Karlsruhe Institute of Technology

Design Digitaler Schaltkreise

System Level introduction and Lab Work Discussion

Asic and Detector Lab - IPE

Prof. Ivan Peric ivan.peric@kit.edu
Richard Leys richard.leys@kit.edu

Ilias: https://ilias.studium.kit.edu/goto_produktiv_crs_430424.html

Lecture Goal

- Introduce System Level Design
- Introduce Digital Implementation flow
  - Rapid overview
  - Some other lectures will go more in details
- Plan Lab Work
  - Look at a base system
  - Discuss work-plan
    - More/Less HDL?
    - More/Less Chip Implementation?
  - Comments are welcome

Lab Work organisation

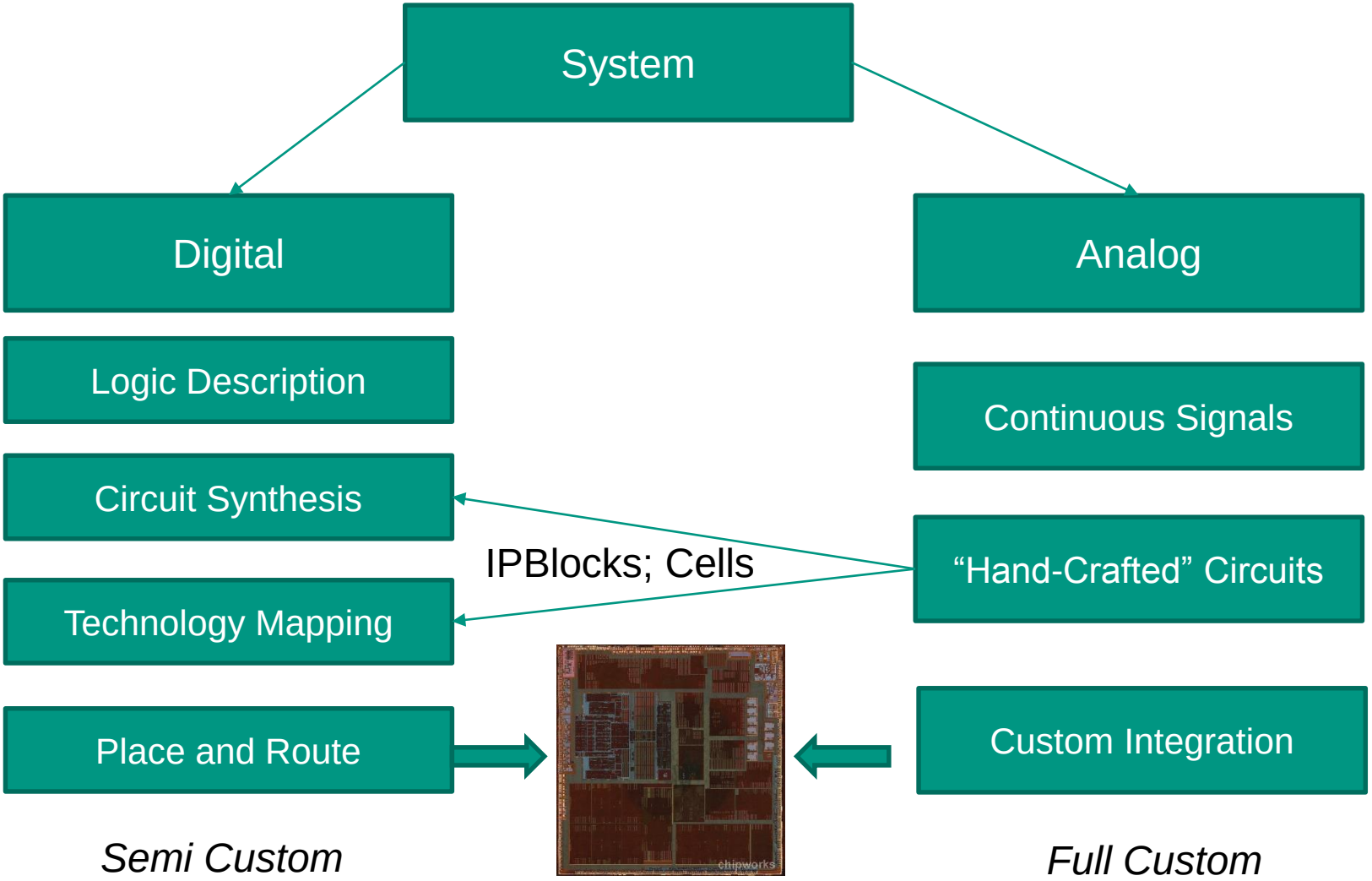
- Groups of 15
- 2/3 groups
- Every 2/3 weeks
- Begin: 7th May
- Where/When: Thursdays 13h30 at IMS (Westhochschule)
- Registration on ILIAS
  - https://ilias.studium.kit.edu/goto_produktiv_crs_430424.html
- Sprechstunde:
  - Nach vereinbarung
  - Gut: Freitags

