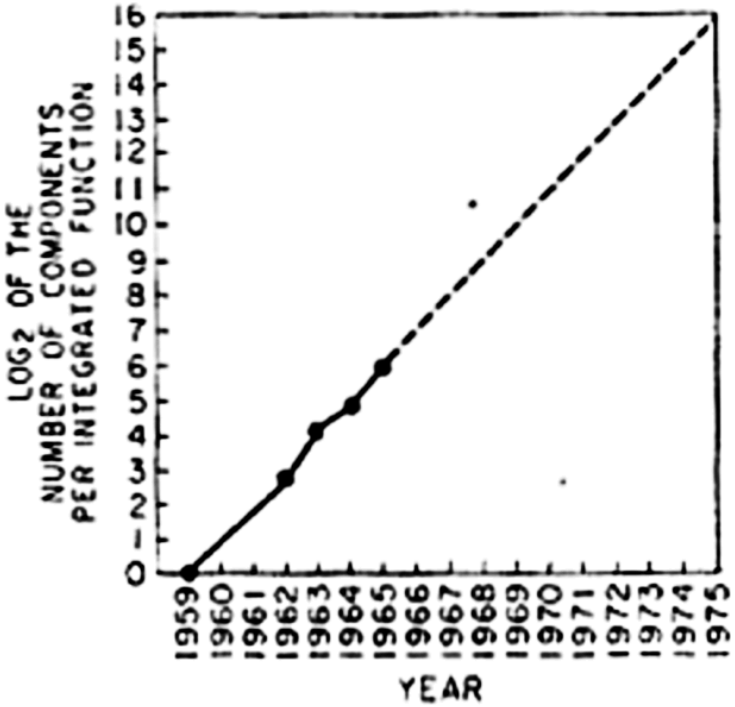
Gliederung

Moore's Law

Moore's Law

Gordon Moore, Mitgründer von Intel, 1965

Speicherkapazität von ICs vervierfacht sich alle drei Jahre



Gordon Moore 1965:
„Cramming more components onto integrated circuits"

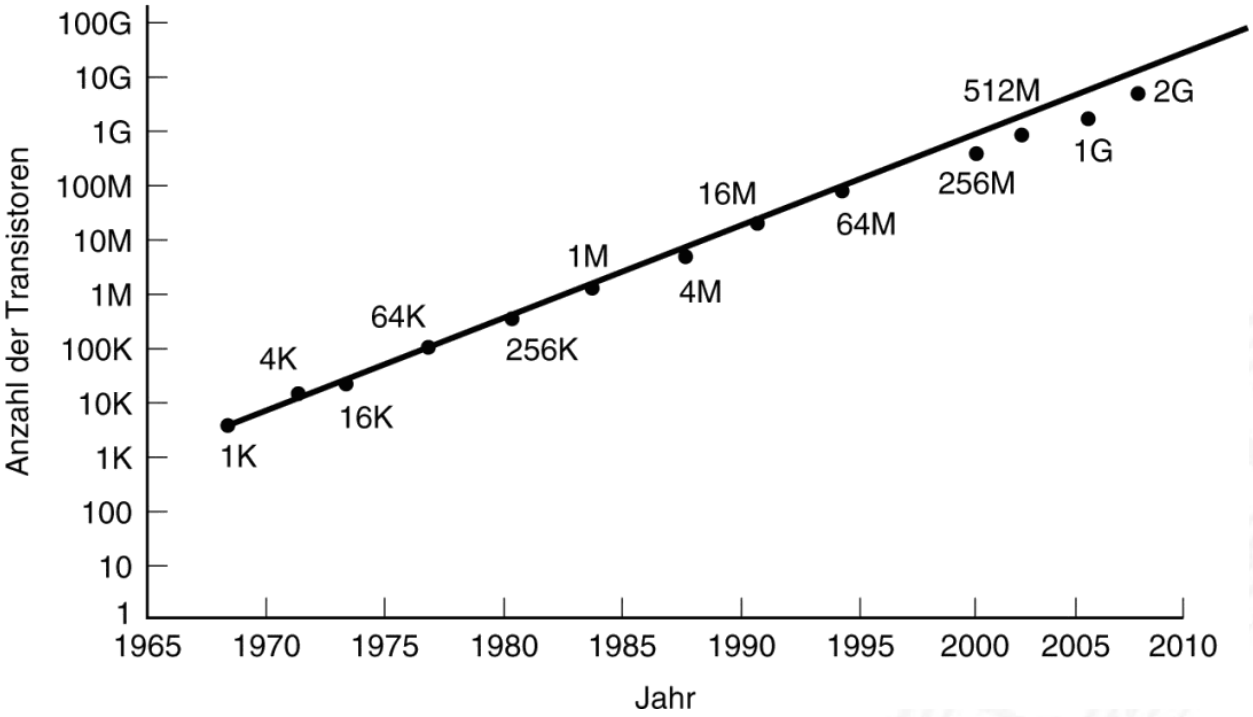
⇒ schnelles exponentielles Wachstum

⇒ bei gleicher Funktion kleinere und billigere Chips

⇒ bei gleicher Größe leistungsfähigere Chips

Moore's Law (cont.)

Transistoren pro Speicherchip

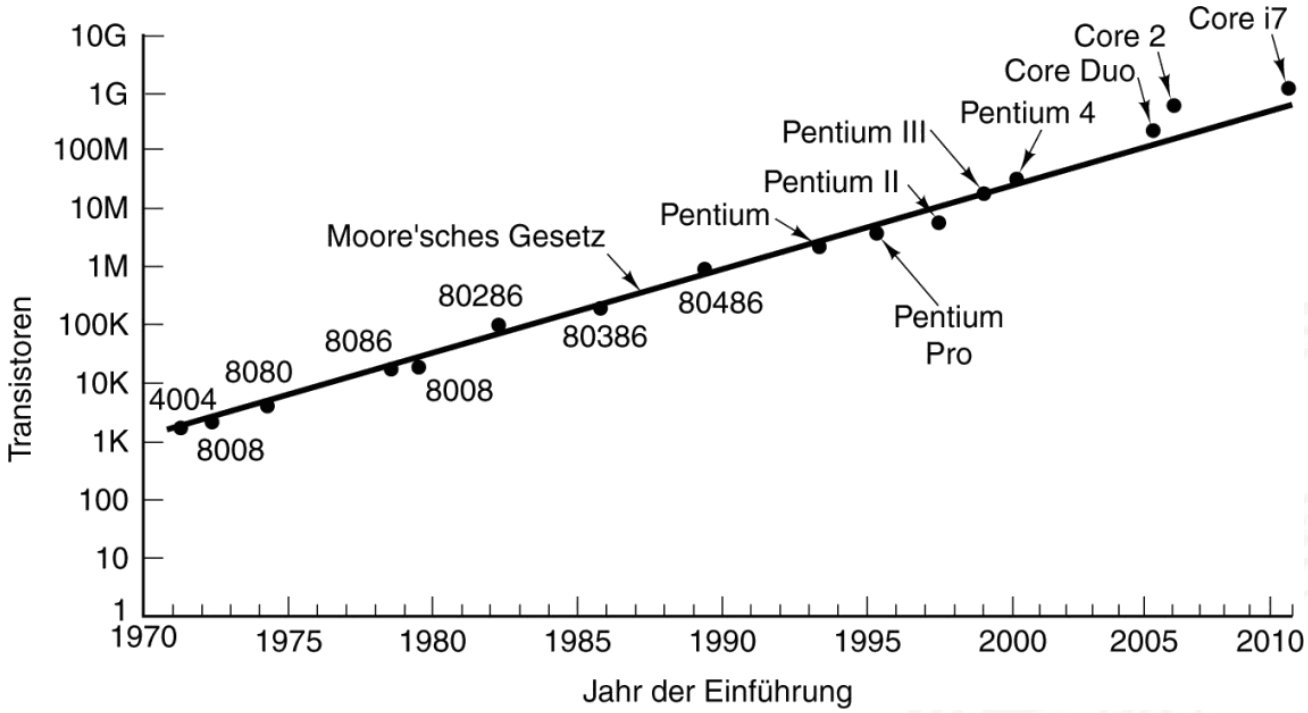


[TA14]

- Vorhersage: 60% jährliches Wachstum der Transistoranzahl pro IC

Moore's Law (cont.)

Evolution der Prozessoren



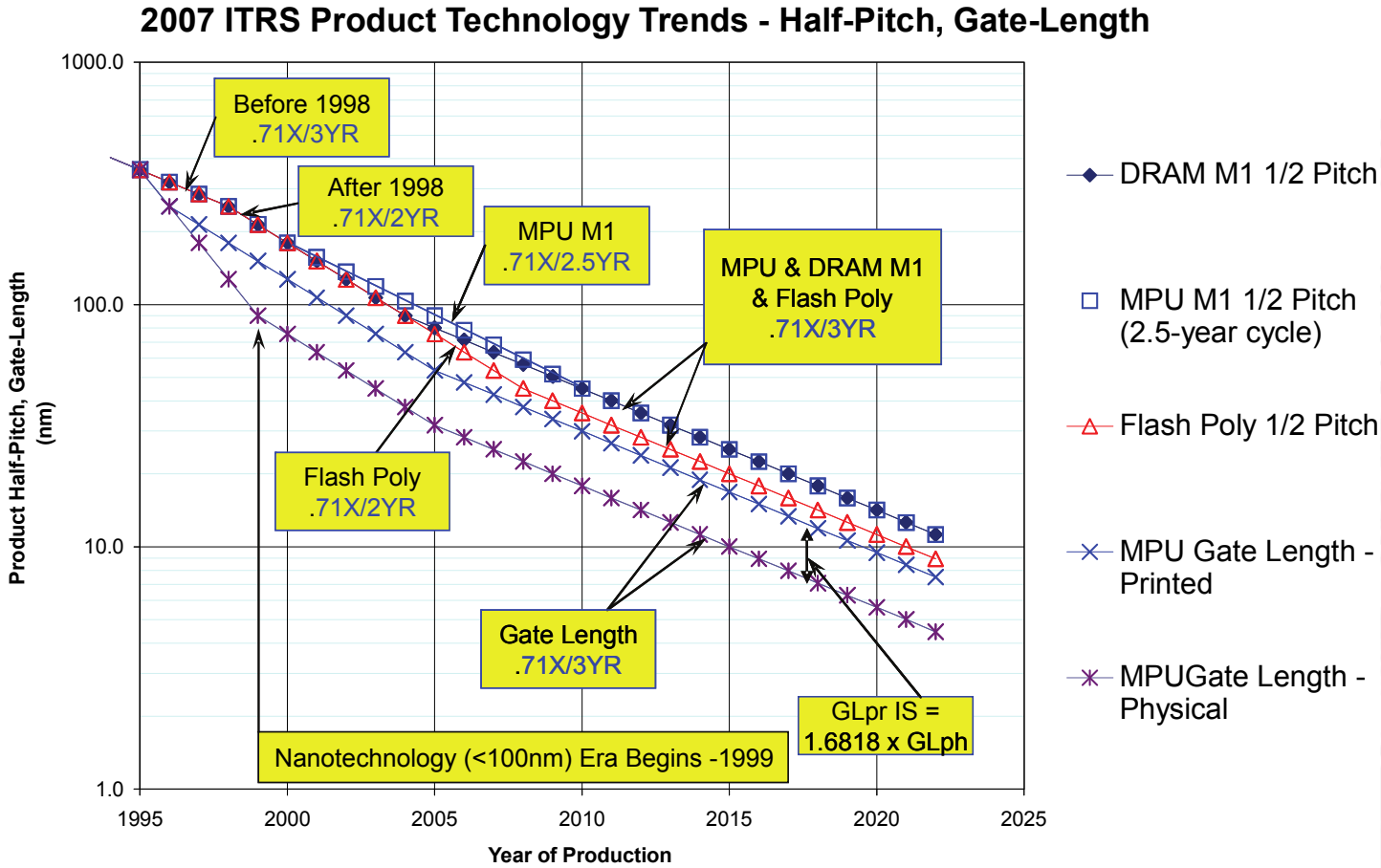
[TA14]

Modell		Typ	Jahr	# Trans.
Xeon Broadwell E5 v4	Intel	CPU	2016	7,2 Mrd.
Sparc M7	Oracle	CPU	2015	> 10,0 Mrd.
GP100 Pascal	Nvidia	GPU	2016	15,3 Mrd.
Stratix 10	Intel (Altera)	FPGA	2016	> 30,0 Mrd.

Moore's Law: Antrieb und Konsequenzen

Technologischer Fortschritt

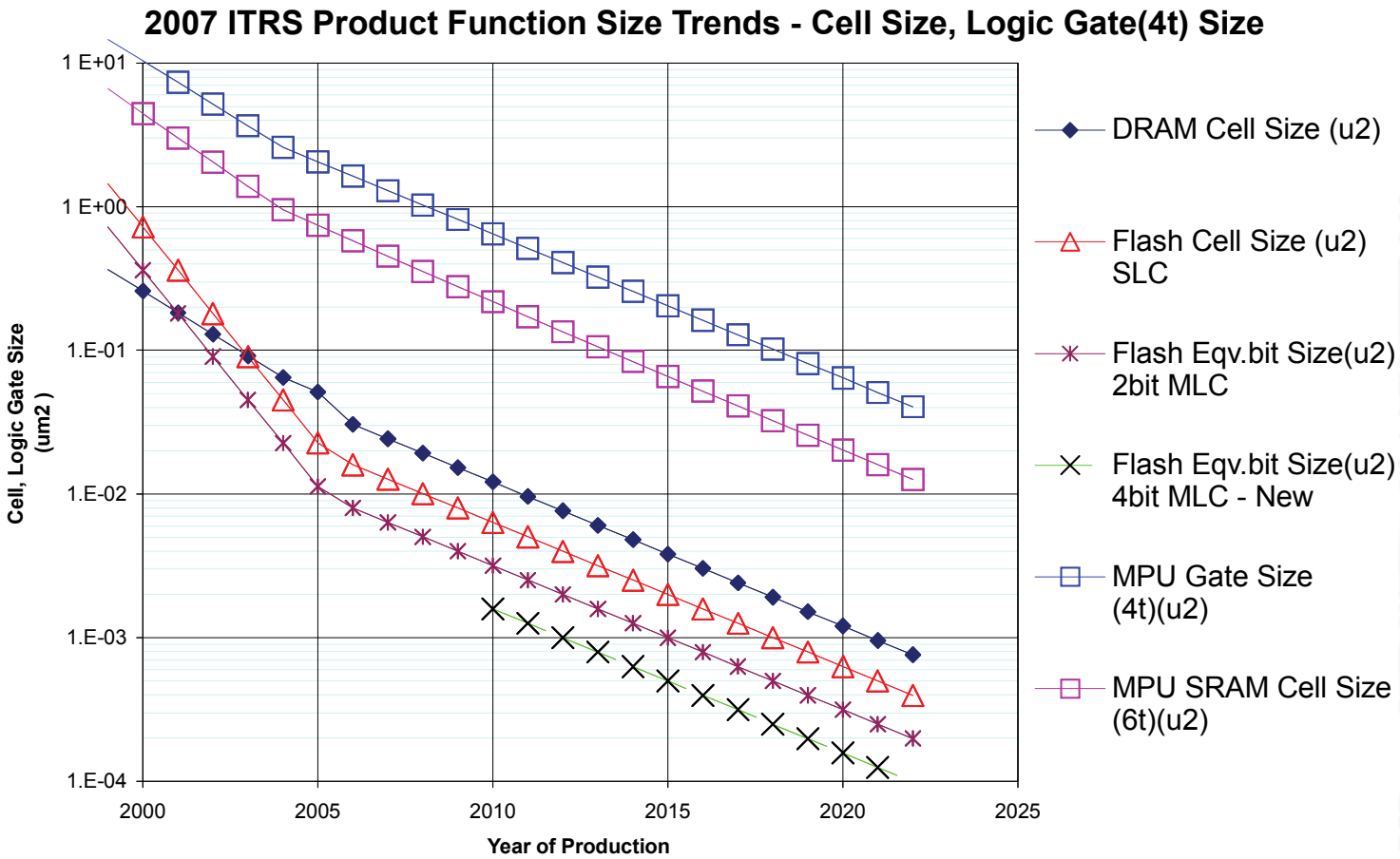
- Verkleinerung der Strukturbreite



[ITRS07]

Moore's Law: Antrieb und Konsequenzen (cont.)

