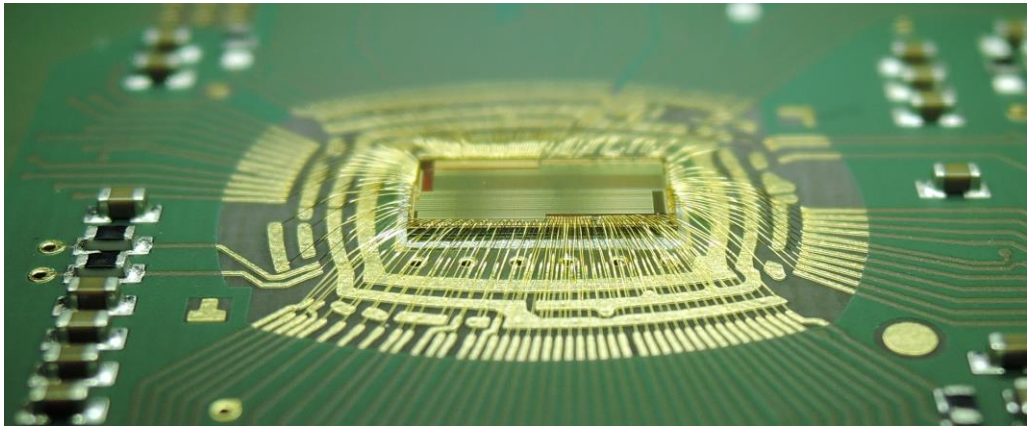
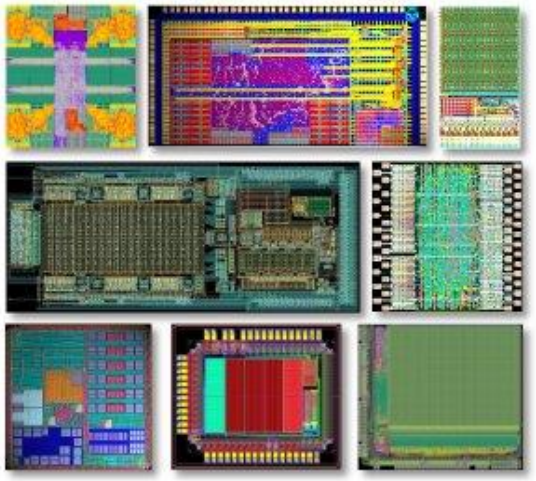
Schaltkreis- und Systementwurf



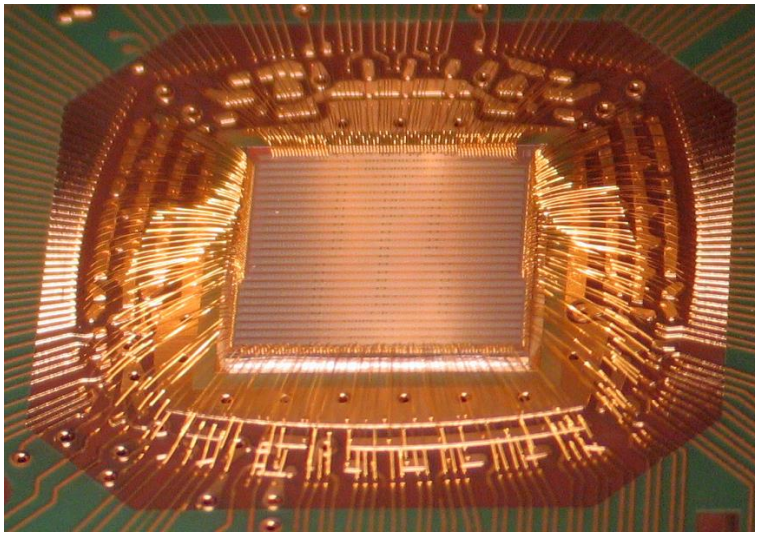
FAKULTÄT **ELEKTROTECHNIK** UND **INFORMATIONSTECHNIK**



- Dr.-Ing. Sebastian Höppner
- Raum TOE216
- sebastian.hoeppner@tu-dresden.de
- https://tu-dresden.de/ing/elektrotechnik/iee/hpsn

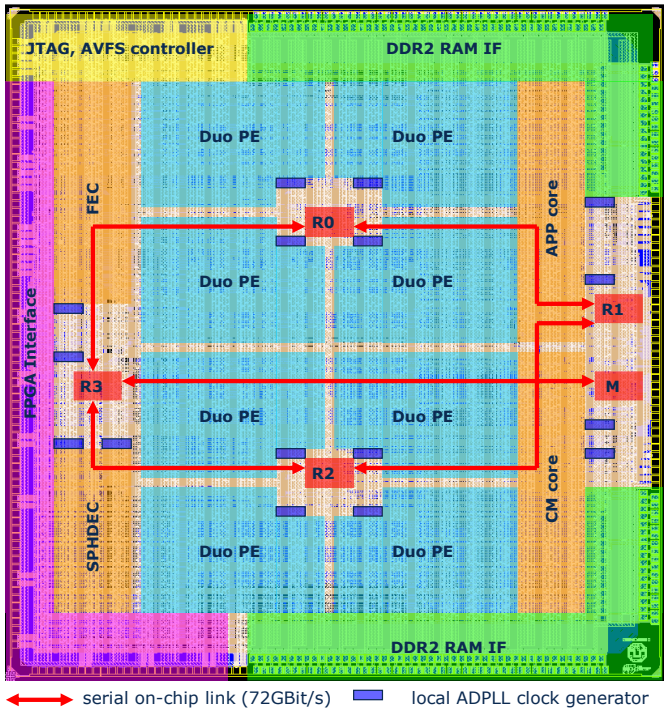
- Vorlesungsfolien unter:
  - https://tu-dresden.de/ing/elektrotechnik/iee/hpsn/studium/materialien

- Entwurf von integrierten Schaltkreisen („Chips“) als Kernbestandteil moderner Elektronikprodukte
- Realisierung komplexer Schaltungen auf einem Chip ermöglicht:
  - Hohe Funktionalität
  - Hohe Integrationsdichte
  - Geringe Verlustleistung
  - Hohe Zuverlässigkeit
- Die Verkleinerung der Fertigungstechnologien ermöglicht die Integration immer komplexerer Systeme
- Herausforderung bei Entwurf und Verifikation



Komplexes Multiprozessor-System-on-Chip

- **„Tomahawk2“** Software-Defined-Radio Basisbandprozessor
  - 36 mm² Chipfläche in 65nm CMOS Technologie
  - 465 Pads
  - 10,2 Millionen Gatter
  - 750kByte SRAM
- 20 Prozessorkerne
- Adaptives und dynamisches Power-Management
- 17 Taktgeneratoren
- Network-on-Chip mit seriellen on-chip links mit bis zu 72GBit/s Datenrate
- DDR2 Speicherinterface
- FPGA Interface mit 10GBit/s

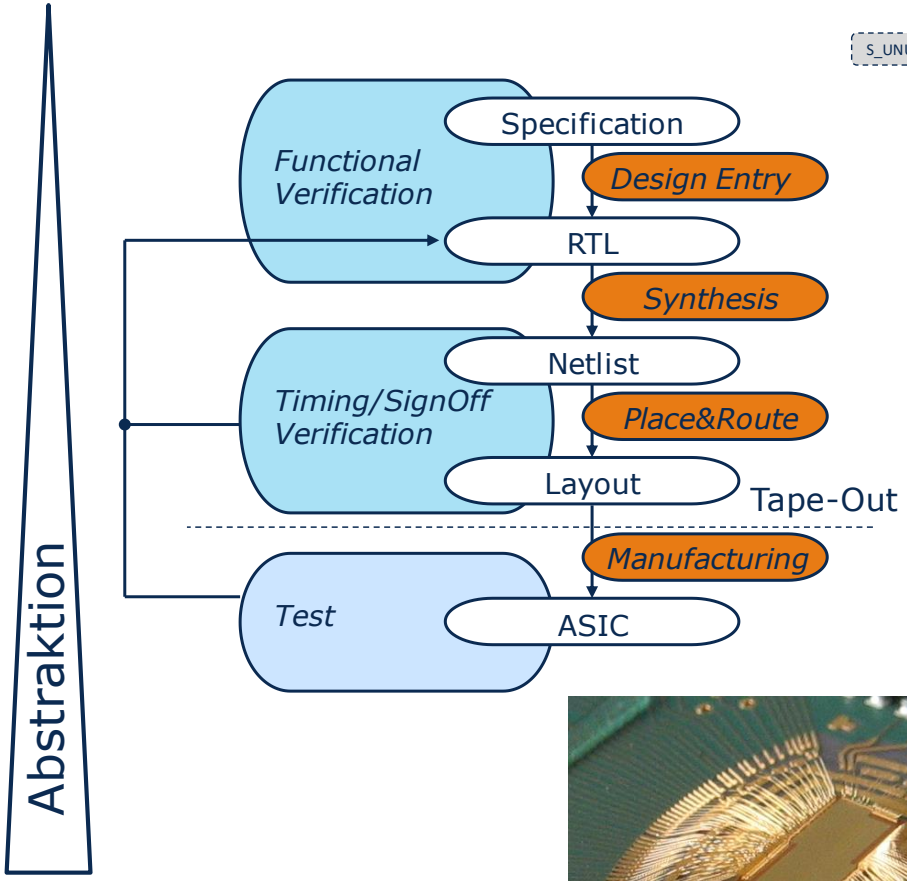


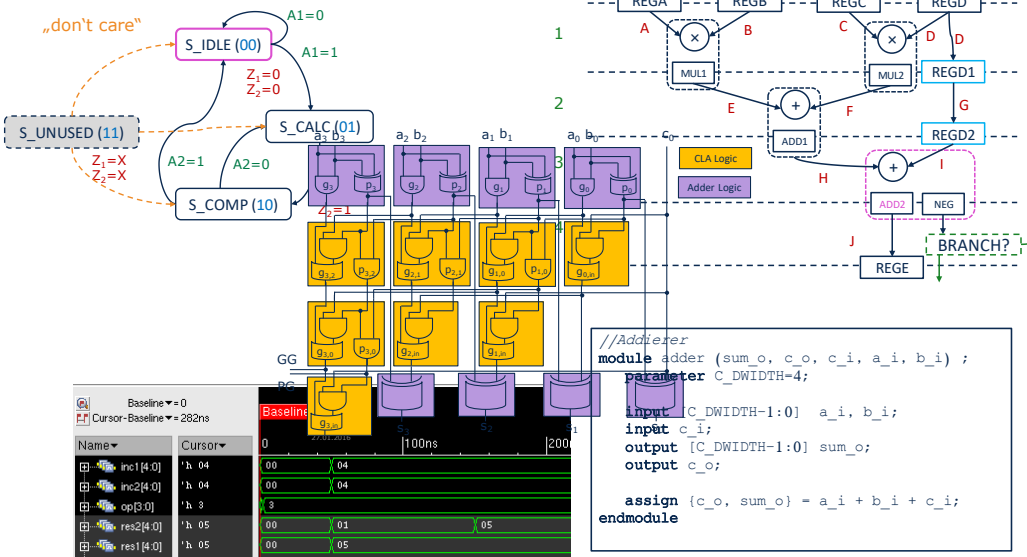
Entwickelt durch die Professuren HPSN, und Vodafone Chair (Prof. Fettweis) im CoolBaseStations Projekt