SYSTEM LEVEL BASICS

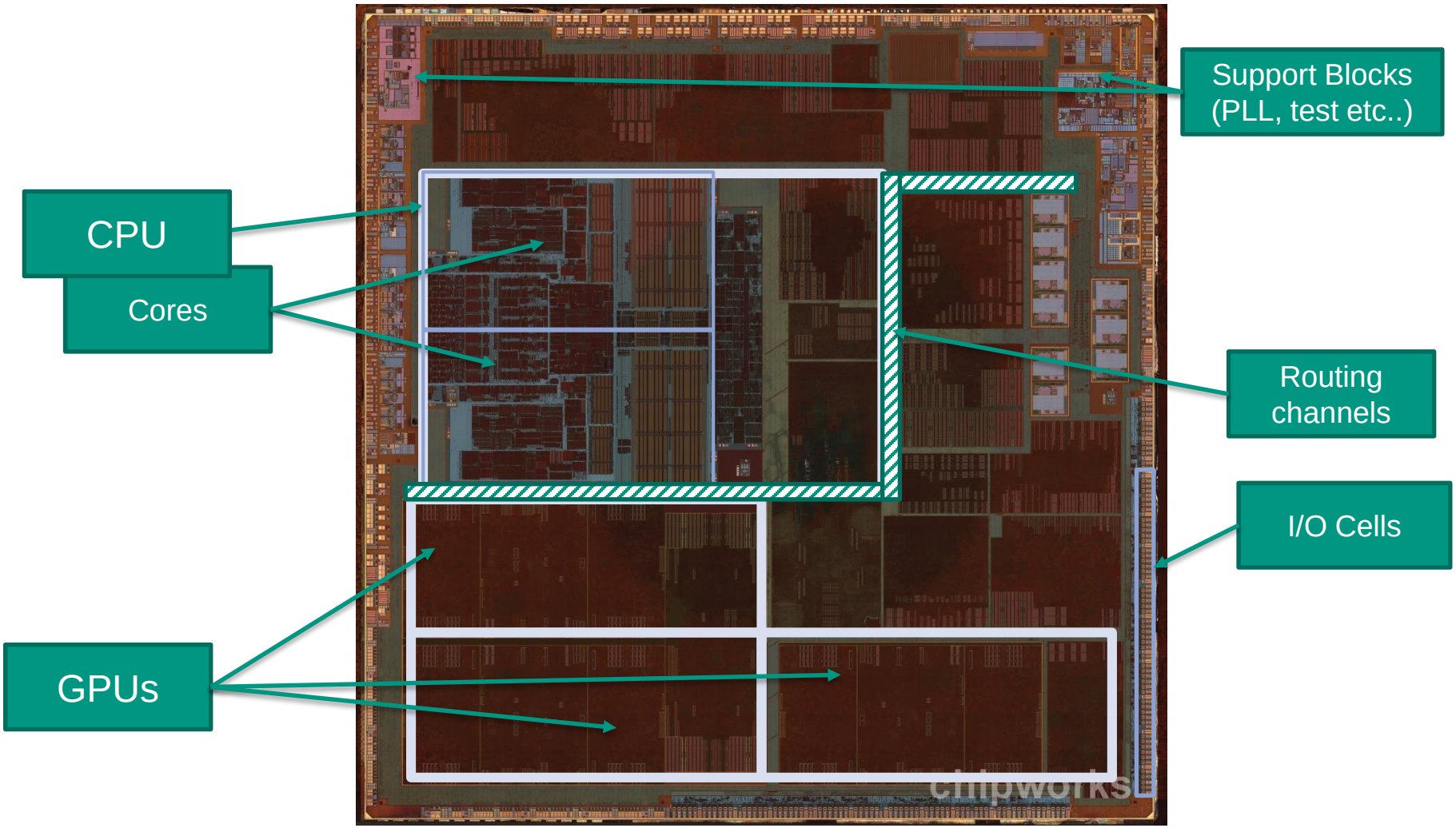
Design Size Growth

- Features Size decreases, now 65nm easy to access, next standard node is around 28nm, maybe 13nm
- Consequences:
  - More features per ASIC -> System on Chip
  - Each feature gets bigger -> Design and Implementation Cost higher
- System Implementation stakes:
  - Difficult: Register Transfer level design and verification
  - Difficult: System Physical Planning and constraining
  - Actual implementations: the tools do the job
- Costs Location:
  - Hardware Designer Time
  - Specification
  - Computation time leads to project-parallelizing

Analog and Digital Flows



Top Level System on Chip Overview

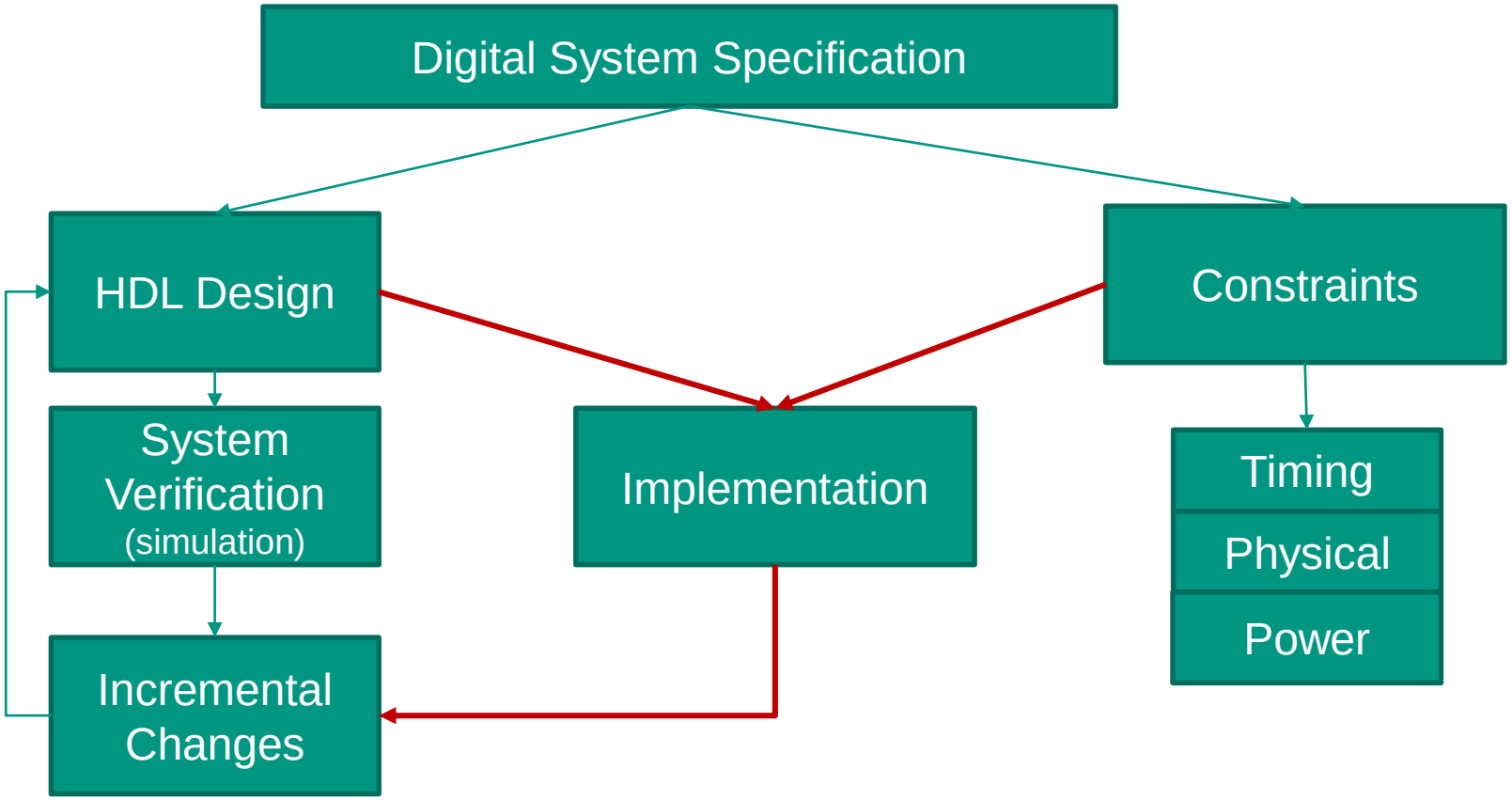


Iphone A6 APU (picture from chipworks.com)

