UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

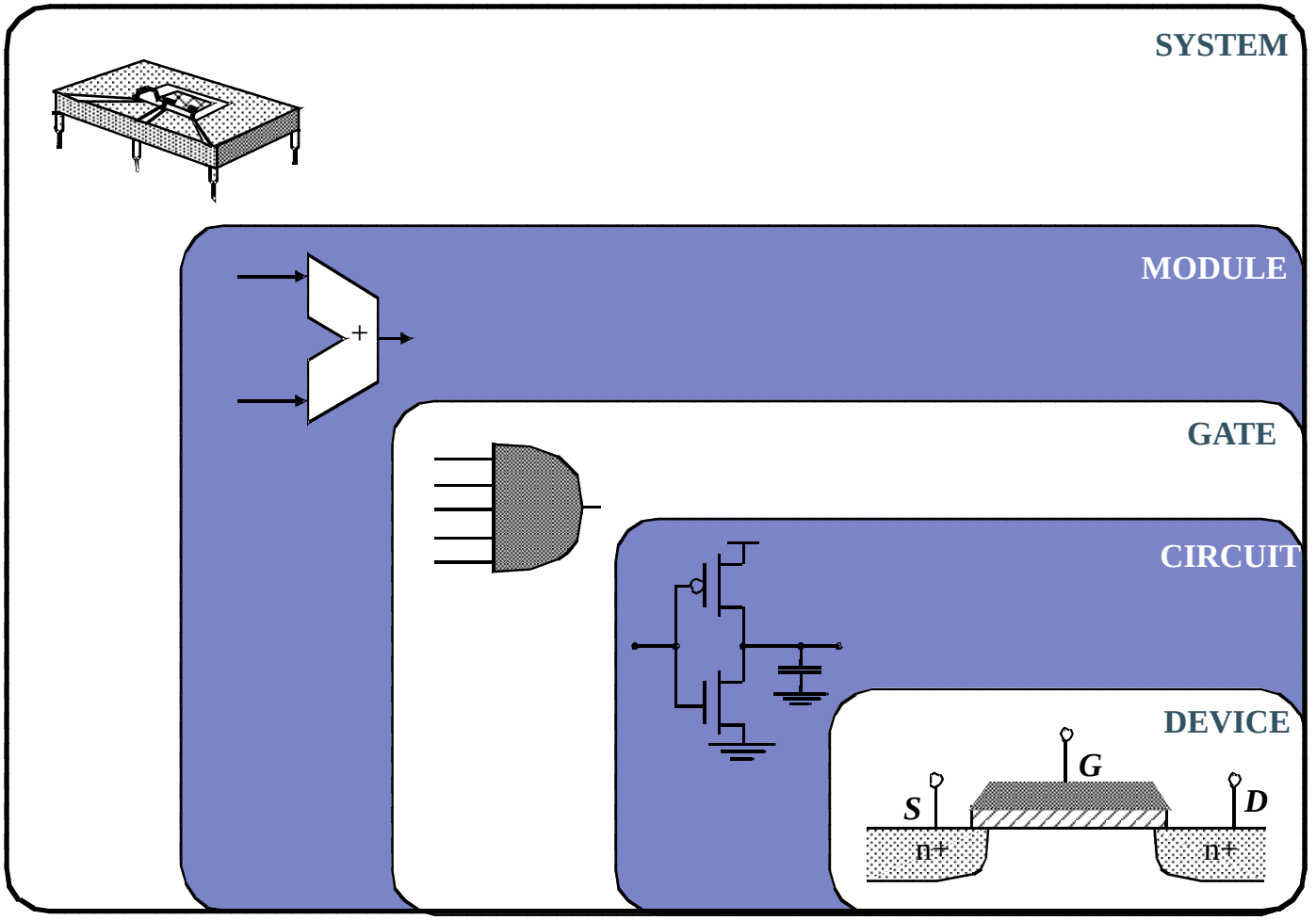
Processor Design

José María Arnau

Technology Scaling

- Technology shrinks 0.7x per generation
  - 2x more functions per chip on each new generation
- However, engineering population does not double every two years
- We need better and more efficient design techniques
  - Take advantage of abstraction levels

Design Abstraction Levels



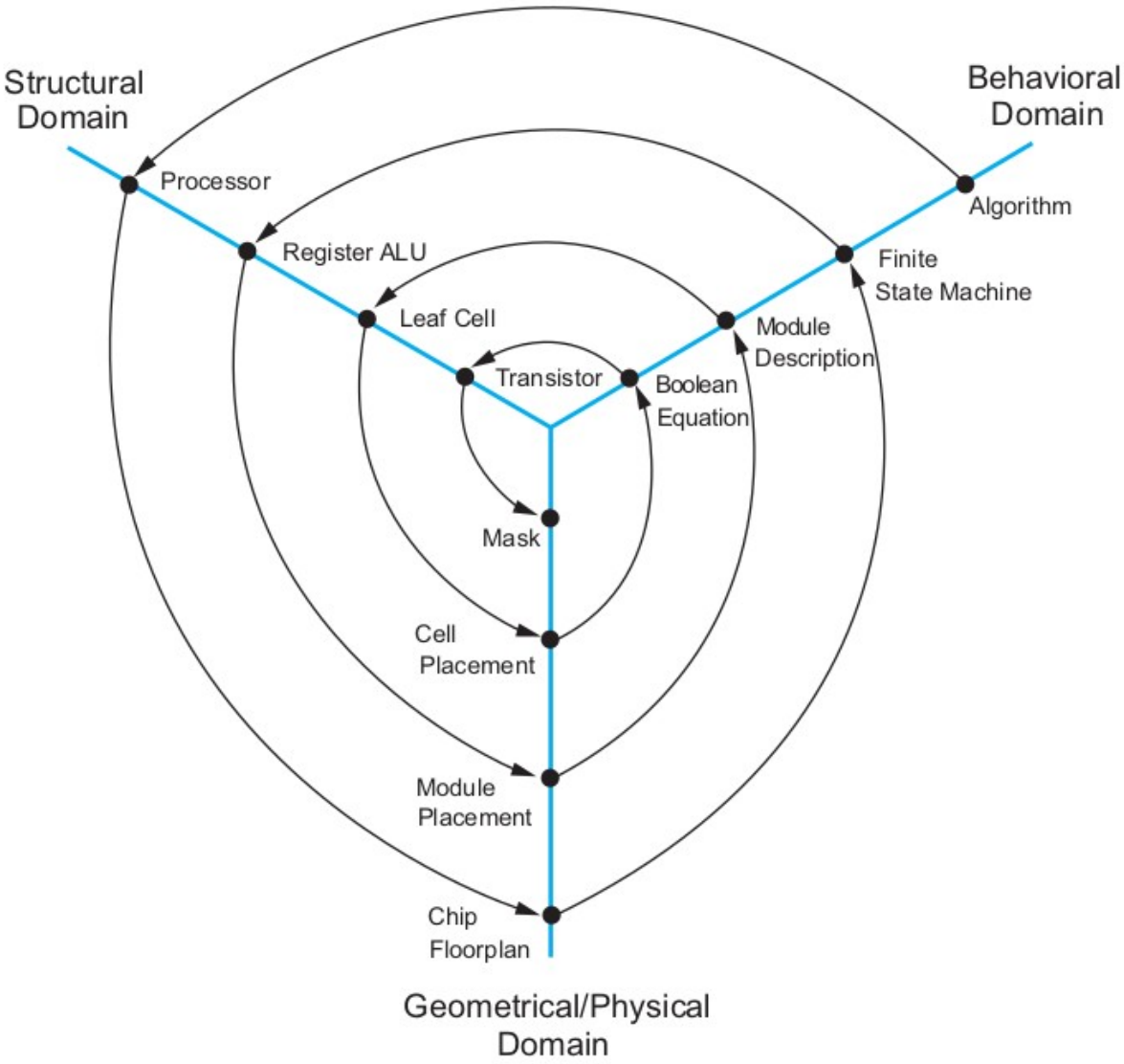
VLSI Design Tools

- VLSI system design is a complex process
  - Requires automatization of the synthesis cycle
- EDA tools:
  - Synthesis
    - Convert HDL to phyisical representation
  - Analysis and Verification
    - Guarantee correctness of the design
  - Testing
    - Check errors in the fabrication process

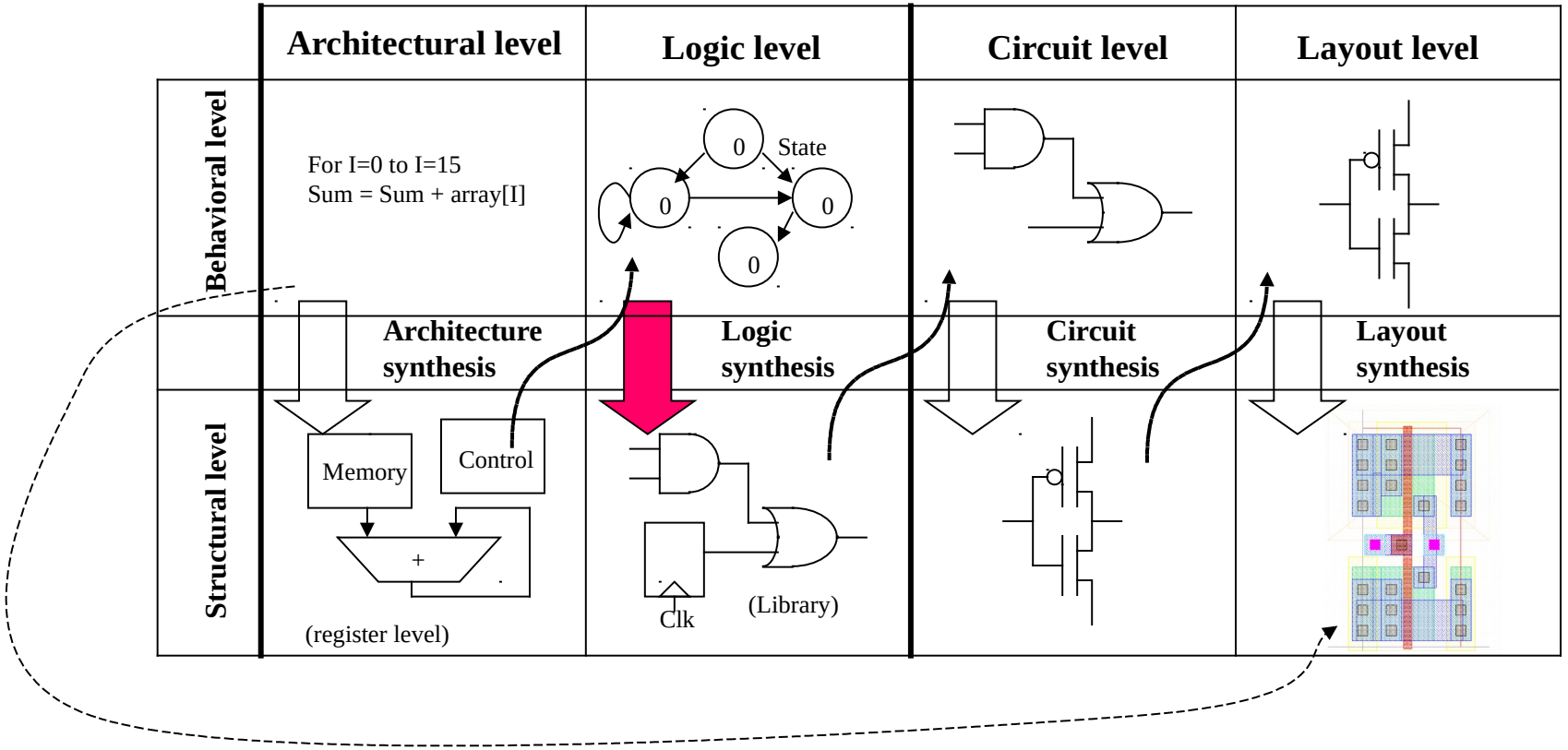
VLSI Design Tools

- Synopsys
  - Design Compiler, IC Compiler, VCS...
- Cadence
  - Virtuoso, Incisive, SoC encounter...
- Mentor
  - Questa/ModelSim, Calibre, Nitro-SoC...
- Qflow (Free and Open Source)
  - Yosys, Graywolf, QRouter....

Design Domains (Gajski i Kuhn)



Abstraction Levels and Synthesis

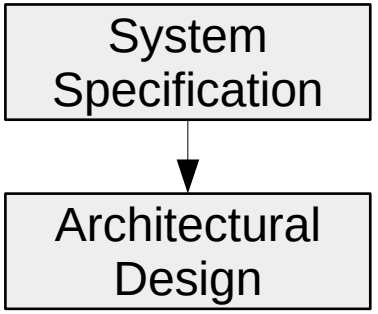


Silicon compilation (not a big success)