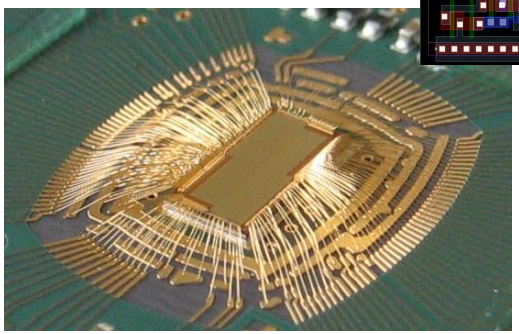
Inhalt der Lehrveranstaltung

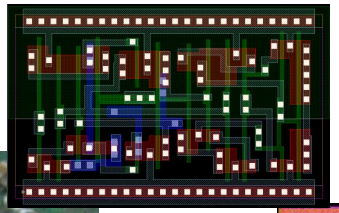
- Entwurfsablauf integrierter digitaler Schaltungen
- Realisierung von Algorithmen in Hardware
- Hardwarebeschreibungssprache Verilog
- Schaltungssimulation und Verifikation
- Digitale Schaltungen in CMOS Technologie

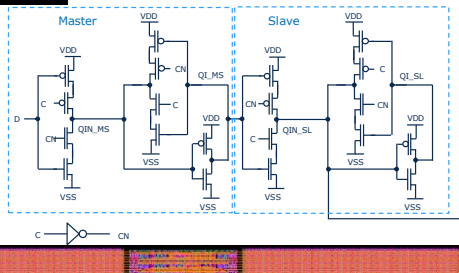
Inhalt der Lehrveranstaltung

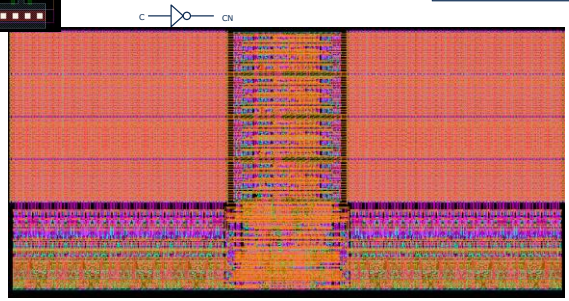












Ablauf

- Wintersemester: 2/1/0, Sommersemester: 0/0/2
- Wintersemester:
  - Vorlesung: Do. 2.DS, Ort: BAR/205/H
  - Übung: Mi. 4.DS, Ort: GÖR/226/H
    - Konsultation zur Belegarbeit
  - Praktikum: Fr. 4.DS, Ort: TOE/201
    - Infos und Termine siehe Webseite
    - Einführungspraktikum und betreutes Praktikum zur individuellen Arbeit am Beleg
- Sommersemester:
  - Praktikum: Zeit: Fr. 4.DS, Mi 4.DS, Ort: TOE/201
    - Betreutes Praktikum zur individuellen Arbeit am Beleg

Einführung ins Praktikum und Belegarbeit:
Mittwoch: 23.10.2019, 4.DS, Ort: GÖR/226/H

Belegarbeit

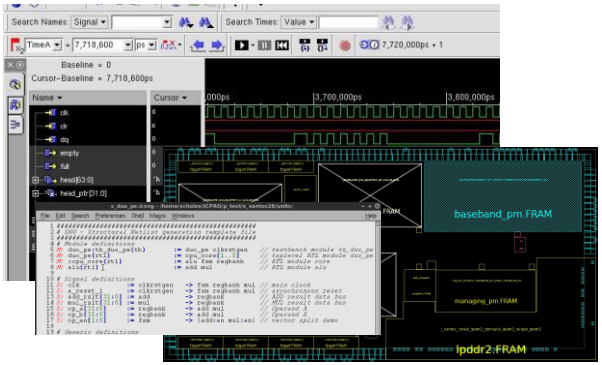
- Realisierung eines selbst gewählten Algorithmus in Hardware
  - Referenzimplementierung in Software
  - Architekturentwicklung (Scheduling, Datenpfad, Ablaufsteuerung)
  - Implementierung der Schaltung mittels Datenpfadbaublöcken
  - Verifikation durch Schaltungssimulation
  - Synthese und Place&Route der Schaltung
- Schriftlicher Beleg (Details in der Übung!)
- Elektronische Abgabe (.pdf)
- Abgabe:
  - Prüfungseinschreibung in dem Semester in dem Abgabe erfolgt!
  - **Abgabe bis 15.09. (SS) bzw. 16.03. (WS)**

Lernziele

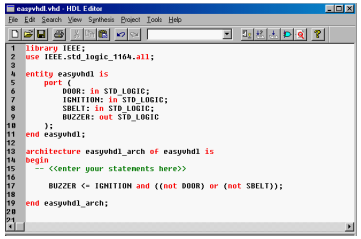
- Diese Lehrveranstaltung wird
  - den Design Flow für digitale integrierte Schaltungen vorstellen
  - Methoden zur Implementierung von Algorithmen in Hardware vorstellen
  - Grundlagen und Konzepte der Verilog HDL vorstellen
  - Verifikationsstrategien vermitteln
  - Vertiefende Grundlagen zur Realisierung digitaler Schaltungen in CMOS Technologien vermitteln
  - Praktische Erfahrung in der Implementierung einer digitalen Schaltung vermitteln
  - Anregung zum Selbststudium geben

Weitere Lehrveranstaltungen der Professur

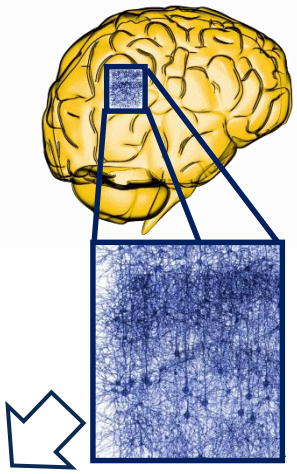
- **VLSI Prozessorentwurf**
  - RTL Entwurf eines Prozessors
  - Simulation und Verifikation
  - Top-Down Design RTL2GDS Flow (Synthese, Timing Analyse, Place&Route, Power Analyse, Sign-off Analyse)
  - Praktikum: Implementierung eines Prozessors



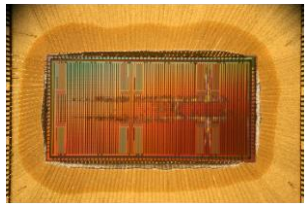
- **VHDL-Entwurf (Oberseminar Informationstechnik)**
  - Einführung in Entwurf, Modellierung, Verifikation mittels VHDL
  - Vorstellung von Aufgaben und Lösungsansätzen aus aktuellen Forschungsarbeiten des Lehrstuhls
  - Referate der Teilnehmer zu ihren Beleg-Projekten



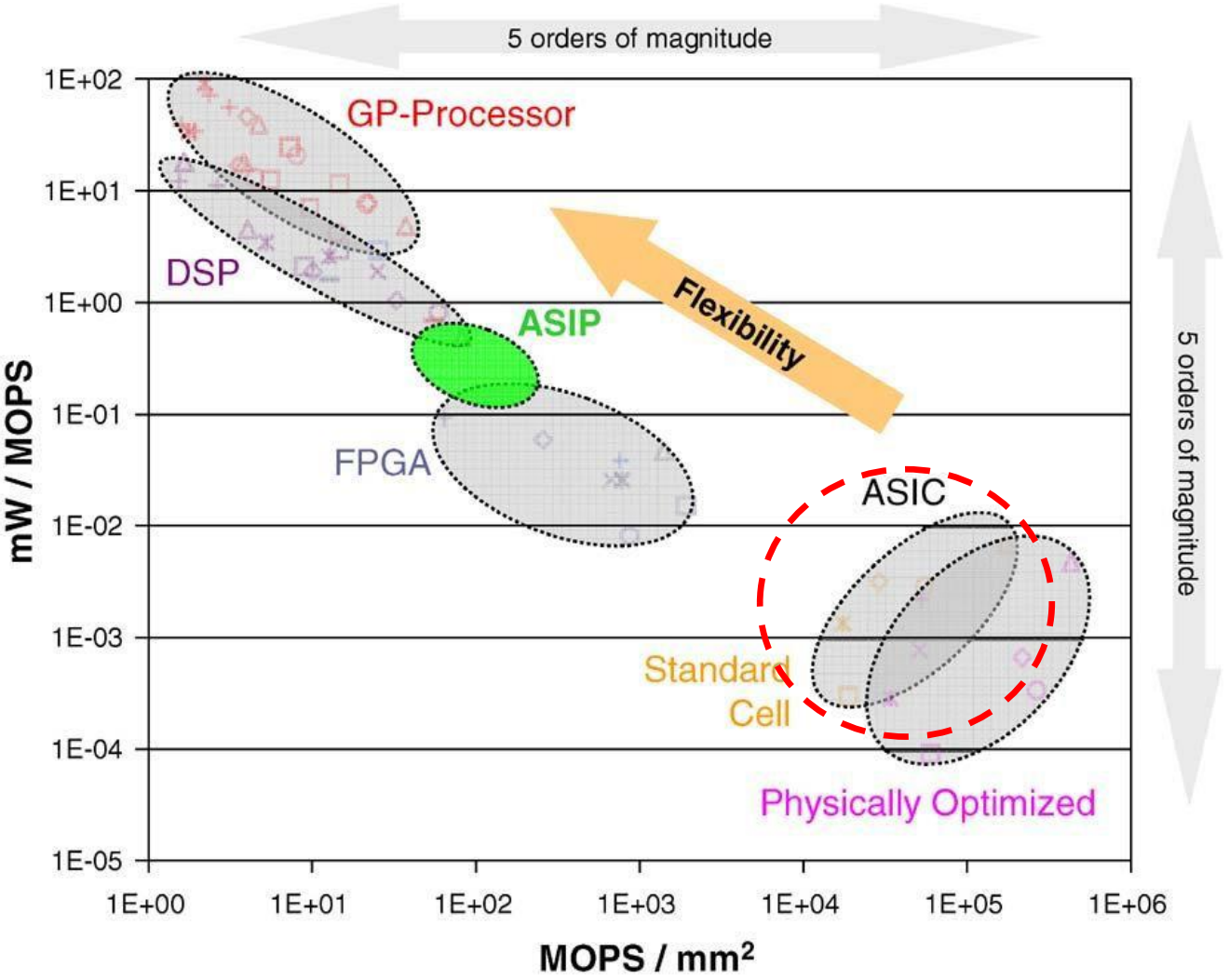
```
library IEEE;
use IEEE.std_logic_1164.all;

entity easyvhdl is
    port (
        DOOR: in STD_LOGIC;
        IGNITION: in STD_LOGIC;
        SBELT: in STD_LOGIC;
        BUZZER: out STD_LOGIC
    );
end easyvhdl;

architecture easyvhdl_arch of easyvhdl is
begin
    -- <<Enter your statements here>>

    BUZZER <= IGNITION and ((not DOOR) or (not SBELT));

end easyvhdl_arch;
```