Difference to FPGA design

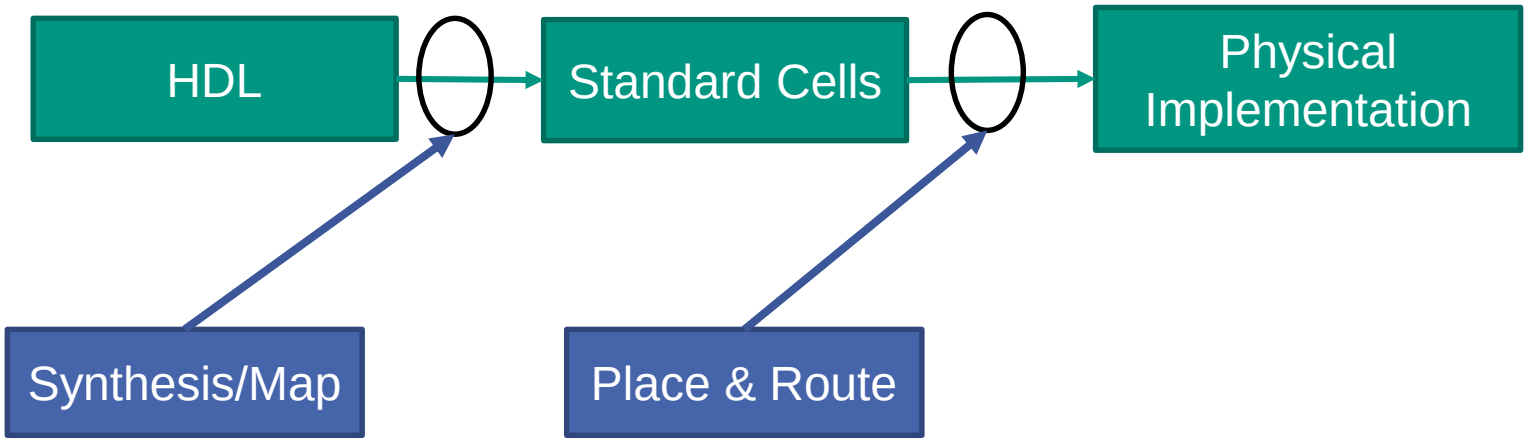
- RTL-Level design is very similar
- FPGA
  - FPGA is a System with pre-placed needed functions
  - Synthesis maps to those functions
  - Place and route creates configuration file for routing
  - Design size and speed is limited
- Digital ASIC:
  - Every component must be provided and placed
  - Example: SRAM memories
  - Example: PLL
  - Very expensive
  - Design can be very big

System Design Phases



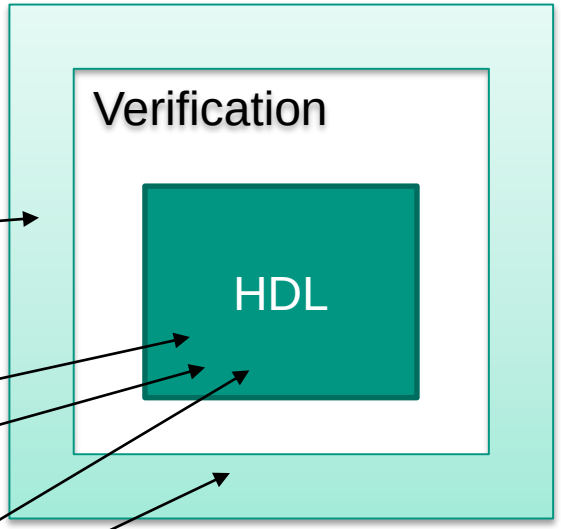
DESIGN FLOW OVERVIEW

Simplified Overview



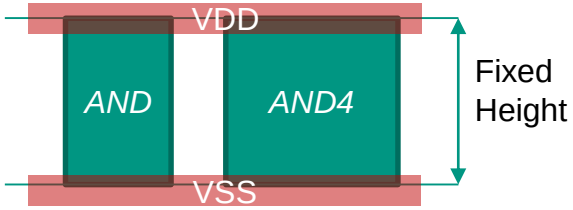
HDL Design reminder

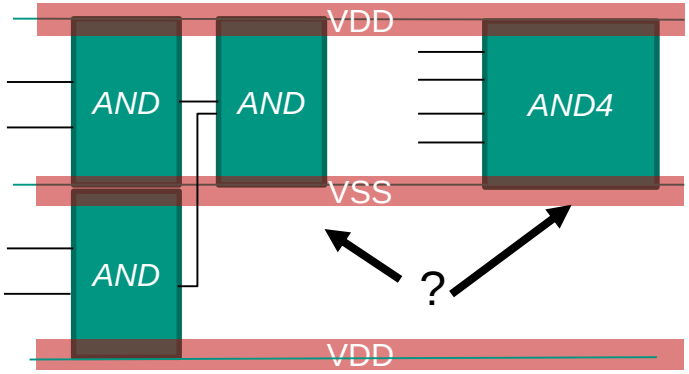
- HDL -> Register Transfer Level
  - Verilog
  - VHDL
- System Verification and Metrics
  - Simple test benches
  - Complex Verification
    - UVM with SystemVerilog/e-Language
  - Assertions
    - “Inline” signal dependencies check
  - Coverage
    - Detect unused logic
- Software Supported design
  - Abstract tools for easy design
    - FSM designers
    - Registerfile generator
    - HDL generator
    - Etc…
  - Mostly TCL language
  - Requires good software skills to develop tools



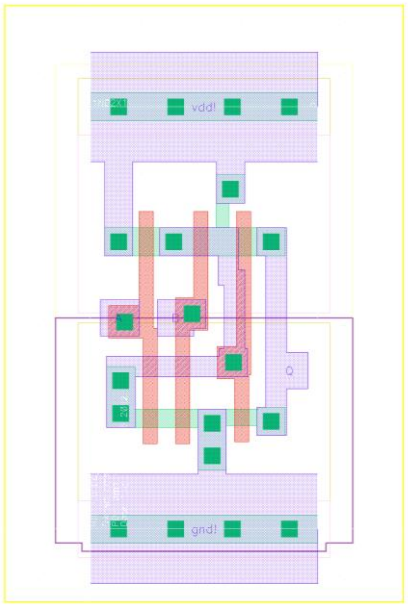
Synthesis: Standard cell mapping

- Start Point: HDL
- Synthesis creates the logic circuit from HDL
- Circuits is mapped to Standard Cells
  - Standard Cells are VLSI implementation of simple logic functions inside a standard sized block
  - Designs Kits provide standard cells
  - Each Cell is a logic function implementation:
    - Example: AND3 -> AND with 3 inputs
  - Timing is provided in Cell description
    - Example: Slow/Typical/Fast corners
- Mapping is optimised:
  - Timing
    - Example: Use Low threshold cells
  - Area
  - Power