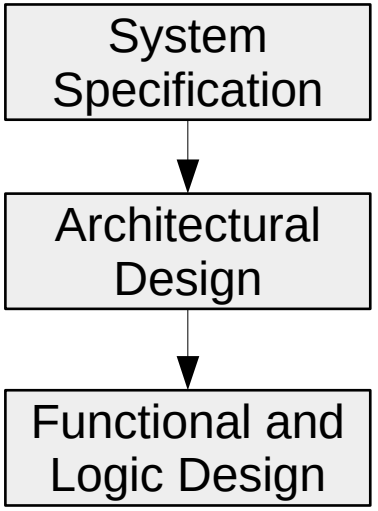
VLSI Design Flow

System Specification

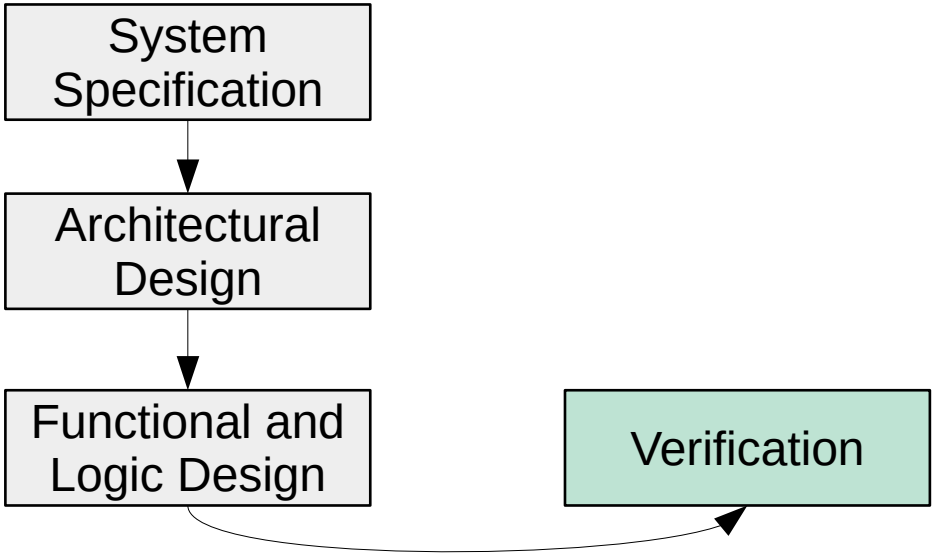
VLSI Design Flow



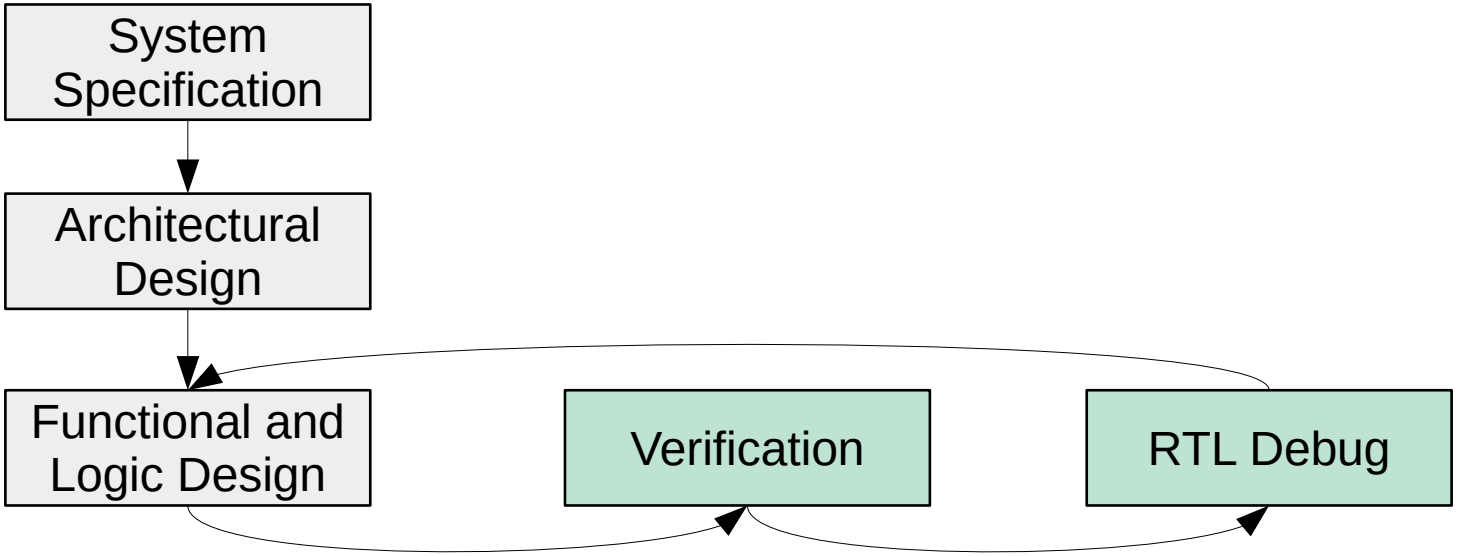
VLSI Design Flow



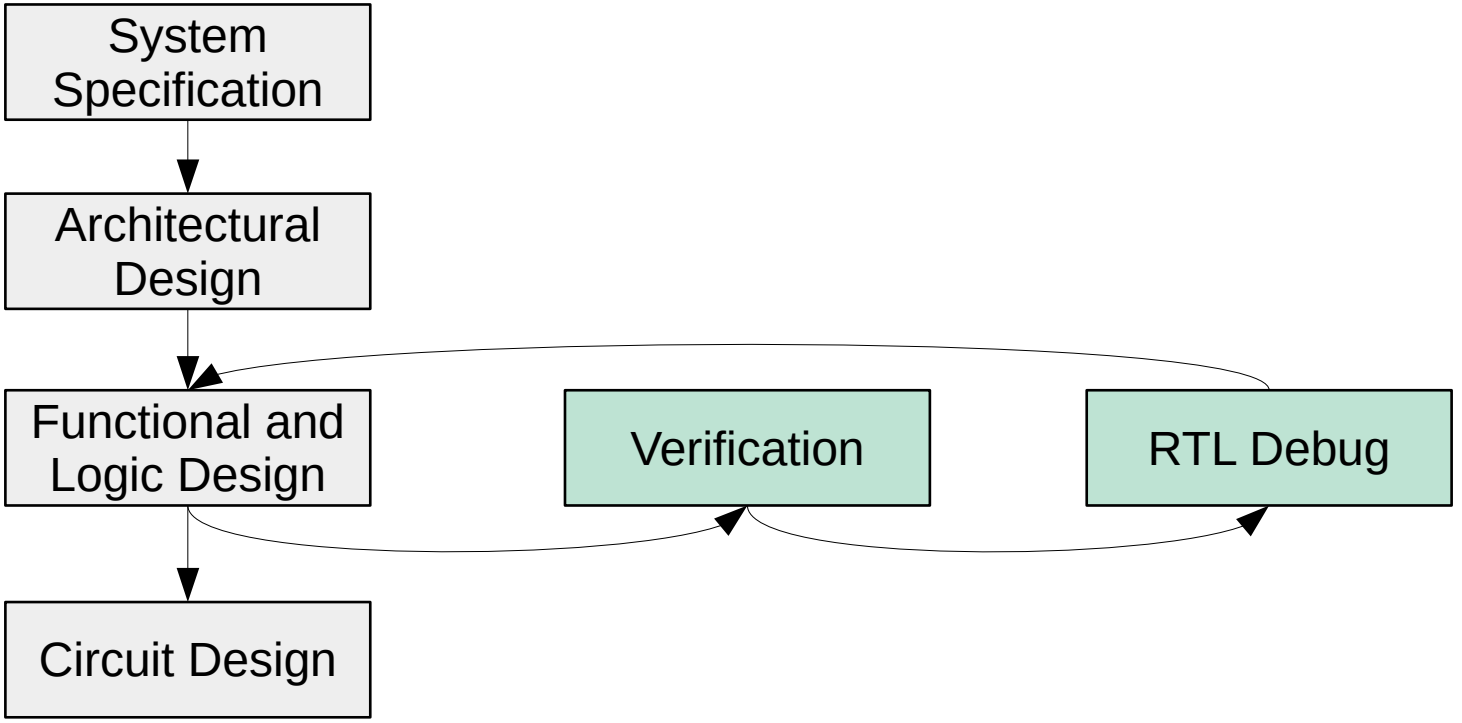
VLSI Design Flow



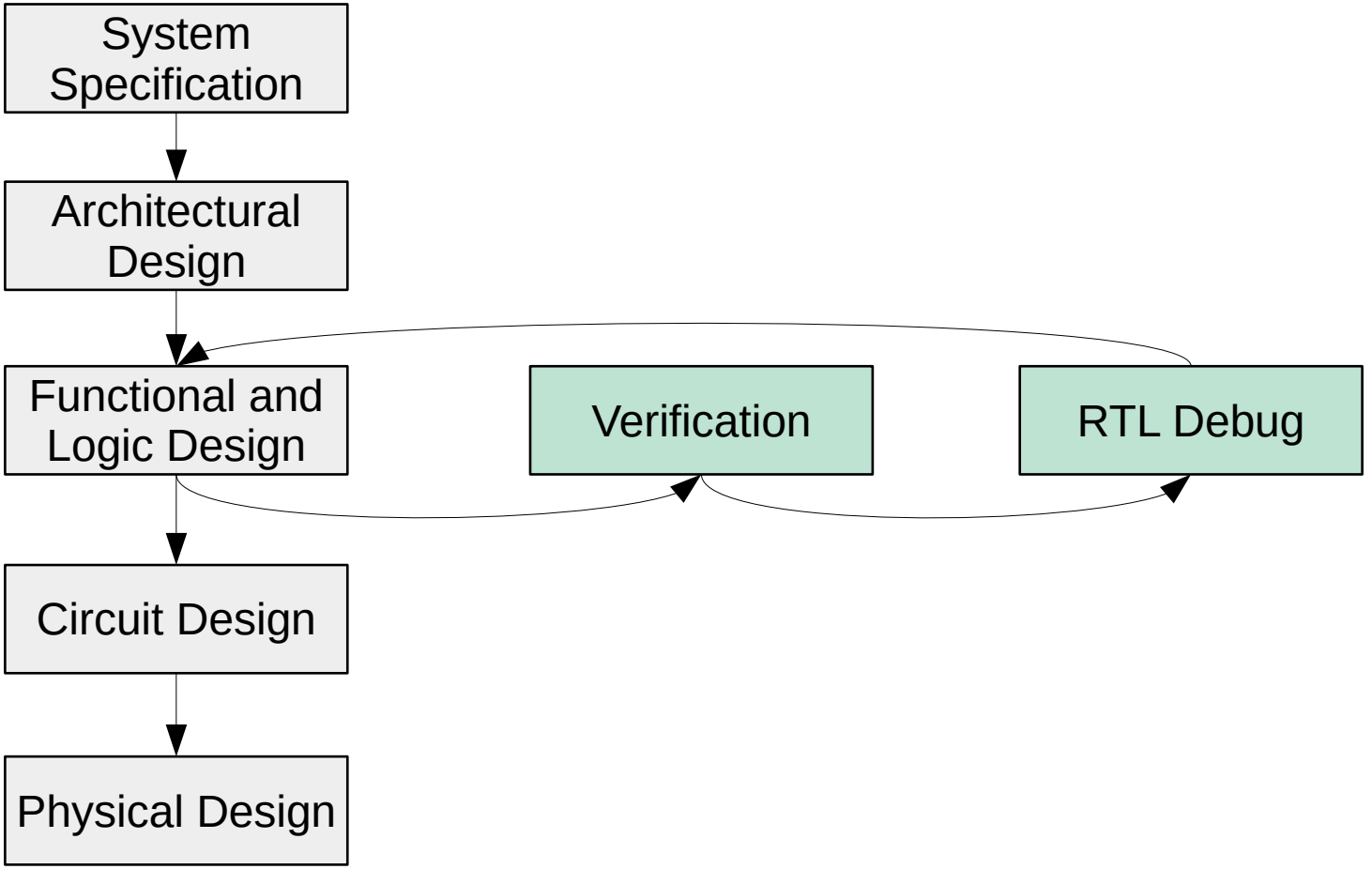
VLSI Design Flow



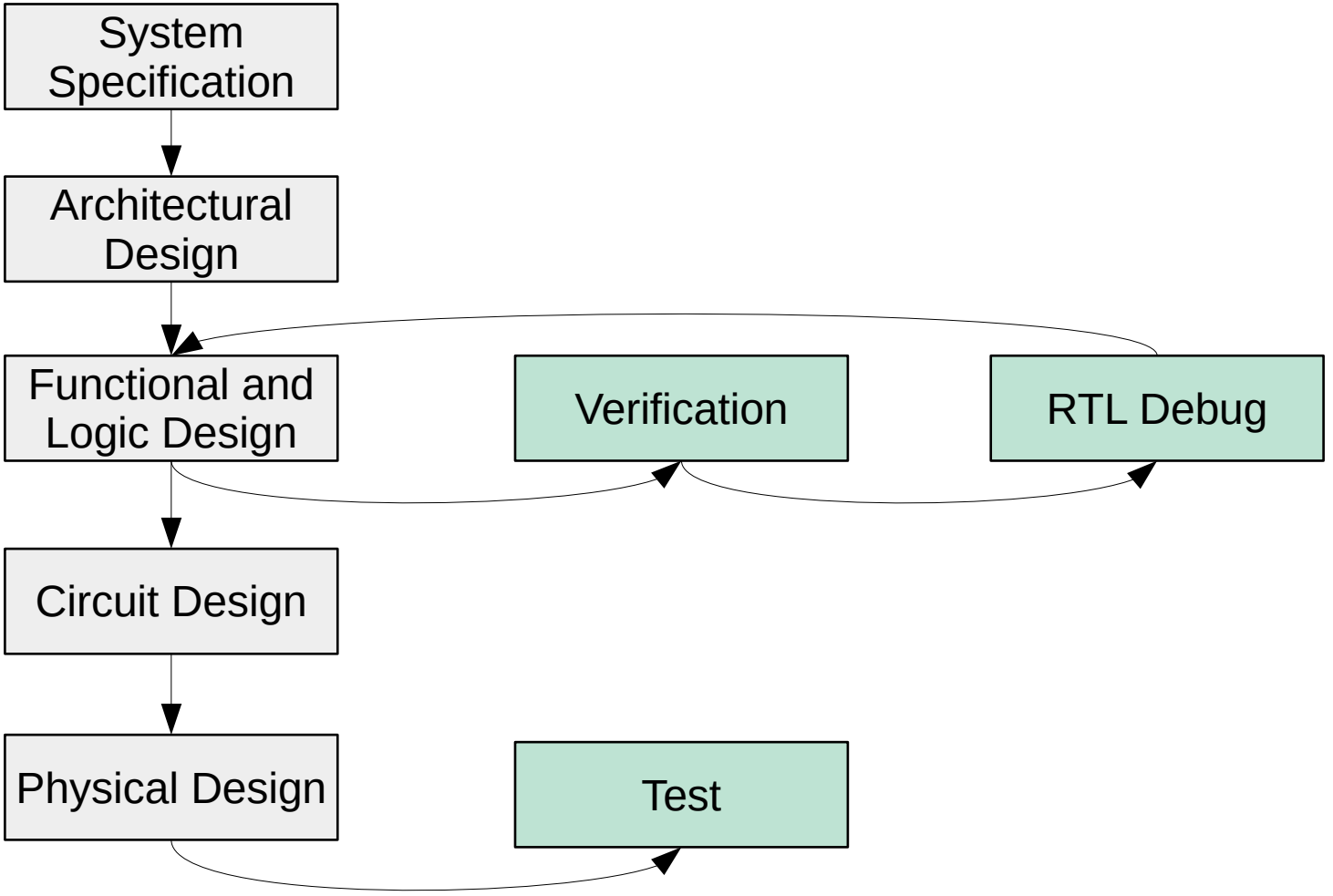
VLSI Design Flow



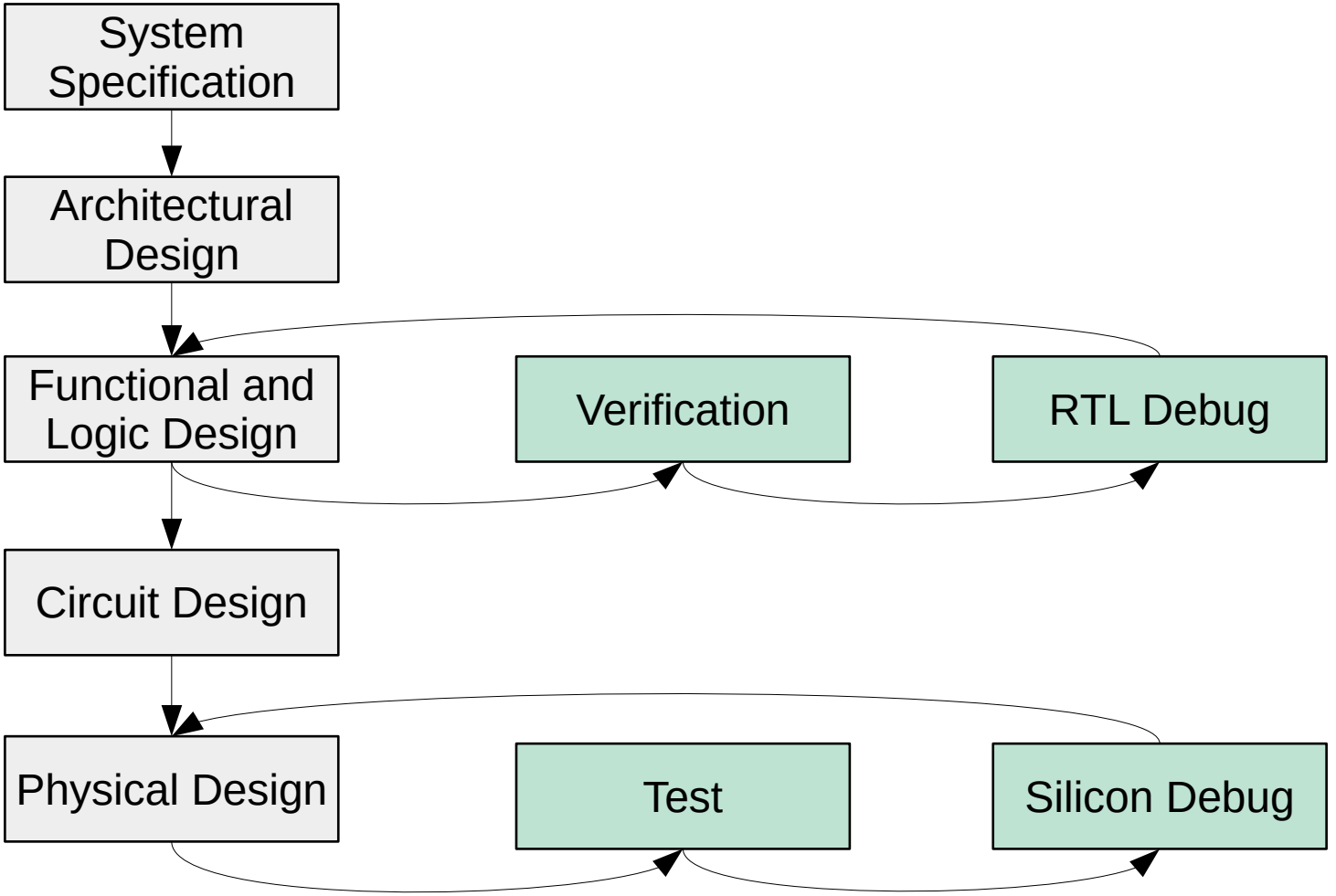
VLSI Design Flow



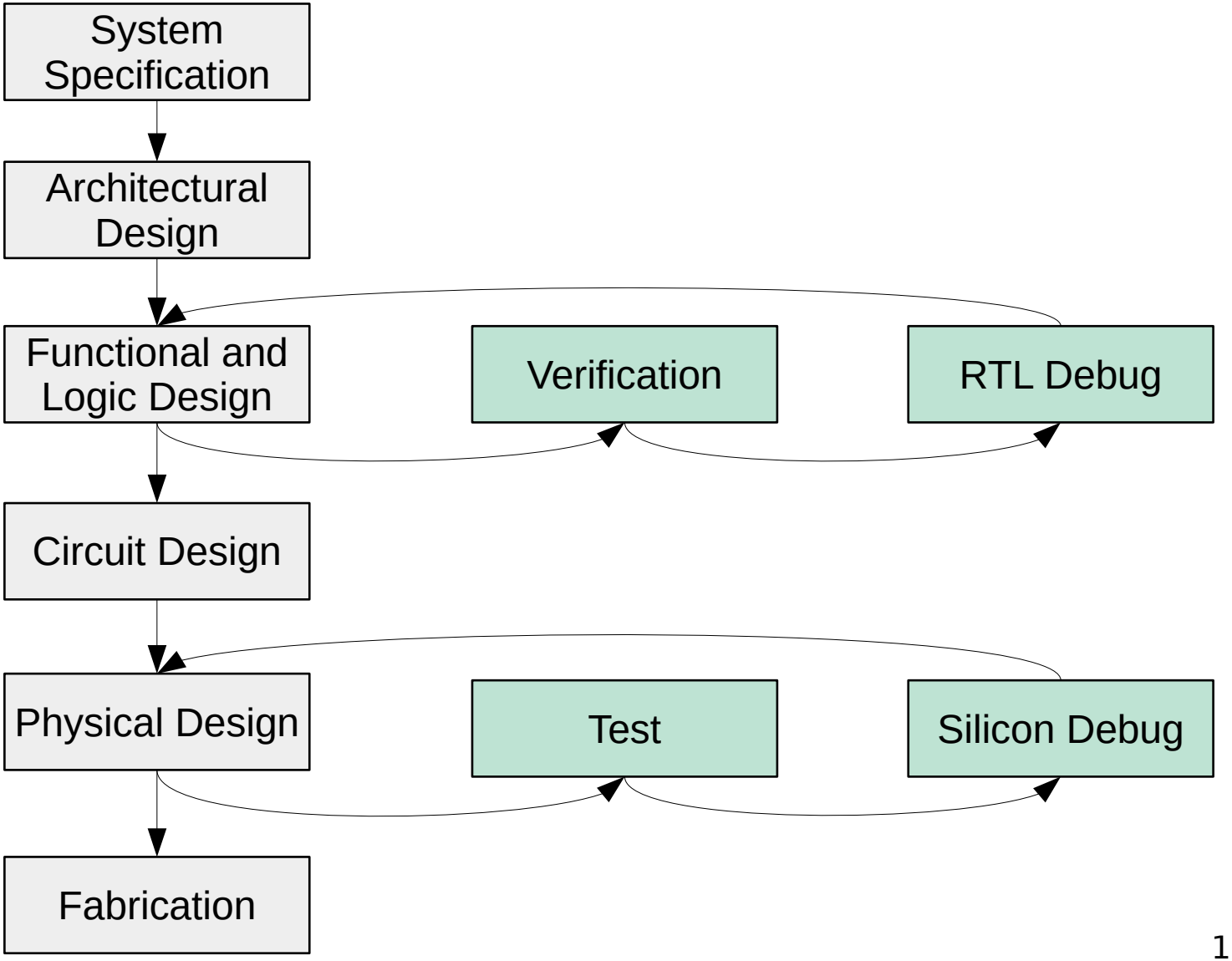
VLSI Design Flow



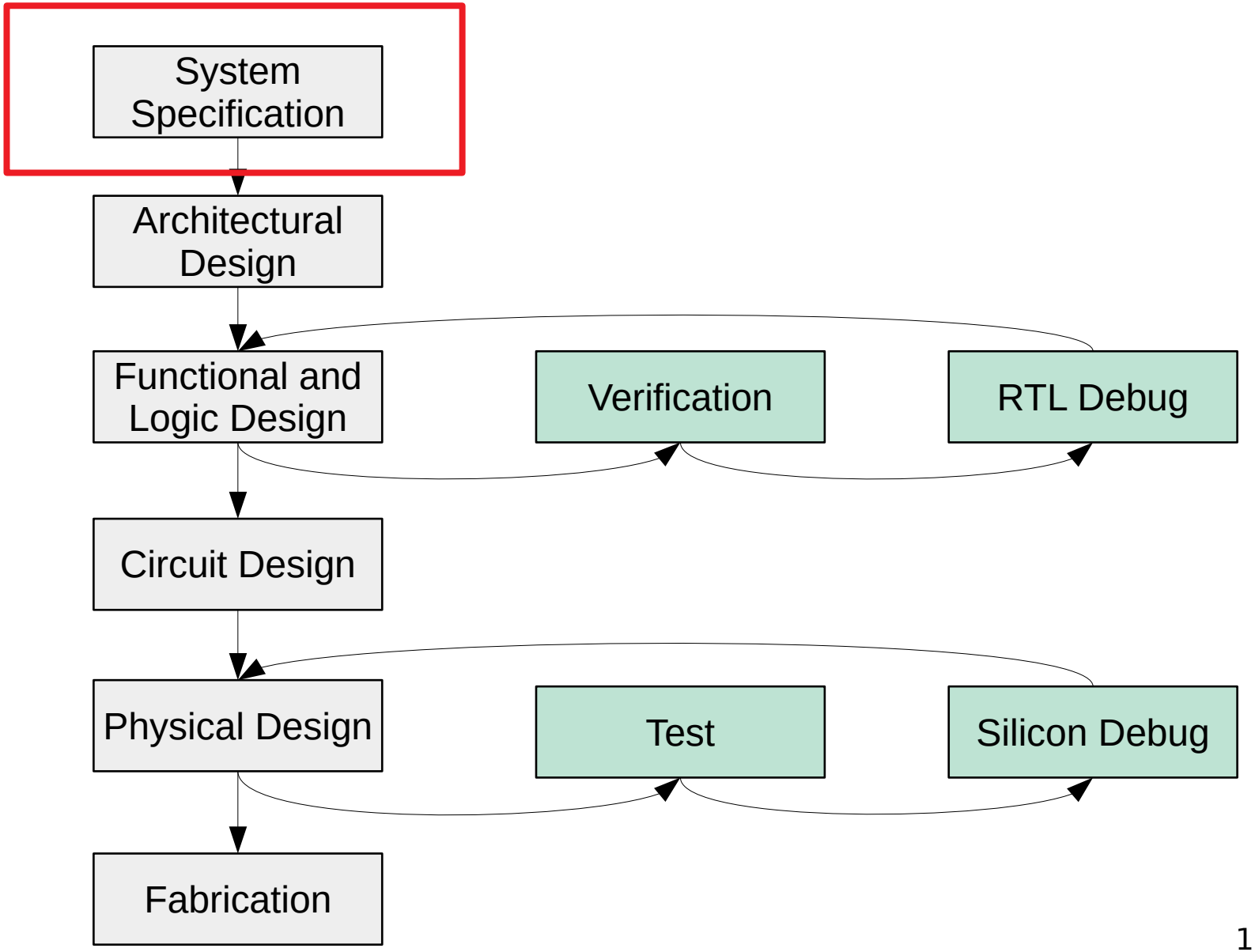
VLSI Design Flow



VLSI Design Flow



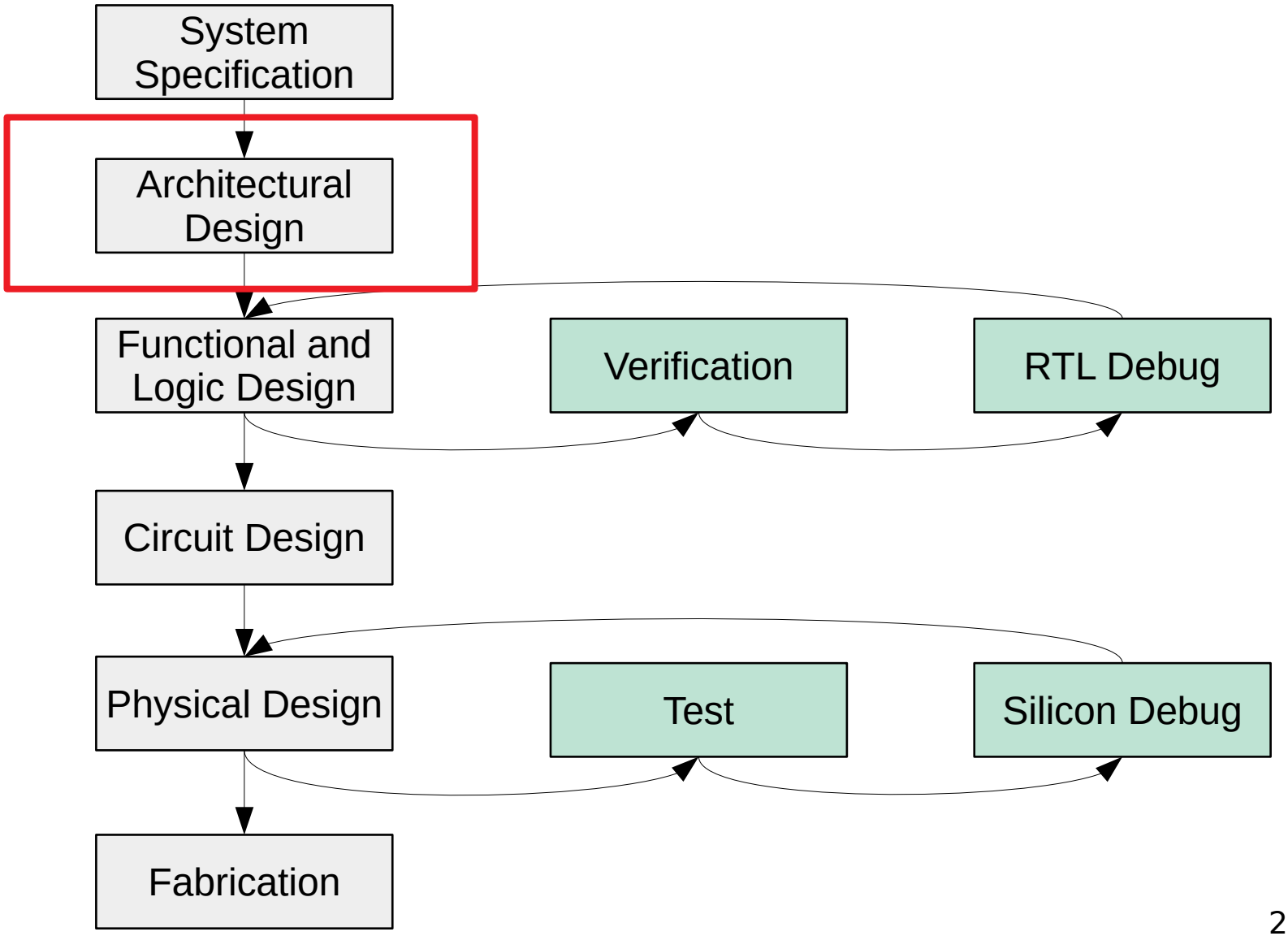
VLSI Design Flow



System Specification

- Define overall goals and requirements
  - Functionality
  - Performance/Power/Area
- Format
  - Textual description (English)
  - Flowcharts
  - State transition graphs
  - Timing charts
  - Executable specification (SystemC)