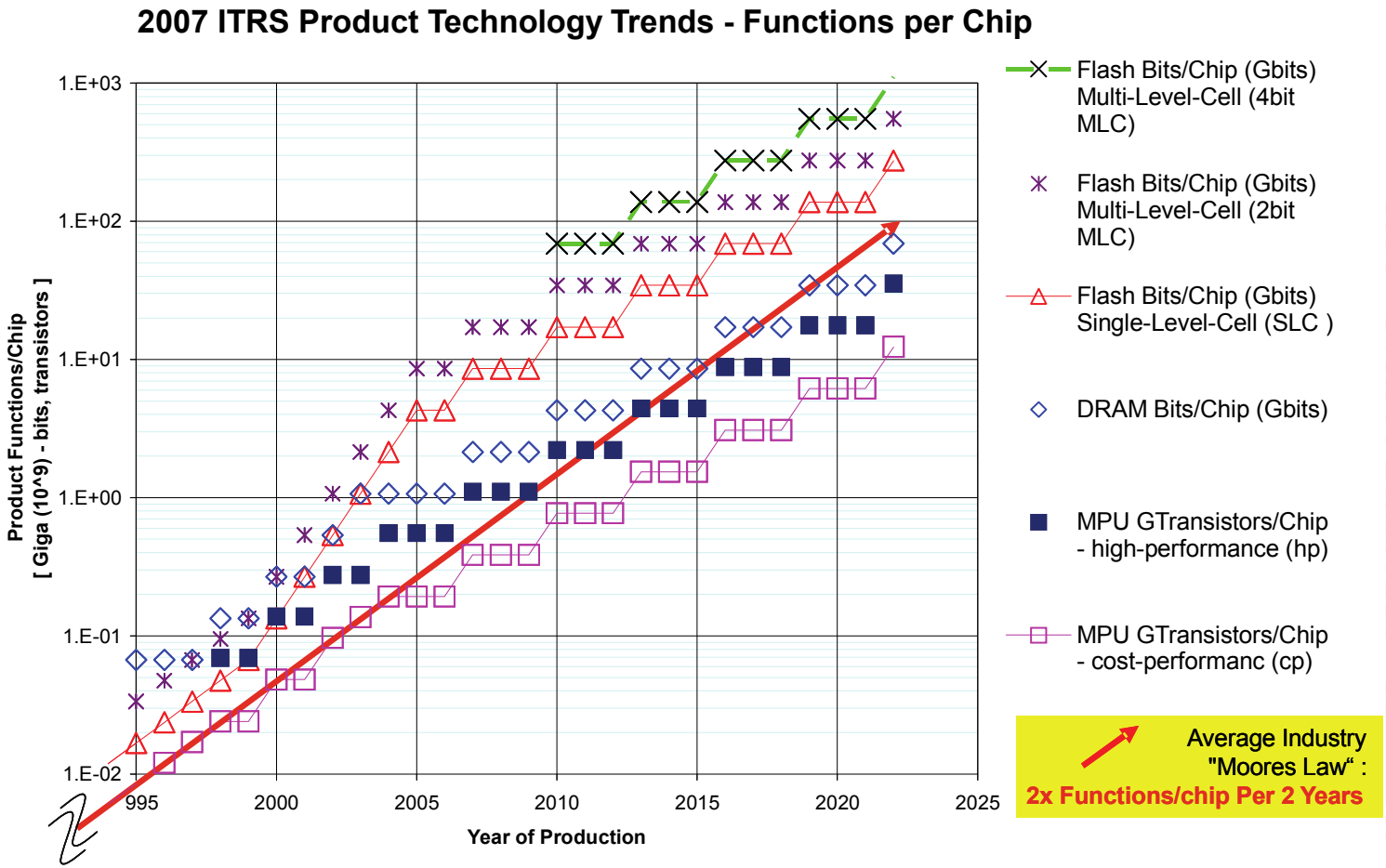
- höhere Integrationsdichte



[ITRS07]

Moore's Law: Antrieb und Konsequenzen (cont.)

- mehr Funktionen pro IC

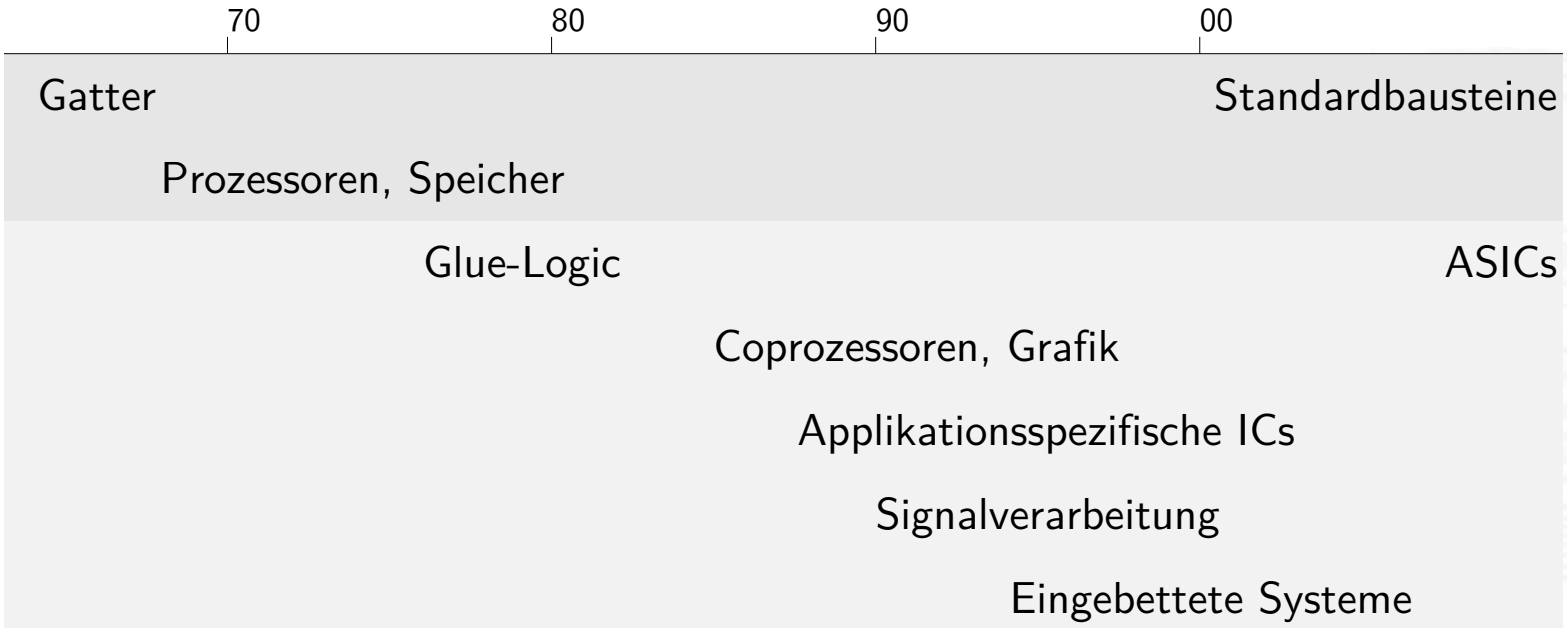


[ITRS07]

Moore's Law: Antrieb und Konsequenzen (cont.)

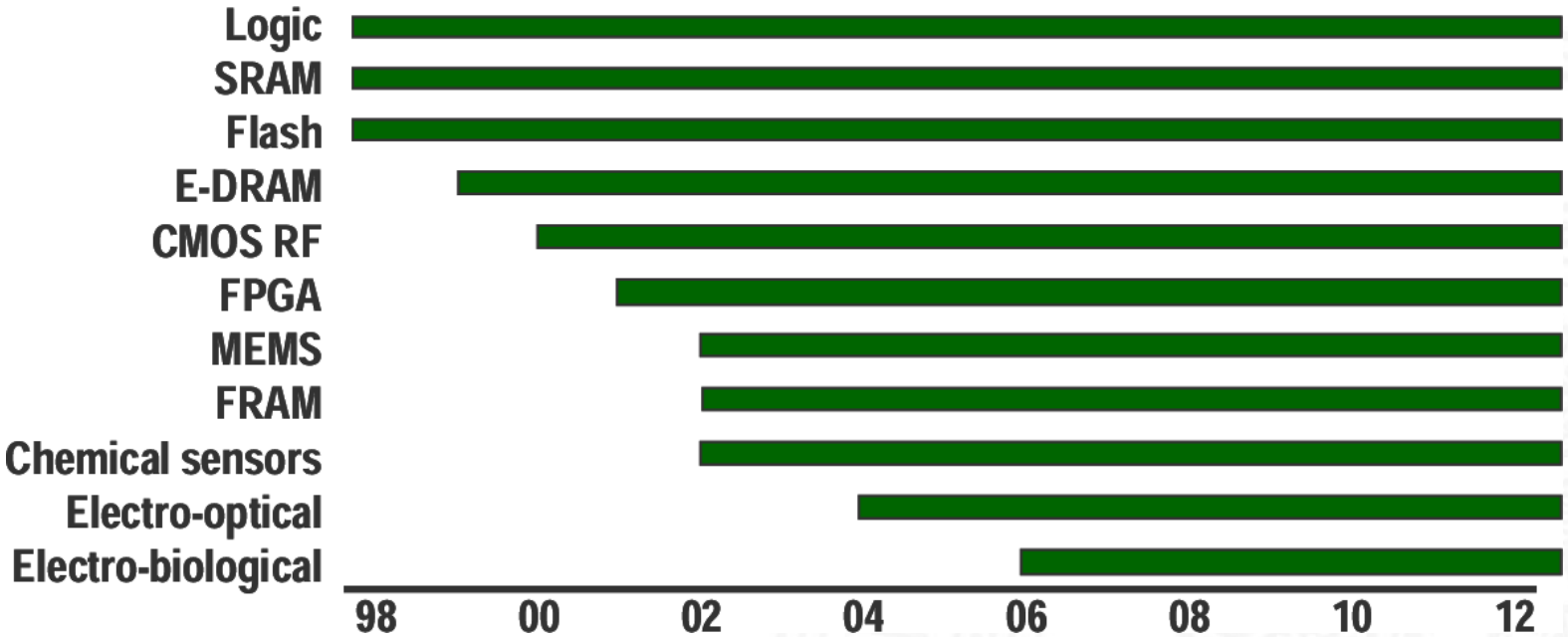
Neue Applikationen

- von Standardbausteinen zu ASICs und Systemen
- digitale Anwendungen



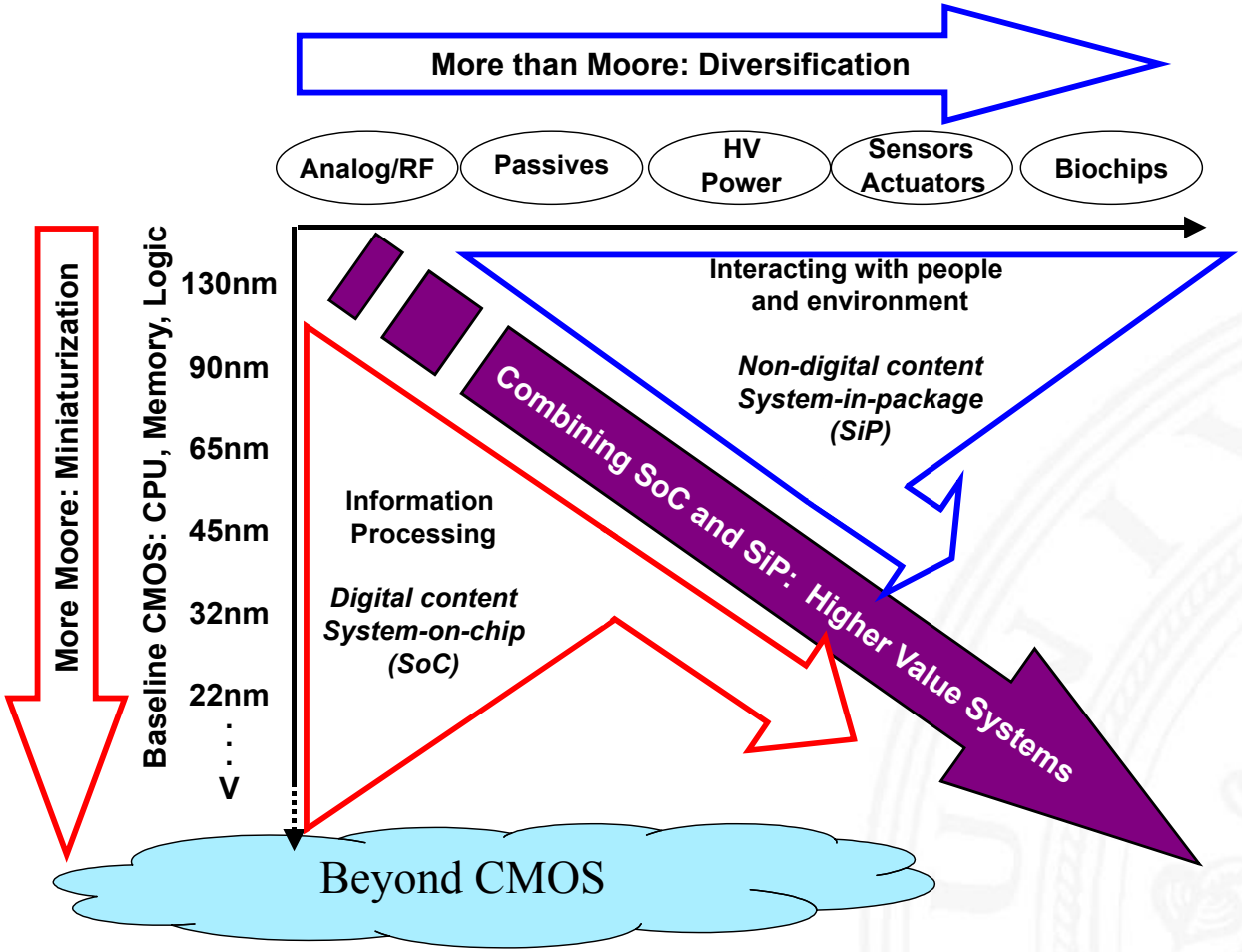
Moore's Law: Antrieb und Konsequenzen (cont.)

- Erweiterungen durch Integration neuer Technologien
  - Speicher
  - analoge Schaltungen
  - Micromechanik / -Sensorik



Moore's Law: Antrieb und Konsequenzen (cont.)

Übergang zu Systemen: SoC (**S**ystem **o**n a **C**hip)



[ITRS07]

Moore's Law: Antrieb und Konsequenzen (cont.)

Neue Anwendungsfelder

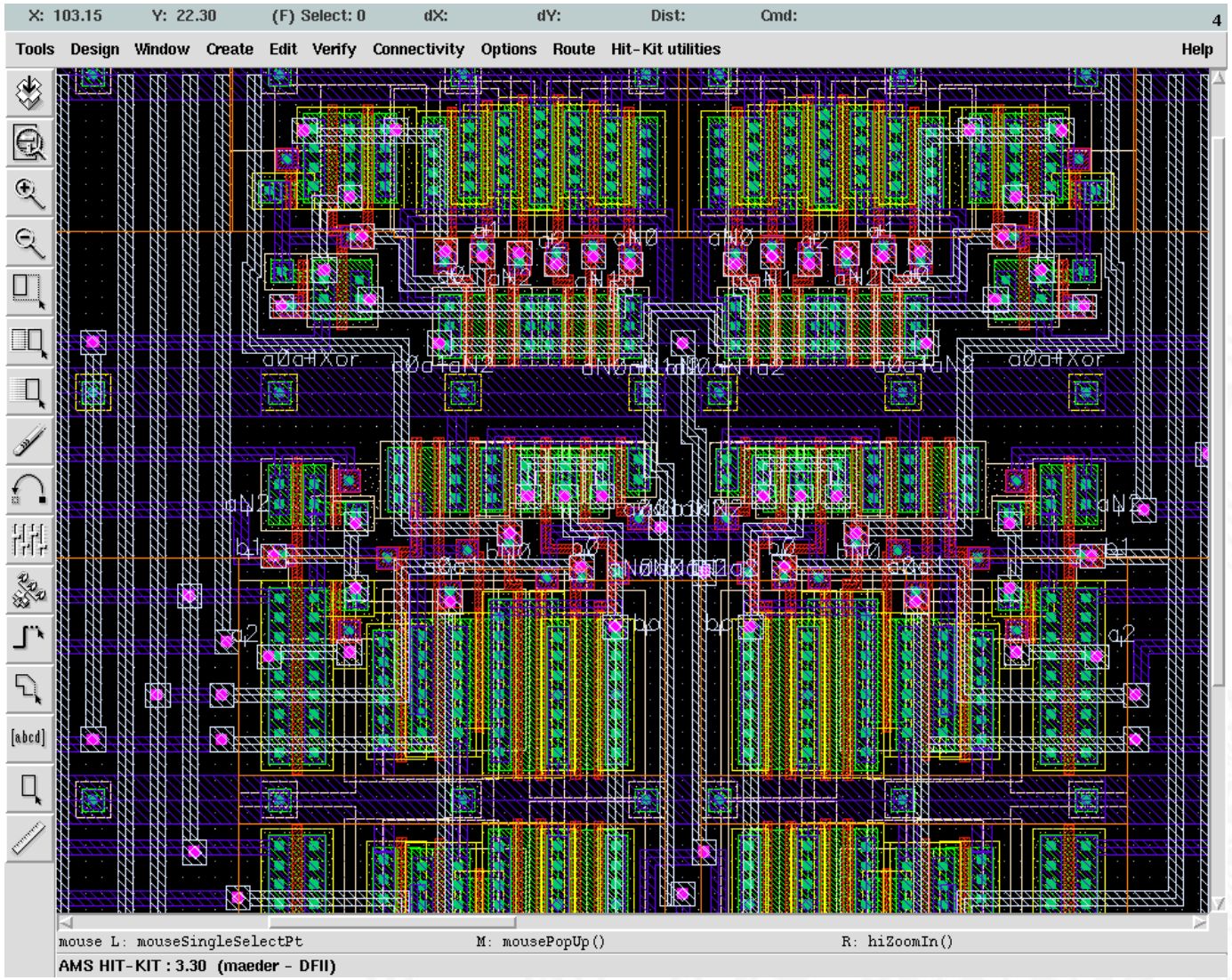
- „Computing“
- „Consumer Products“
- „Automotive“
- „Telecommunication“
- „mobile Applications“

Neue Methoden und Werkzeuge im Chipentwurf

- enges Zusammenwirken mit der technischen Entwicklung und den Anforderungen durch die Applikationen

Wie wird Entworfen?