- **Neuromorphe VLSI-Systeme**
  - Entwurfsmethoden für integrierte analoge CMOS-Schaltungen und deren Schaltungsdimensionierung
  - Neuromorphe VLSI-Systeme und deren neurobiologische Grundlagen, Anwendungen z.B. in Brain-Machine-Interfaces
  - Praktikum zur Erstellung und Analyse von analogen und neuromorphen CMOS-Schaltungen mit der Entwurfssoftware Cadence DF2

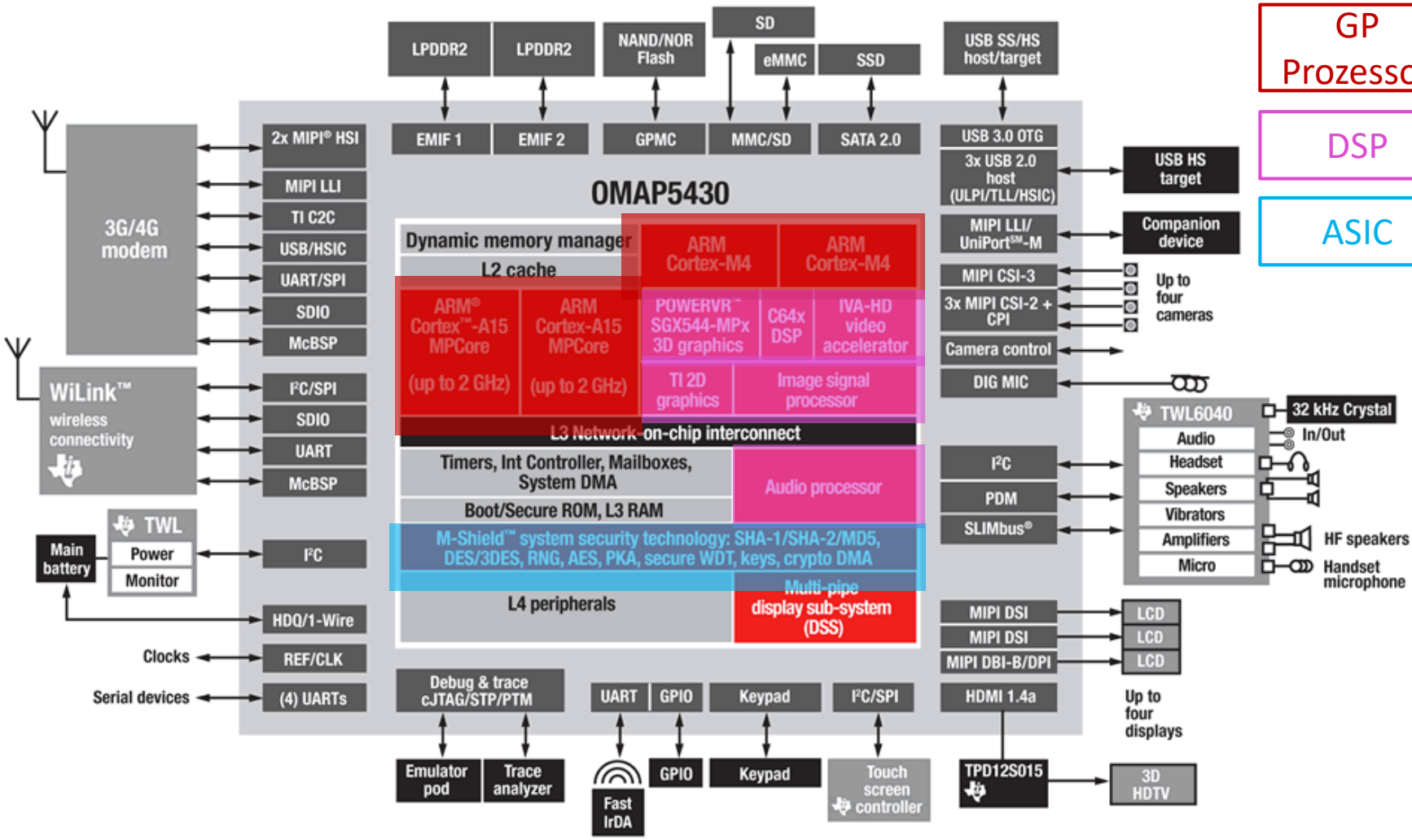


Anwendungsspezifische Integrierte Schaltungen (ASICs)



Quelle: Tobias G. Noll, Thorsten von Sydow, Bernd Neumann, Jochen Schleifer, Thomas Coenen, and Gotz Kappen "Reconfigurable Components for Application-Specific Processor Architecture" in Dynamically ReconfigurableSystems, Springer, 2010

Beispiel System-on-Chip: TI OMAP5430

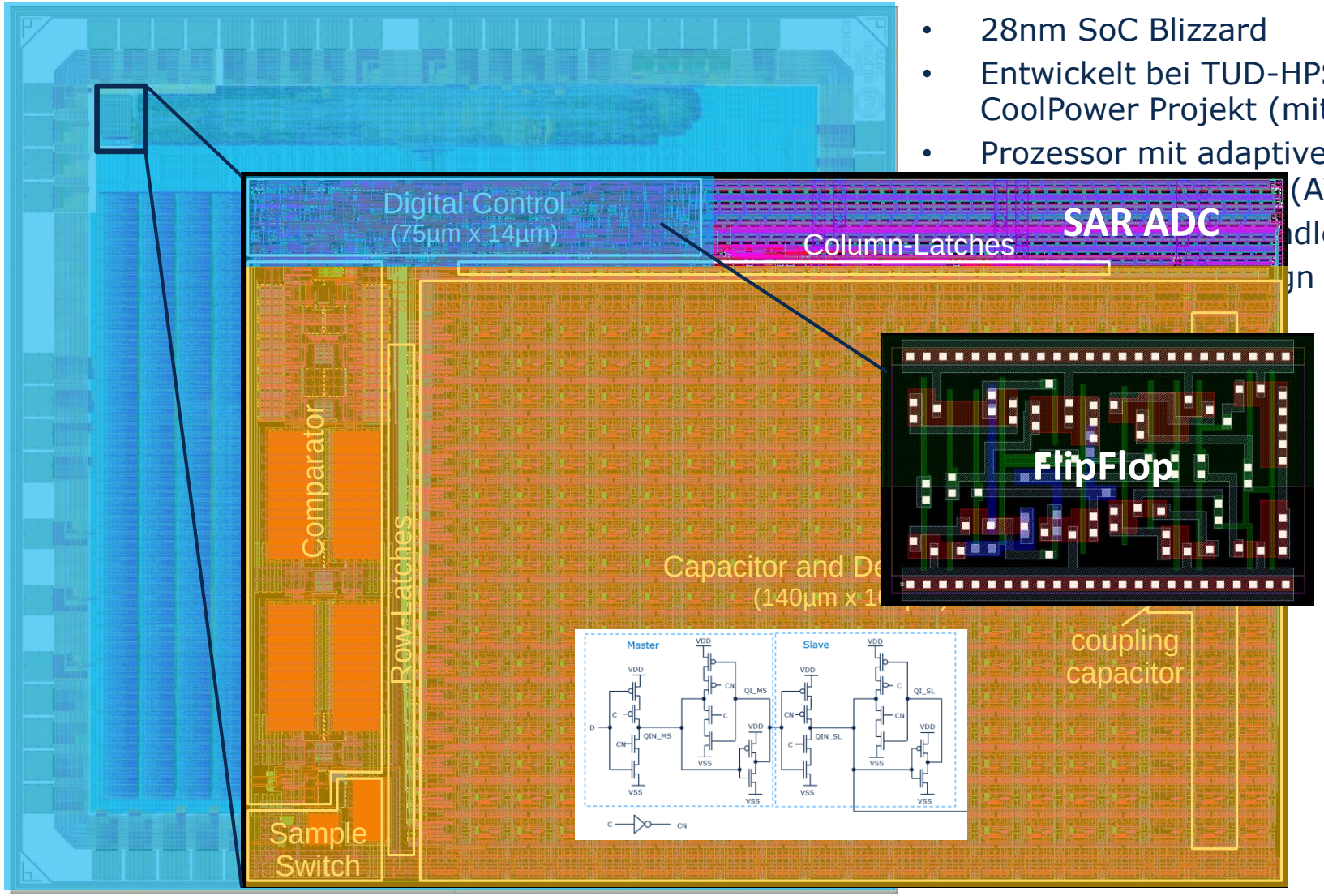


Quelle: anandtech.com

Entwurfsstile

- **Full-Custom Design**
  - Manuelle Eingabe von Schaltplan (schematic) und Layout
  - Vorteile:
    - Detaillierte Optimierungen möglich
  - Nachteile:
    - Geringe Produktivität
  - Anwendungen:
    - Analoge und Mixed-Signal Schaltungen
    - Grundelemente digitaler Schaltungen (Standardzellen, I/O Zellen, Speicher)
    - Speziell optimierte digitale Schaltungen (z.B. High-speed, Ultra-Low-Power)
- **Semi-Custom Design**
  - Manuelle Eingabe einer Hardwarebeschreibung (HDL)
  - (Teil-) automatisierte Erzeugung der Entwurfsdaten (Netzliste, Layout)
  - Vorteile:
    - Sehr hohe Produktivität
  - Nachteile:
    - Eingeschränkte Optimierung von Schaltungseigenschaften
  - Anwendungen:
    - Komplexe Digitale Schaltungen
    - Systems-on-Chip

Beispiel: Entwurfsstile (System-on-Chip Blizzard)



- 28nm SoC Blizzard
- Entwickelt bei TUD-HPSN im CoolPower Projekt (mit ZMDI)
- Prozessor mit adaptivem (AVFS)
  ndler
  gn

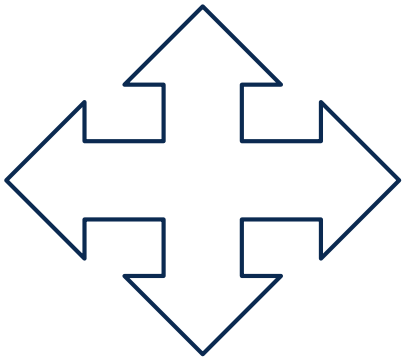
Kommerzielle Aspekte des ASIC und SoC Designs

Markt

- Anforderungen (Spezifikation)
- Kunden
- Preis des Chips
- Entwicklungszeit (Time-to-Market)

Mehrwert

- Neue Funktionalität (z.B. neuer Funkstandard)
- Bessere Funktionalität (z.B. längere Akkulaufzeit)
- Kleinerer Form-Faktor (Gehäuse Größe)
- Reduzierte Anzahl der Chips im System
- Zuverlässigkeit