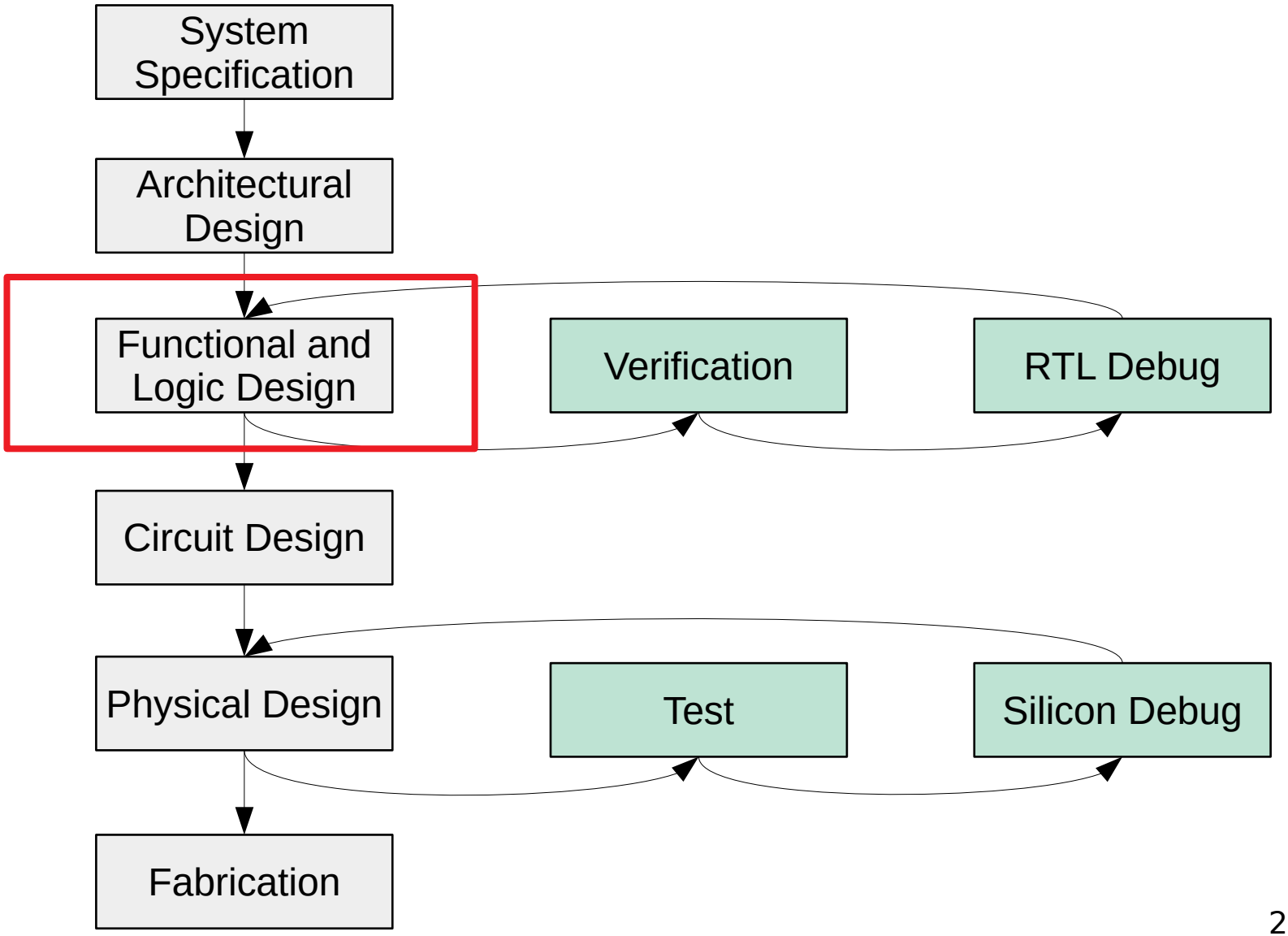
VLSI Design Flow



Architectural Design

- Determine the architecture to meet the specifications
  - Microarchitecture details
    - Branch predictor
    - Caches
    - Datapath and Control Unit
  - Usage of hard IP blocks
  - Choice of process technology
- Simulators used to validate architecture
  - Check functional correctness and perform DSE

VLSI Design Flow



Functional and Logic Design

- System described using HDL
  - Programming language to describe hardware
  - Behavioral description
  - Register-Transfer Level (RTL) description
- Two most common HDLs
  - VHDL
  - Verilog

Why HDL?

- Allows write-run-debug cycle for hardware development
  - Similar to software development
- Higher productivity than using schematics
  - Easier to describe complex circuits with millions of gates
- Input to synthesis EDA tools
  - Only the synthesizable subset
- Design space exploration with simulation

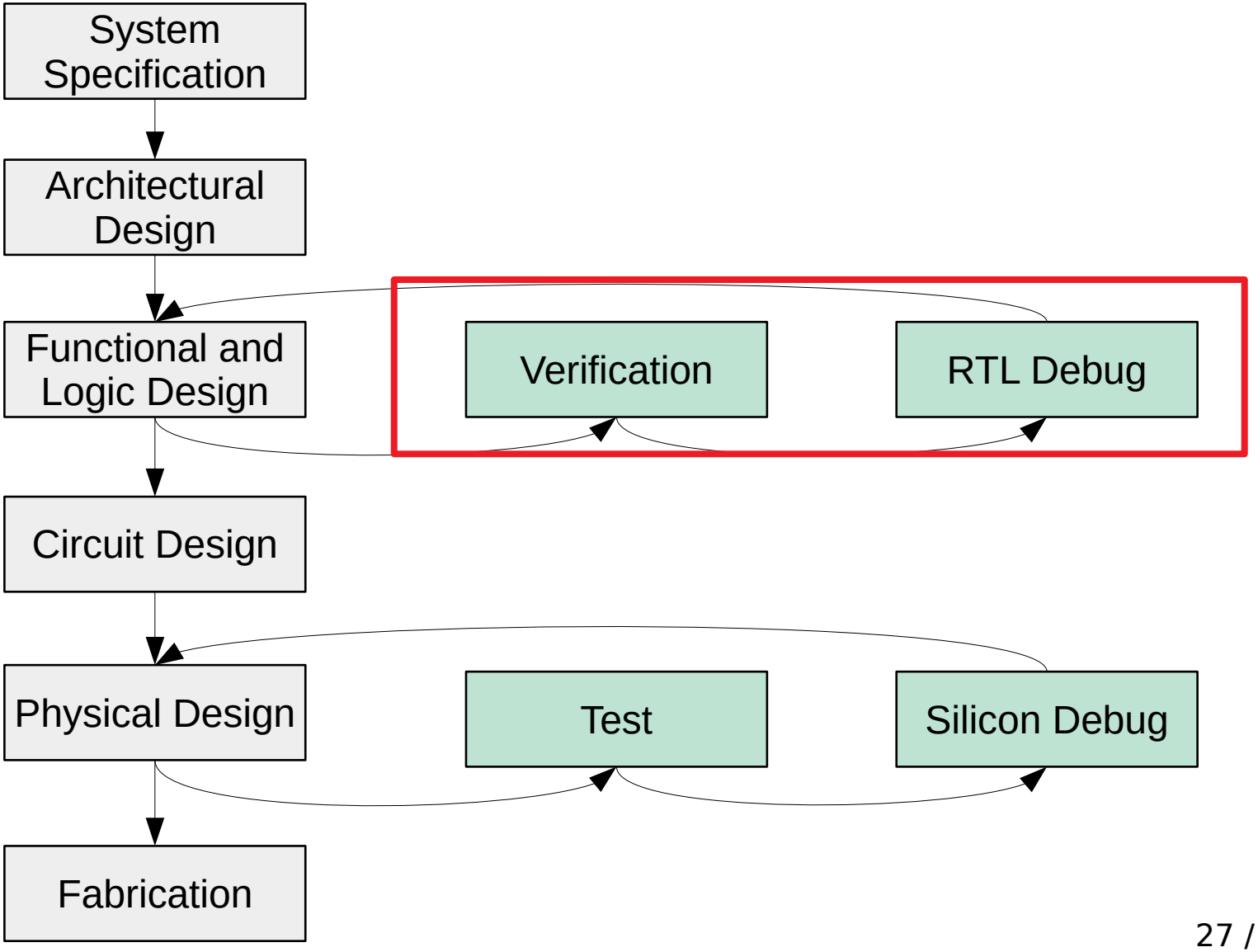
Why HDL?

- Why not use a general purpose language?
  - Support for structure and **instantiation**
  - Support for describing **bit-level** behavior
  - Support for **timing**
  - Support for **concurrency**

HDLs

- Verilog
  - Close to C
  - More than 60% of the world digital design market (larger share in US)
  - IEEE standard since 1995
  - Extensive support for Verification (SystemVerilog)
- VHDL
  - Close to Ada
  - Large standard developed by the US DoD

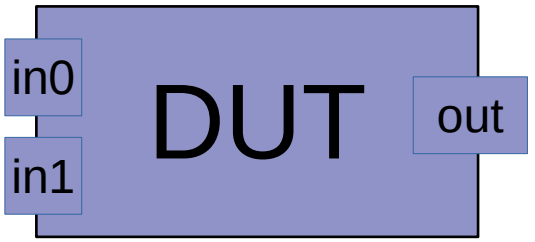
VLSI Design Flow



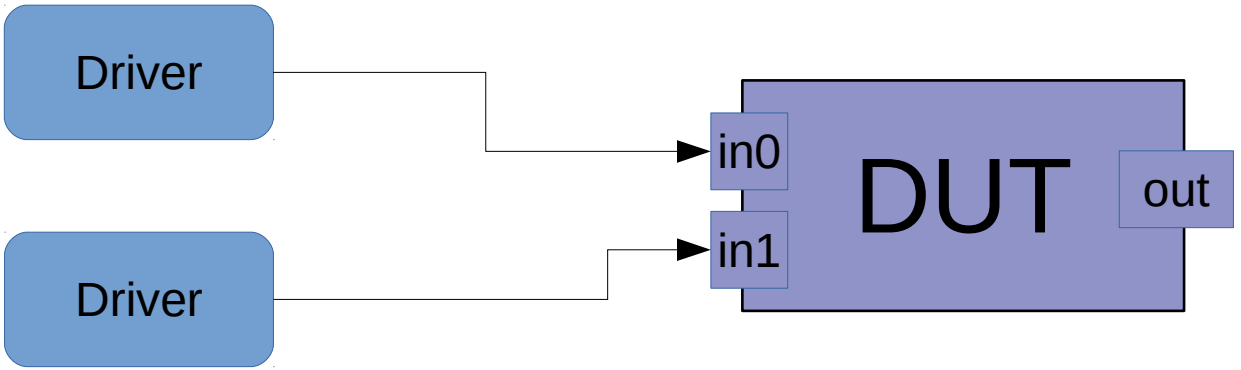
Hardware Verification

- Verify the functional correctness of the design
- Verification methodology
  - Prepare verification plan from specification
  - Prepare testbenches
    - Instantiate Design Under Test (DUT)
    - Provide stimuli to the DUT
    - Check DUT’s output
  - Run testbenches using HDL simulator
    - Verilog: icarus verilog, verilator
    - VHDL: GHDL
    - VCS, ModelSim