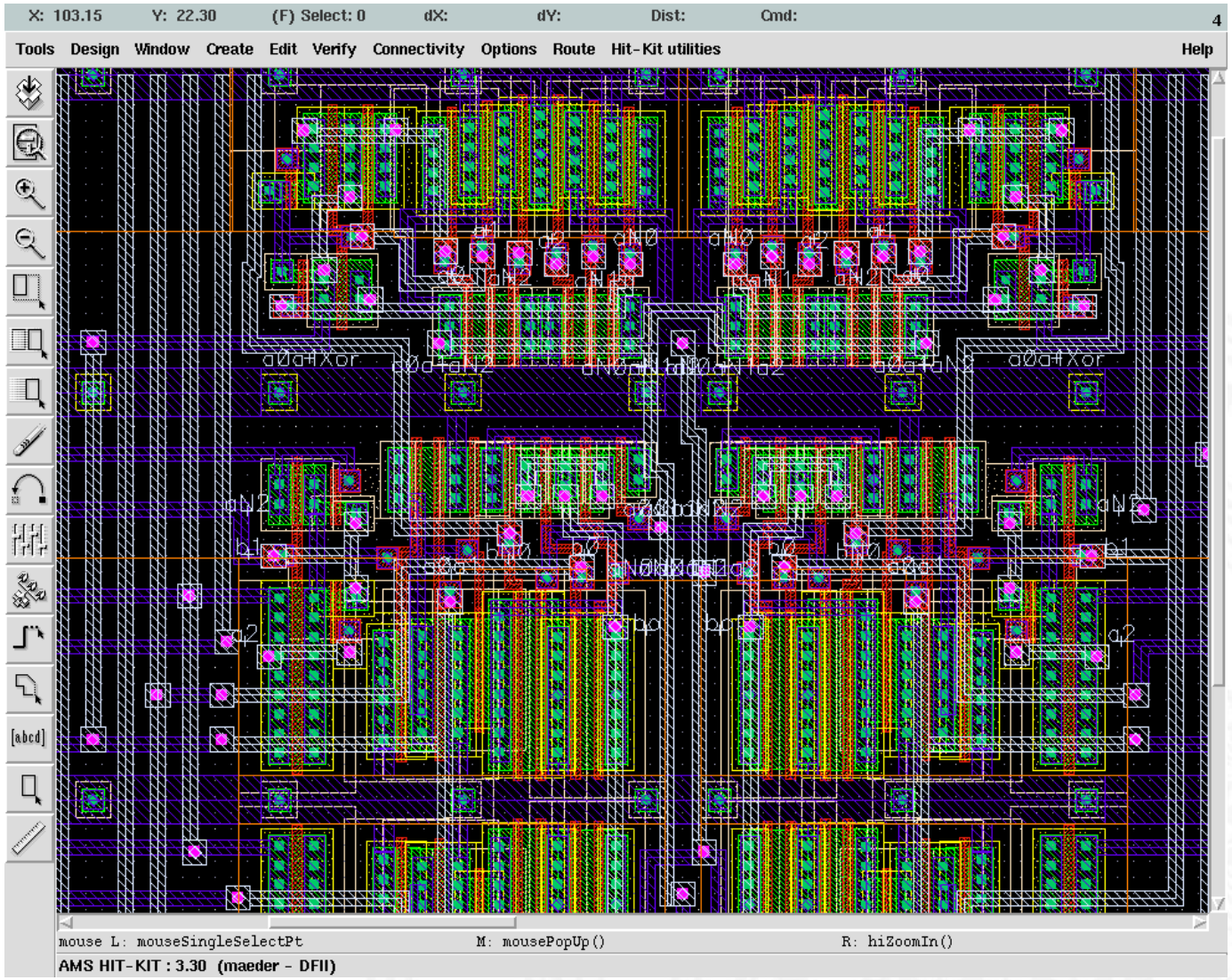
Hardwareentwurf



Wie wird Entworfen?

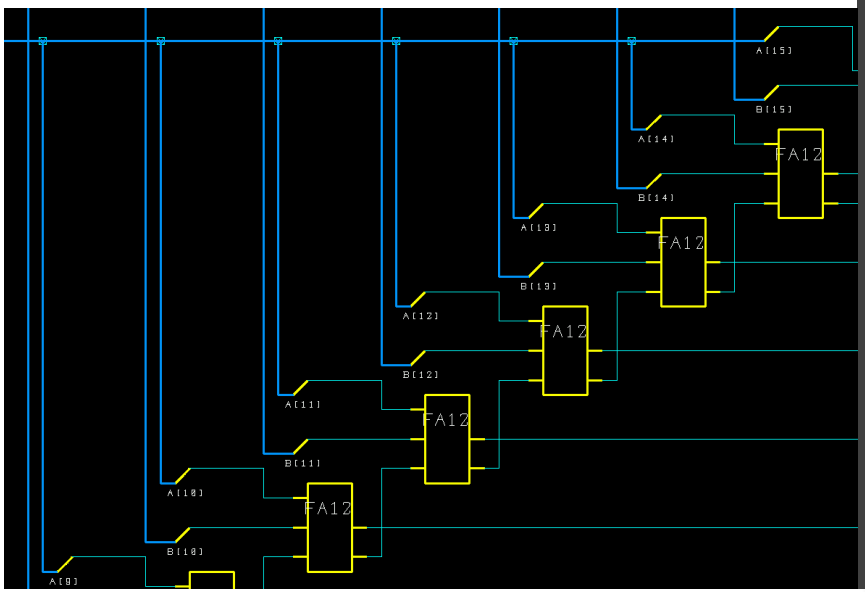
Hardwareentwurf

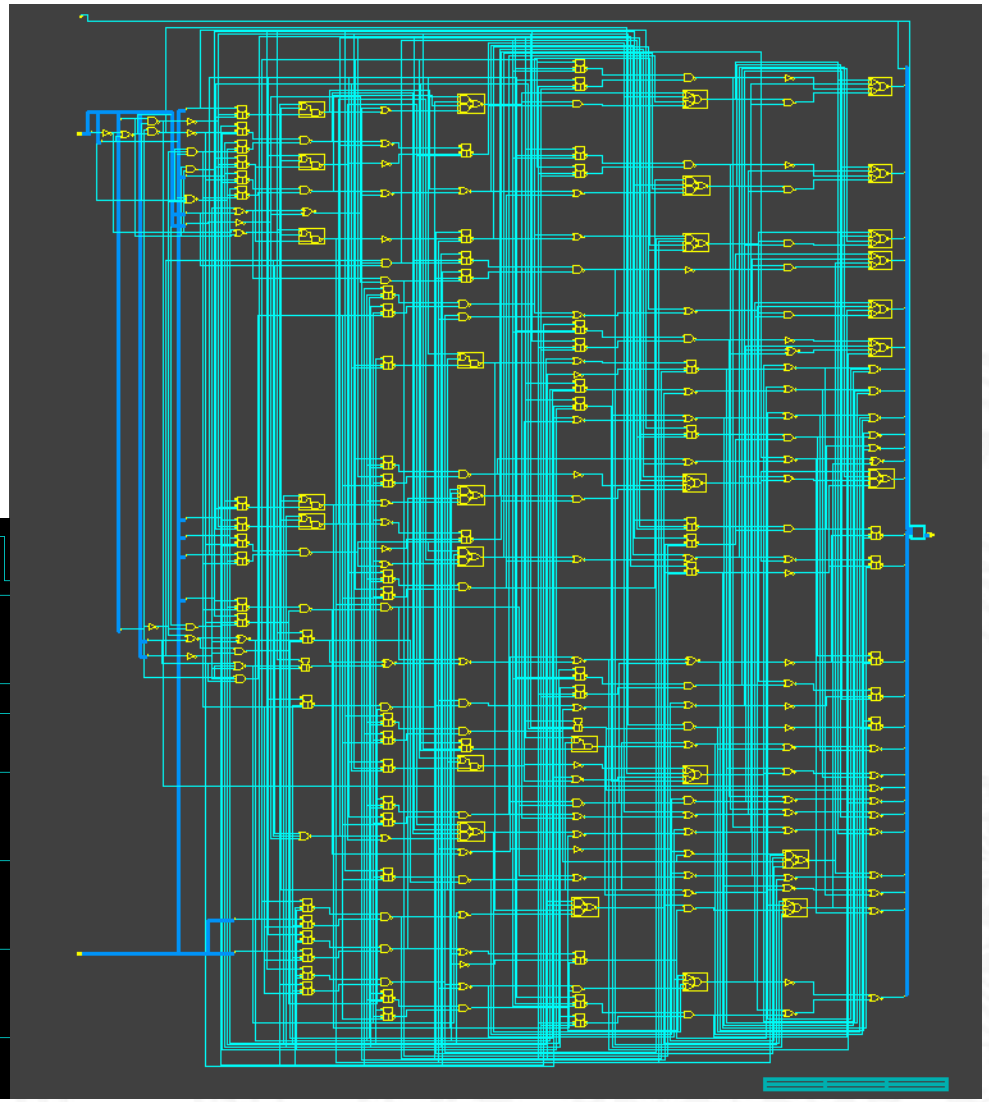
. . . so nicht
meistens



Wie wird Entworfen?

Hardwareentwurf

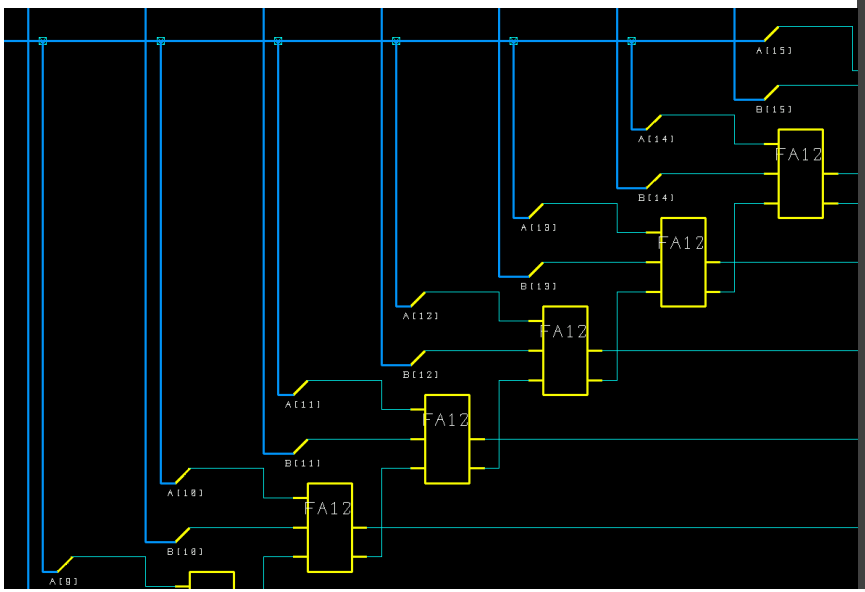


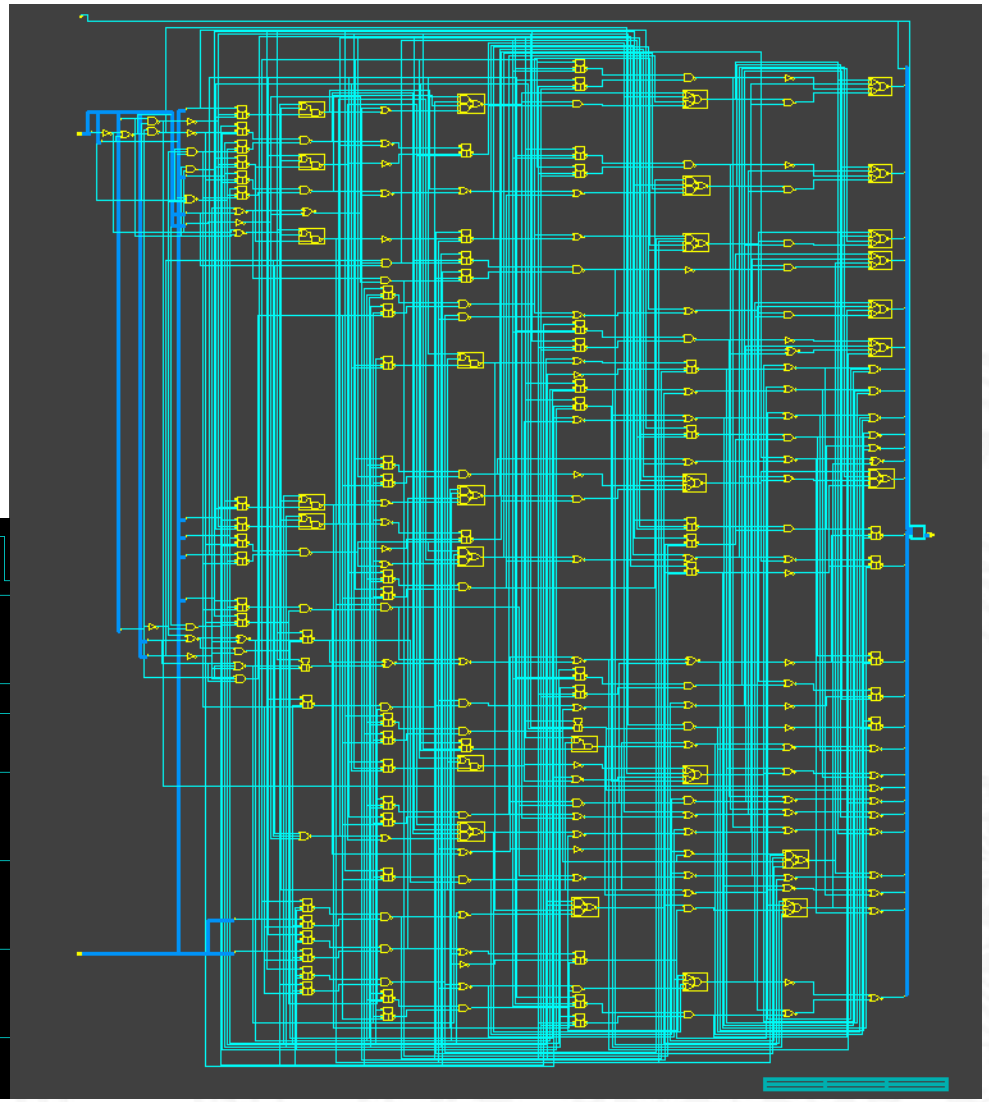


Wie wird Entworfen?