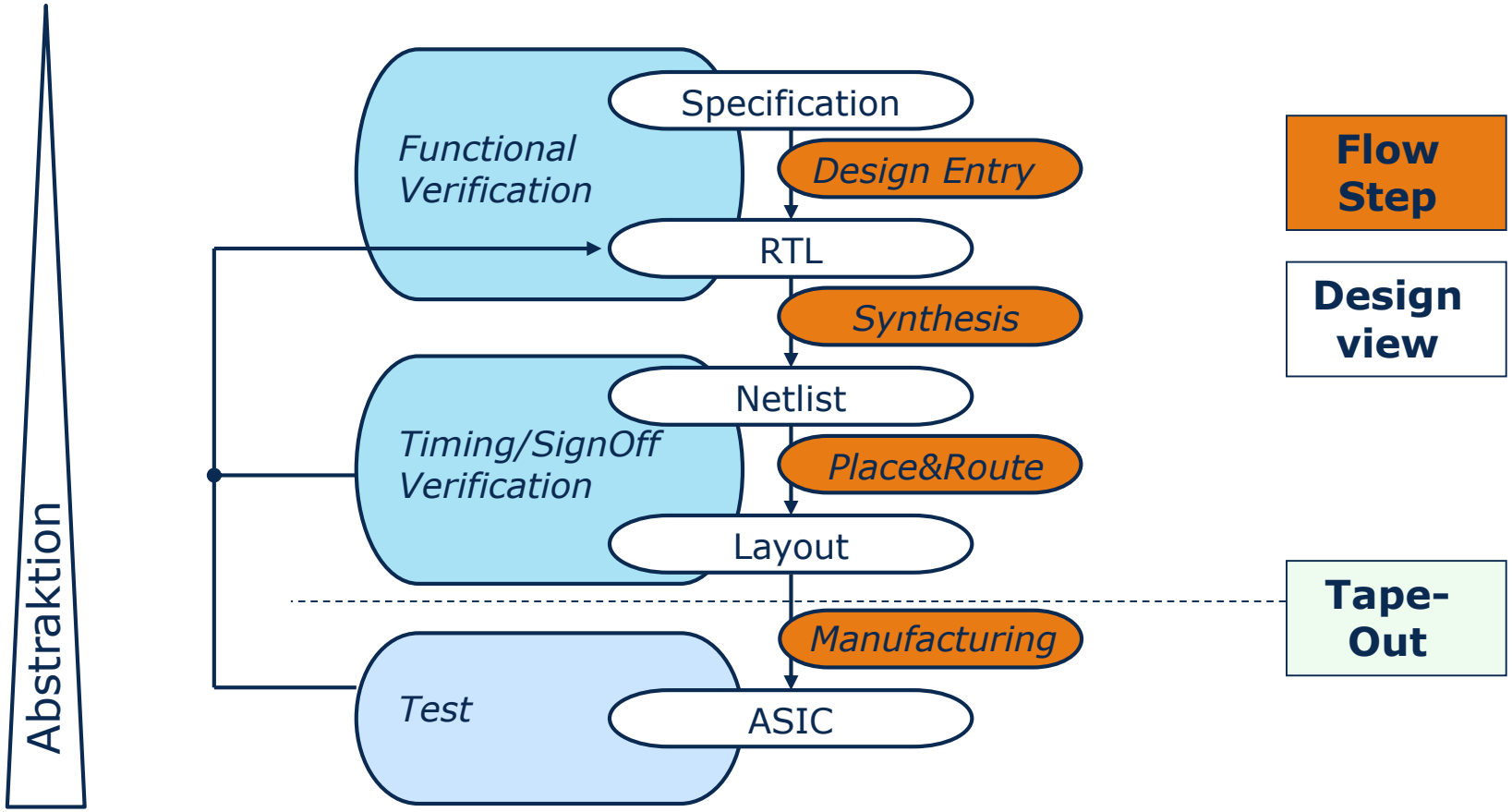
Kosten

- Entwicklung (Design, Implementierung, Verifikation)
- Einkauf von IP
- Produktion (Masken, Waferherstellung, Packaging)
- Test der produzierten Chips

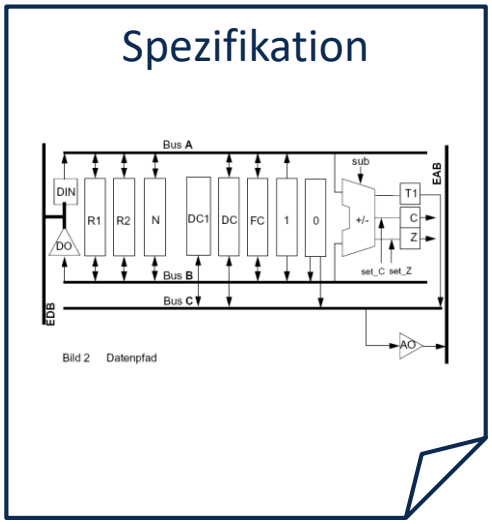
- Architekturentscheidung
- Technologieentscheidung
- Package
- Entwurfsmethodik

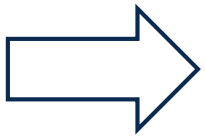
Entwurfsablauf integrierter Digitalschaltungen

Design Flow: Semi-Custom Digital Design



- Entwurf von Schaltungsmodulen basierend auf Systemspezifikation als Register-Transfer-Level (RTL) Beschreibung
- Verwendung von Hardwarebeschreibungssprachen (HDL) wie z.B. Verilog, VHDL
- Synthesegerechte Beschreibung
- Check von RTL Coding Guidelines

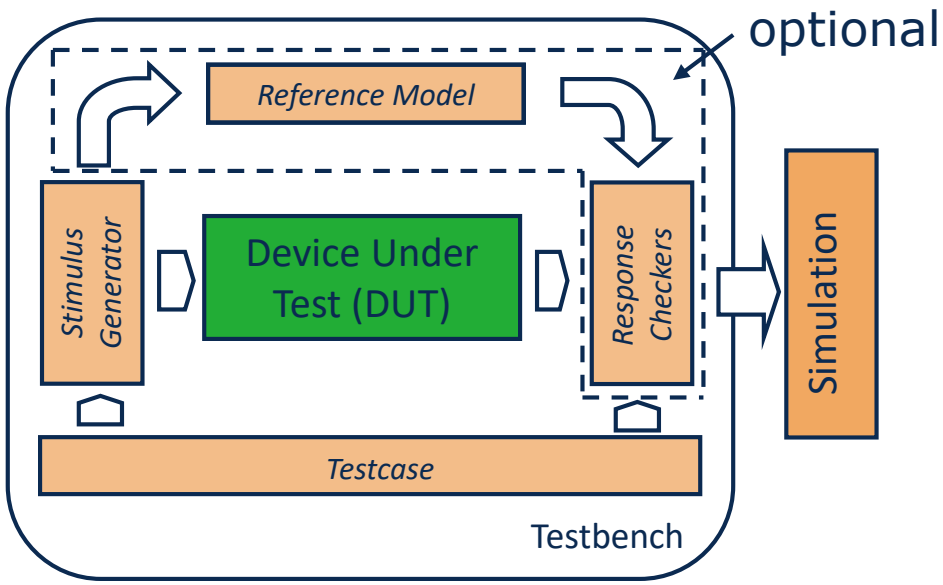




```
module fsm (
module decoder (
module adder_reg (clk_i, a_i, b_i, c_o);
   input clk_i;
   input [7:0] a_i, b_i;
   output [8:0] c_o;
   reg [8:0] c_r;
   always @(posedge clk_i) begin
      c_r<=a_i+b_i;
   end
   assign c_o=c_r;
endmodule
```

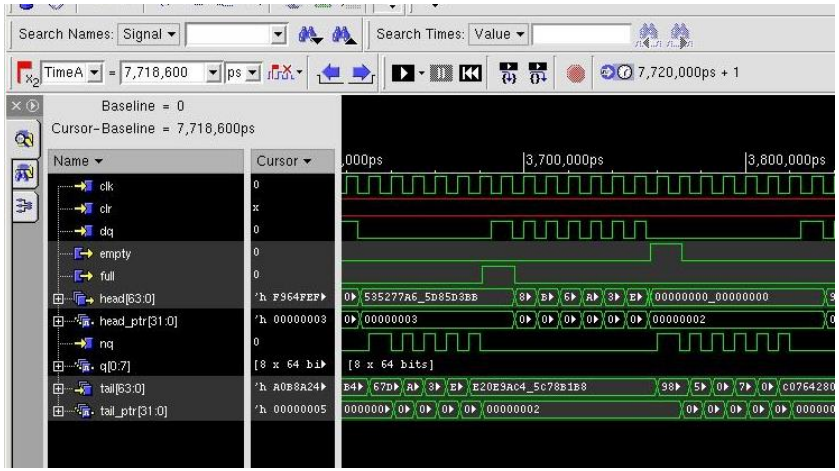
Testbench Implementierung

- Testbench: Simulationsumgebung für ein Systemmodul
- Stimulus Generator erzeugt Eingangssignale
- Response Checker prüft Ausgangssignale gemäß Spezifikation
- Optional: Vergleich mit Referenzmodell möglich
- Testbench üblicherweise mit HDL als Verhaltensbeschreibung implementiert

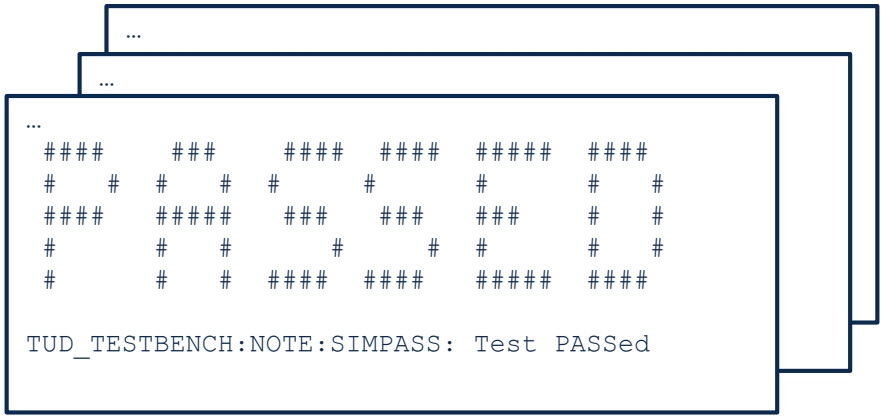


- Simulation der Testbench mit eventbasiertem Digitalsimulator
- Tools, u.a.
  - Cadence NCSIM
  - Mentor Questa

Interaktive Analyse/Debugging



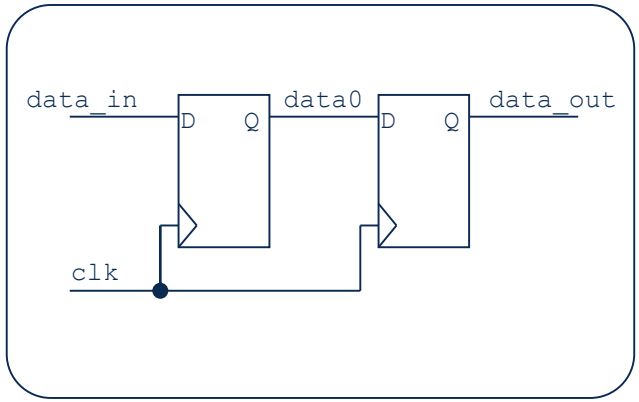
Self-checking Testbench
Test PASS/FAIL



- Abbildung einer synthesegerechten RTL Beschreibung in eine Gatternetzliste basierend auf Standardzell Bibliotheken
- Library File (.lib) beinhaltet Informationen zu Funktionalität, Timing, Power
- Vorgabe von Constraints, z.B. Taktfrequenz, Eingangs- und Ausgangsdelay
- Tools, u.a.
  - Synopsys DesignCompiler
  - Cadence RTLCompiler

```
always @(posedge clk)
begin
  data0 <= data_in;
end
always @(posedge clk)
begin
  data_out <= data0;
end
```