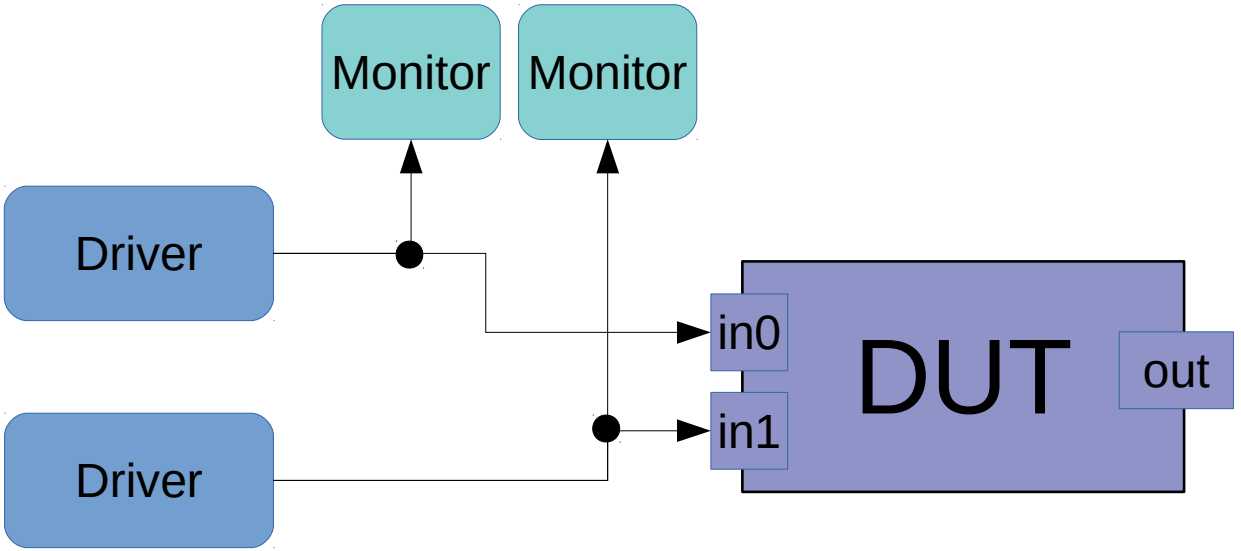
Testbench Architecture



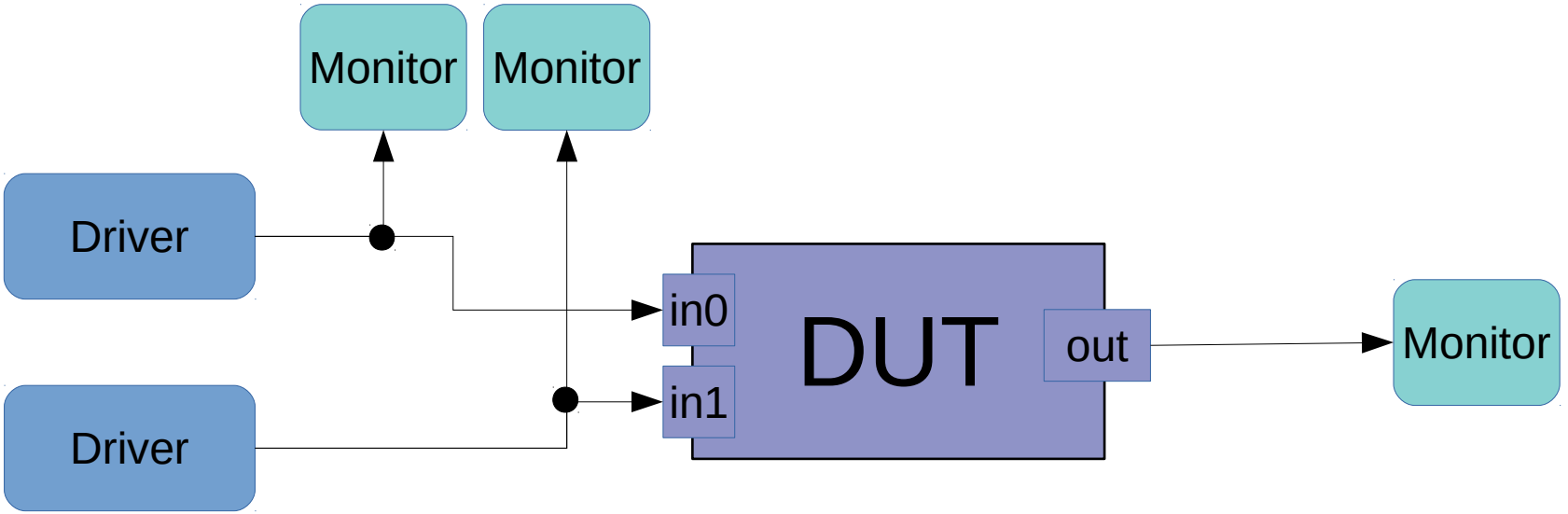
Testbench Architecture



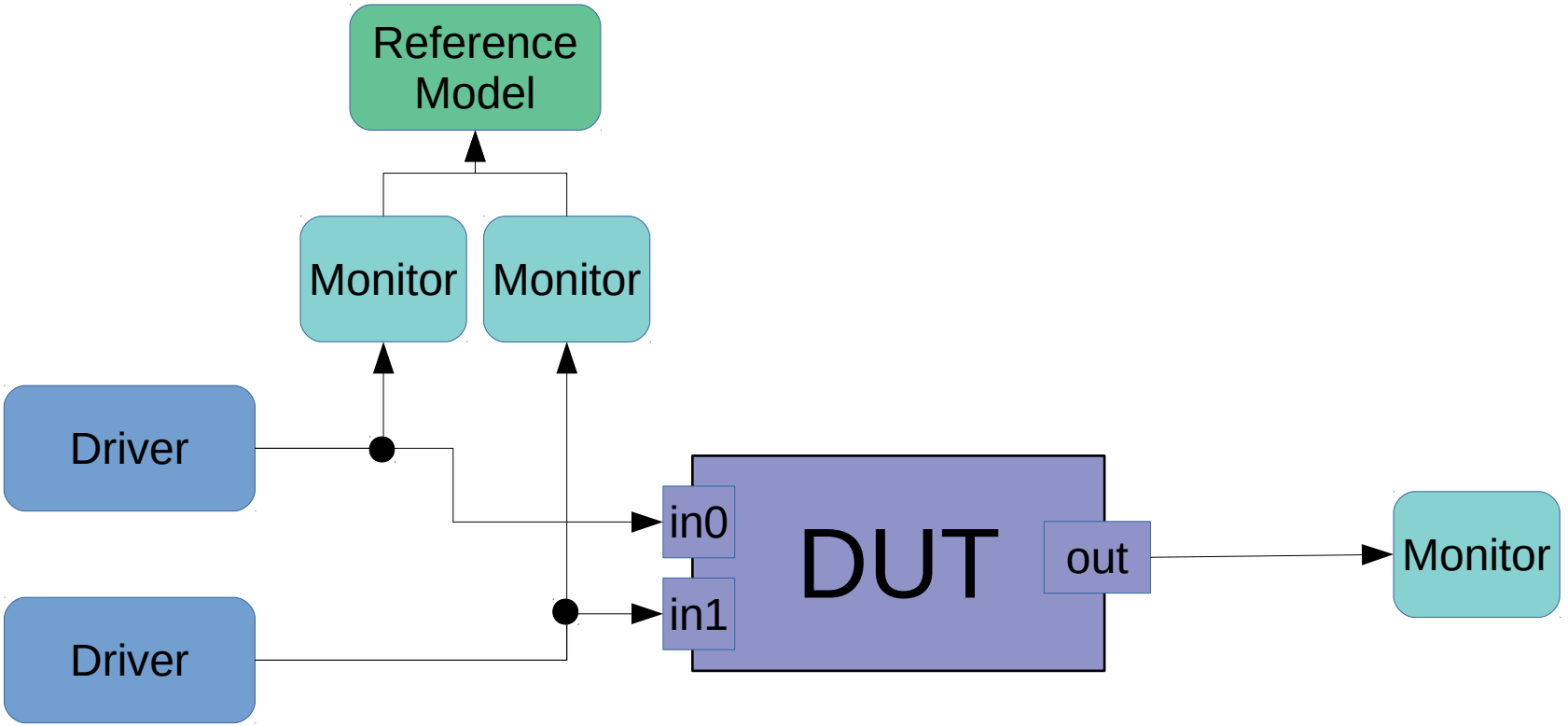
Testbench Architecture



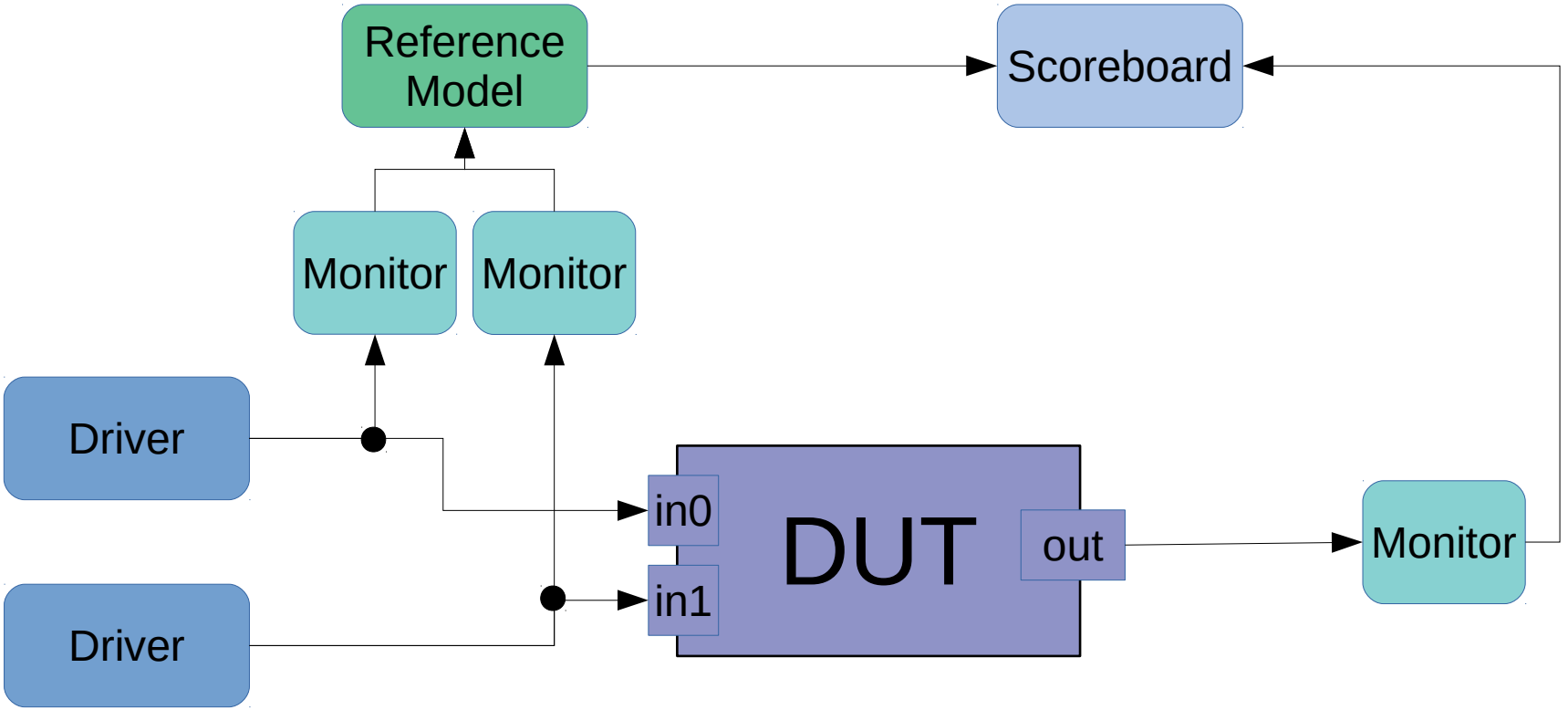
Testbench Architecture



Testbench Architecture



Testbench Architecture



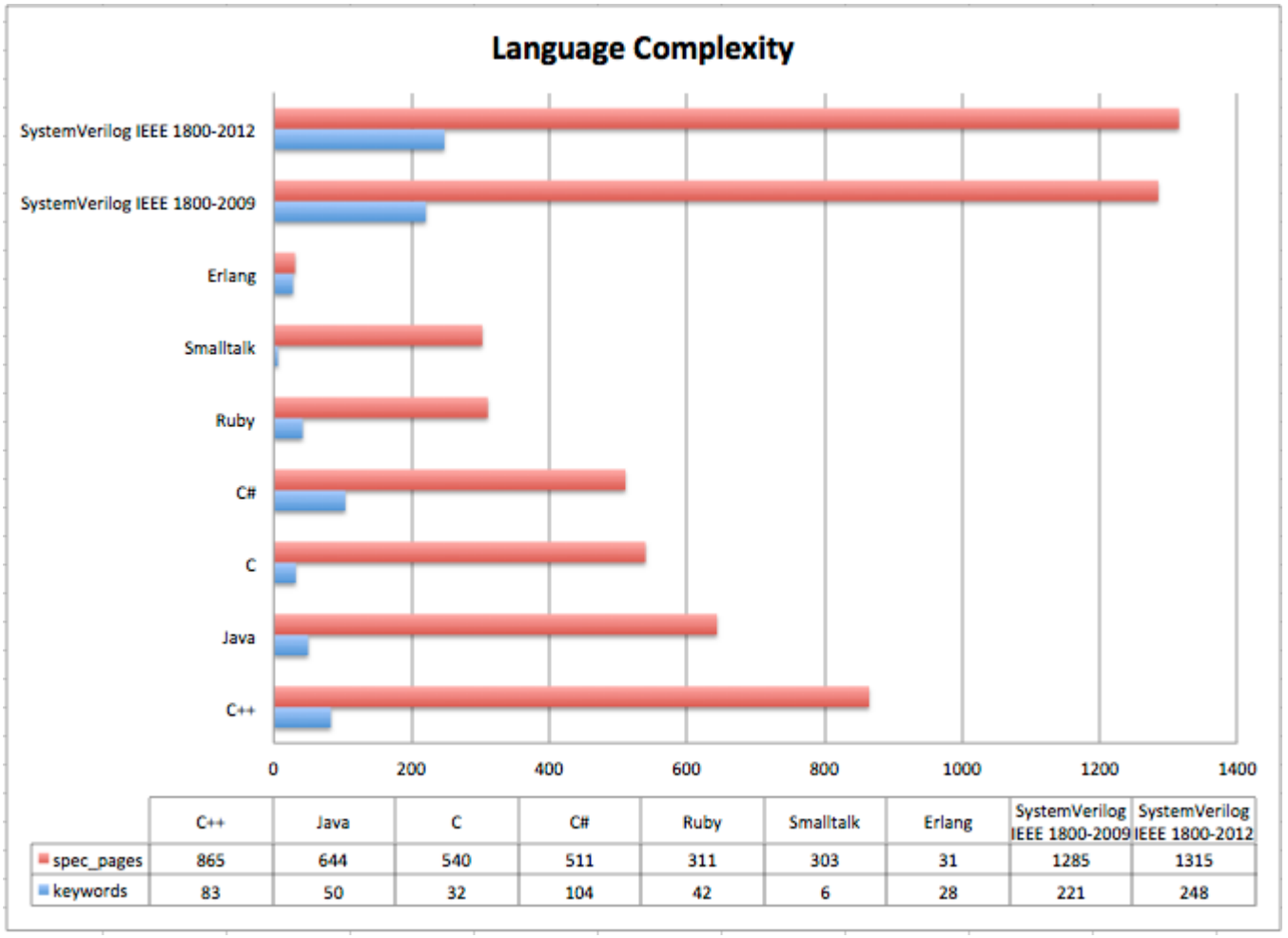
Hardware Verification

- Directed tests
  - “Manually” prepare input stimuli (test vectors)
  - Only find bugs you are looking for
- Constrained Random Verification
  - Randomize input stimuli with restrictions
  - How do we know which functionality has been tested?
- Coverage
  - Code coverage
  - Functional coverage

Verification tools

- Universal Verification Methodology (UVM)
  - Class library written in SystemVerilog
  - Very powerful, but very complex
    - Over 300 classes in UVM!
  - Grad students unlikely to have prior experience with SystemVerilog
- Open Source VHDL Verification Methodology (OSVVM)
  - Library written in VHDL
  - Similar to UVM

SystemVerilog Complexity

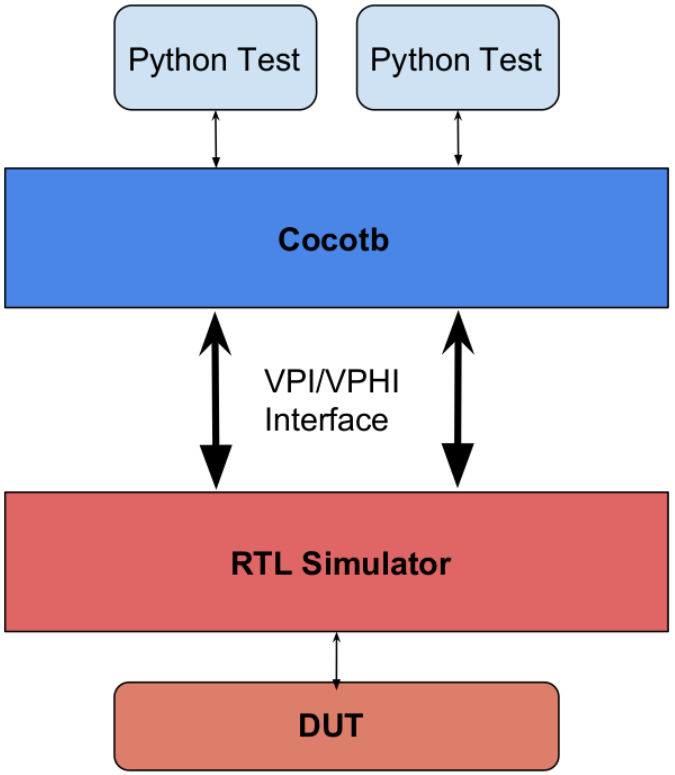


	C++	Java	C	C#	Ruby	Smalltalk	Erlang	SystemVerilog IEEE 1800-2009	SystemVerilog IEEE 1800-2012
spec_pages	865	644	540	511	311	303	31	1285	1315
keywords	83	50	32	104	42	6	28	221	248

http://www.fivecomputers.com/language-specification-length.html

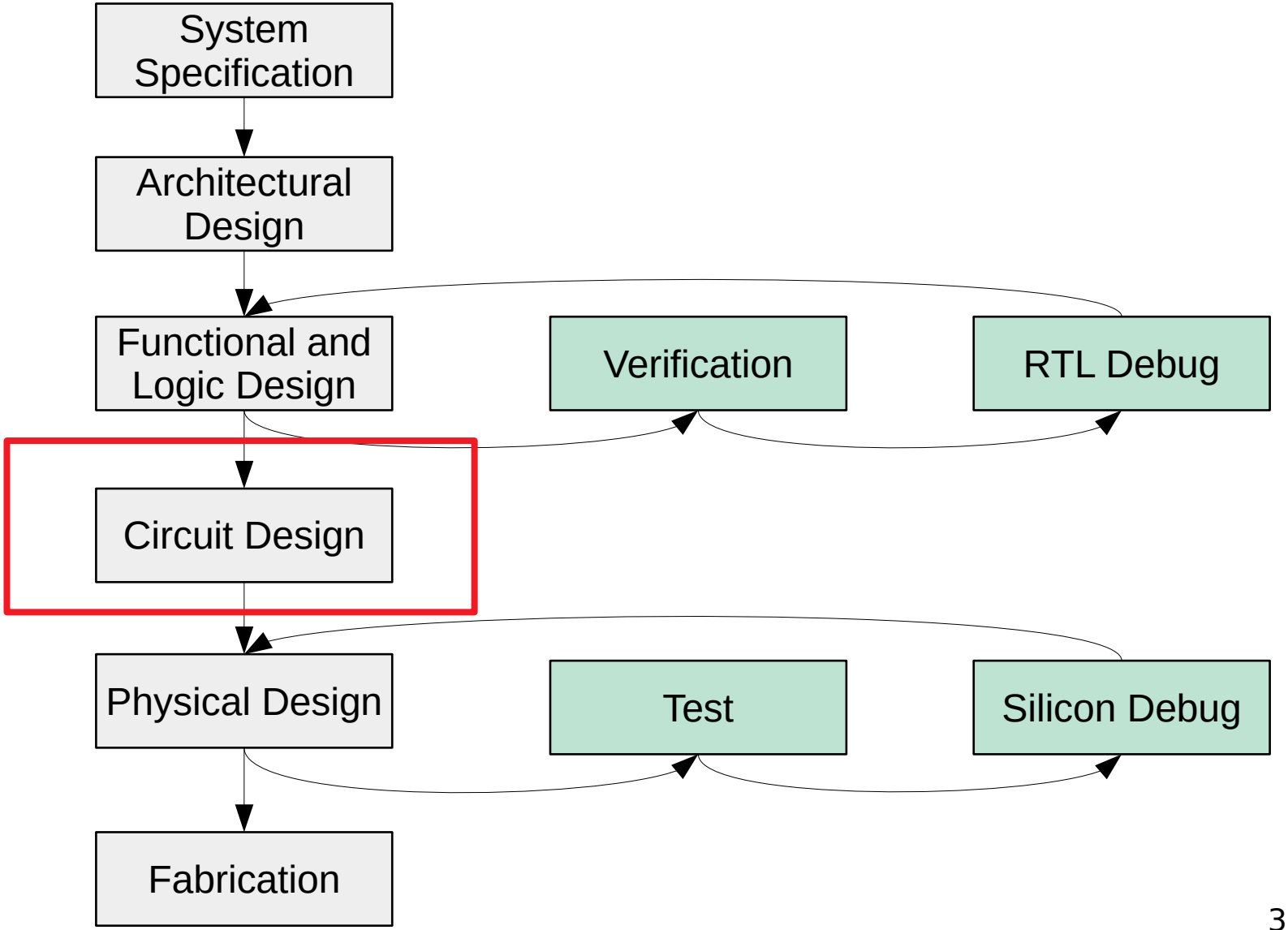
Cocotb

- Coroutine Cosimulation TestBench
  - Write testbenches in Python!
  - VHDL and Verilog
  - Interface to RTL simulators
    - Icarus, GHDL, ModelSim, VCS...
- Cocotb-coverage
  - Constrained Random Verification
  - Functional Coverage
  - Regression Tests



https://github.com/cocotb/cocotb

VLSI Design Flow



Logic Synthesis

- Converts RTL description to gate-level netlist
  - Automatically performed by EDA tool
    - Synopsys Design Compiler
    - Yosys
  - Standard cell library
    - Collection of low-level electronic logic functions
      - AND, OR, INVERT, flip-flops, latches...
      - Layout, schematic, timing, power...
    - Process Development Kit (PDK)

Standard Cell Design

- Design circuits using standard cells
  - Cells: gates, latches...
- Technology mapping selects cells
- EDA tools
  - Place and route the cells
  - Wiring in channels
  - Minimize area, delay

Qflow - Yosys

- Framework for Verilog RTL synthesis
- Implements multiple optimization passes
- Input:
  - Synthesizable Verilog code (Verilog-2005)
  - Standard cell library
- Output:
  - Gate-level netlist: BLIF, Verilog
  - Post-synthesis verification using same testbenches

Example: 2-bit adder

- RTL code

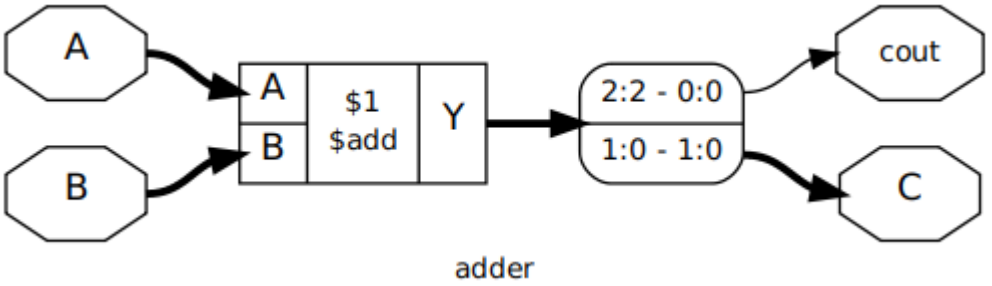
```
module adder (input wire [1:0] A,
              input wire [1:0] B,
              output wire [1:0] C,
              output wire cout);

    assign {cout, C} = A + B;

endmodule
```