Hardwareentwurf

. . . so auch nicht

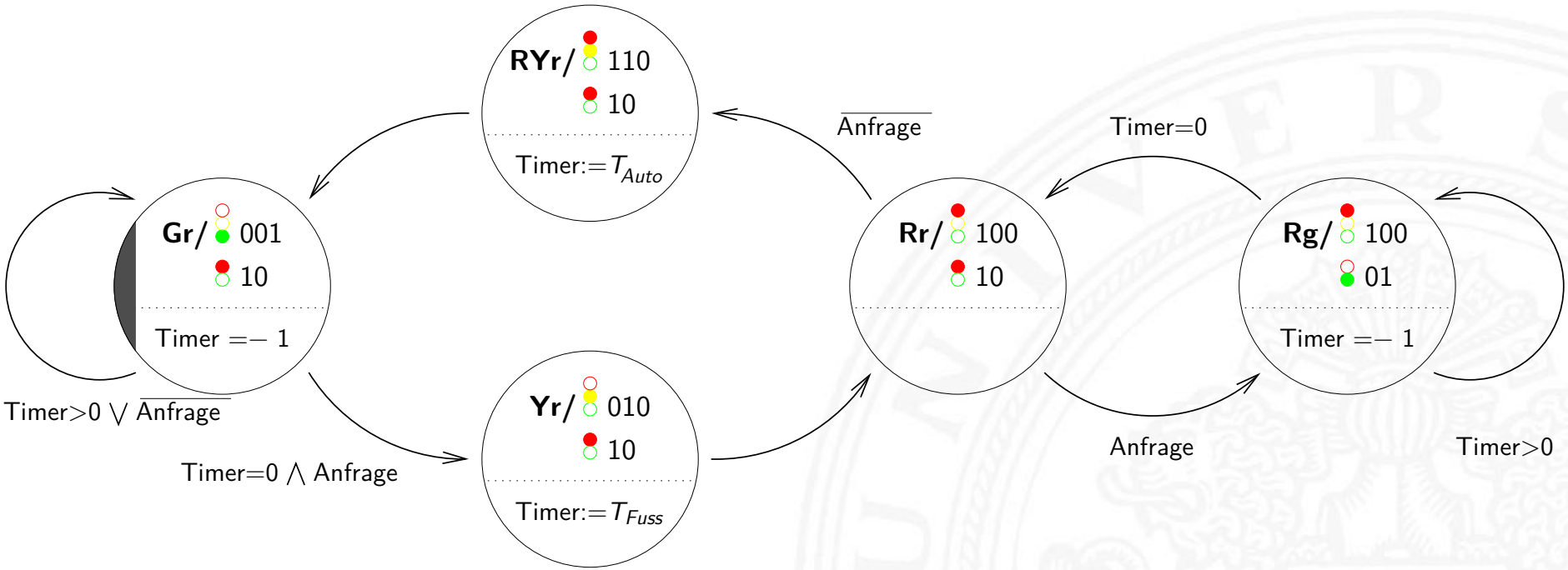




Wie wird Entworfen?

Hardwareentwurf

. . . sondern so



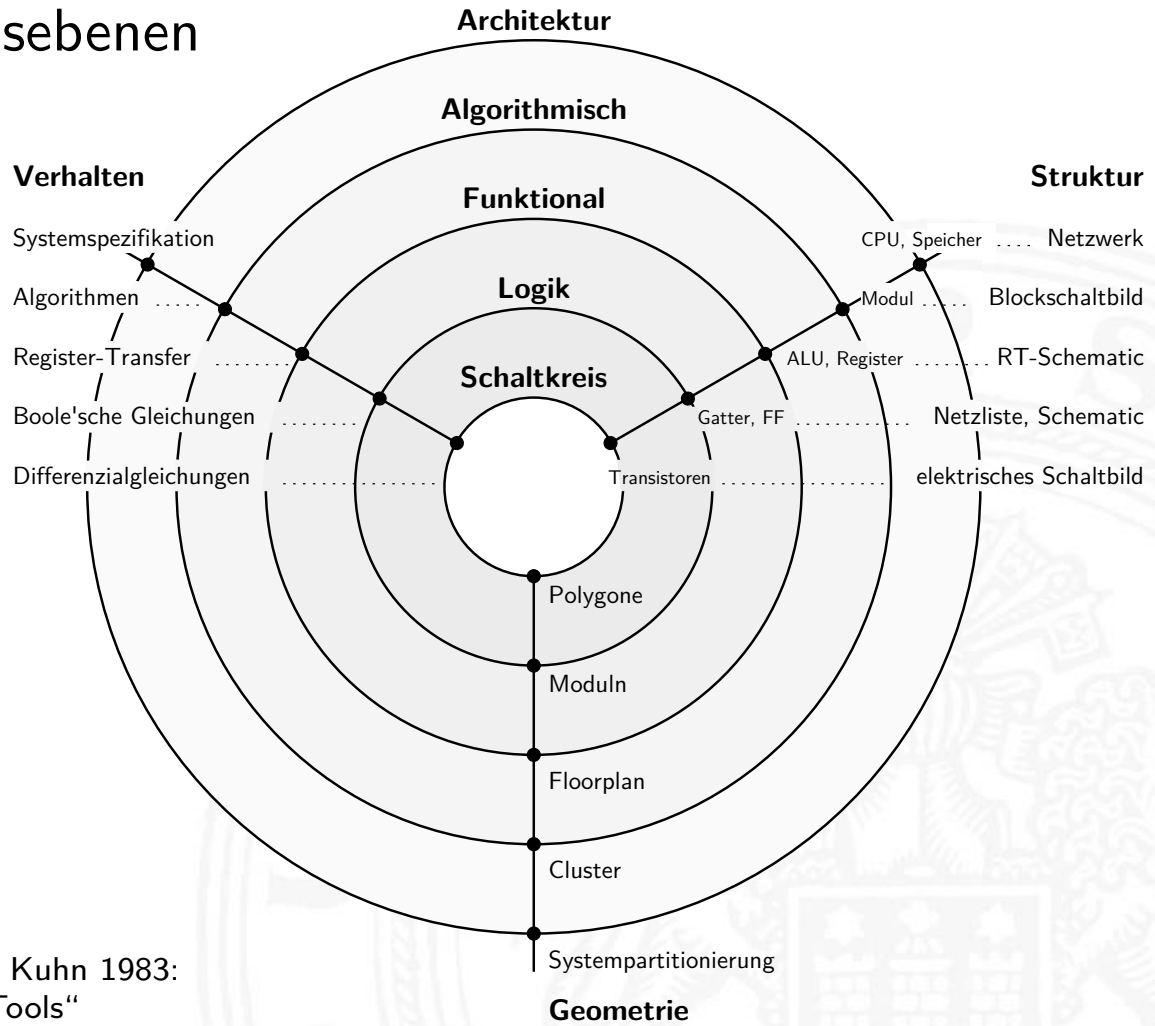
Wie wird Entworfen? (cont.)


```
mainP: process (clk, rst) is
  type stateTy is (Gr, Yr, Rr, Rg, RYr);
  variable      timer           : integer range 0 to maxWalkC;
  variable      state           : stateTy;
  variable      request         : boolean;
begin
  if rst = '0' then             ------------------------- async. reset
      liCar     <= "001";       liWalk  <= "10";
      state     := Gr;
      timer     := 0;
      request   := false;
  elsif rising_edge(clk) then   ------------------------- clock edge
    case state is
    when Gr =>  ---------------------------------------- Green + red
      liCar     <= "001";       liWalk  <= "10";
      if (reqWalk = '1')  then  request := true;        -- store request
      end if;
      if    (timer > 0)   then  timer   := timer - 1;   -- no timeout
      elsif request       then  state   := Yr;          -- timeout and request
      end if;
    when Yr =>  ---------------------------------------- Yellow + red
      liCar     <= "010";       liWalk  <= "10";
      timer     := maxWalkC-1;                          -- init. timer
      state     := Rr;
    when Rr =>  ---------------------------------------- Red + red
      ...
```

Abstraktion im VLSI-Entwurf

Y-Diagramm / Gajski-Diagramm

- visualisiert Abstraktionsebenen
- Sichtweisen
  1. Funktion / Verhalten
  2. Struktur
  3. Geometrie

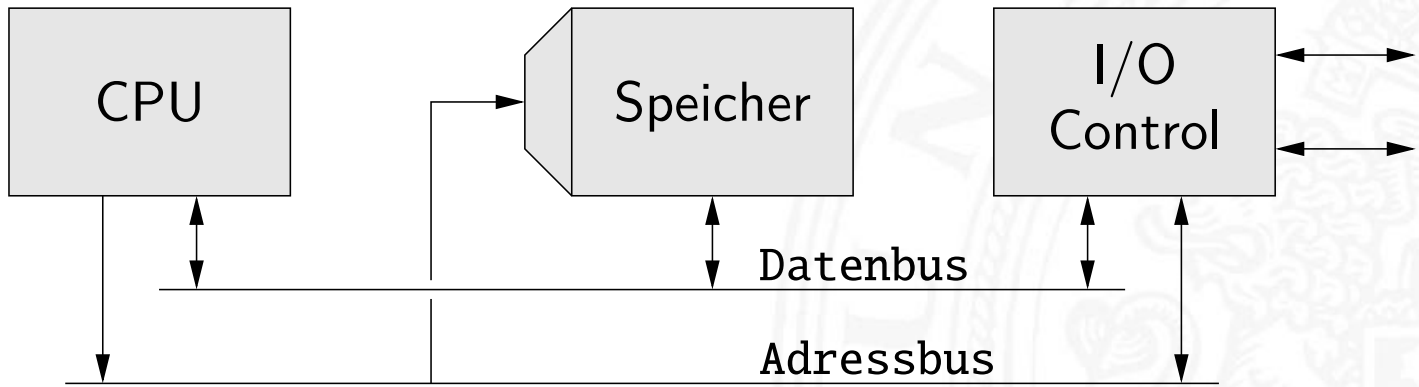


D. Gajski, R. Kuhn 1983:
„New VLSI Tools“

Abstraktion im VLSI-Entwurf (cont.)

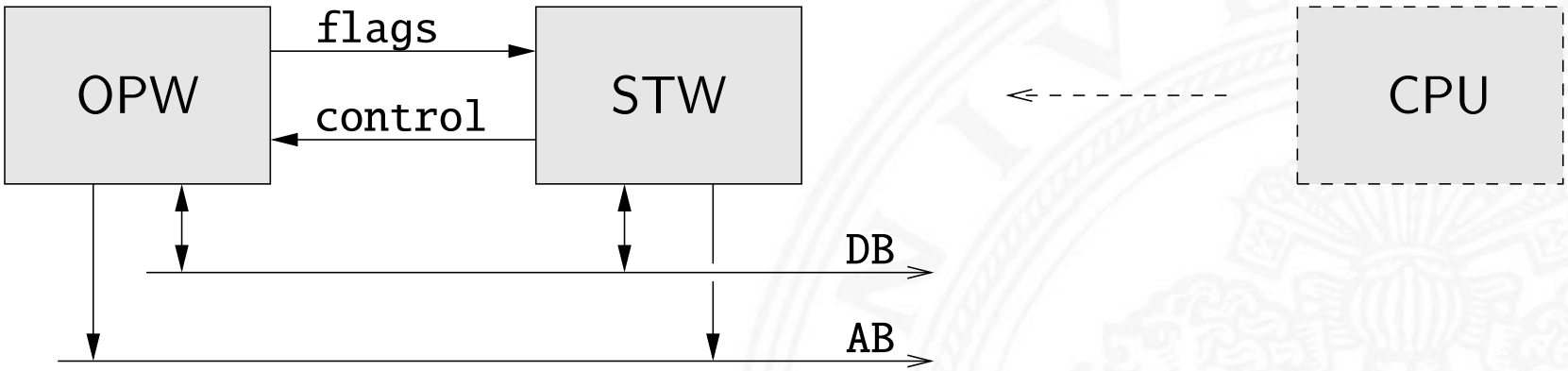
Abstraktionsebenen

- keine einheitliche Bezeichnung in der Literatur
- Architekturebene
  - Funktion/Verhalten Leistungsanforderungen
  - Struktur Netzwerk
    aus Prozessoren, Speicher, Busse, Controller ...
  - Nachrichten Programme, Protokolle
  - Geometrie Systempartitionierung



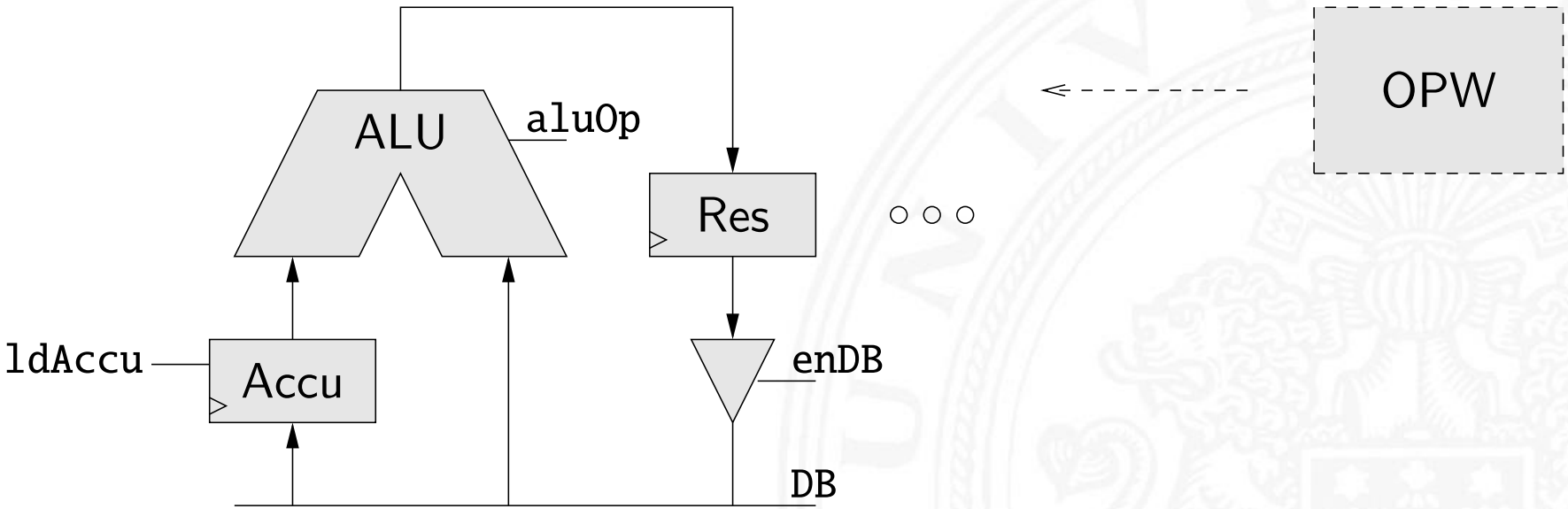
Abstraktion im VLSI-Entwurf (cont.)

- Hauptblockebene (Algorithmenebene, funktionale Ebene)
  - Funktion/Verhalten Algorithmen, formale Funktionsmodelle
  - Struktur Blockschaltbild
    aus Hardwaremodule, Busse ...
  - Nachrichten Protokolle
  - Geometrie Cluster



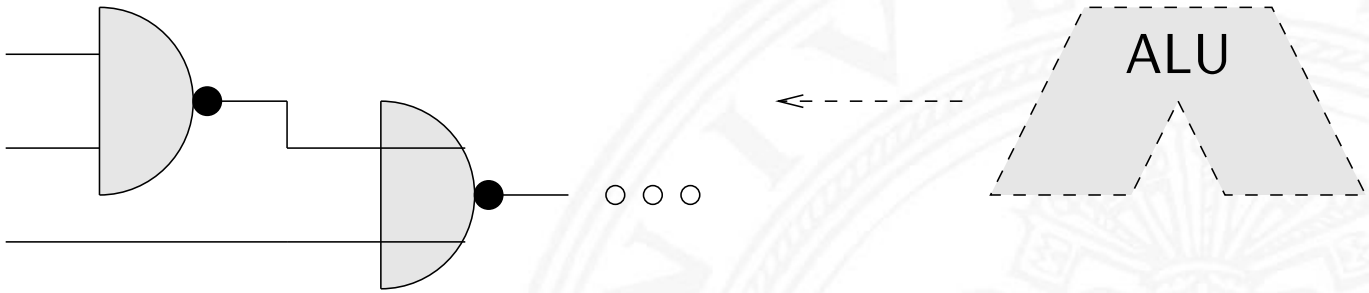
Abstraktion im VLSI-Entwurf (cont.)

- Register-Transfer Ebene
  - Funktion/Verhalten Daten- und Kontrollfluss, Automaten ...
  - Struktur RT-Diagramm
    aus Register, Multiplexer, ALUs ...
  - Nachrichten Zahlencodierungen, Binärworte ...
  - Geometrie Floorplan



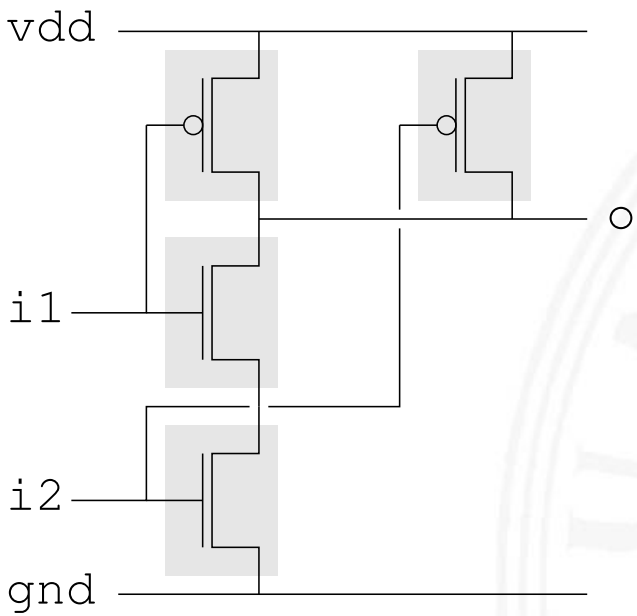
Abstraktion im VLSI-Entwurf (cont.)