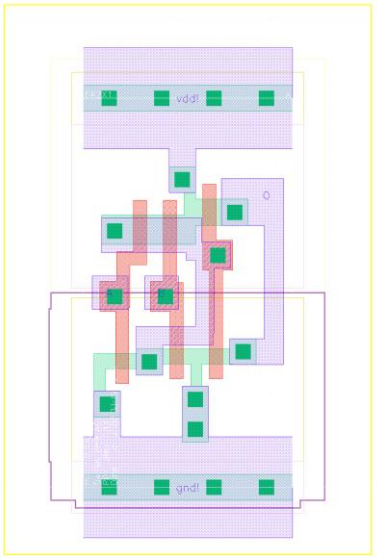


More standard cells

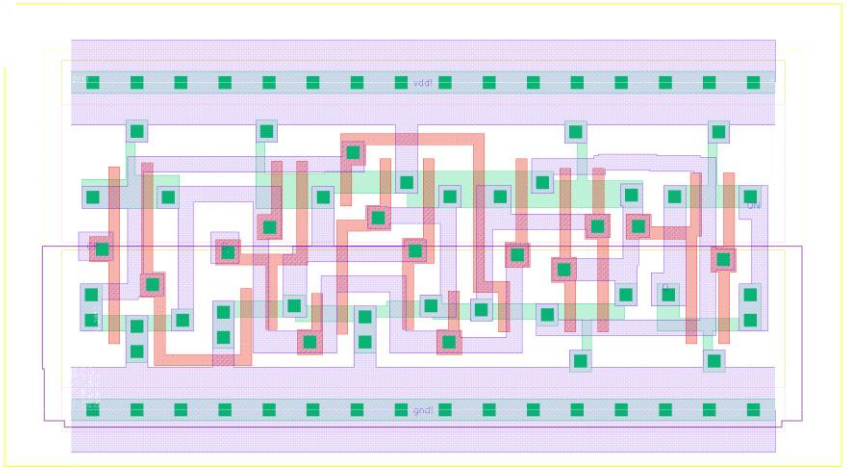


AND

OR



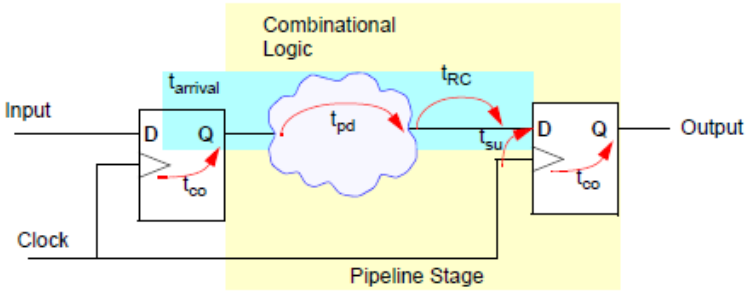
Flip Flop



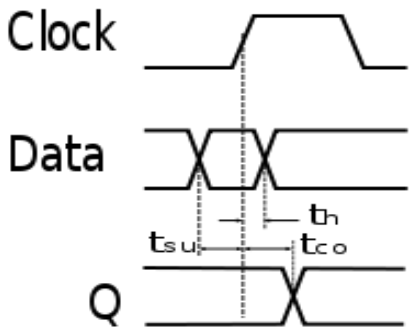
Synthesis: Static Timing Analysis

- Lowest Level in Register Transfer Level
  - Register
  - Logic
  - Register
- Timing Constraints:
  - Clock cycle definition
  - Input/Output Delays
- STA:
  - D->Q time
  - Logic (gates + interconnect) time
  - Must fit in the clock cycle
    - Check setup time: Too slow
    - Check hold time: Too Fast
- <u>Fixes happen along the whole implementation process</u>

$t_{cyc} - t_{arrival} \geq t_{su}$ must be satisfied by all pathes



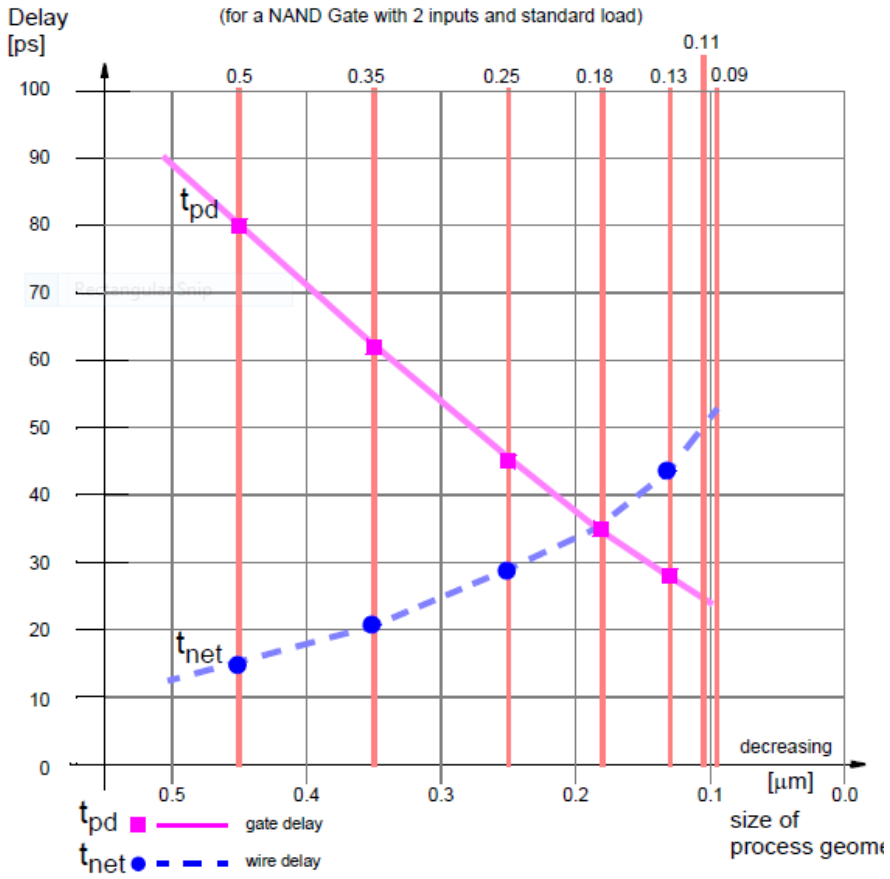
where: $t_{arrival} = t_{co} + t_{pd} + t_{RC}$