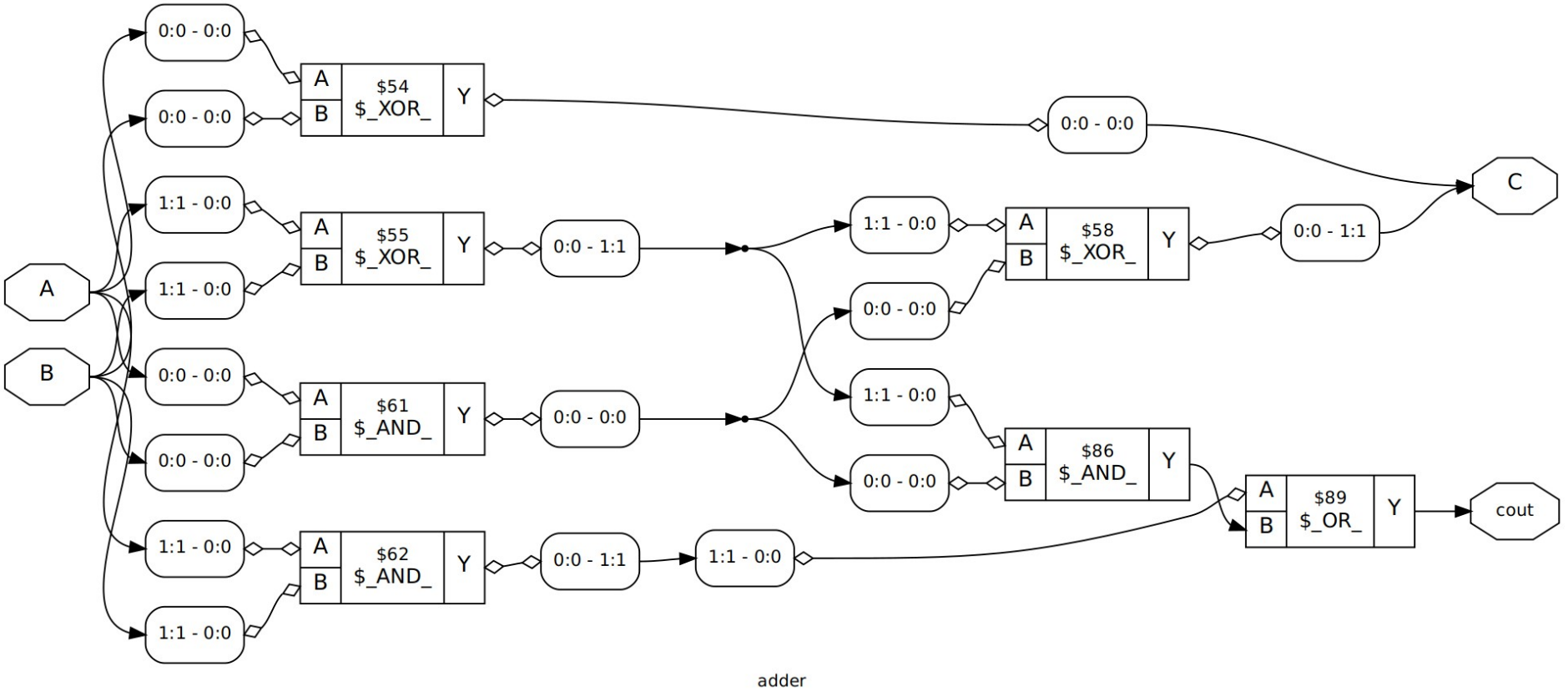
Example: 2-bit adder

- Yosys – Read and process Verilog



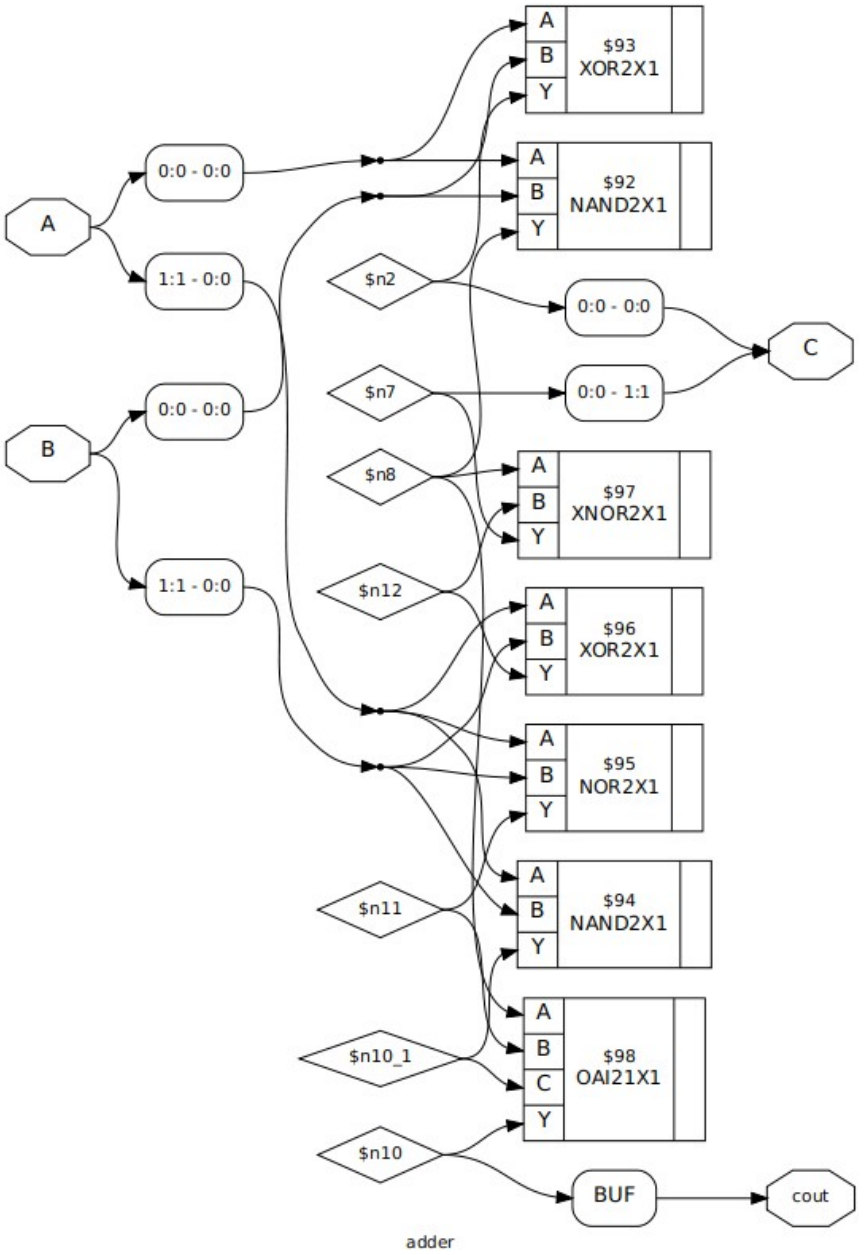
Example: 2-bit adder

- Yosys – Map to internal technology library



Example: 2-bit adder

- Yosys/ABC:
  - Map to standard cell library

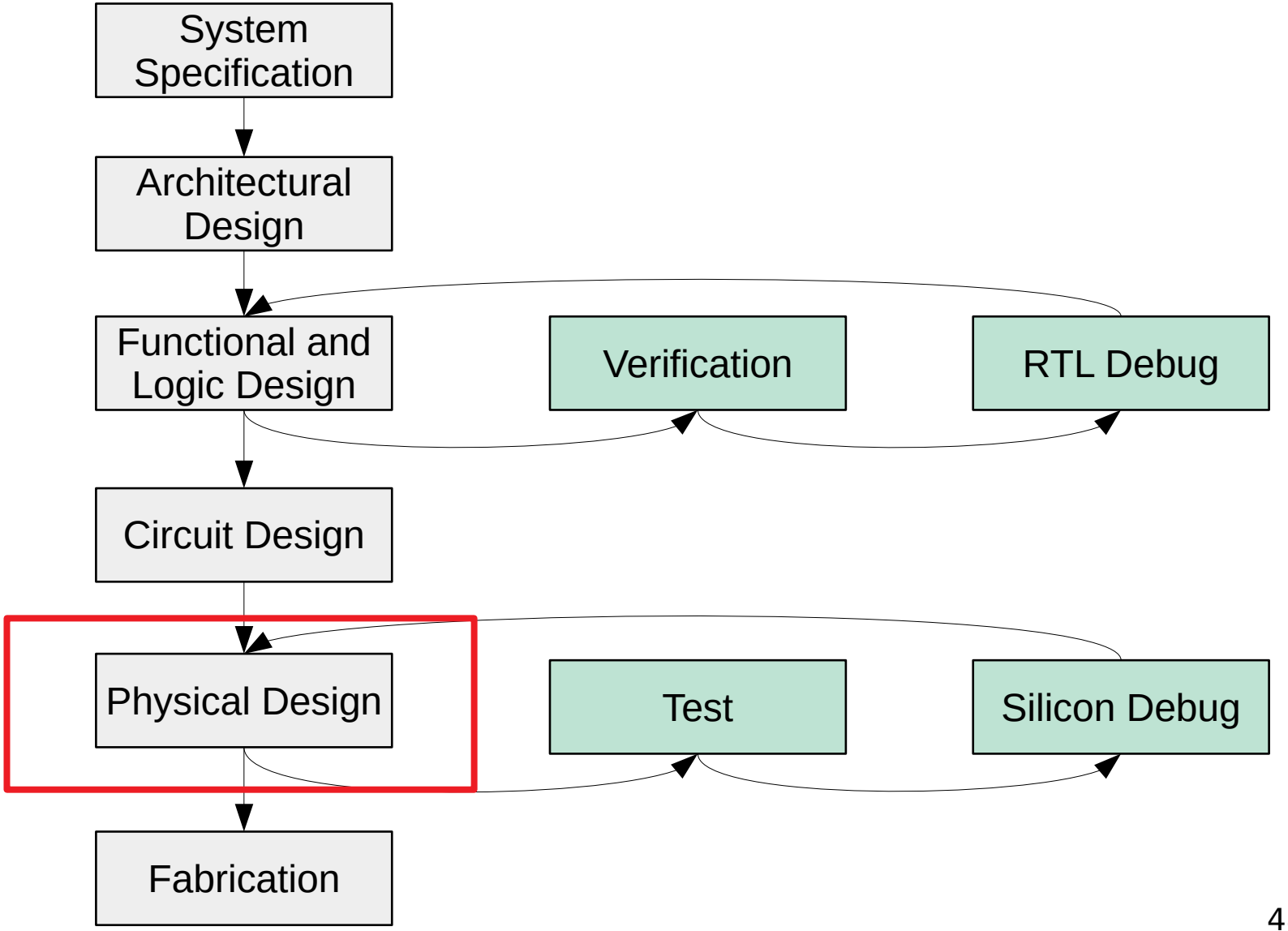


Example: 2-bit adder

- Yosys – Gate-level netlist in Verilog

```
module adder(A, B, C, cout);
  wire _00_;
  wire _01_;
  wire _02_;
  wire _03_;
  input [1:0] A;
  input [1:0] B;
  output [1:0] C;
  output cout;
  NAND2X1 _04_ (.A(A[0]), .B(B[0]), .Y(_03_) );
  XOR2X1 _05_ (.A(A[0]), .B(B[0]), .Y(C[0]) );
  NAND2X1 _06_ (.A(A[1]), .B(B[1]), .Y(_00_) );
  NOR2X1 _07_ (.A(A[1]), .B(B[1]), .Y(_01_)  );
  XOR2X1 _08_ (.A(A[1]), .B(B[1]), .Y(_02_) );
  XNOR2X1 _09_ (.A(_03_), .B(_02_), .Y(C[1]) );
  OAI21X1 _10_ (.A(_03_), .B(_01_), .C(_00_), .Y(cout) );
endmodule
```

VLSI Design Flow



Physical Design

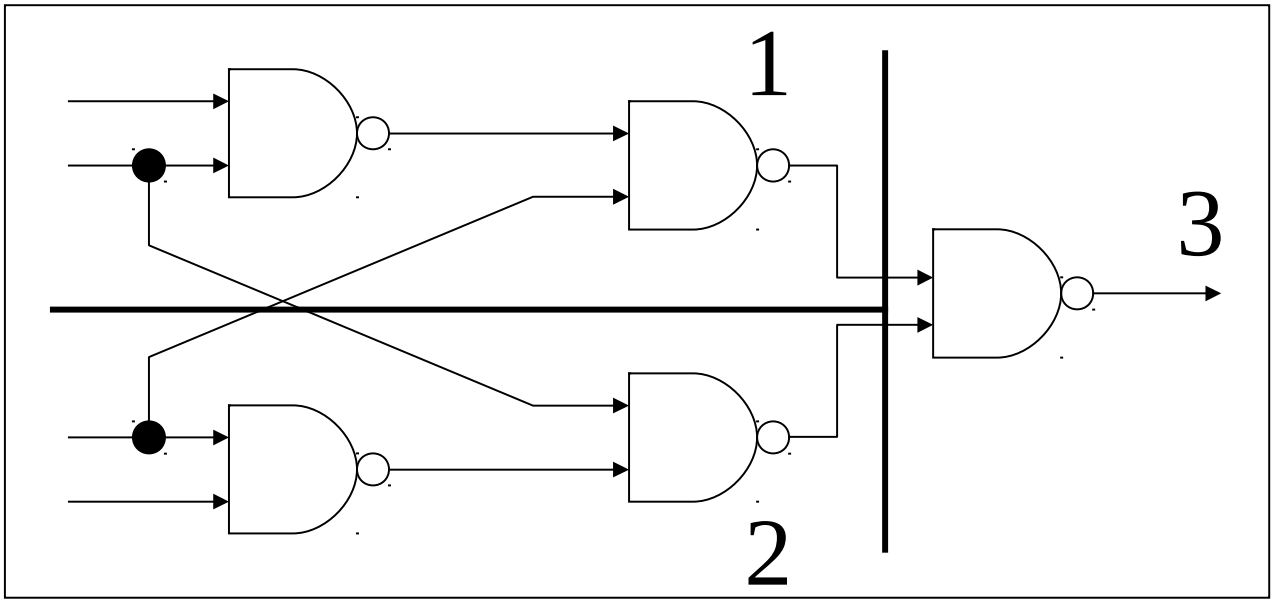
- Produce a geometric chip layout from an abstract circuit design
- Inputs:
  - Gate-level netlist
  - Standard cell library
  - Constraints
- Output:
  - Silicon layout (geometric description of the chip)

Physical Design - Steps

- Partitioning
- Chip planning
- Placement
- Global routing
- Detailed routing

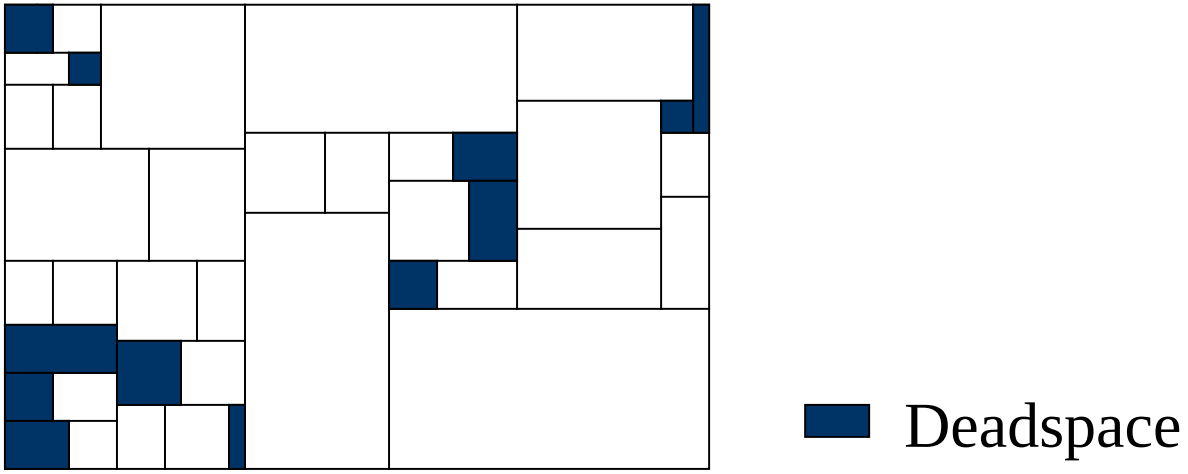
Partitioning

- Breaks up a large circuit into smaller sub-circuits (blocks)
  - Number of blocks
  - Interconnection between blocks



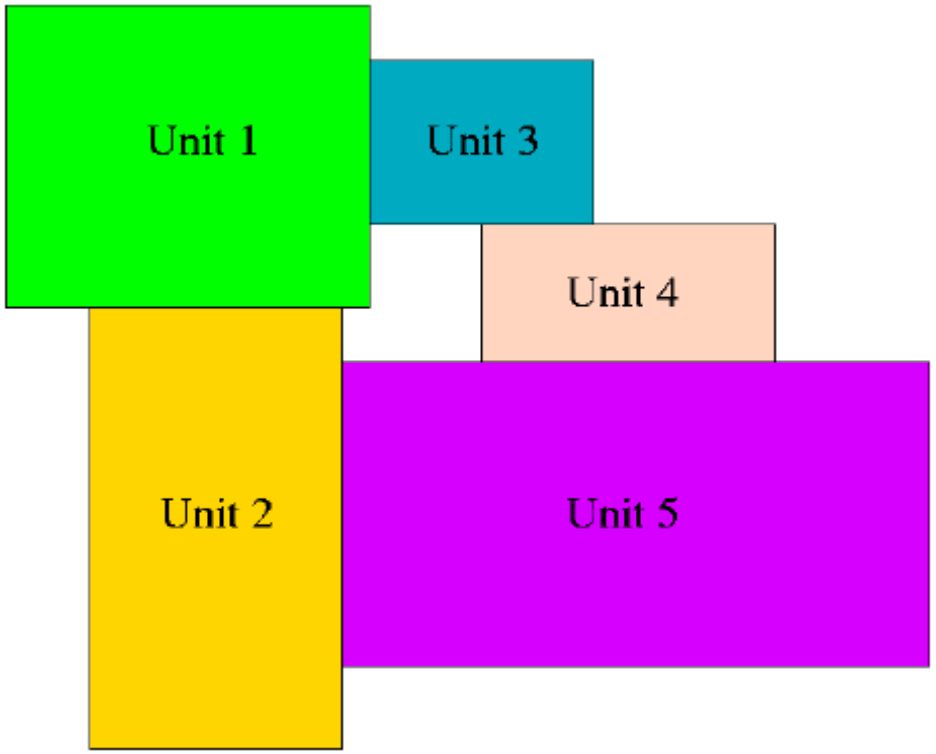
Floorplanning

- Set up a plan for a good layout
- Place the blocks at early stage when details like shape, area, position of I/O pins… are not yet fixed



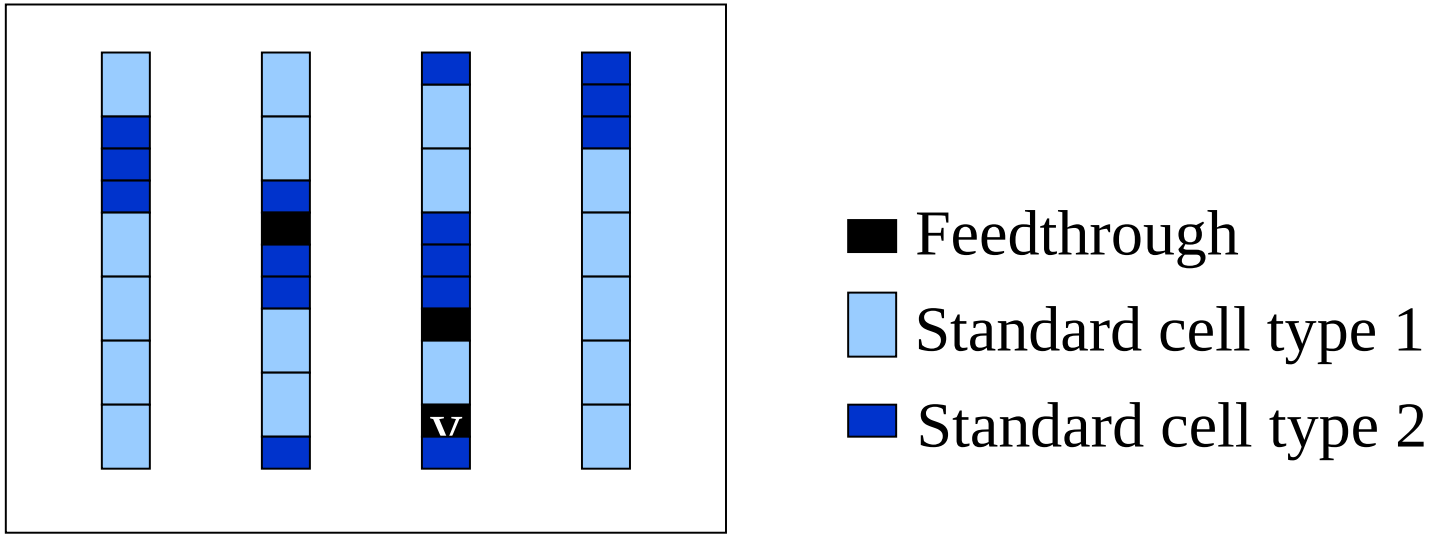
Floorplanning

- Blocks are placed in order to minimize area and the connections between them



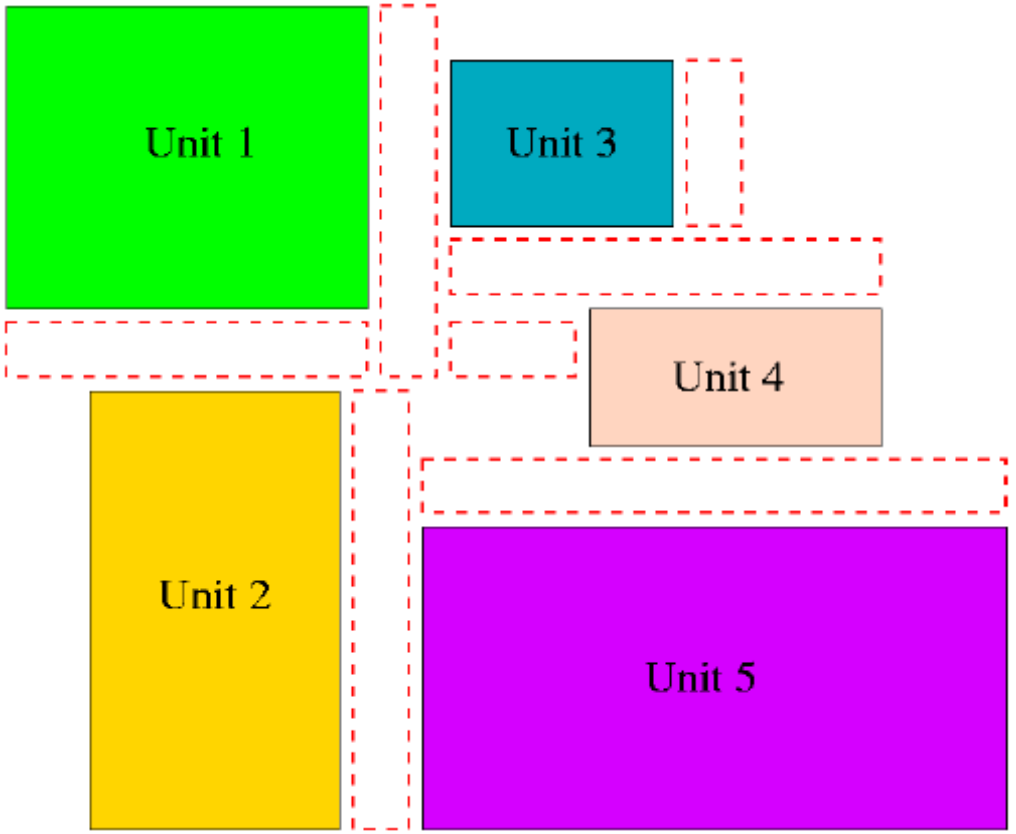
Placement

- Exact placement of the modules
  - Modules can be gates, standard cells...
- Details of design are known
  - Goal is to minimize total area and interconnect cost