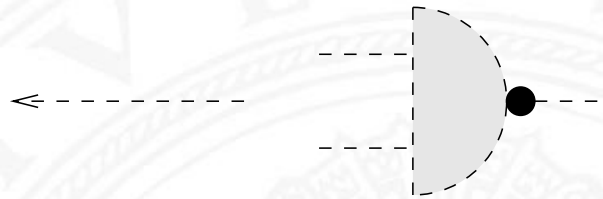
- Logikebene (Schaltwerkebene)
  - Funktion/Verhalten Boole'sche Gleichungen
  - Struktur Gatternetzliste, Schematic
    aus Gatter, Flipflops, Latches ...
  - Nachrichten Bit
  - Geometrie Moduln



Abstraktion im VLSI-Entwurf (cont.)

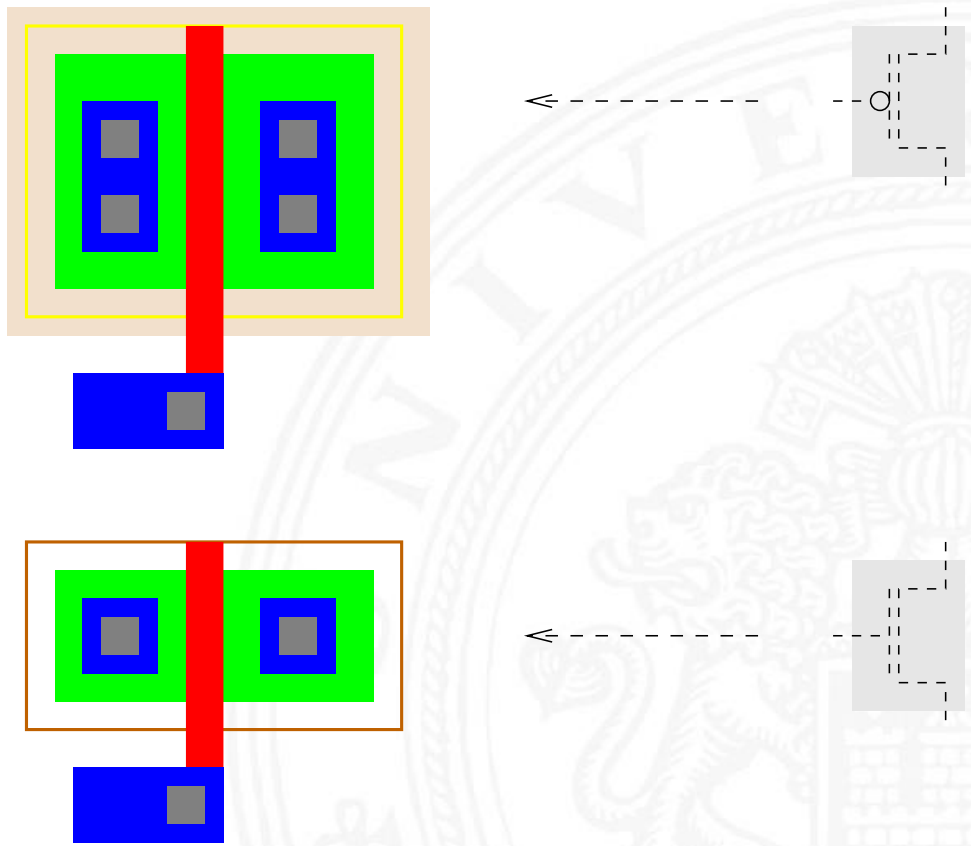
- elektrische Ebene (Schaltkreisebene)
  - Funktion/Verhalten Differentialgleichungen
  - Struktur elektrisches Schaltbild
    aus Transistoren, Kondensatoren ...
  - Nachrichten Ströme, Spannungen
  - Geometrie Polygone, Layout → physikalische Ebene





Abstraktion im VLSI-Entwurf (cont.)

- physikalische Ebene (geometrische Ebene)
  - Funktion/Verhalten partielle DGL
  - Struktur Dotierungsprofile



Entwurfsvorgehen

- Unterscheidung von *Struktur* und *Verhalten*
- Auf jeder Abstraktionsebene gibt es *elementare Einheiten* mit definiertem Verhalten
- Entwurfsaufgabe
  - ein gegebenes Verhalten in eine Strukturbeschreibung (aus elementaren Einheiten) der jeweiligen Ebene umzusetzen
  - jede dieser Einheiten ist ihrerseits in der nächst niedrigeren Abstraktionsebene entsprechend zu realisieren

  $\Rightarrow$ hierarchischer Entwurf, top-down

$\Rightarrow$ top-down: typisches Entwurfsvorgehen

$\Rightarrow$ bottom-up: Einflüsse auf höhere Abstraktionsebenen

- Zeitverhalten
- Schaltungstechniken
- Arithmetiken
- ...

Entwurfsvorgehen (cont.)

- Zentrale Bedeutung der Simulation, bzw. der Verifikation
- Entwurf als iterativer Prozess
  - Alternativen: „exploring the design-space“
  - Versionen
  - Teamarbeit

Gliederung

1. Entwurfsmethodik
2. EDA-Werkzeuge
   - Hierarchischer Entwurf
   - Werkzeuge
   - Probleme
3. Entwurfsstile

Hierarchischer Entwurf

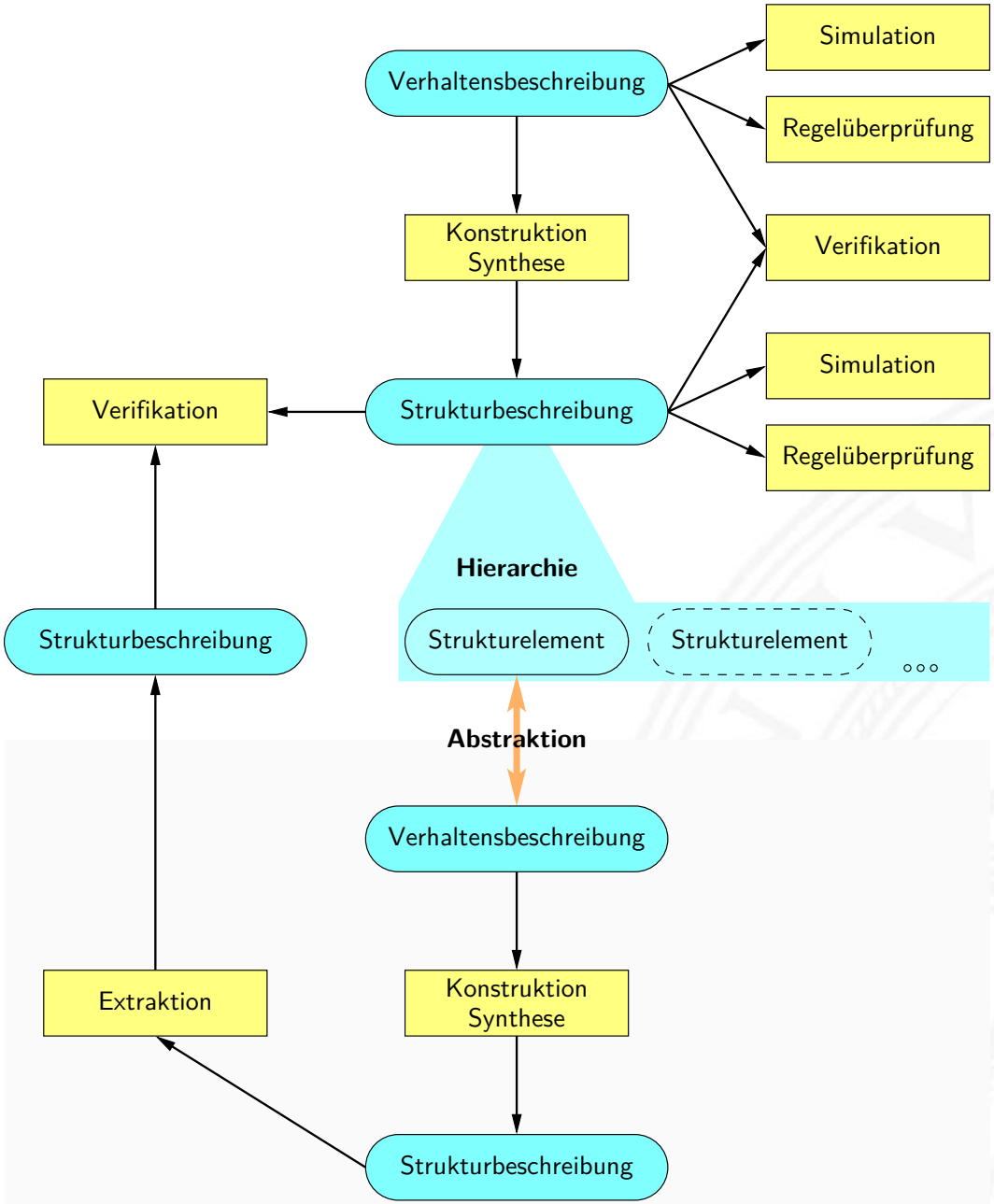
Nur durch neue Methoden und Werkzeuge konnte die Produktivität beim Chipentwurf während der letzten Jahre mit Moore's Law mithalten

- Änderungen in der Entwurfsmethodik

  Struktur $\Rightarrow$ Verhalten
  grafische Eingabe $\Rightarrow$ Hardwarebeschreibungssprachen

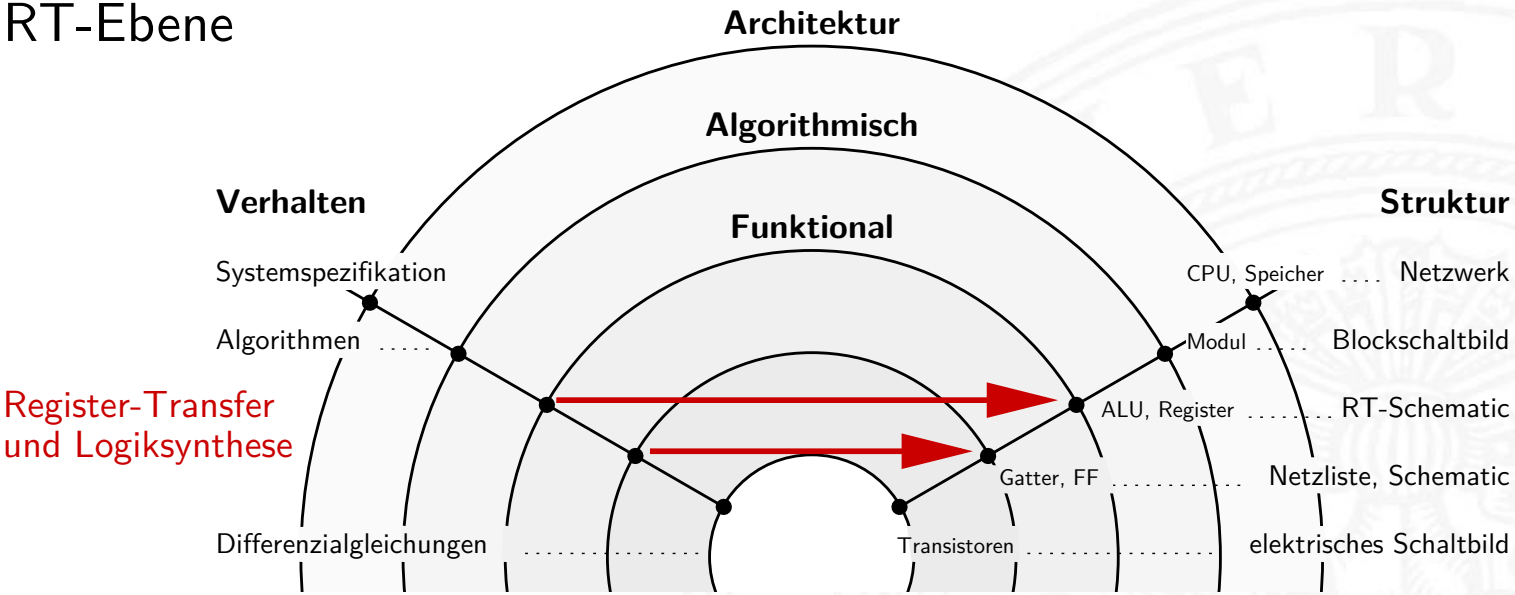
- Entwurf auf höheren Abstraktionsebenen
- Automatische Transformationen bis zum Layout
  - Synthese: Register-Transfer, High-Level
  - Datenpfad-/Makrozellgenerierung
  - Zellsynthese
  - Platzierung & Verdrahtung

Entwurfswerkzeuge



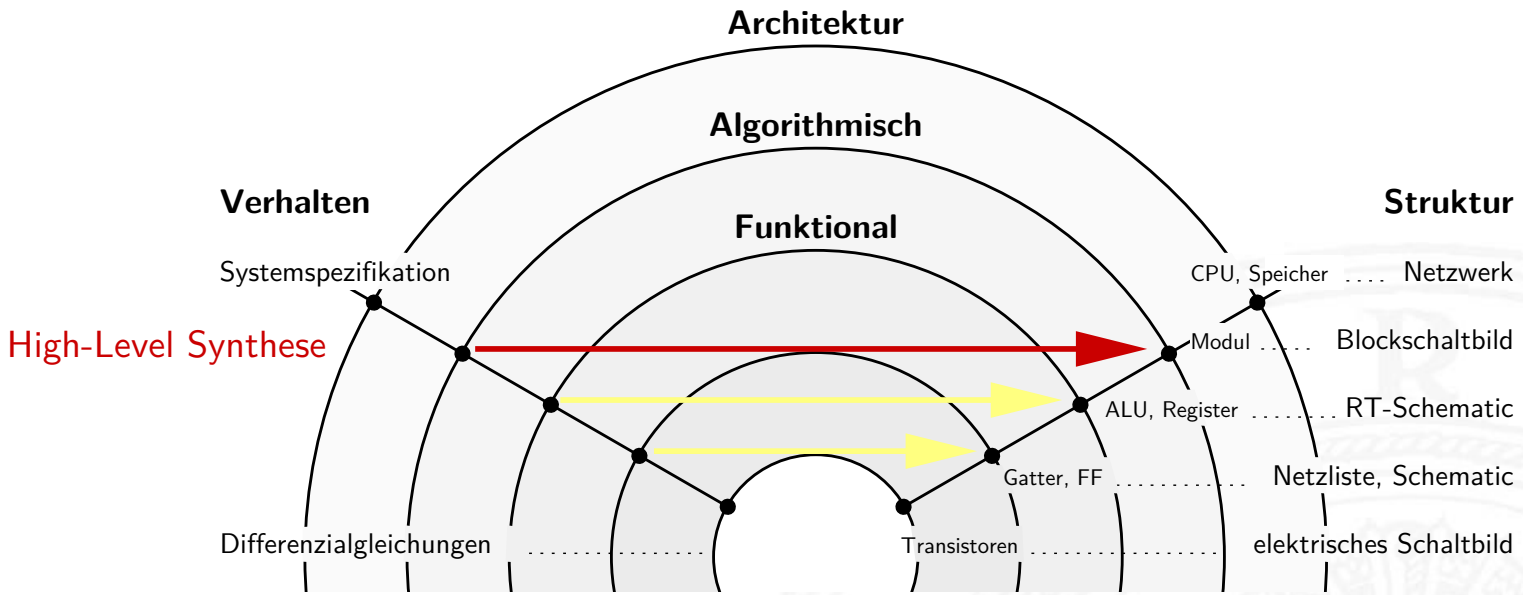
Entwurfswerkzeuge (cont.)

- Synthese
  - = automatische Generierung von Strukturbeschreibungen aus Verhaltensmodellen
  - Trend: IP-Komponenten (**I**ntellectual **P**roperty) und „behavioral Code“
  - RT-Ebene



Entwurfswerkzeuge (cont.)