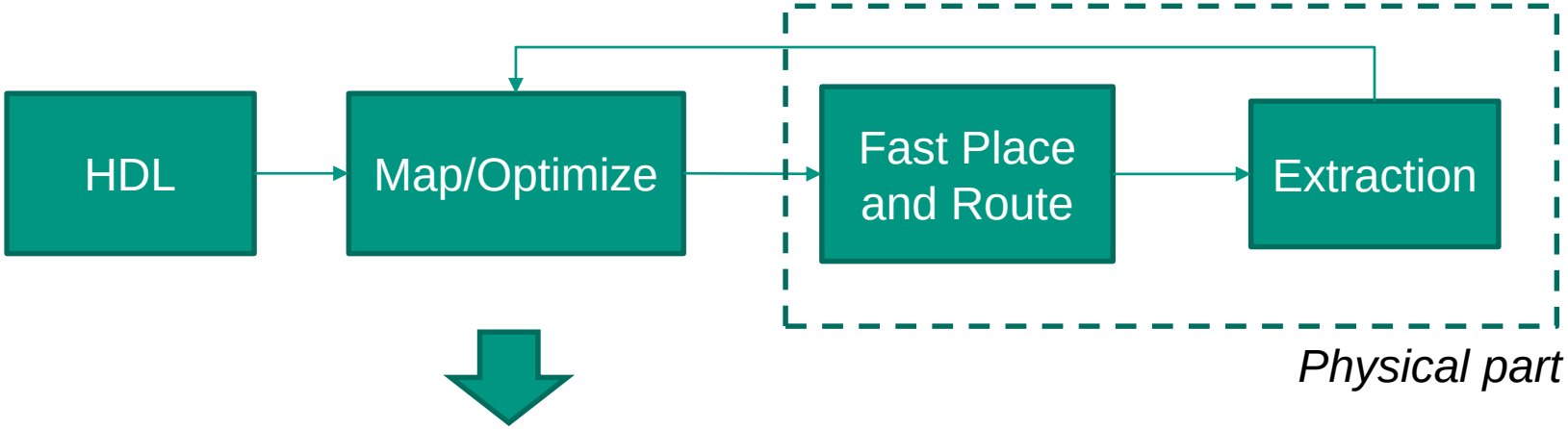
Picture: Uni. Heidelberg – LS Rechnerarchitektur

Synthesis: Static Timing Analysis

- During Circuit mapping, the STA checks the timing paths
- Two delay paths:
  - Gate Delay: Given by Standard Cell Specification
  - Wire Delay: Interconnection
- Wire Delay:
  - Small feature size makes wire delay predominant
  - Need for physical synthesis



Synthesis: Physical Synthesis

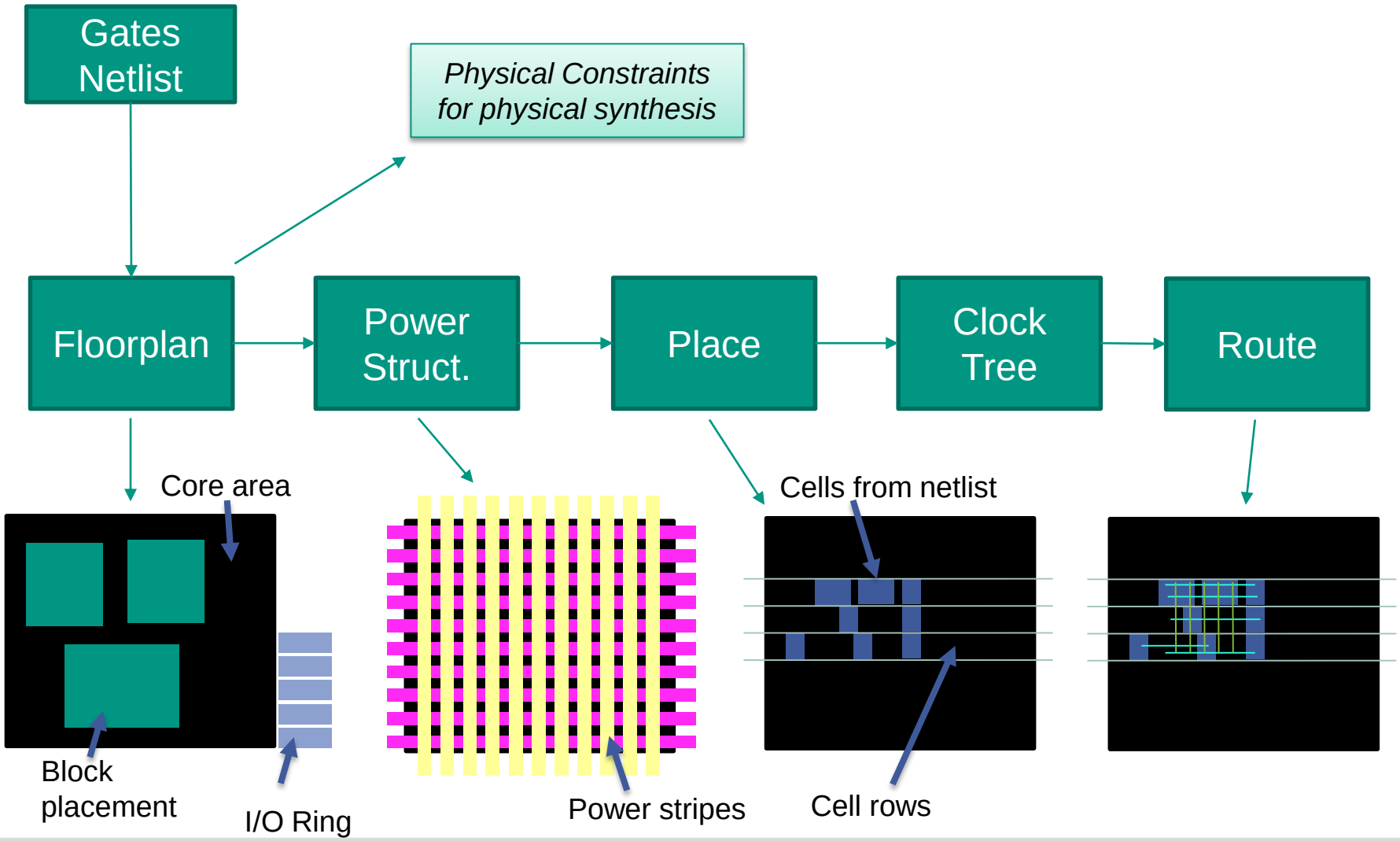


Gates Netlist

- Verilog file with module instances
- Modules are Standard Cell instances

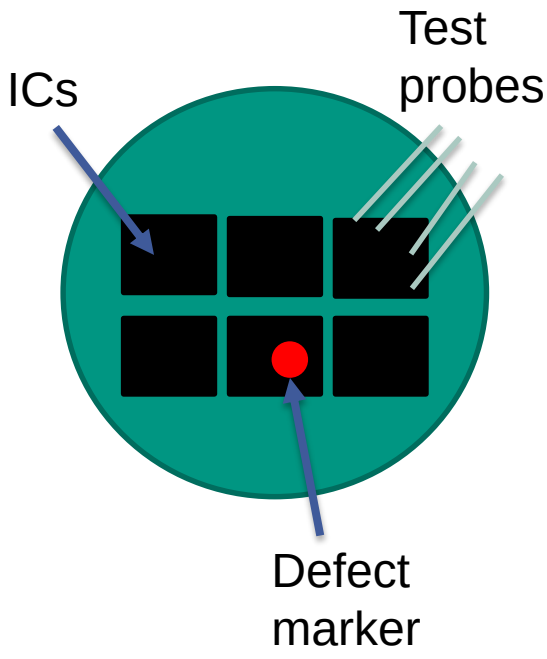
.AND(a,b,net1);
.OR(c,net1,net2);