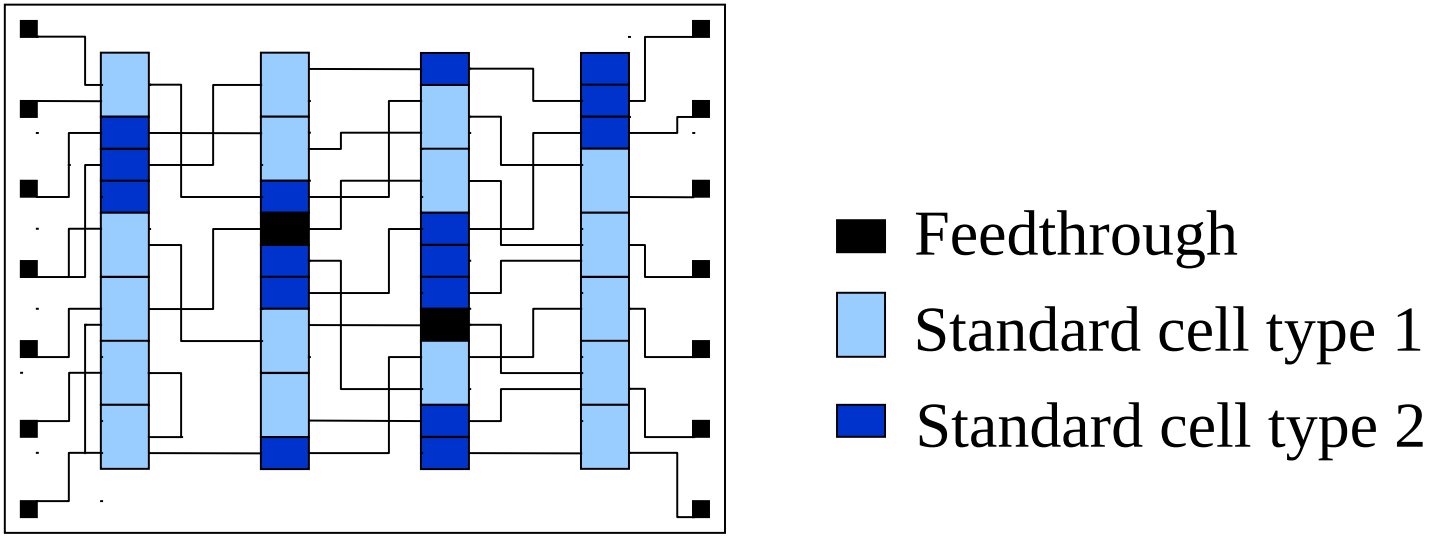
Placement

- Blocks are placed so empty rectangular spaces are left between them
- These spaces will be later used to make the interconnection



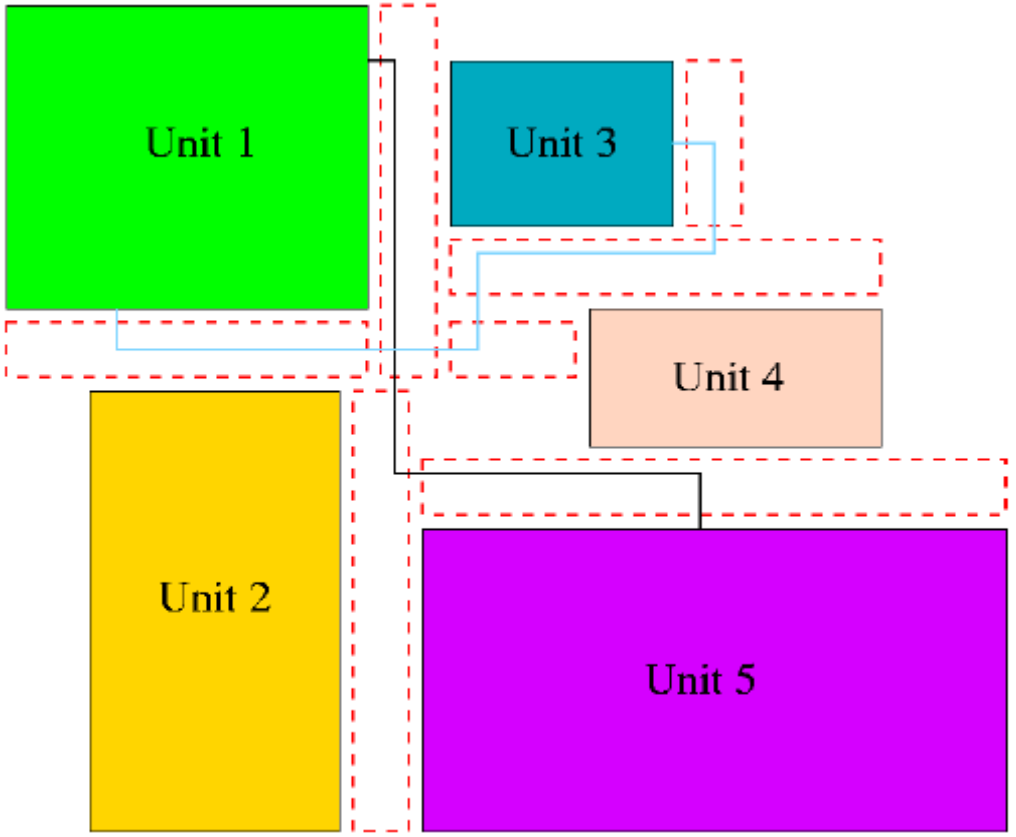
Routing

- Completes the interconnections between modules
- Considers delay of critical path, wire spacing...
- Global routing and detailed routing



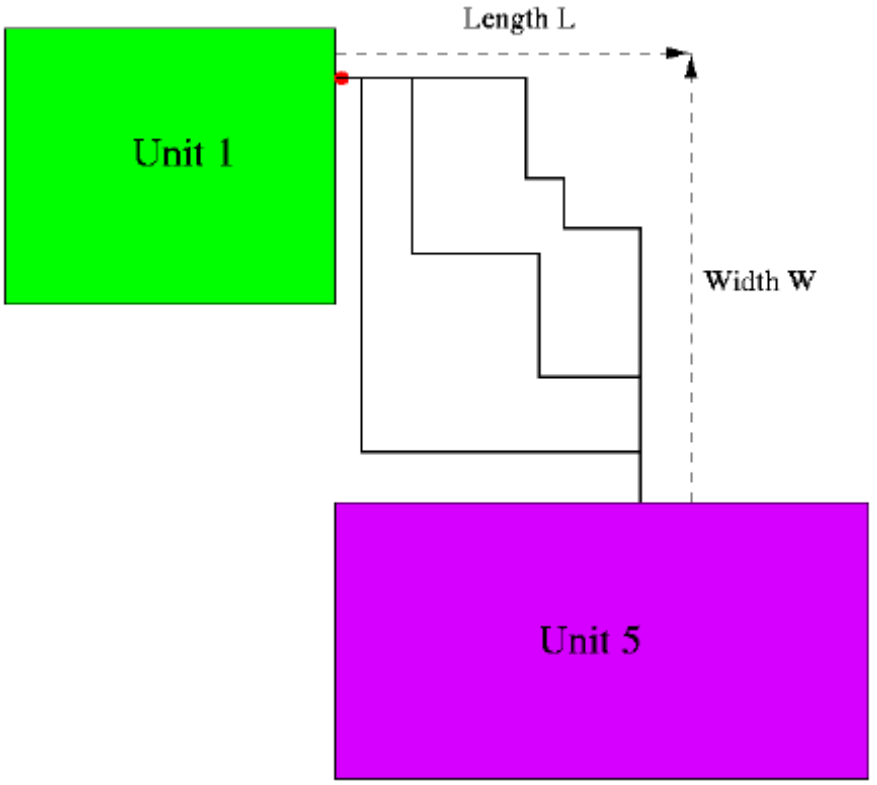
Global Routing

- Each connection will go from each origin block, through the channels until the end block



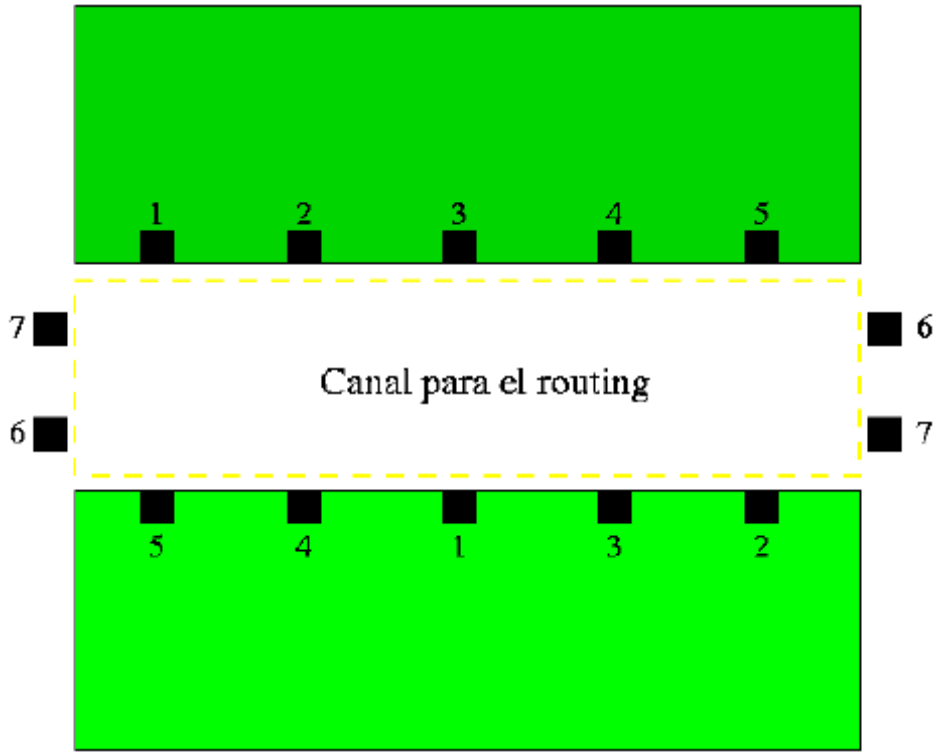
Global Routing

- The length of the connections will depend on the situation of the blocks rather than the way the routing is done



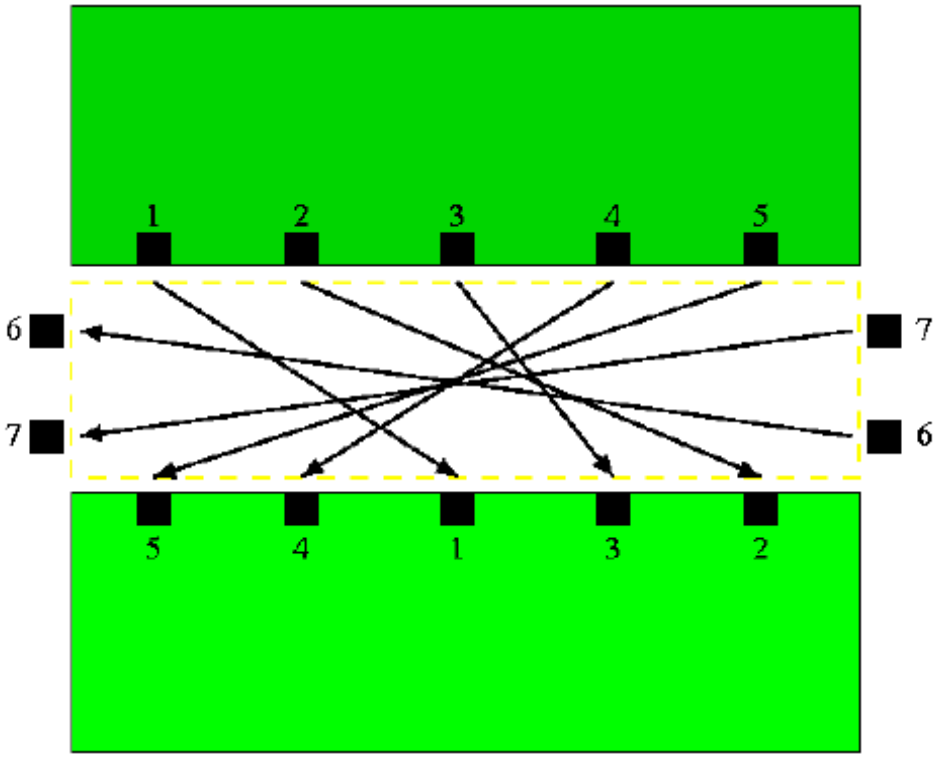
Detailed Routing

- The space required for each channel will depend on the complexity and density of the connections



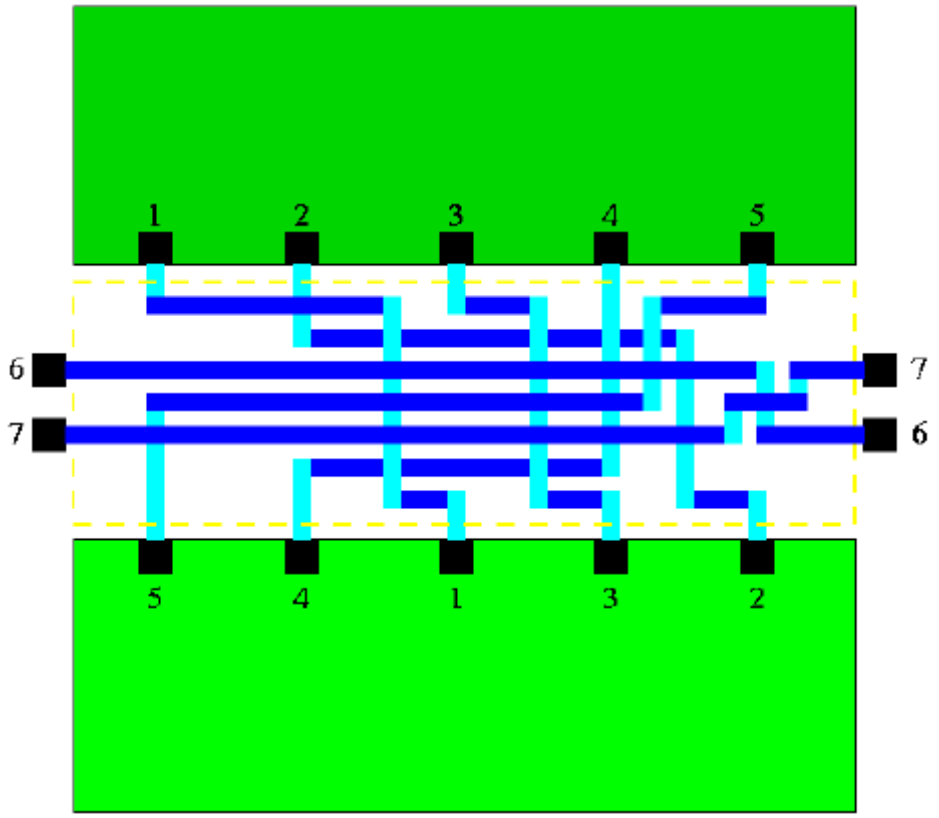
Detailed Routing

- Non-aligned connections increment the complexity of the routing, increasing the amount of space required



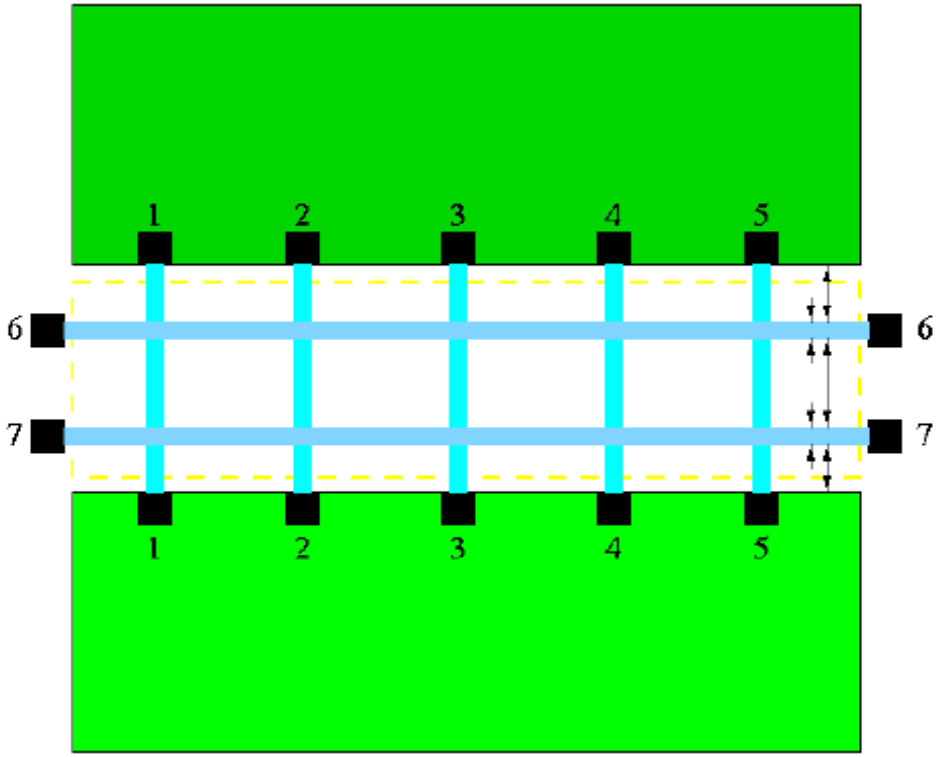
Detailed Routing

- The number of crossings determines the space required



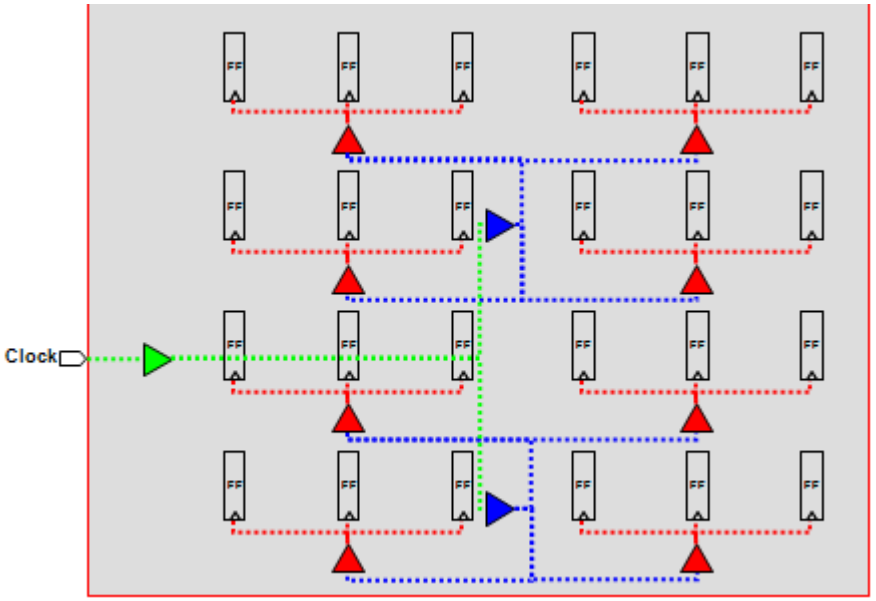
Detailed Routing

- Aligned connections reduce dramatically the area required of the channel from completing routing