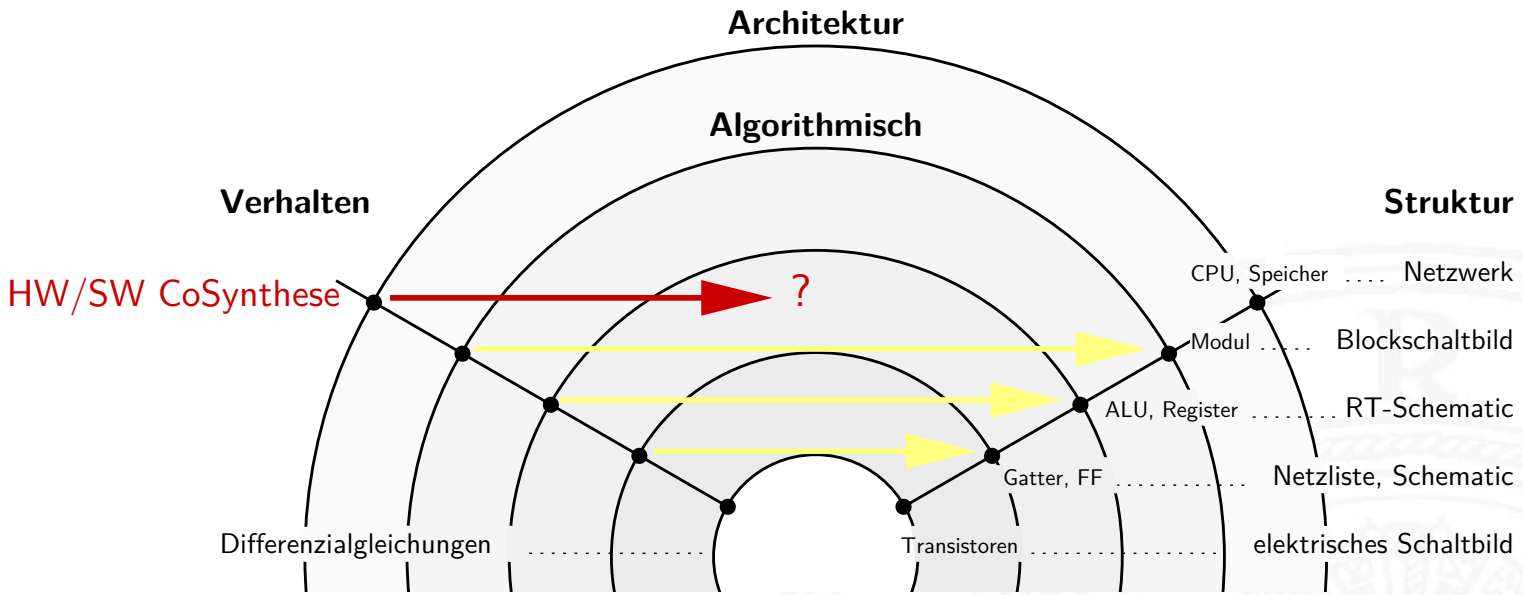
- High-Level Synthese



- Einschränkung des „Suchraums“
- spezielle Zielarchitekturen
- spezielle Anwendungsfelder
- Datenflussdominiert DSPs
  Kontrollflussdominiert Prozessoren

Entwurfswerkzeuge (cont.)

- CoDesign → CoSynthese



- Partitionierung Hardware / Software ?
- nur manuell möglich

Entwurfswerkzeuge (cont.)

- Simulation
  - Trend: wachsender Aufwand, Systemsimulation
  - Problem der Simulationsauswertung $\Rightarrow$ auch dort Abstraktion
    - Programmiersprachen-Schnittstellen (VHPI, Verilog-PLI ...)
      Beispiele: ▸ Signalverarbeitung ▸ Bildverarbeitung
  - Hardwarebeschleunigung
    - Emulation von Gatternetzlisten durch FPGA-Boards
    - Beispiel: Betriebssystem auf Simulationsmodell vom Mikroprozessor booten (Sun Microsystems)
  - gemischte Simulation
    - Hardware- und Software
    - auf verschiedenen Abstraktionsebenen
    - + IP-Modelle
    - + analoge Modelle
- Analysewerkzeuge
  - Leistungsverbrauch
  - Timing
  - jeweils: statisch, geschätzt oder in Verbindung mit Simulation

Entwurfswerkzeuge (cont.)

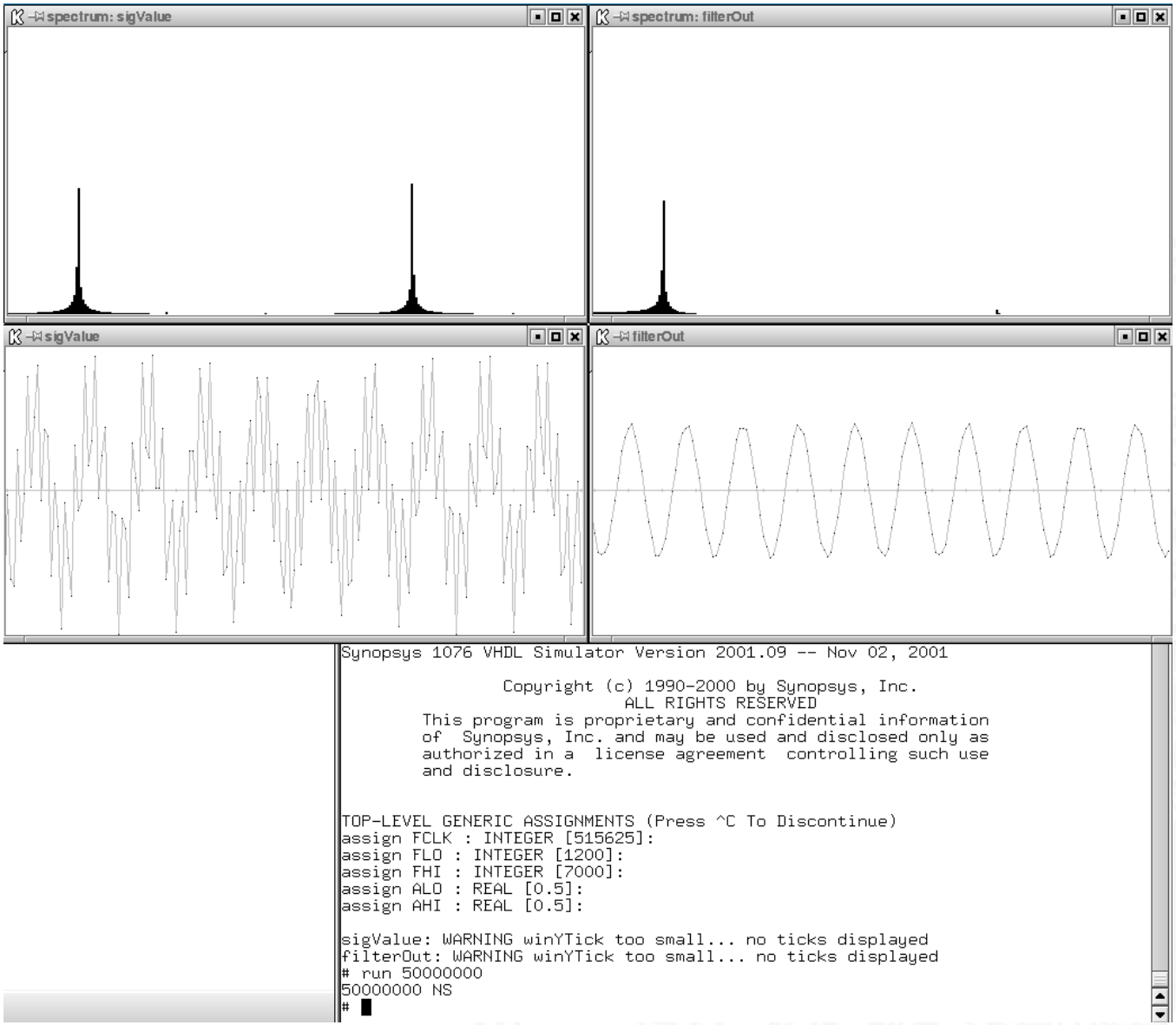
- Verifikation, wenn möglich
  - = Verifikation: Aussagen gelten *für alle möglichen* Eingaben
    Simulation: Beschränkung auf Stimuli
  - formale Methoden, um Eigenschaften zu überprüfen
  - meist Vergleich verschiedener Modelle
    - in Verbindung mit Extraktion
    - Referenzmodell, woher?
  - Ersatz von Simulationen
- Layoutwerkzeuge / Platzierung & Verdrahtung
  - NP-vollständige Probleme
  - $\Rightarrow$ Heuristiken
  - $\Rightarrow$ sehr starke Spezialisierung, z.B. Routing bei Standardzell Entwürfen:
    1. Verdrahtung der Spannungsversorgung: Power-Routing
    2. Clock-Tree Synthese / -Routing
    3. zeitkritische Netze bearbeiten: „constraint driven“ Routing
    4. normale Verdrahtung
    5. nachträgliche Optimierung: DRC-Fehler, thermische Modelle ...

Entwurfswerkzeuge (cont.)

- Test des Entwurfs
  - = Testbarkeit: Fertigungsfehler (physikalisch) feststellen
    Simulation: Überprüfung der Funktion
  - Ziel: defekte ICs aussortieren, *vor* Verpackung in Gehäuse
  - Problem
    - *alle internen Leitungen/Gatter ansprechen*
    - *nur die Padzellen sind direkt zugänglich*
  - Fehlermodelle: „stuck-at“, bridging, open ...
  - Verfahren um Testbarkeit zu gewährleisten
    - Selbsttest, z.B. BIST (**B**uild **I**n **S**elf **T**est)
    - Scan-Path: Flipflops als Schieberegister
    - ...
    - Dabei wird zusätzliche Logik integriert (bis zu 30%)
    - (teil-)automatisch bei der Synthese
  - Fehlersimulation: überprüft die Fehlerüberdeckung
    „Wie viele Fehler können erkannt werden?“
  - Testmustergenerierung: erzeugt automatisch Testvektoren

Simulation: Beispiel

Signalverarbeitung – digitales Filter



◂ Simulation

Simulation: Beispiel

Bildverarbeitung – Segmentierung

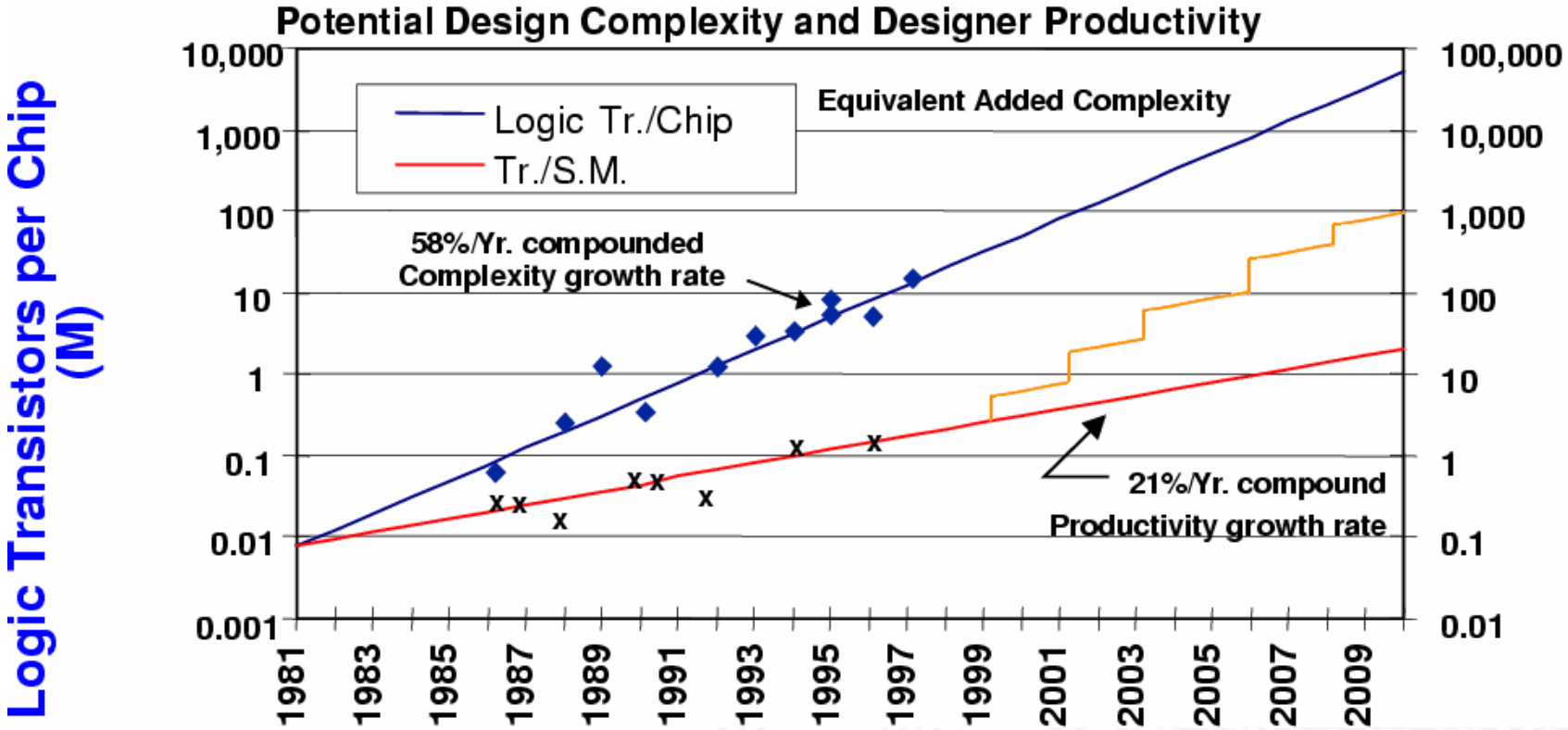

```
</proVhdl%xv
level 3: allocated 256 colors, 1 pixel per color
level 4: allocated 256 colors, 1 pixel per color
level 5: allocated 256 colors, 1 pixel per color
level 6: allocated 256 colors, 1 pixel per color
# run
iter. change:   link   pixel:    1-2     3-5    6-10  11-255
   1   27300   21840    5460      0       0       0    5460
   2   11515    6059    5456   2245    1178     730    1303
   3    6255    2252    4003   2626     752     352     273
   4    3493     956    2537   1966     339     150      82
   5    1908     457    1451   1141     208      72      30
   6    1027     203     824    668     115      24      17
   7     493      75     418    339      55      15       9
   8     226      30     196    170      15       4       7
   9     106      15      91     76      11       2       2
  10      46       6      40     30       6       2       2
  11      18       3      15     11       2       1       1
  12       8       2       6      4       0       2       0
  13       6       0       6      5       1       0       0
  14       0       0       0      0       0       0       0
(vhdlsim): Simulation complete, time is 1490944 NS.
#
```

◂ Simulation

Probleme

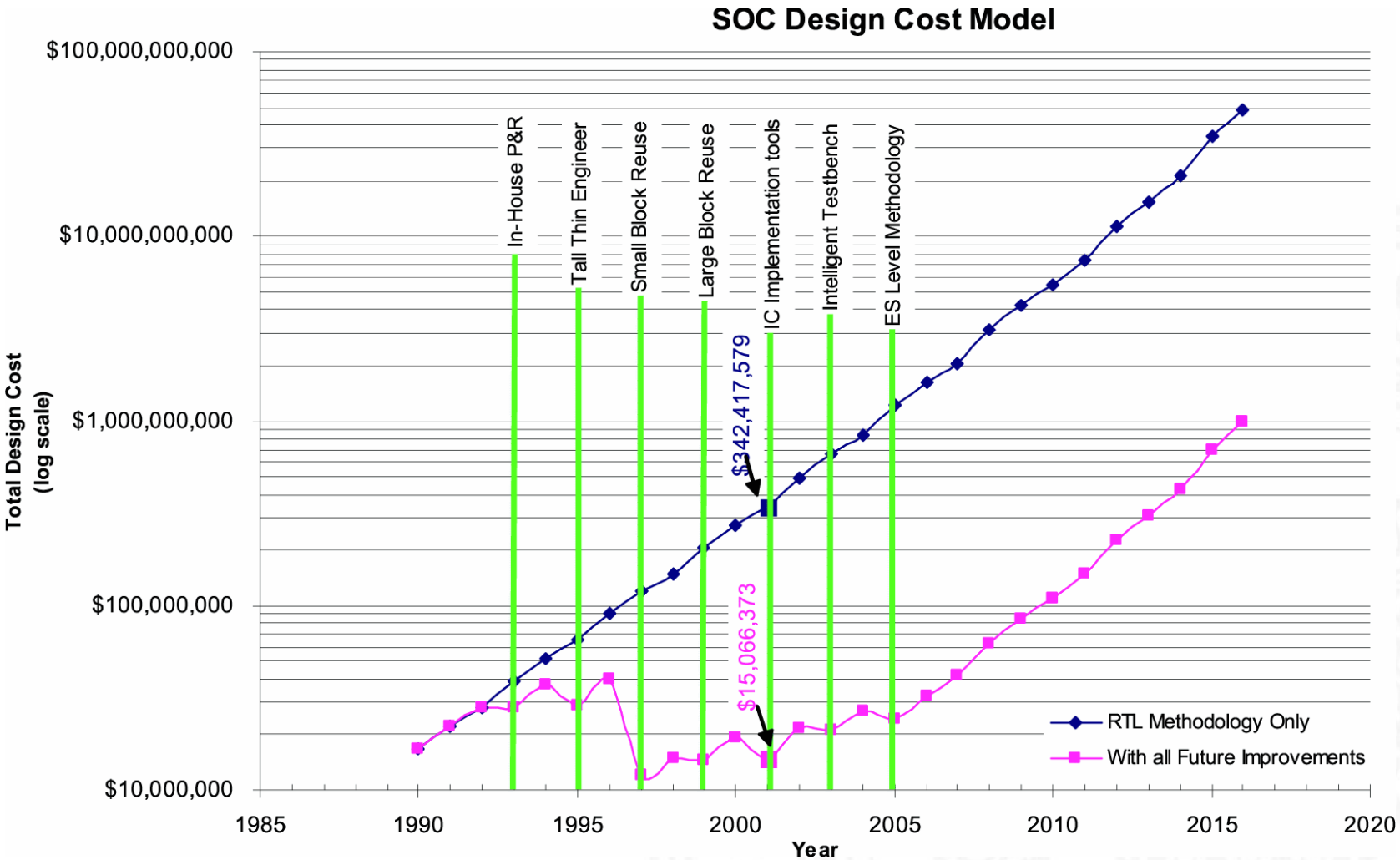
Moore's Law heißt in der Praxis

- Entwurf immer größerer und komplexerer Systeme
- Produktivitätssteigerungen



Probleme (cont.)