Place and Route overview



Design For test

- After Production
  - Working Ics
  - Defect Ics
  - Defects/Working = Yield
- Tests:
  - Possible: Optical defect detection
  - Flip-Flop test patterns: JTAG
  - Self test for some blocks
    - SRAM memories have self testing
  - System level test function
- When to test
  - Wafer level during manufacturing
    - Expensive
  - In Lab/Facility using system level tests
- How to implement tests
  - Test patterns can be added during synthesis
  - System Level:
    - Embedded test functionality
    - System test: Setup a system and see what happens



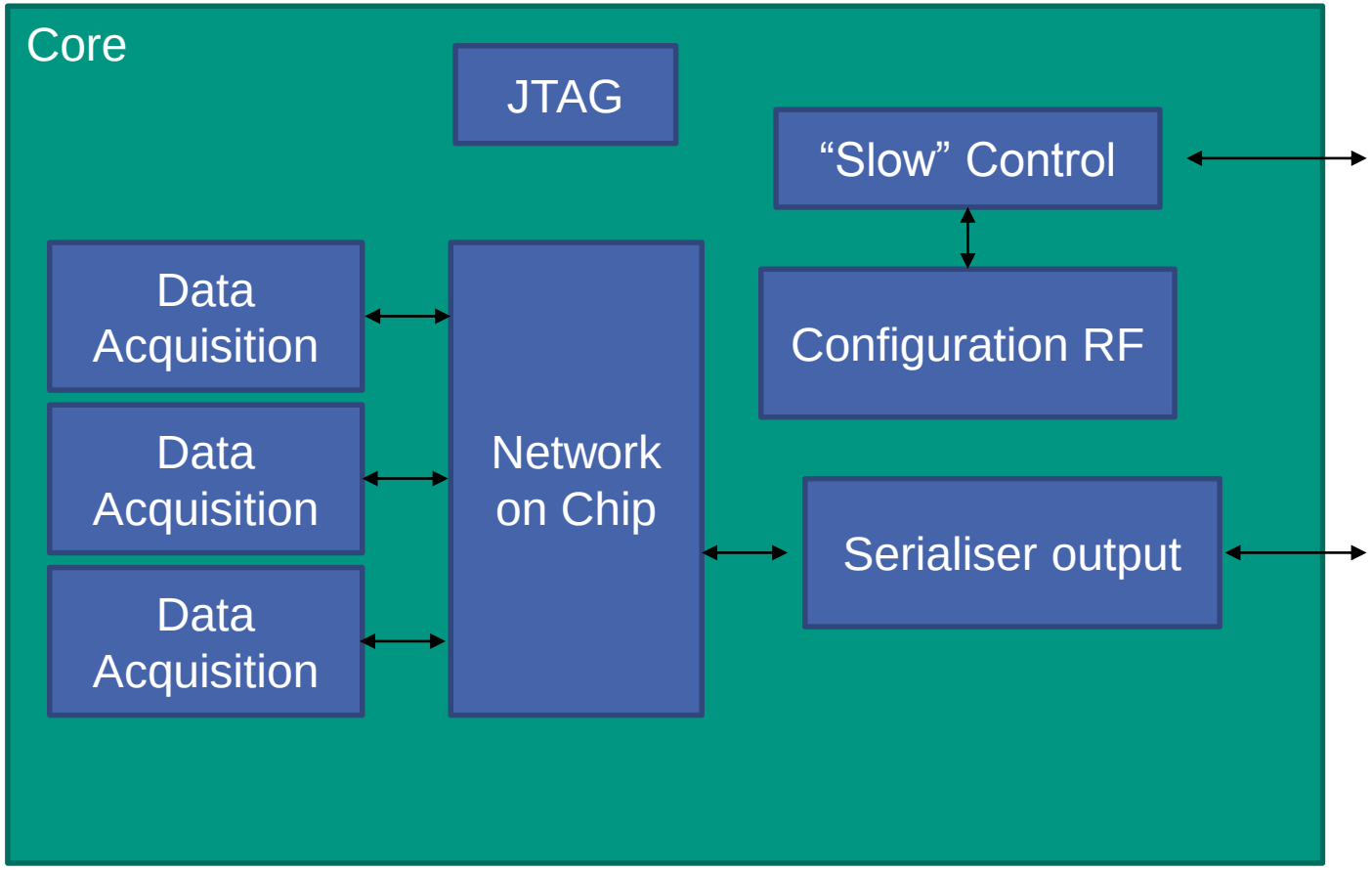
Wafer level test example