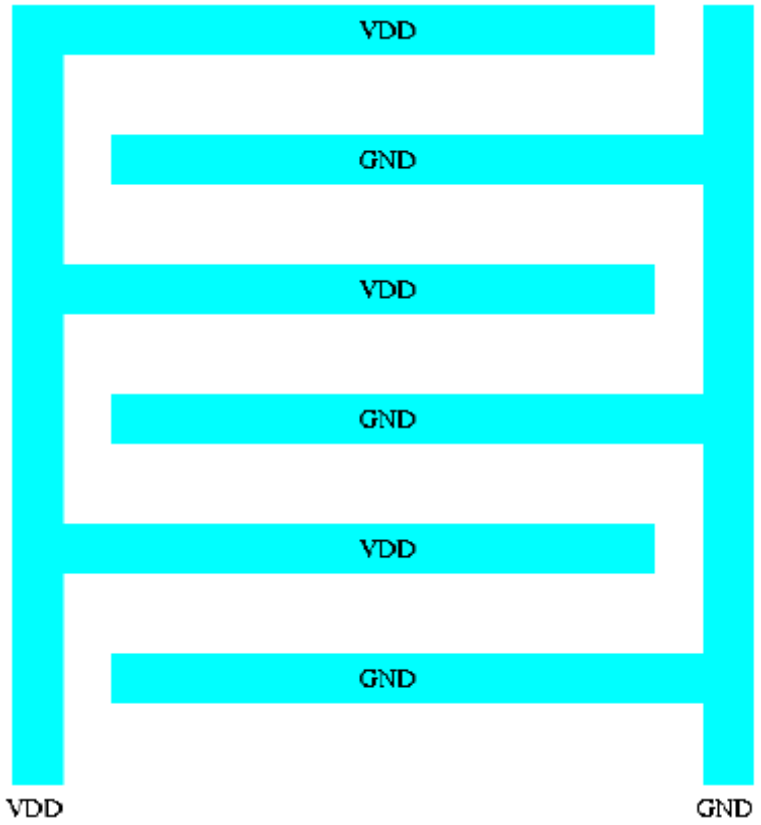
Clock Network Distribution

- Determines routing of the clock signal to meet the delay requirements

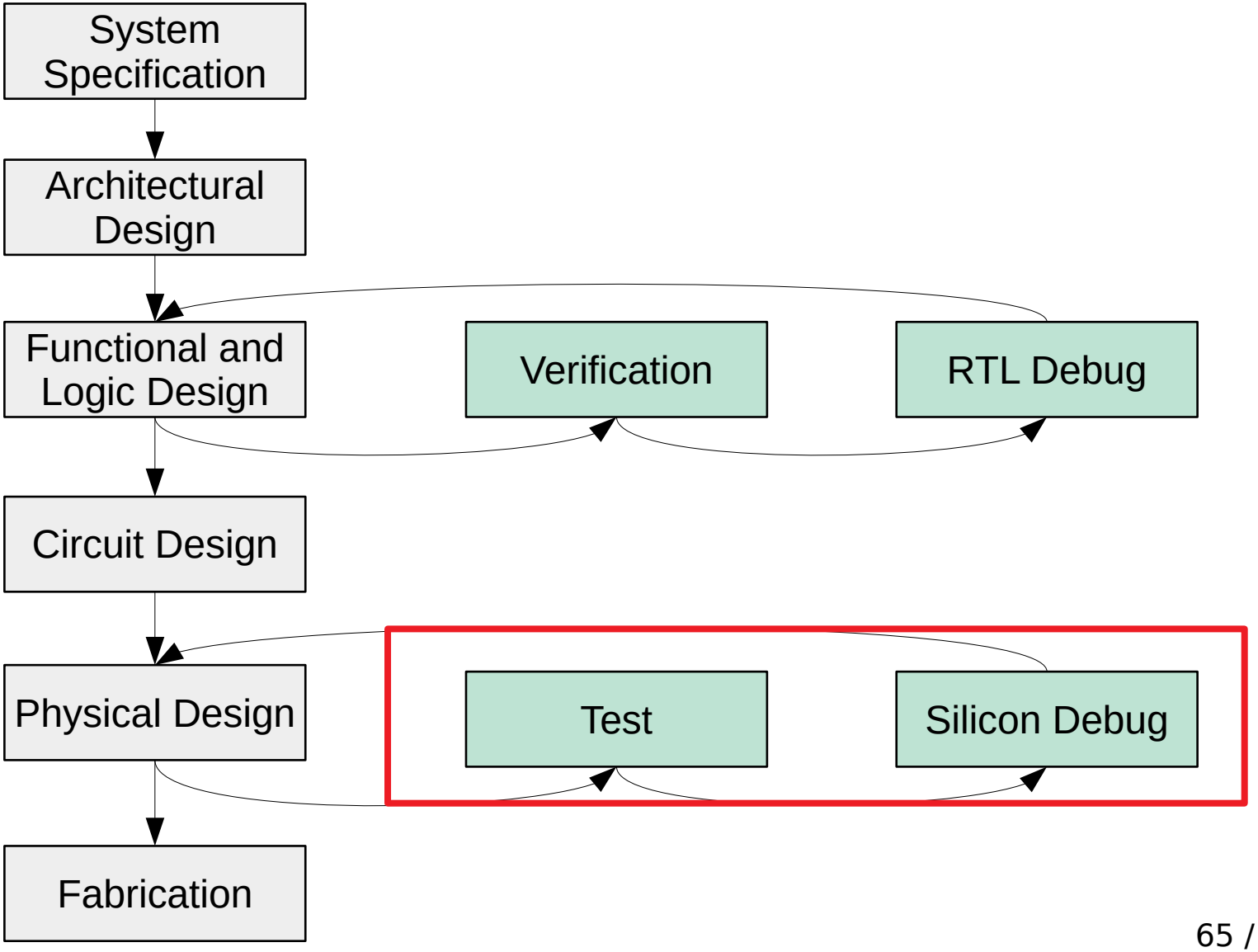


Power Distribution

- Interlaced distribution minimizes number of metal levels
- Thickness will vary according to power consumption



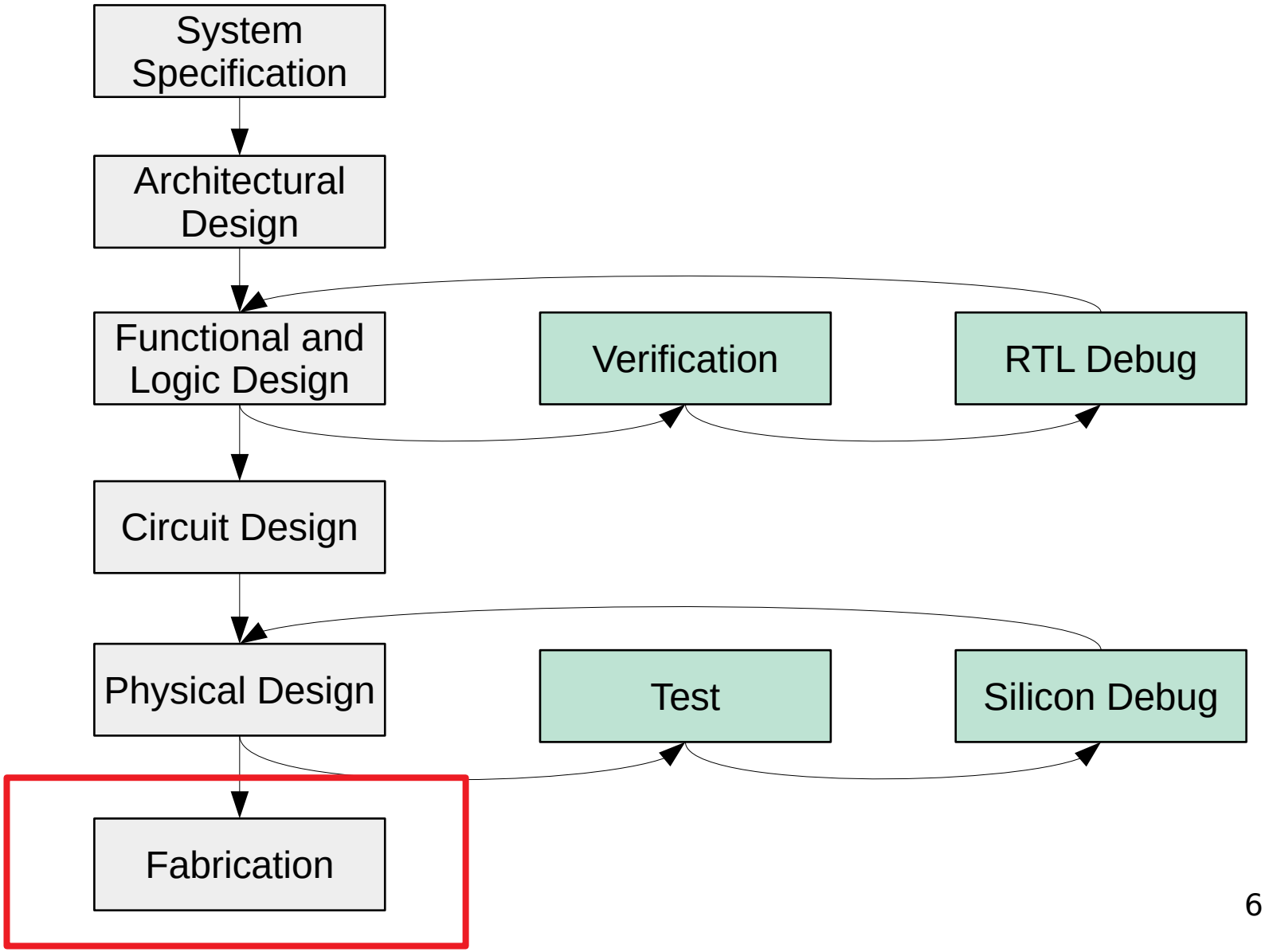
VLSI Design Flow



Testing

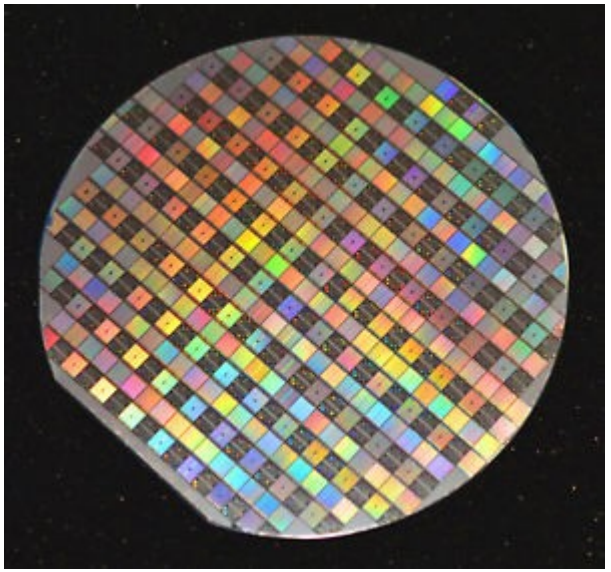
- Check the correctness of the layout
- Design rule checking (DRC)
  - Verifies that layout meets all technology-imposed constraints
- Layout vs schematic (LVS)
  - Verifies the functionality of the design
  - A netlist from the layout is derived and compared with original netlist
- Electrical rule checking (ERC)
  - Verifies the correctness of power and ground connections, signal transition times...

VLSI Design Flow

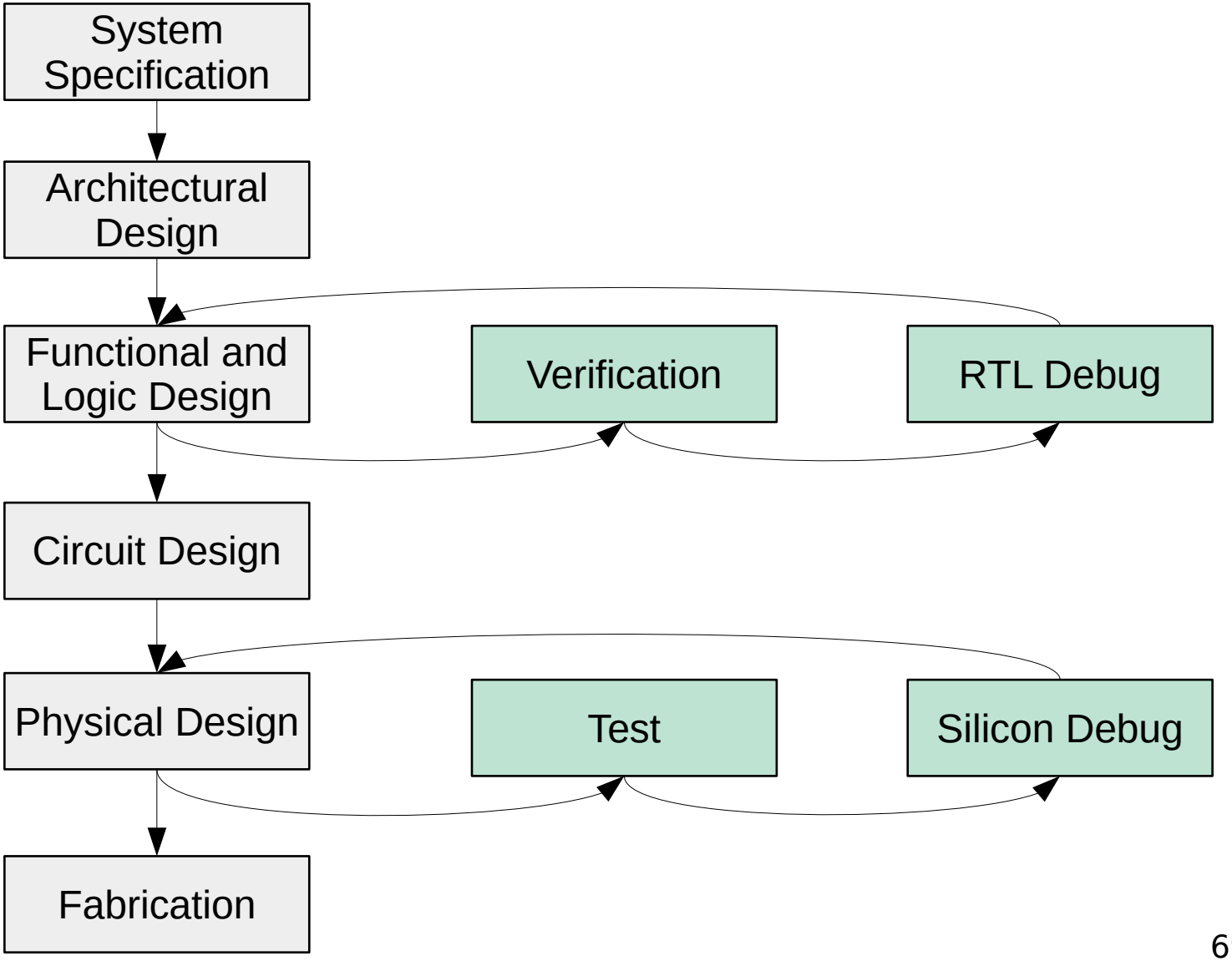


Fabrication

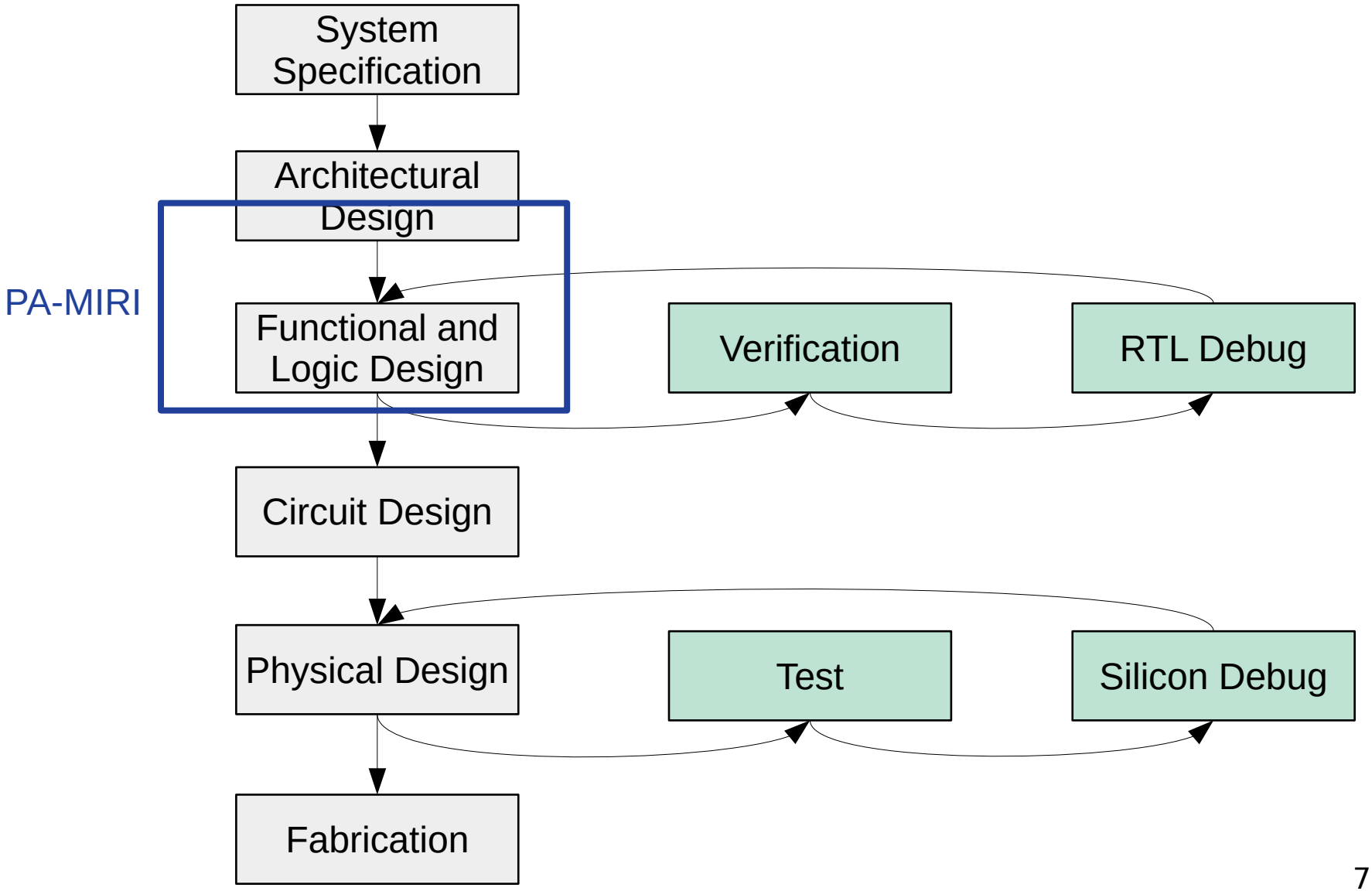
- The final layout (GDSII Stream format) is sent to a dedicated silicon foundry (fab)
- Tapeout
  - Final result of the design process for integrated circuits before they are sent for manufacturing