– Entwurfskosten



Probleme (cont.)

- Geänderte Systemanforderungen
  - Performanz
  - Größe
  - ökonomische Randbedingungen
  - Low-Power: Leistungsaufnahme, Abwärme ...
  - Umgebung: EMV, Temperatur, mechanische Eigenschaften ...

  – Wie können all diese Anforderungen (formal) spezifiziert werden?

Gliederung

1. Entwurfsmethodik
2. EDA-Werkzeuge
3. Entwurfsstile
   - Full-Custom
   - Makro- und Standardzellentwurf
   - Gate-Array Entwurf
   - programmierbare Logik: PLDs, FPGAs
   - Vergleich

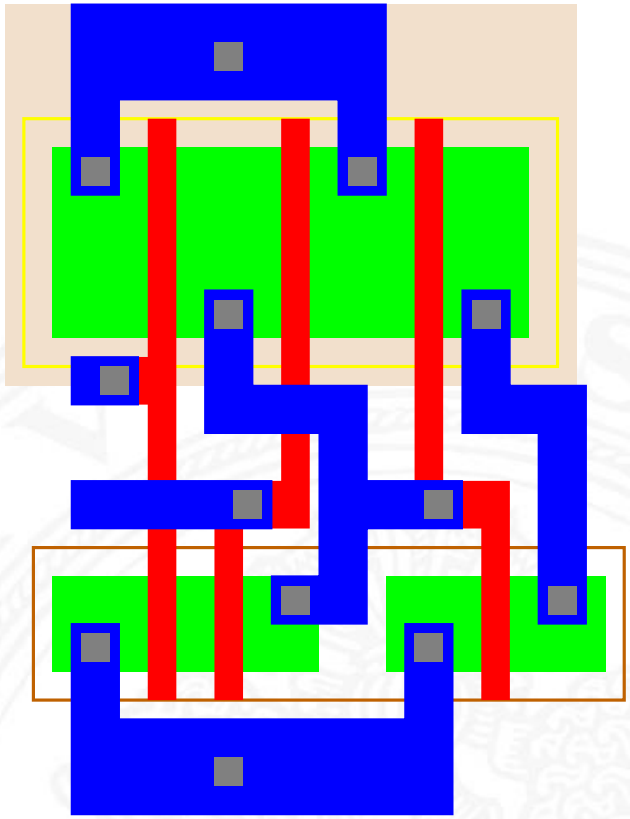
Entwurfsstile

- mehrere Möglichkeiten Schaltungen zu entwerfen
- Unterscheidungsmerkmale
  - Zeitaufwand: Entwurfsdauer, Fertigungszeit
  - Kosten: Fertigung, pro Stück, EDA-Werkzeuge
  - IC-Eigenschaften: Größe, Taktfrequenz, Leistungsaufnahme ...
- Entwurfsstile
  - Full-Custom
  - Standardzell
  - Gate-Array
  - FPGA / programmierbare Schaltungen

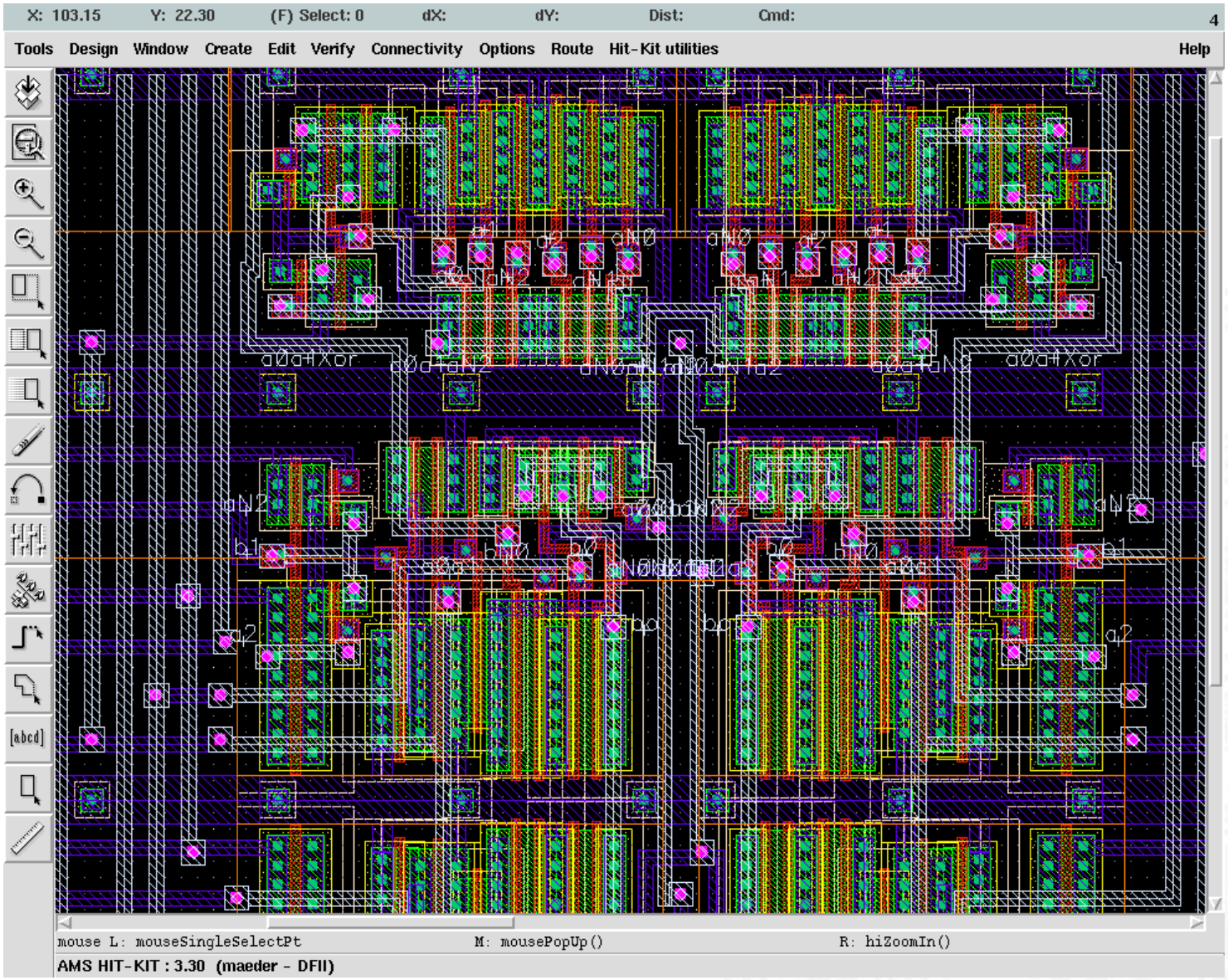
Full-Custom

Vollkundenspezifischer Entwurf / Full-Custom

- Layout aller geometrischer Strukturen
- viel manuelle Arbeit mit Layout-Editoren
- optimal kleine, schnelle Entwürfe
- sehr lange Entwurfsdauer (Effizienz)
- Ausnutzen von Regularität
- Teamarbeit nötig, Schnittstellen
- erfordert erfahrene Entwerfer



Full-Custom (cont.)

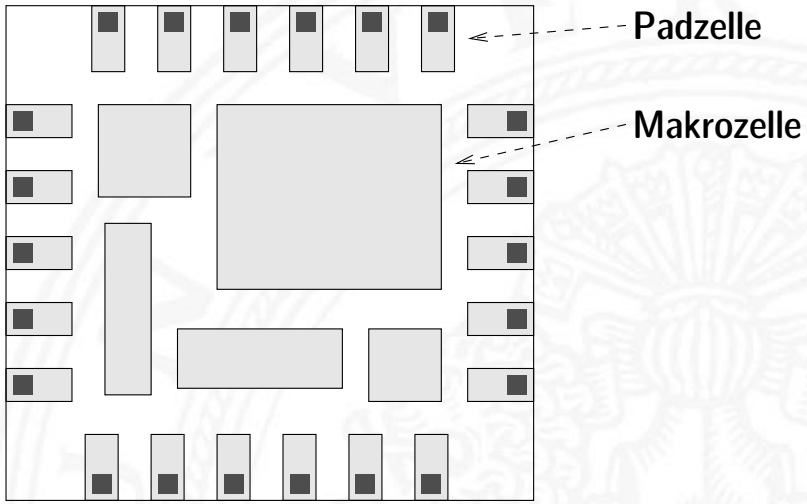


Makrozellentwurf

Makrozellentwurf

- Zellen wie Speicher, ALUs oder Datenpfade werden über Generatoren erzeugt
- Makrozellen in Full-Custom Qualität
- meist in Verbindung mit Standardzellentwurf

Chipgröße	variabel
Zellenanzahl	variabel
Zellengröße	variabel
Anschlusslage	variabel
Leiterbahnkanäle	variabel

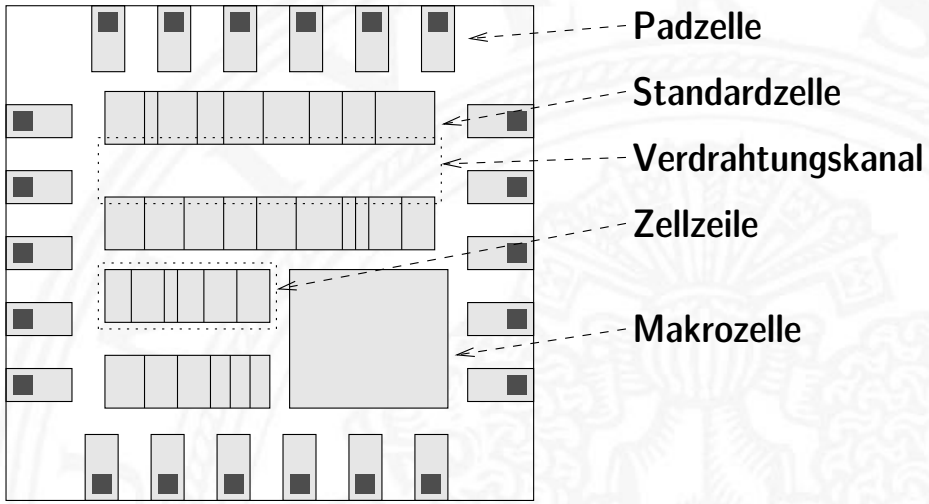


Standardzellentwurf

Standardzellentwurf

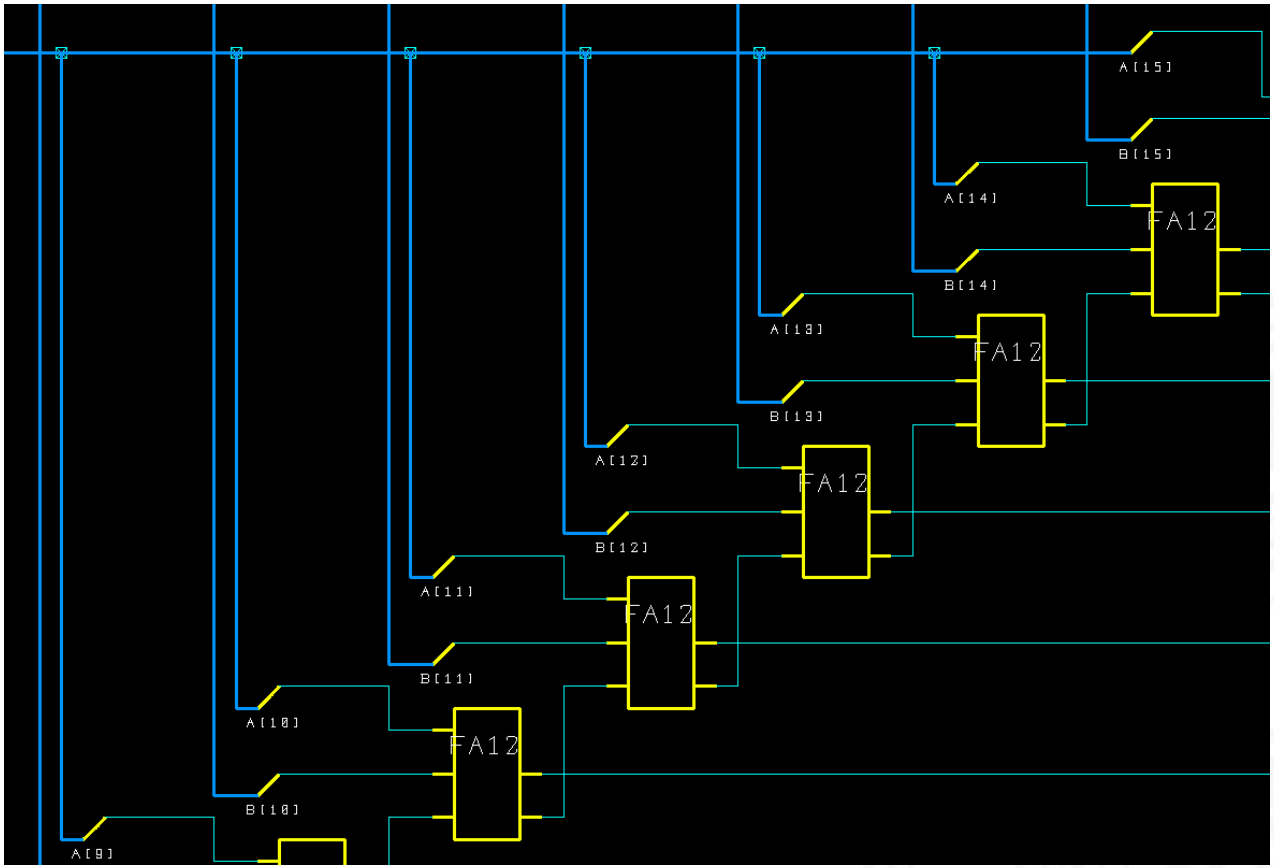
- vorgefertigte Zellen aus Bibliotheken benutzen
- Layout der Standardzellen in Full-Custom Qualität
- schneller flexibler Entwurf
- meist in Verbindung mit Makrozellgeneratoren

Chipgröße	variabel
Zellenanzahl	variabel
Zellenhöhe	fest
Zellenbreite	variabel
Anschlusslage	variabel
Leiterbahnkanäle	variabel

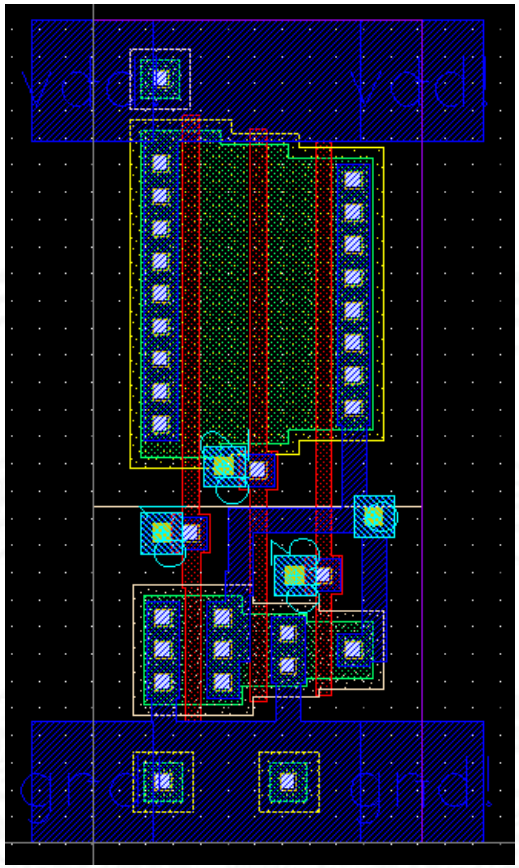


Standardzellentwurf (cont.)