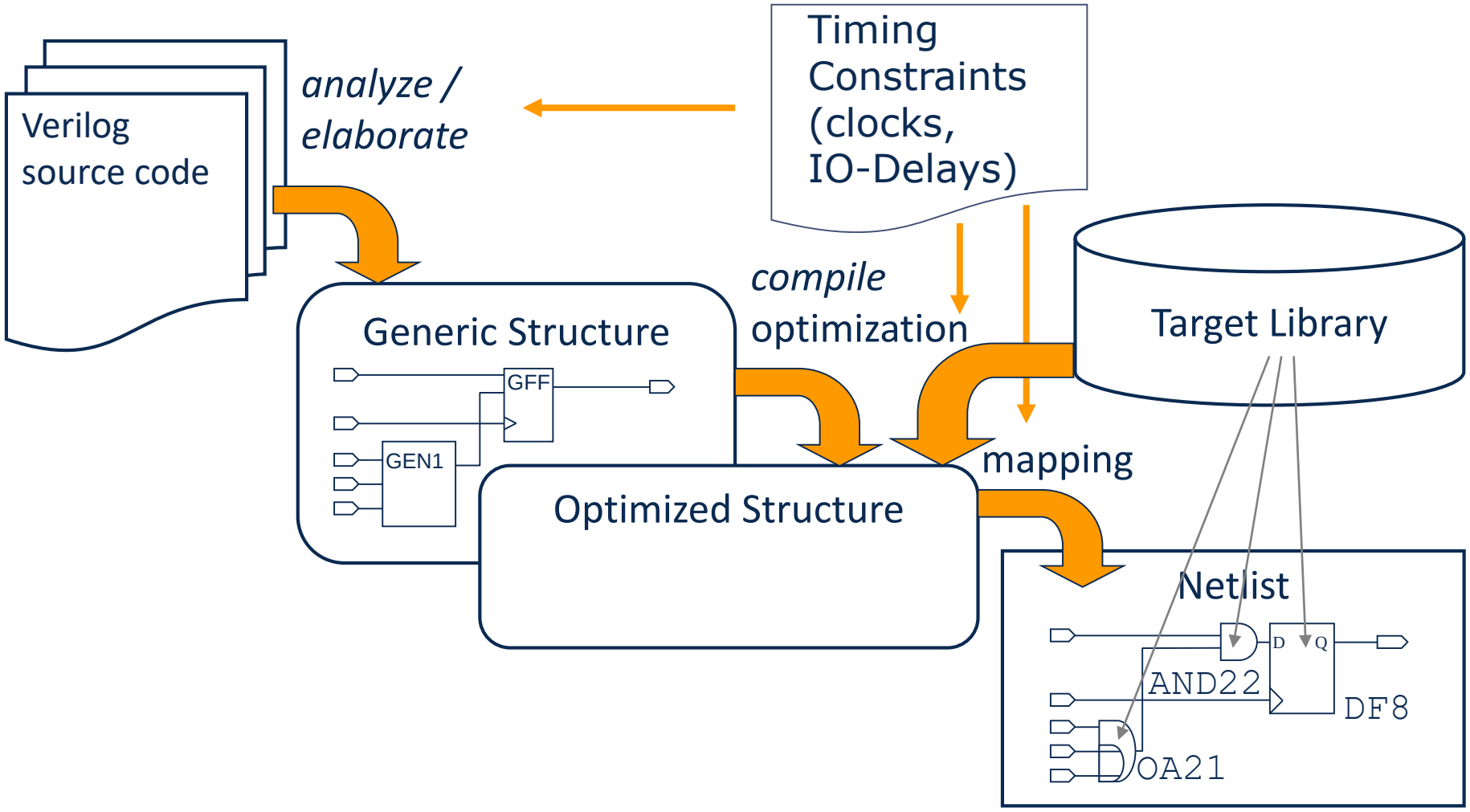
RTL Synthese



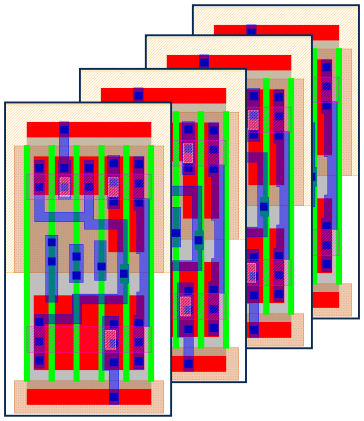
Verilog RTL

Äquivalente Gatter-Netzliste

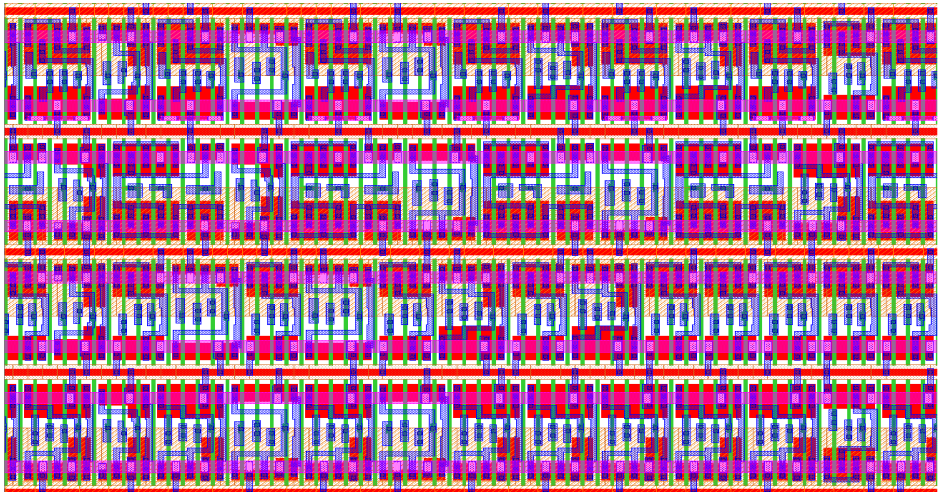
Logiksynthese -Ablauf



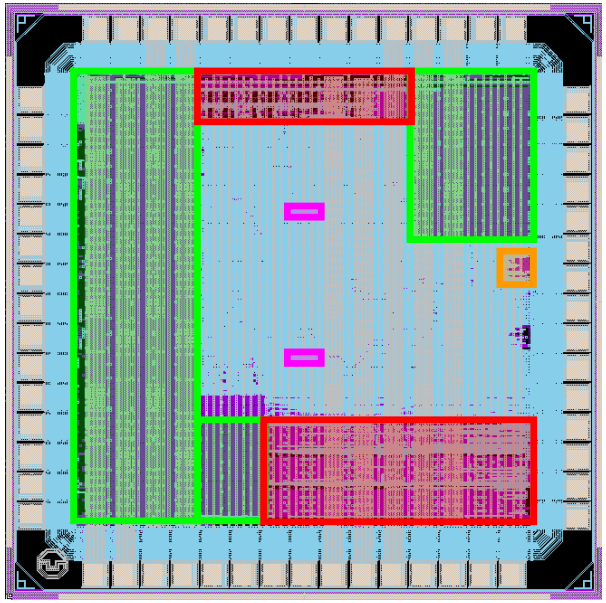
- Automatisiertes Platzieren und Verdrahten der Standardzellen
- Synthese von Clock Trees
- Extraktion von parasitären Layout Elementen



Standardzellen

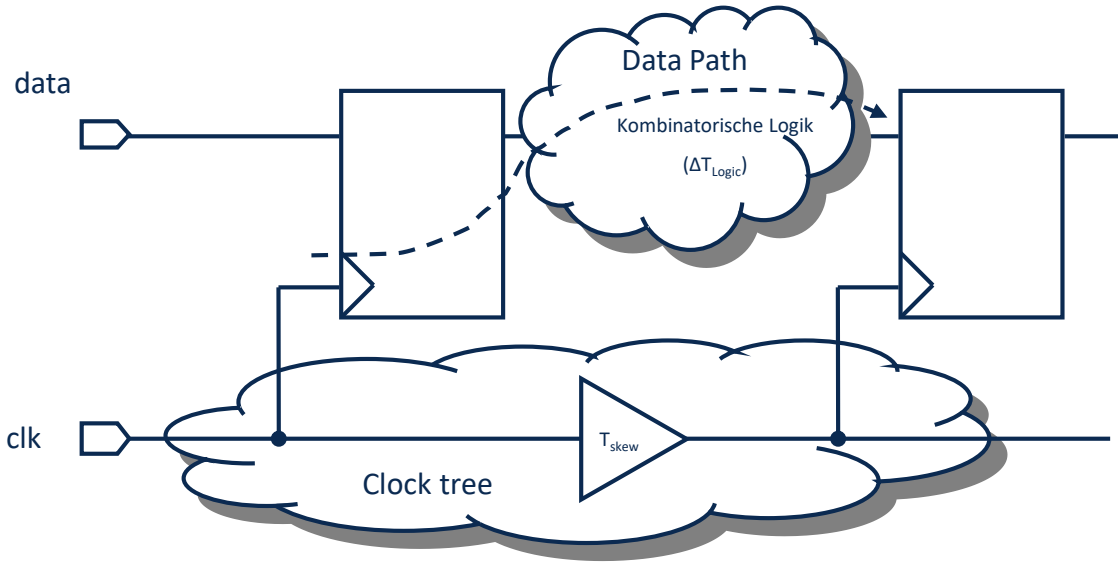


Schaltungsblock



Chip

- Check von Timing Constraints

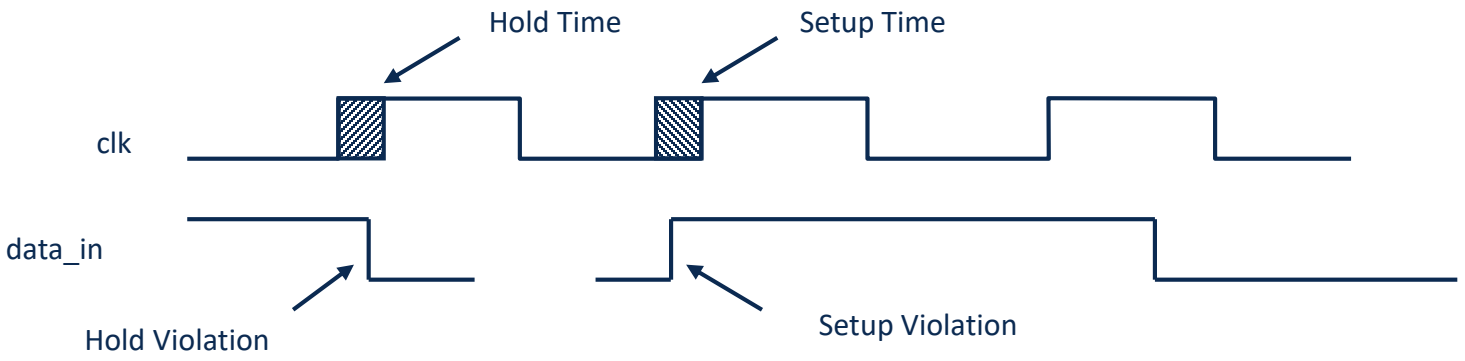


Setup Check:

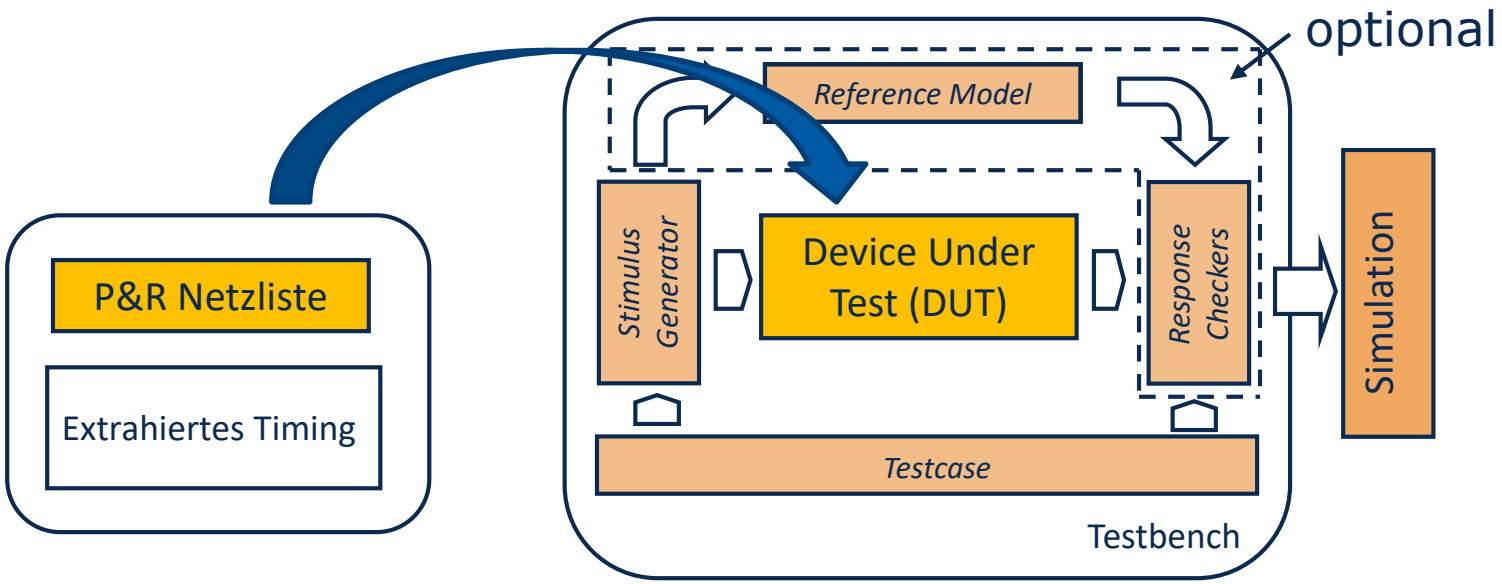
$T_{clk} + T_{skew} \geq \Delta T_{Logic} + T_{setup}$

Hold Check:

$T_{skew} + T_{hold} \leq \Delta T_{Logic}$



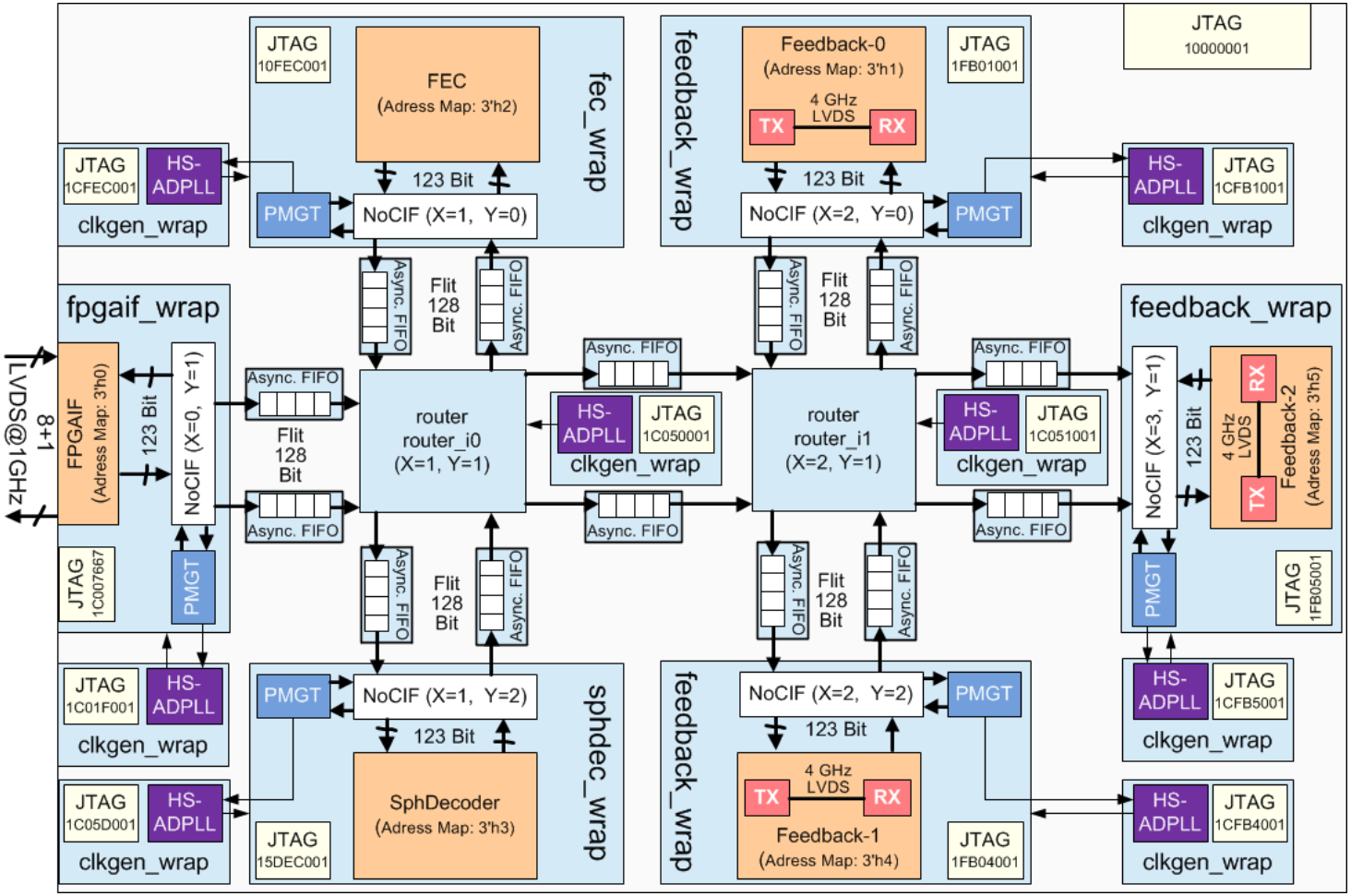
- Annotation von Verzögerungszeiten an die Gatternetzliste
- Individuelle Delays und Constraint Checks (Setup, Hold) für jedes Gatter
- zeitaufwändig



Design Views

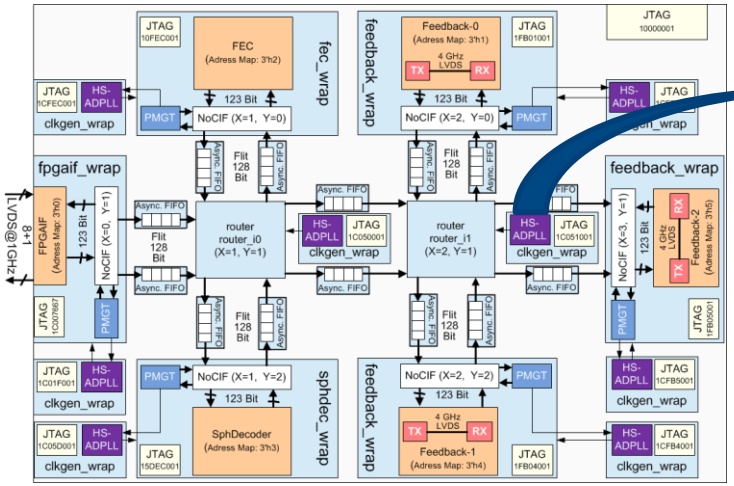
- Ein Schaltungsmodul kann verschieden repräsentiert sein:
  - Verhaltensmodel (behavioral)
    - Verhaltensmodel für Digitalsimulator
    - „Executable System Specification“
  - RTL
    - Synthesefähige RTL Beschreibung
  - Gatter Netzliste
    - Gatternetzliste als Ergebnis von Synthese/P&R
  - Transistor-Netzliste (schematic netlist)
    - Netzliste auf Transistorlevel
  - Transistor-Netzliste (extracted netlist)
    - Netzliste mit Transistoren und parasitären Layout Elementen
  - Layout
    - Physische Darstellung der zu fertigenden Schaltungsgeometrien

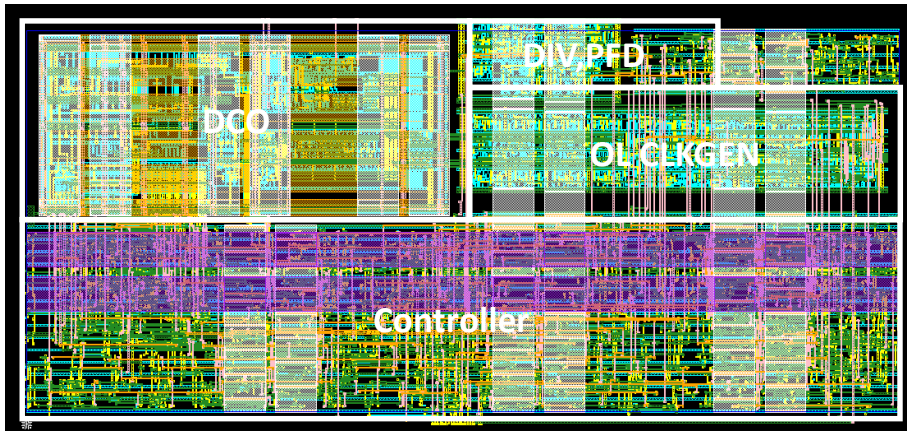
- Hierarchische Gliederung von komplexen Systemen