LAB WORK PLAN

Design Phases

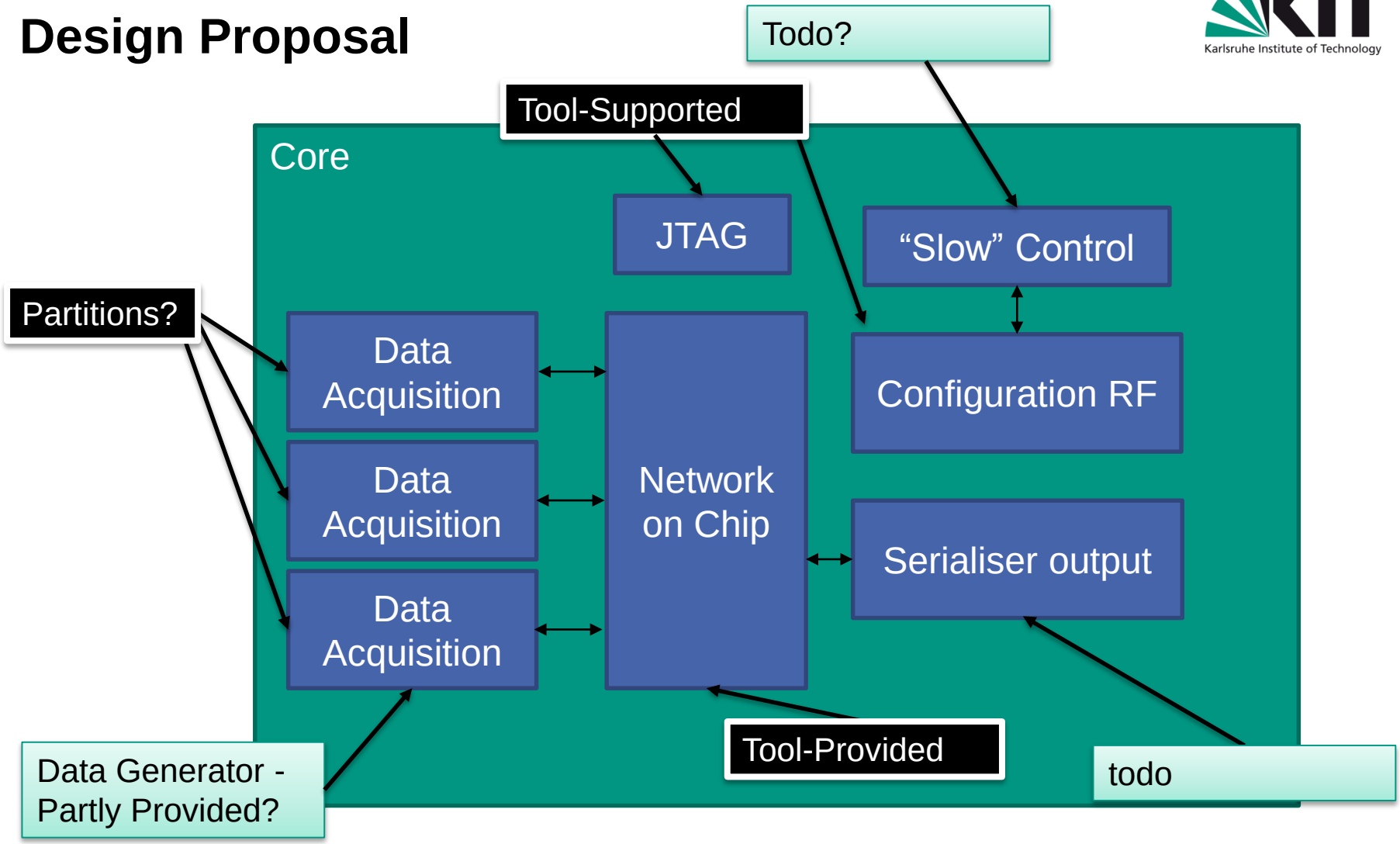
- HDL Building Blocks
  - Simple HDL Blocks to get used to basic building blocks
  - Simple Synthesis to check design characteristics
- System Design
  - Create a Chip Hierarchy
  - Use some tools to add special components
  - Add some system verification constructs for fun
- Implementation
  - Prepare Chip Constraints:
    - Target Clocks
    - I/O Delays
    - Floorplan
  - Synthesise: Cadence RTL Compiler
  - Place and Route: Cadence First Encounter

Design Proposal



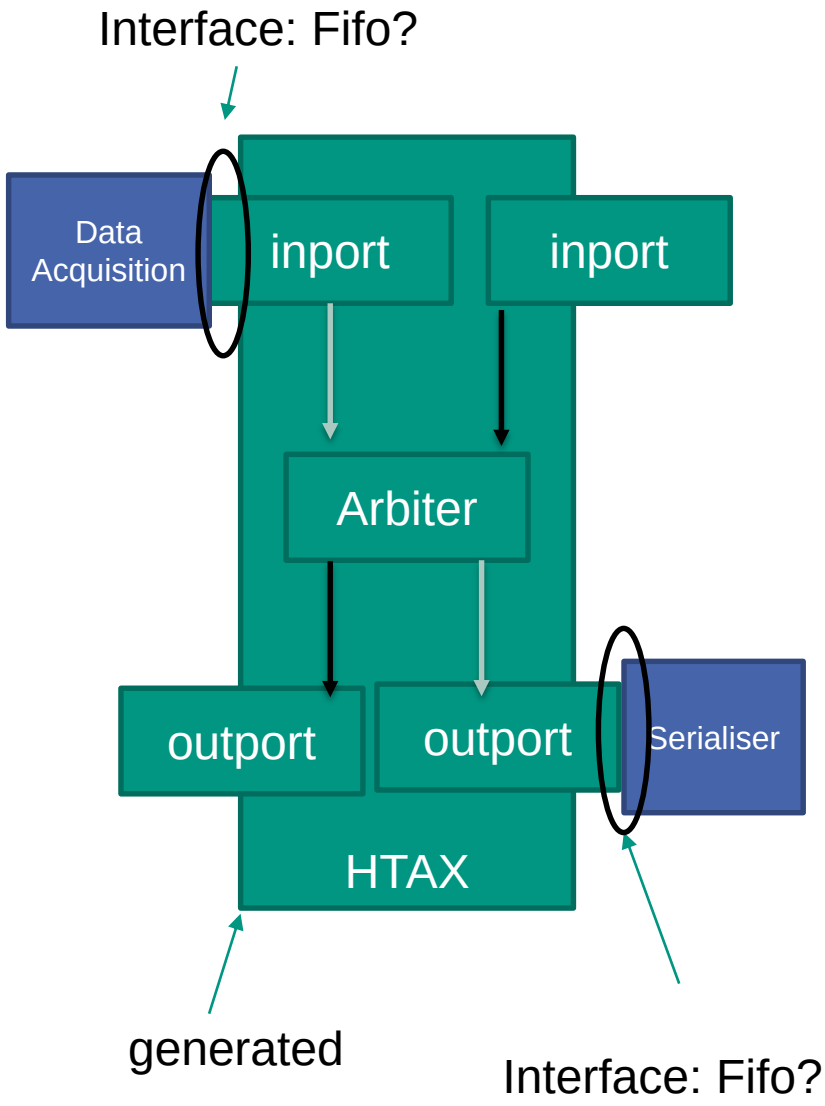
Ambitious -> don’t care too much about functional correctness ?