Schematic

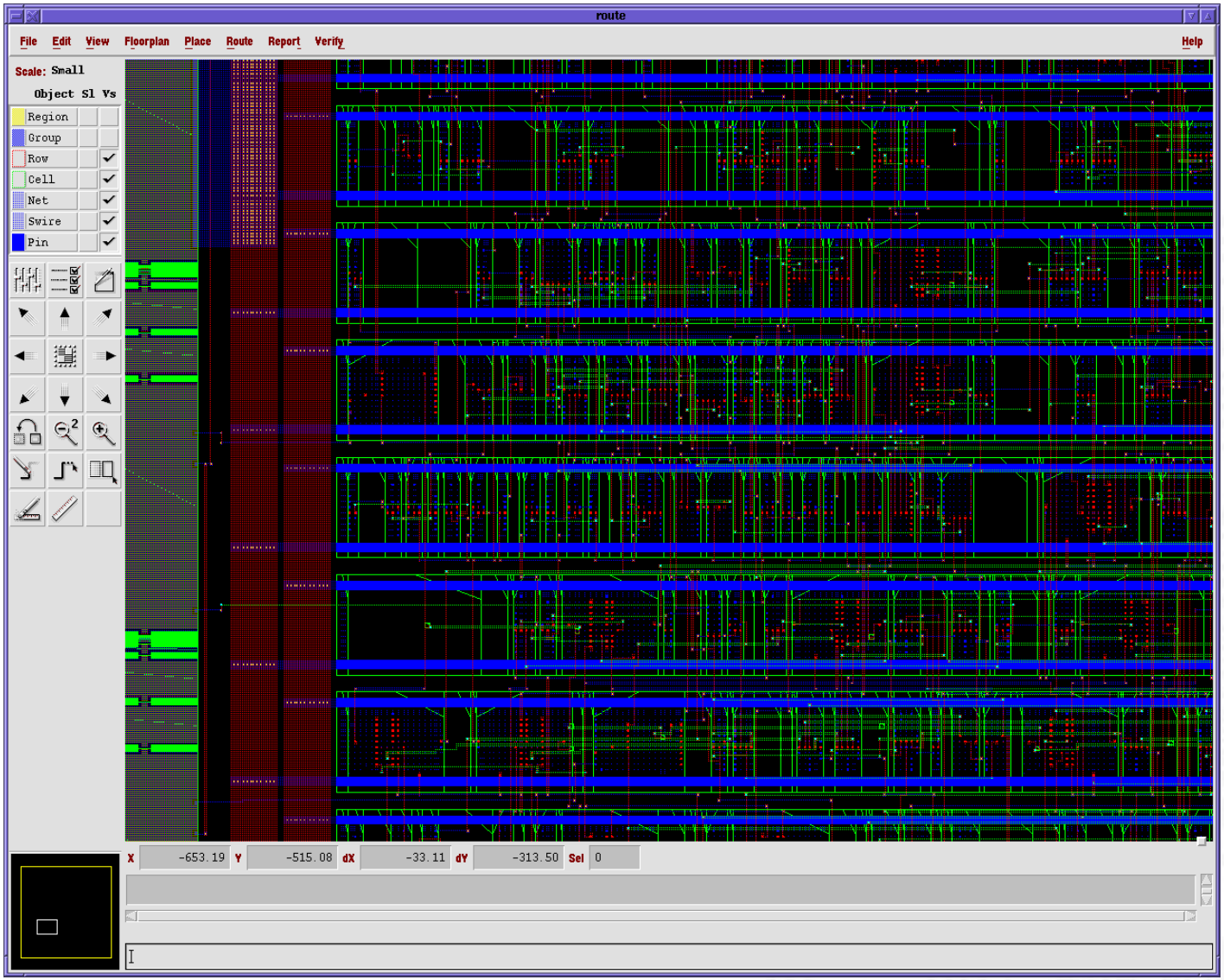


Zell-Layout



Standardzellentwurf (cont.)

Standardzell Layout

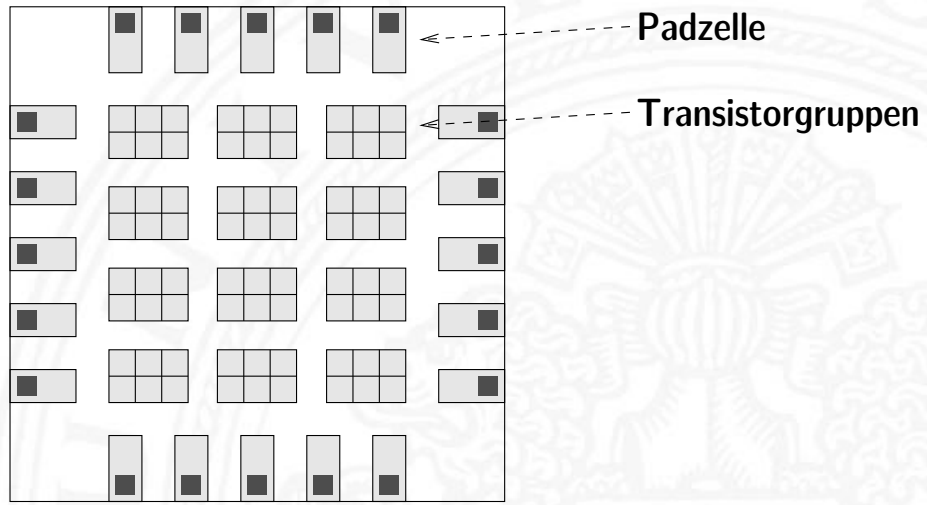


Gate-Array Entwurf

Gate-Array / Sea-of-Gate Entwurf — abgelöst durch FPGA

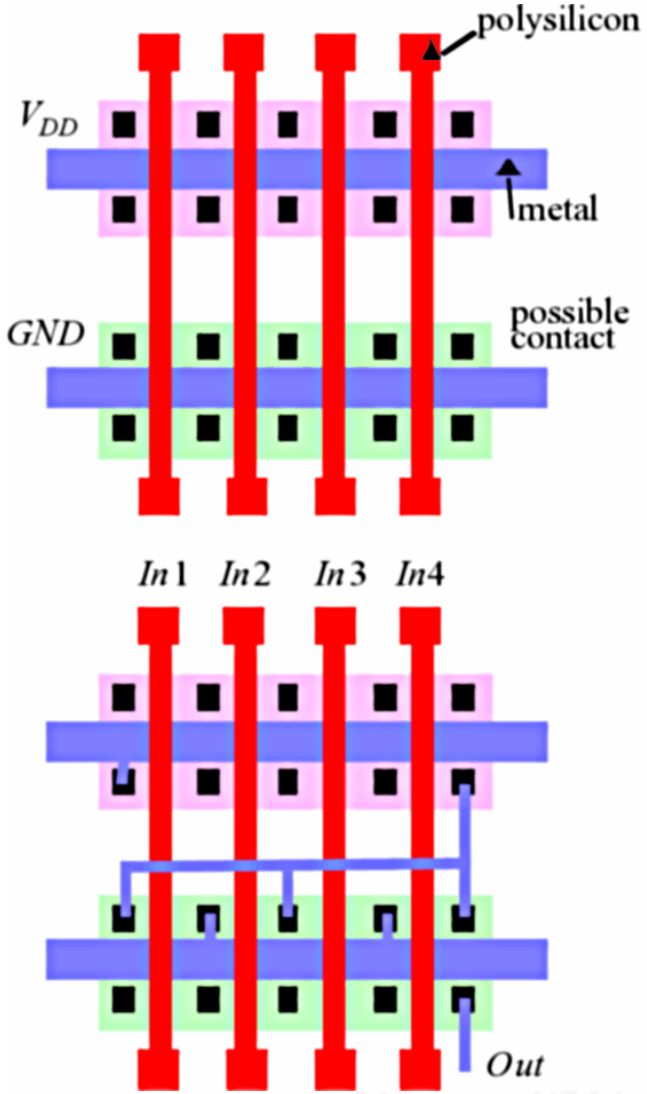
- vorgefertigte Transistoren
- Layout durch Verbindungsstruktur (Verdrahtung, Kontakte)
- intra-Zell Verdrahtung aus Zellbibliotheken
- vorgegebene Master: Komplexität eingeschränkt, Verschnitt
- schnelle Verfügbarkeit

Chipgröße	fest
Zellenanzahl	fest
Zellengröße	fest
Anschlusslage	fest
Leiterbahnkanäle	fest



Gate-Array Entwurf (cont.)

Gate-Array



Uncommited Cell

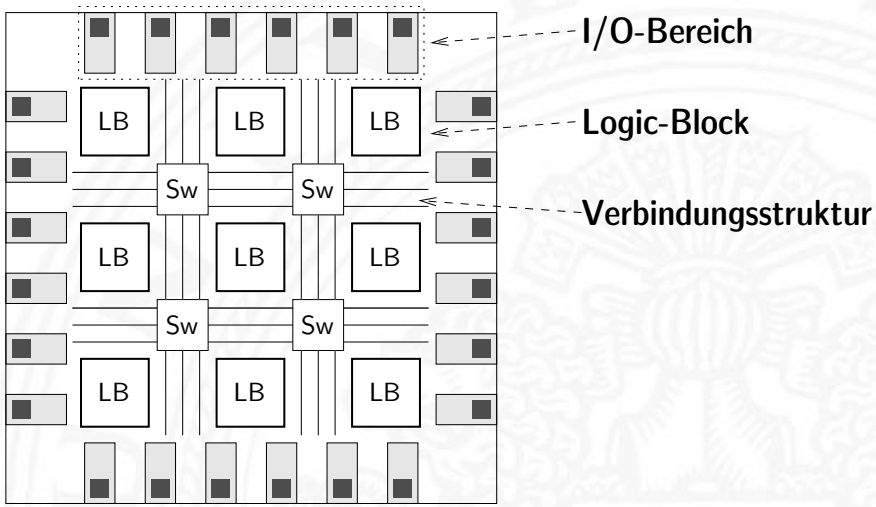
Committed Cell (4-input NOR)

programmierbare Schaltungen

programmierbare Schaltungen: FPGA, PLD, LCA ...

- fertig vorgegebene Schaltung: Logik und Verbindungsstruktur
- Entwurf: Programmierung durch Anwender $\Rightarrow$ sofort verfügbar
- Einschränkung durch vorgegebene Struktur
- Rekonfiguration möglich
- in-Circuit programmierbar

Chipgröße	fest
Blockanzahl	fest
Anschlusslage	fest
Verbindungsnetz	fest
Blockfunktion	progr.
Verbindungen	progr.



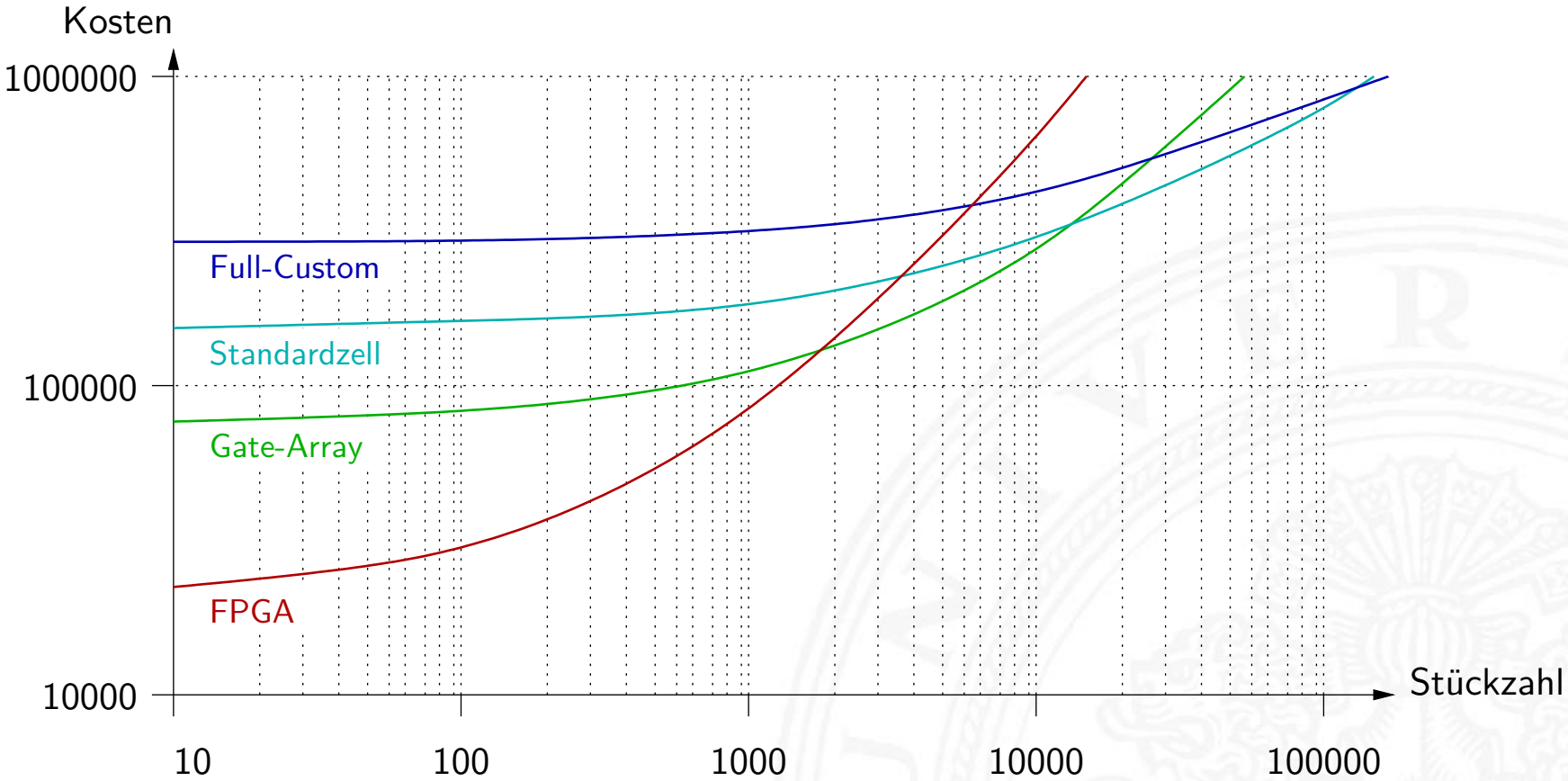
Vergleich der Entwurfsstile

Tabellarische Übersicht

Stil	*Performanz*	*Fläche*	*Kosten (IC)*	*Kosten (Design)*	*time-to-Market*	*Prozessschritte*	*Stückzahlen*
Full-Custom	$+++$	$+++$	$+++$	$---$	$---$	voll	10^5
Standard-/Makrozell	$++$	$++$	$++$	$--$	$--$	voll	10^4
Gate-Array	$+$	$\circ$	$+$	$\circ$	$\circ$	4-10	10^3
programmierbare Logik	$-$	$--$	$--$	$++$	$+++$	0	$< 10^3$

Vergleich der Entwurfsstile (cont.)

Wirtschaftlichkeitsüberlegungen – Tendenz !



Vergleich der Entwurfsstile (cont.)

Faktoren bei der Auswahl

- Kostenüberlegungen
- Entwurfsdauer: „time-to-Market“
- technische Randbedingungen, oft als K.O.-Kriterium
  - Fläche
  - Leistungsaufnahme
  - Sicherheitsaspekte
- organisatorische Randbedingungen
  - vorhandene Werkzeuge
  - Know-How
  - „Faktor: Mensch“ (Erfahrungen, Vorlieben)

$\Rightarrow$ vielfältige Wechselwirkungen

Literaturliste

[ITRS07] *International Technology Roadmap for Semiconductors – 2007 Edition.*
Semiconductor Industry Association.

[ITRS15] *International Technology Roadmap for Semiconductors – ITRS Reports.*
Semiconductor Industry Association, 2015.
URL `www.itrs2.net/itrs-reports.html`

[IRDS20] *International Roadmap for Devices and Systems (IRDS) – 2020 Edition.*
IEEE International Roadmap for Devices and Systems, 2020.
URL `irds.ieee.org/editions/2020`

Literaturliste (cont.)

[BE95] Abdellatif Bellaouar, Mohamed I. Elmasry:
Low-power digital VLSI design: circuits and systems.
Kluwer Academic Publishers; Boston, MA, 1995.
ISBN 0–7923–9587–5

[MC80] Carver Mead, Lynn Conway:
Introduction to VLSI systems.
Addison-Wesley; Reading, MA, 1980.
ISBN 0–201–04358–0

[She95] Naveed A. Sherwani:
Algorithms for VLSI physical design automation.
Kluwer Academic Publishers; Boston, MA, 1995.
ISBN 0–7923–9592–1

Literaturliste (cont.)

[TA14] Andrew S. Tanenbaum, Todd Austin:
Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner.
Pearson Deutschland GmbH, Hallbermoos, 2014.
ISBN 978–3–86894–238–5

[T+90] Donald E. Thomas [u. a.]:
Algorithmic and register-transfer level synthesis: the system architect's workbench.
Kluwer Academic Publishers; Boston, MA, 1990.
ISBN 0–7923–9053–9

[WE94] Neil H. E. Weste, Kamran Eshraghian:
Principles of CMOS VLSI design: a systems perspective.
Addison-Wesley; Reading, MA, 1994.
ISBN 0–201–53376–6