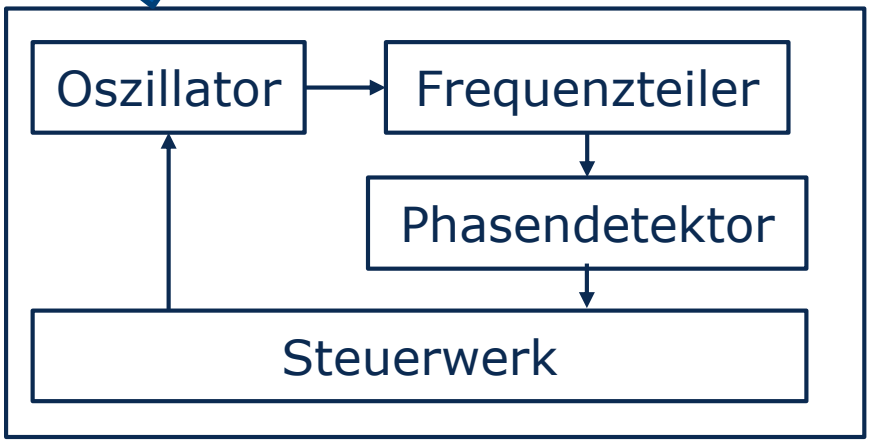


Beispiel: Multiprozessor System „Tommy“

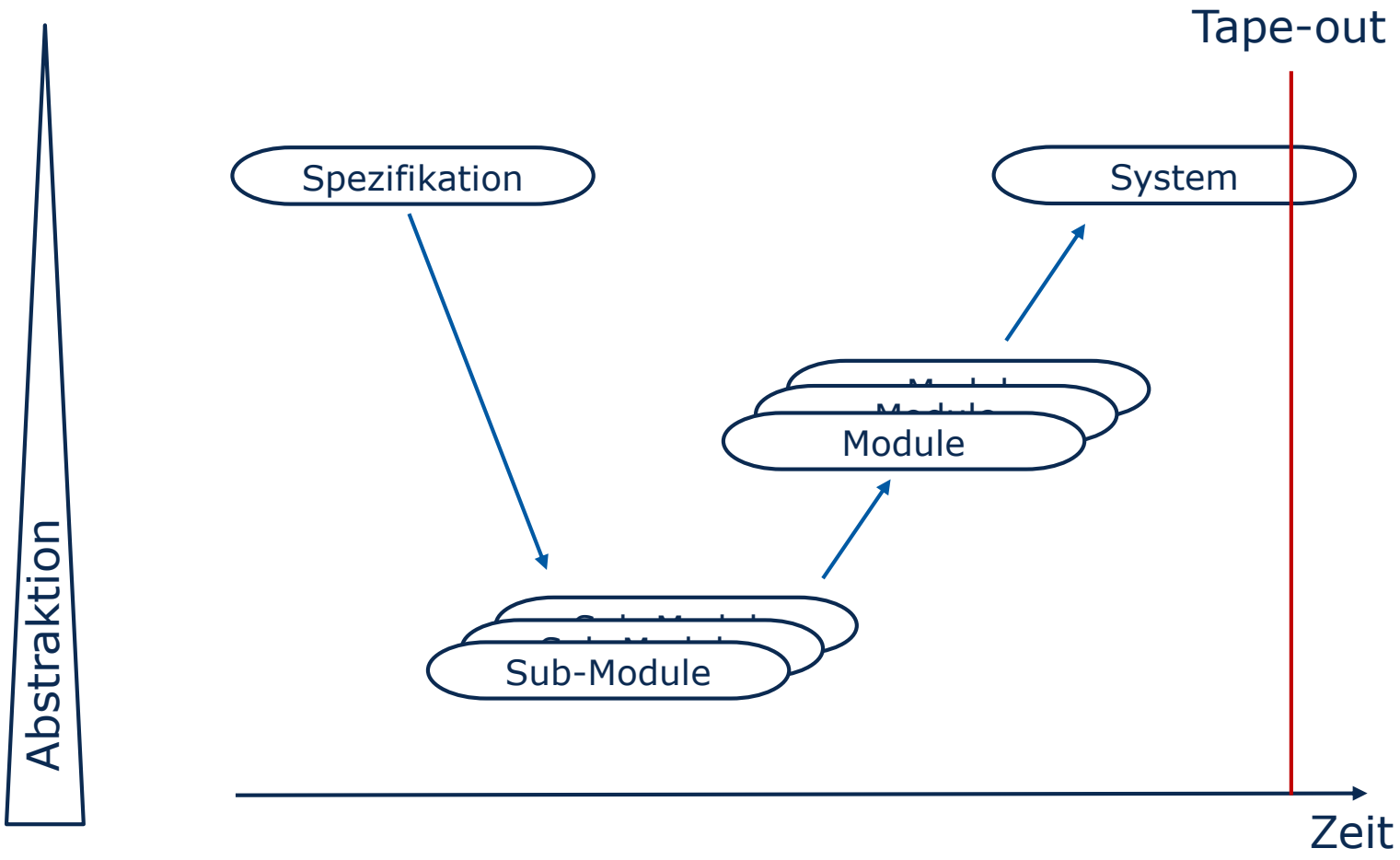
- Beispiel: Volldigitaler Phasenregelkreis (ADPLL) Taktgenerator







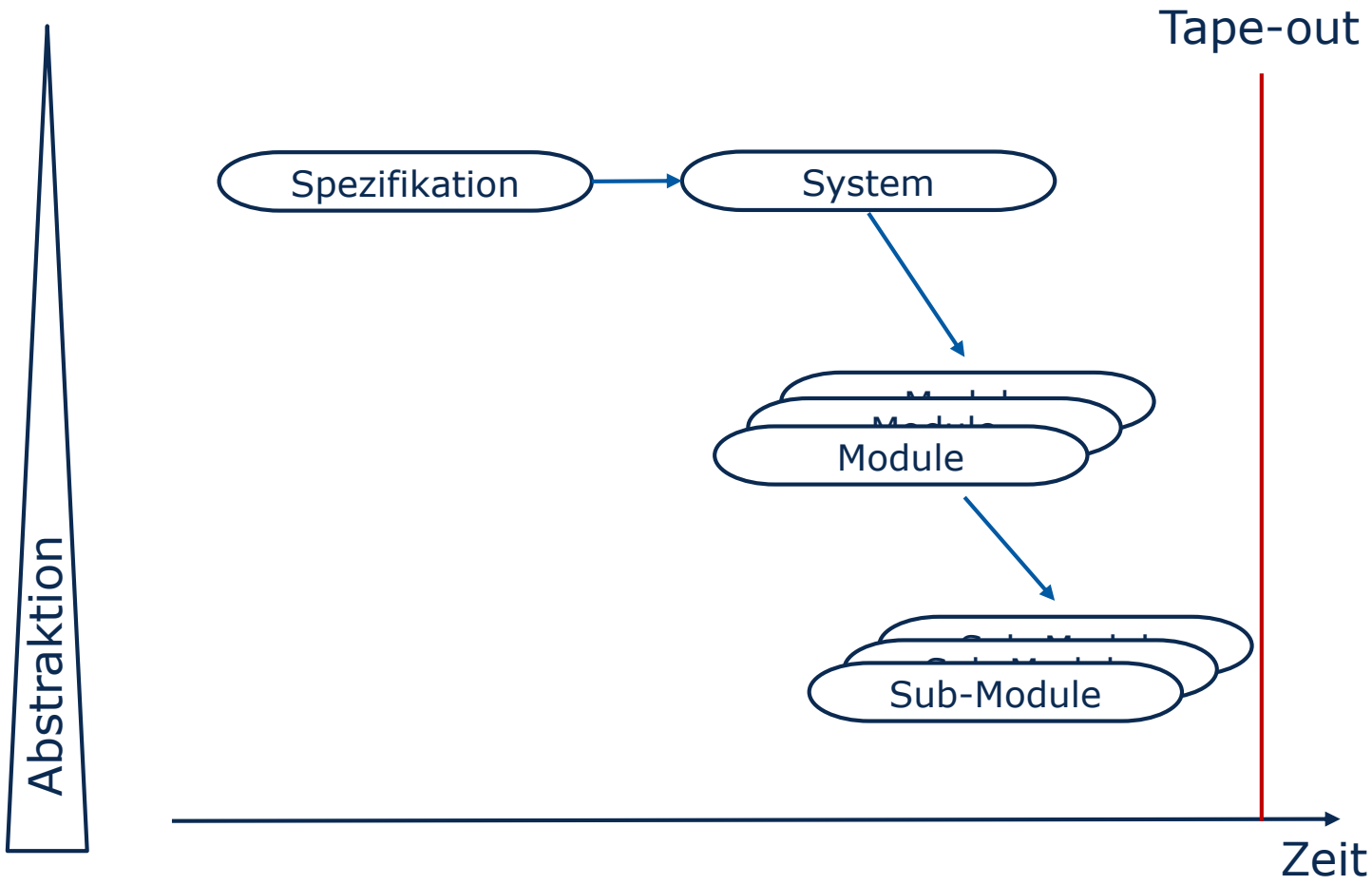
Bottom-Up Design Flow



Bottom-Up Design

- Entwurf von Sub-Modulen basierend auf Systemspezifikation
- Zusammenfügen der Module zum Gesamtsystem
- Vorteile ☺
  - Strukturierte Herangehensweise für kleinere Baublöcke in kleinen Entwurfsteams
  - Geeignet für Sub-Blöcke mit erhöhtem Entwurfsaufwand (z.B. Analog, Mixed-Signal)
  - Geringer zusätzlicher Modellierungsaufwand
  - Benötigt keine vollständige Spezifikation bei Entwurfsbeginn
- Nachteile ☹
  - Ggf. aufwändige Re-Design Zyklen nötig
  - Implementierungsarbeit im Team erfordert sehr genaue Spezifikation
  - Nur sequentielle Entwurfsabläufe möglich

Top-Down Design



Top-Down Design

- Entwurf des Systems basierend auf Systemspezifikation
- Modellierung von Modulen- und Sub-Modulen
- Implementierung von Modulen- und Sub-Modulen
- Vorteile ☺
  - Strukturierte Herangehensweise für komplexe Systeme in größeren Entwurfsteams
  - Parallele Entwurfsarbeiten möglich
    - RTL Implementierung
    - Synthese
    - Place&Route
    - Verifikation
  - Schnelle Verfügbarkeit eines simulierbaren Gesamtsystems (virtueller Prototyp) für
    - Performance Abschätzungen
    - Applikationsentwurf (PCB, Firmware)
- Nachteile ☹
  - Detaillierte Spezifikation bei Entwurfsbeginn benötigt
  - Modellierungsaufwand für Module und Sub-Module