Design Proposal



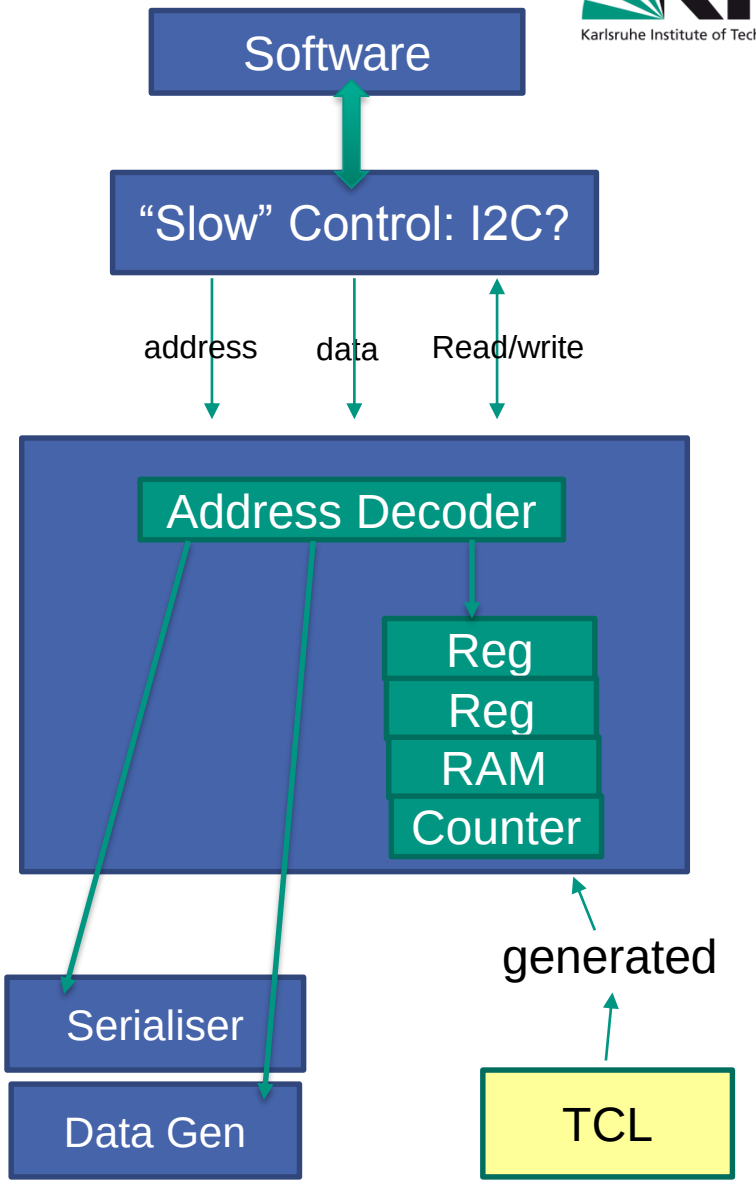
Component: Network on Chip

- Idea:
  - Have various design functions exchange data
- Network form: Crossbar
  - In port requests a target output port
  - Arbiters grants when channel is free
- Content:
  - Input/Output ports
  - Arbiter
- Tool supported:
  - A script can be used to generate the Verilog
- Software name: HTAX (uni Heidelberg; LS Rechnerarchitektur)



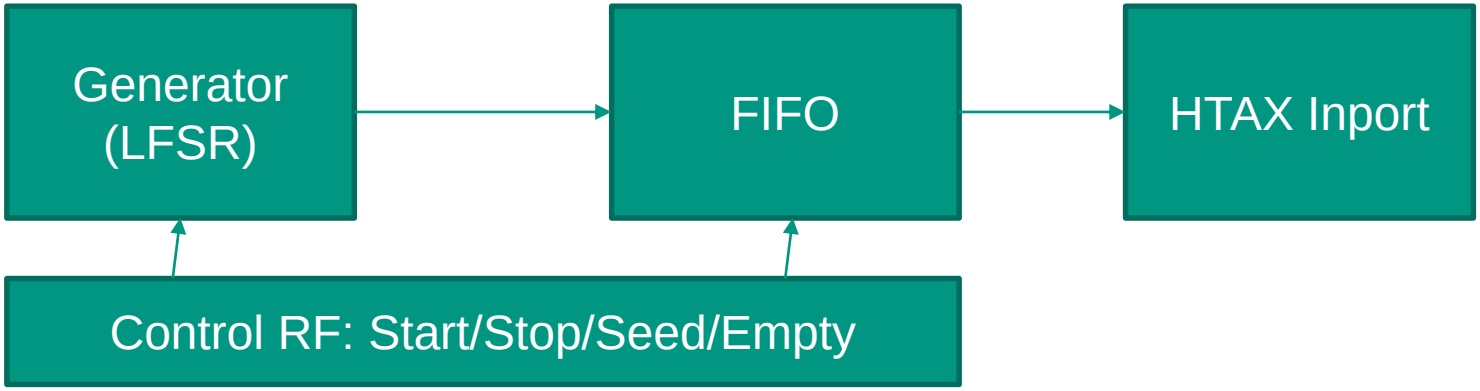
Component: Configuration RF

- Control registers are specified in a TCL script
- Address map and Verilog is generated
- Software sees all registers as an addressed array
- Generator supports more features:
  - Special components
  - Hierarchies
- Slow control:
  - I2C or something else provided



Component: Data Generator

- Make a simple data generator
- Store data in a FIFO
- Sent through HTAX
- Other suggestion?
- We can provide more or less code



Tools: Cadence RTL Compiler and Encounter

- Synthesis:
  - Cadence RTL Compiler
- Place and route
  - Cadence First Encounter
  - Also called during physical synthesis
- Unlike Analog Design tools:
  - Write TCL scripts
  - TCL contains function calls for various steps:
    - read_design
    - elaborate
    - read_constraints
    - synthesize –to_placed
- Some Support software can be used in the tools directly
- GUI View:
  - See the results of scripts mostly
  - Do Timing debug

Discussion

Discussion

Thanks…and HiWi offer ☺

- Some help on supporting tools and designs would be welcome
- Mostly TCL and Scala (Java) programming
- Tools are:
  - FSM designer
  - Design Data generators
  - Script helpers to help fast design analysis
  - Top-level planning helpers
  - Other good ideas

End Slide!

Content

- Top Level ASIC view
  - Floorplan, partitions
  - Mixed Systems
    - Analog for discrete signals needs
    - ADC -> Digital
  - I/O : Area IO, Boundary I/O
- Semi-custom Flow overview
  - HDL Code – RTL level
  - Not So different from FPGA in the end
  - Synthesis/Map
    - IP Blocks / Std. Cell
    - Physical Synthesis (gate/wire delay)
  - Place and Route
    - Die Setup
    - Place
    - Power
    - Clock tree etc..
- System Level Design
  - Design Tools
    - FSM
    - Register File
  - Advanced Abstract Modeling?
    - Software Assisted Design?
  - Verification
    - Coverage
    - Assertions
    - System Level Verification: UVM (System Verilog/E-Language)
- Lab Work Plan
  - What should we do there?

Prof. Ivan Peric – Design Digitaler

Prof. Ivan Peric – Design Digitaler