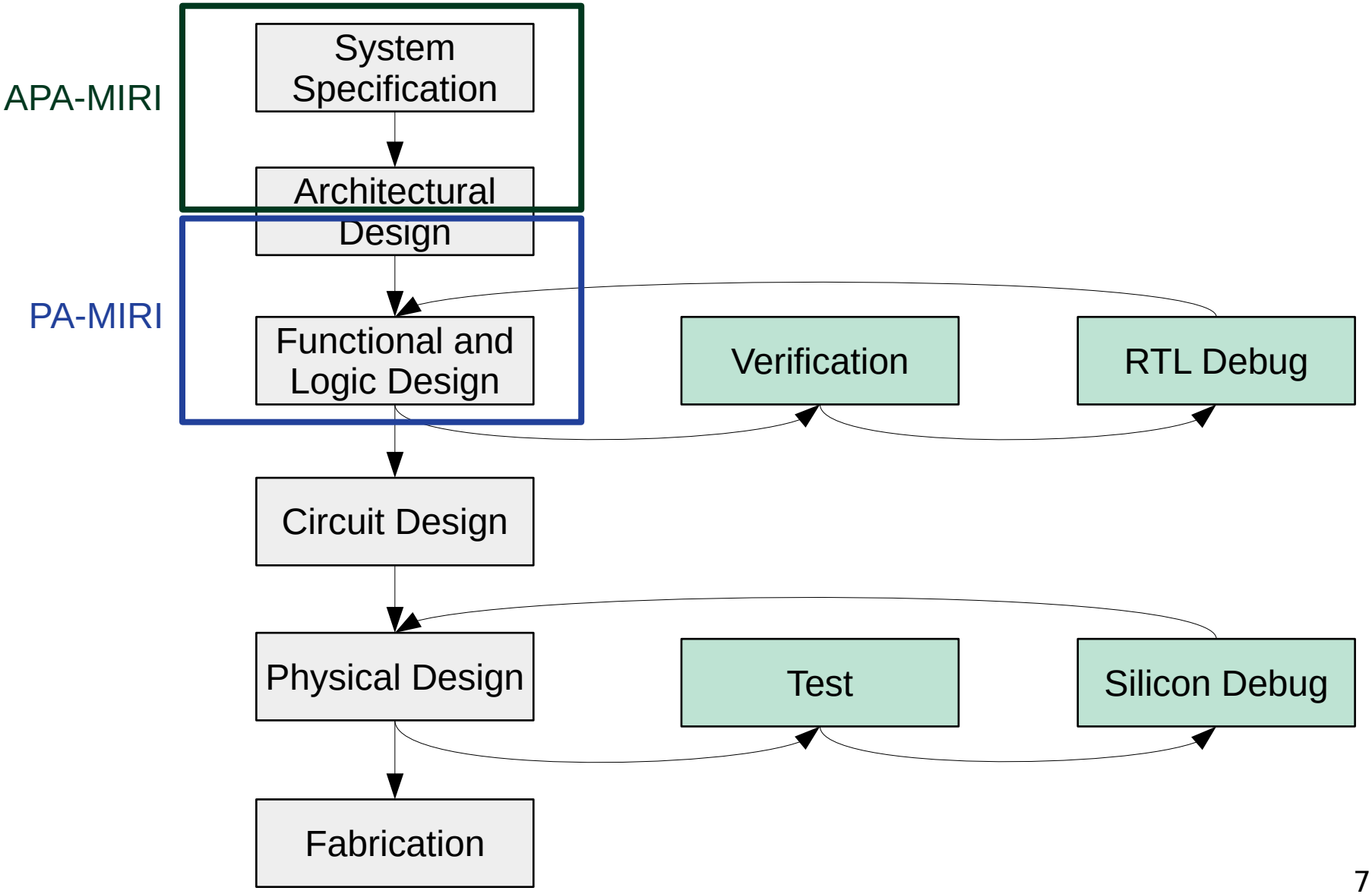
VLSI Design Flow



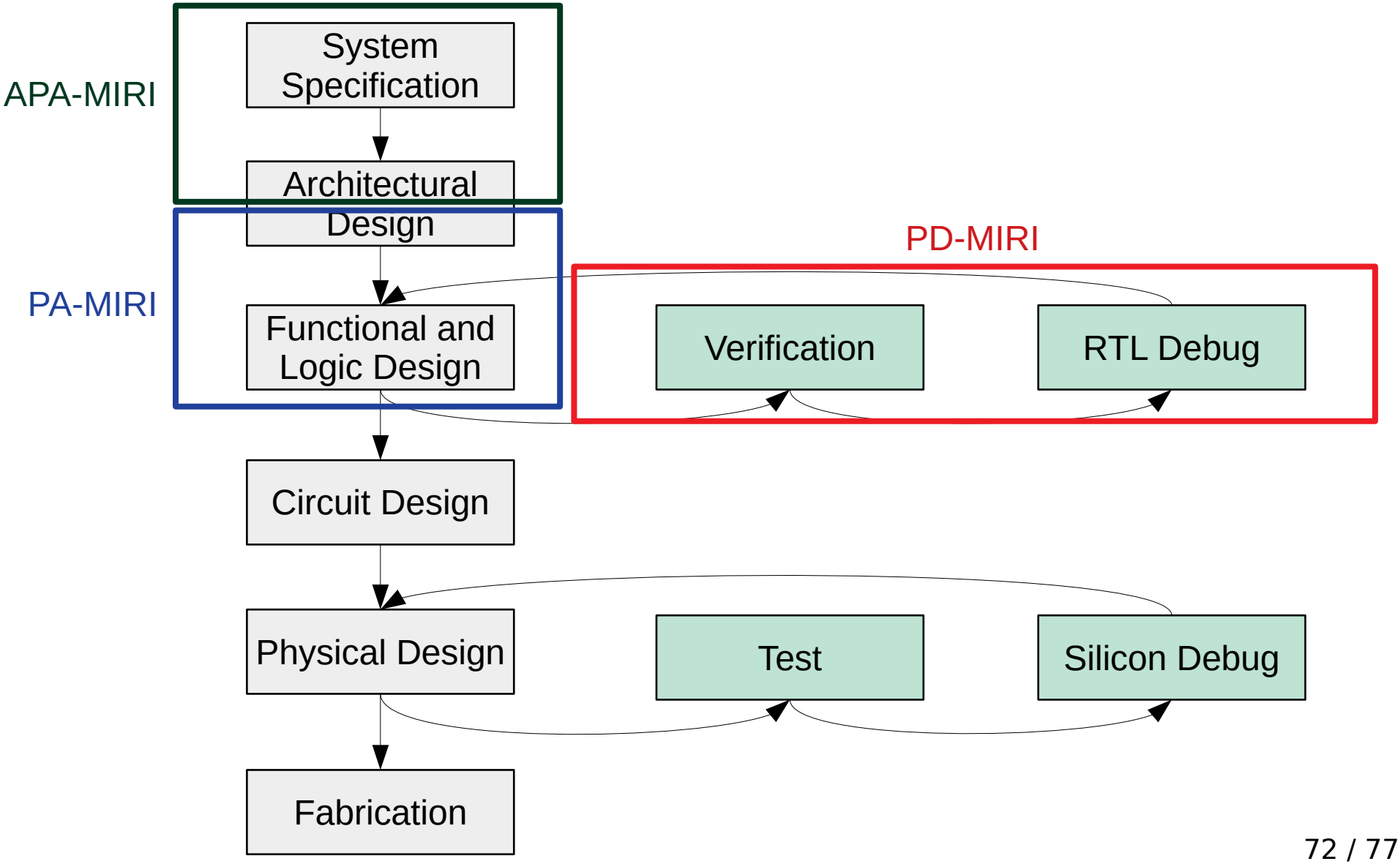
VLSI Design Flow



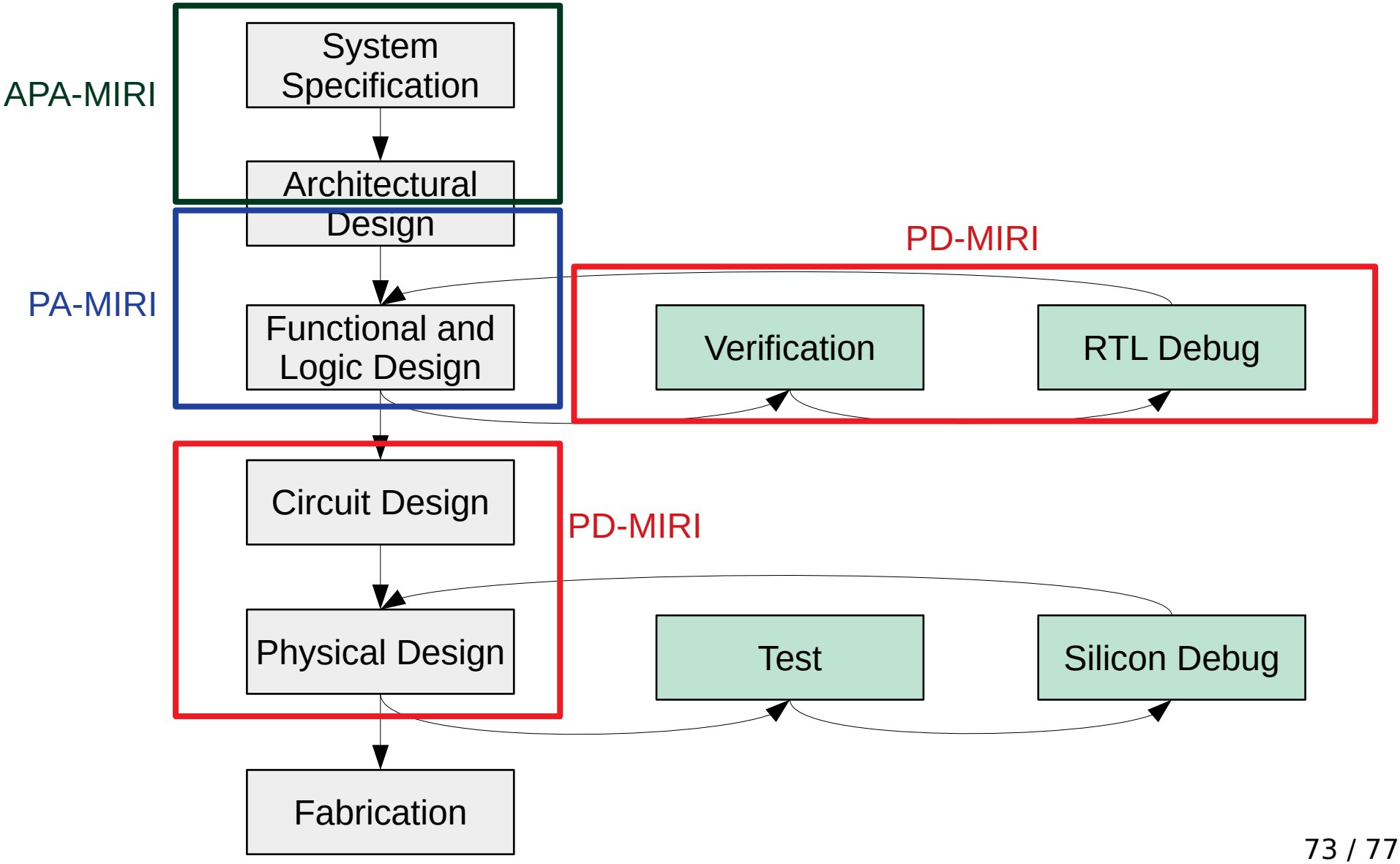
VLSI Design Flow



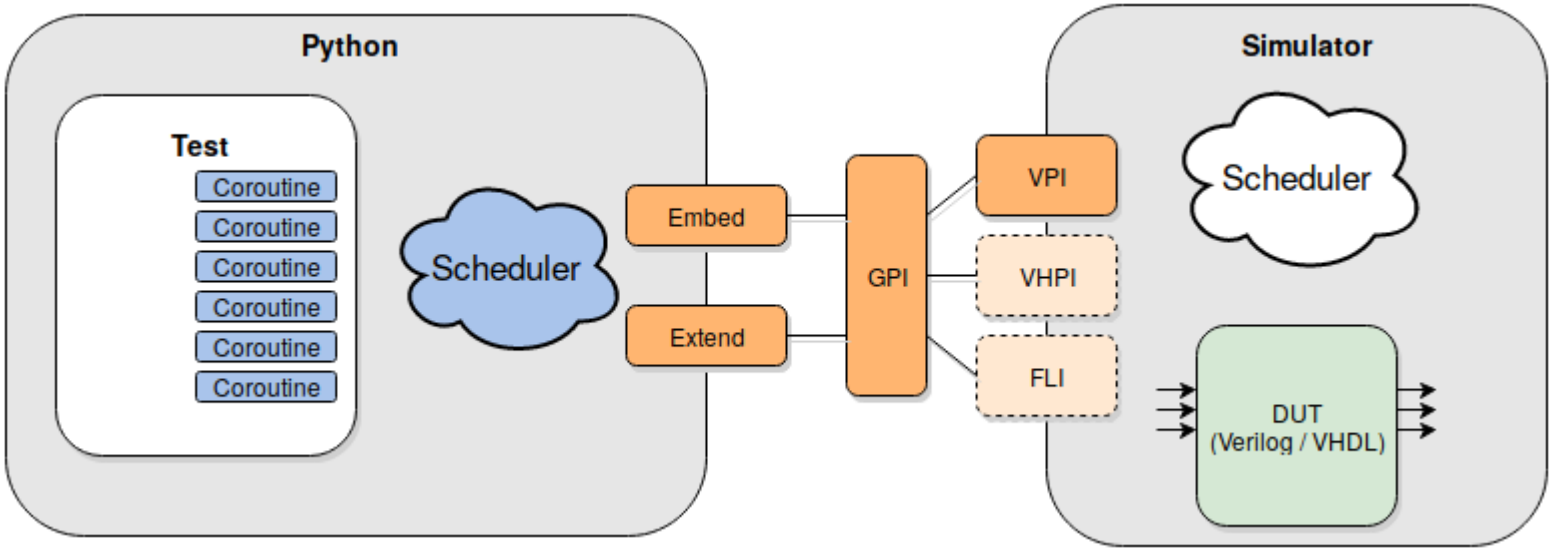
VLSI Design Flow



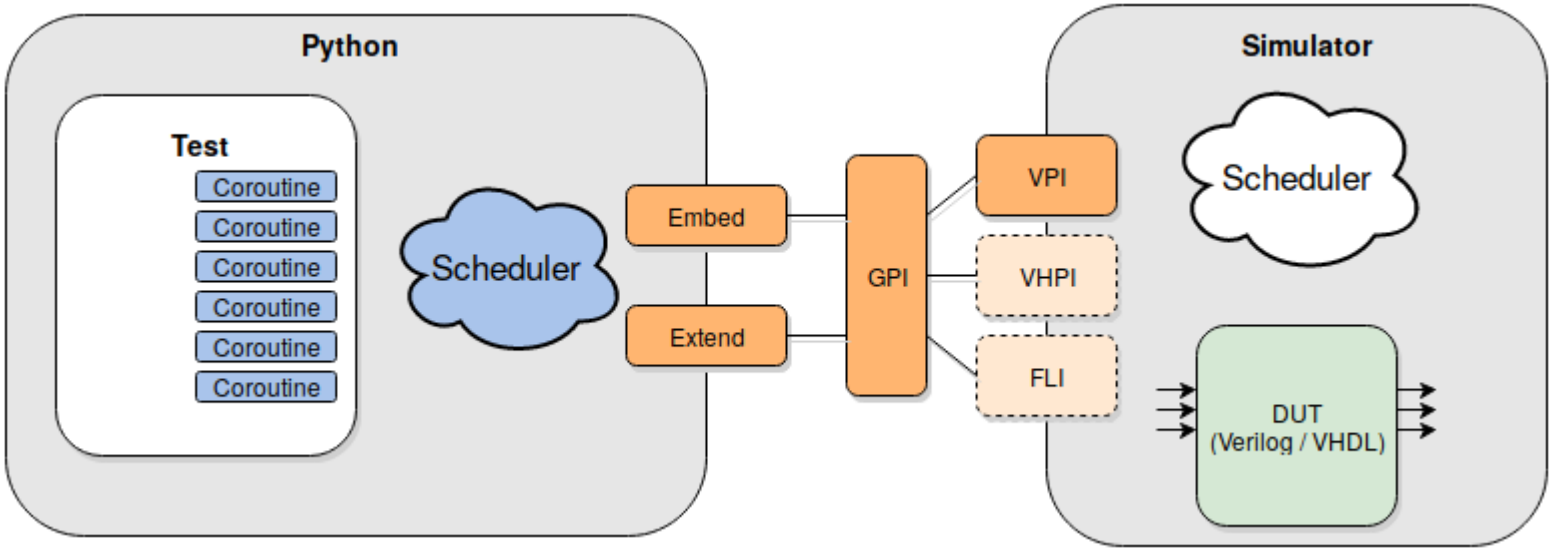
VLSI Design Flow

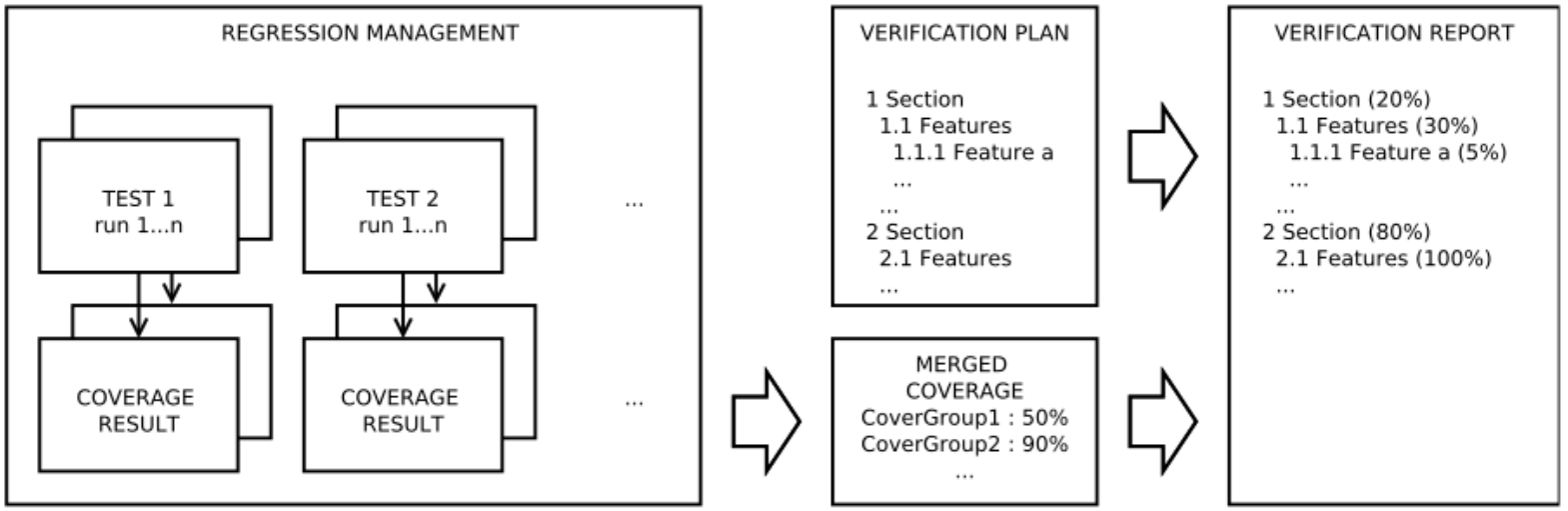


Cocotb - Verification Tool

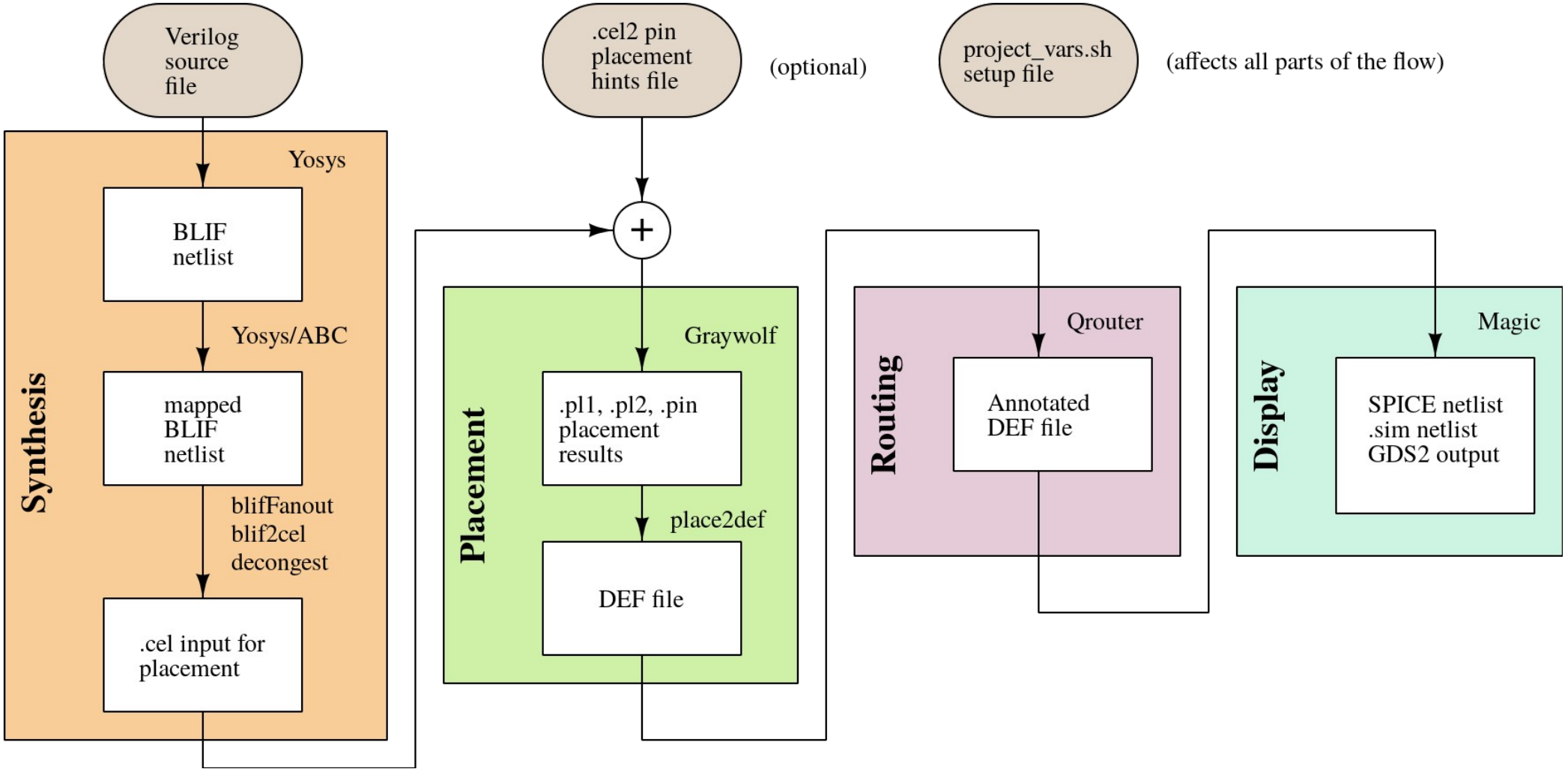


Cocotb - Verification Tool





Qflow - Synthesis Tool



http://opencircuitdesign.com/qflow/

Bibliography

- Weste, N. H., & Harris, D. “CMOS VLSI Design : A Circuits and Systems Perspective”. 4th Edition, 2010.
- Kahng AB, Lienig J, Markov IL, Hu J. “VLSI Physical Design: from Graph Partitioning to Timing Closure”. Springer Science & Business Media; 2011 Jan 27.
- Mead, C., & Conway, L. (1980). Introduction to VLSI systems (Vol. 1080). Reading, MA: Addison-Wesley.