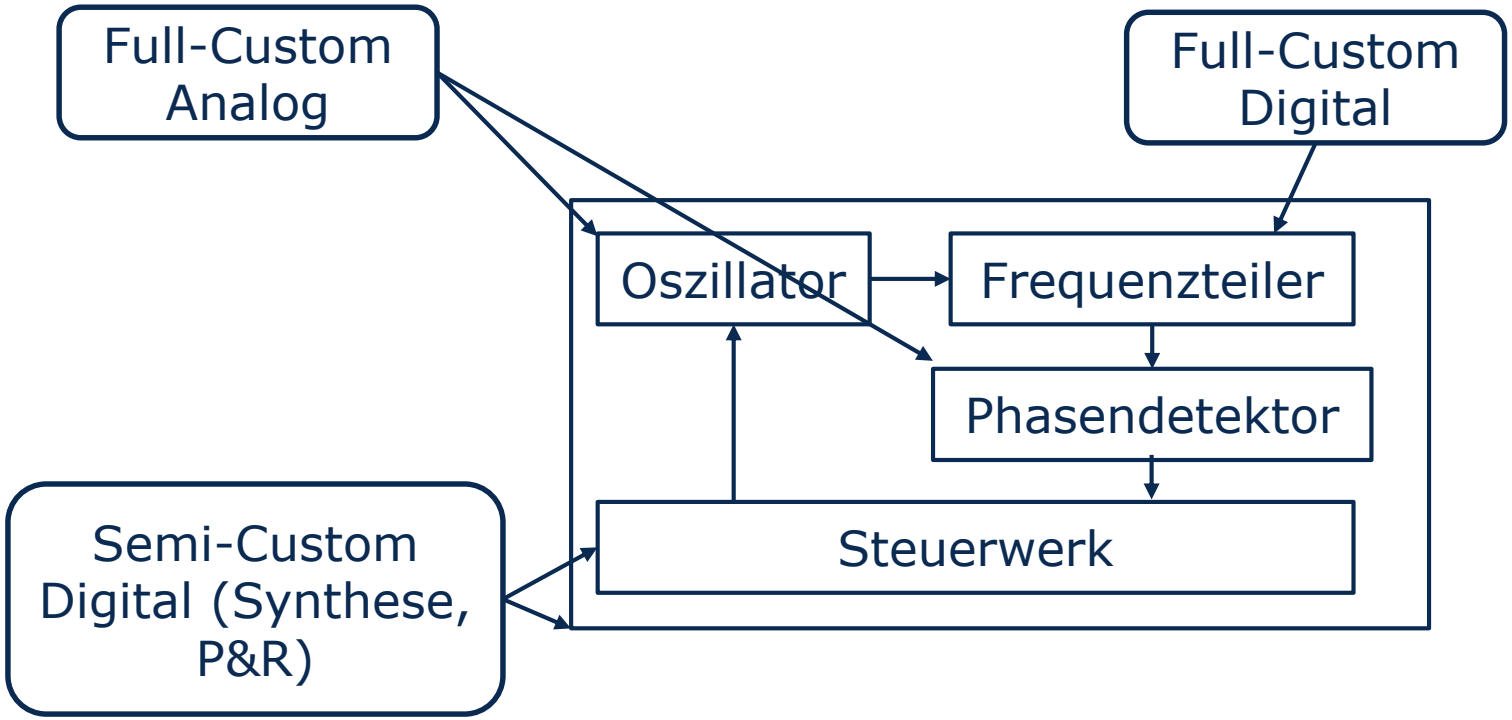
- Möglichkeit der Simulation von Systemkomponenten in unterschiedlichen Abstraktionsebenen (Views)
- Grundlage des Top-Down Designs
- Beispiel: Implementierung eines ADPLL Taktgenerators:

behavioral

RTL

gate

transistor



- Mixed-Level Repräsentationen in unterschiedlichen Entwurfsphasen

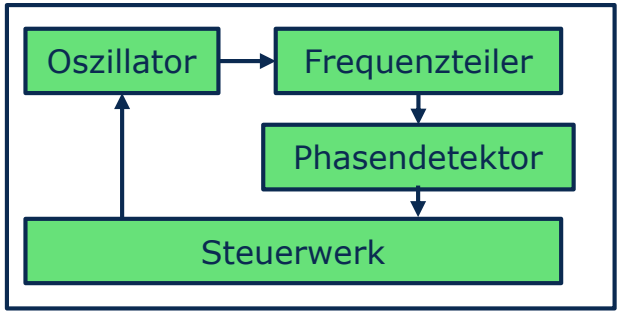
behavioral

RTL

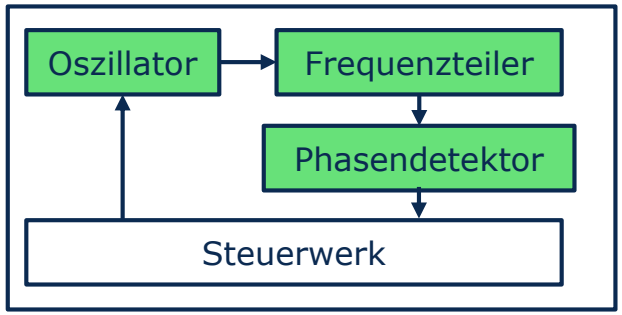
gate

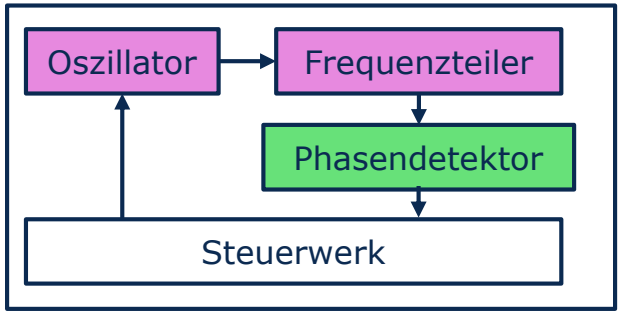
transistor

Systementwurf

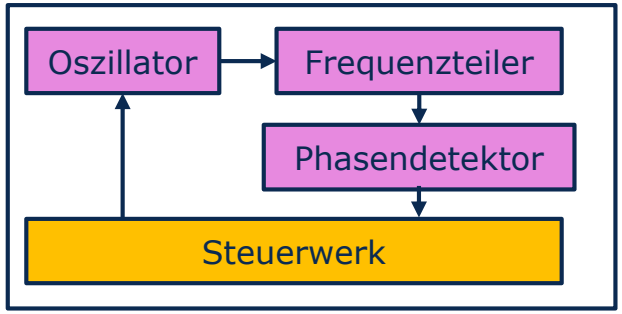


RTL Verifikation



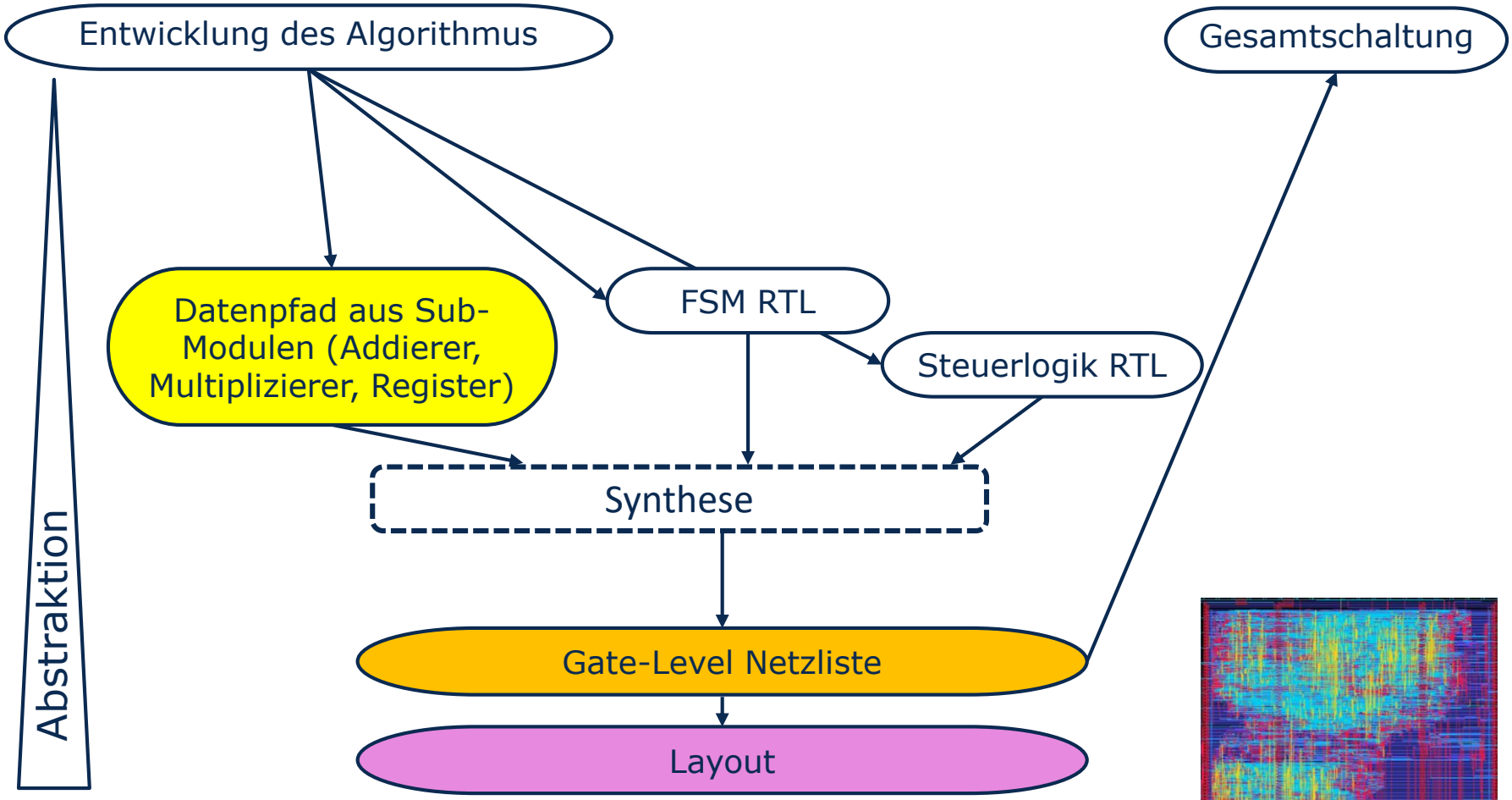


Oszillator Verifikation



Post-Layout Simulation

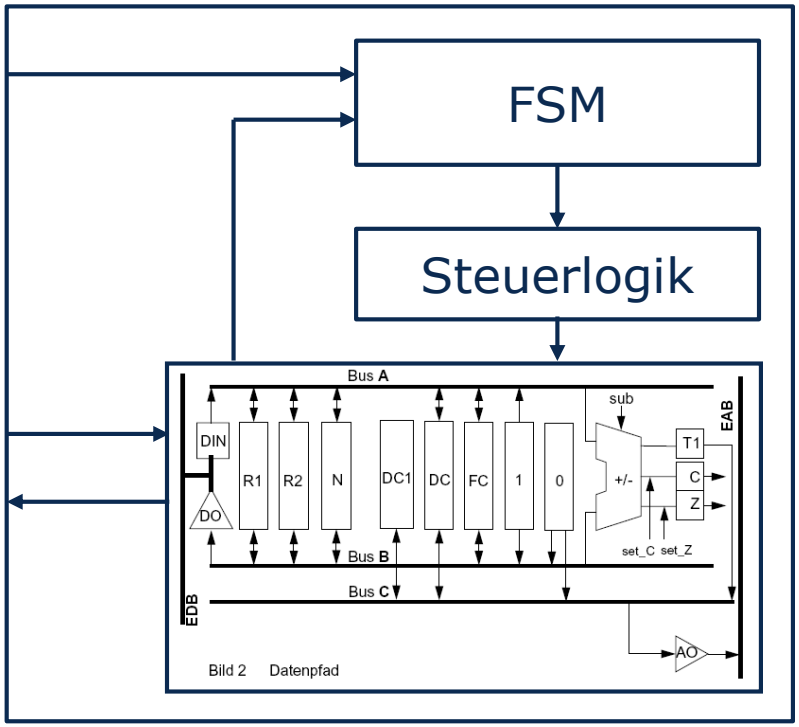
Design Flow im Praktikum Schaltkreis- und Systementwurf



- Design Struktur

RTL

Gate Netlist



Place&Route Netzliste

- Mehr Infos: → siehe Praktikumsanleitung