Praktikum ASIC Design von Hardwarebeschleunigern für RISC-V

Patrick Karl[1]

[1]Technical University of Munich
TUM School of Computation, Information and Technology
Chair of Security in Information Technology

October 25, 2024

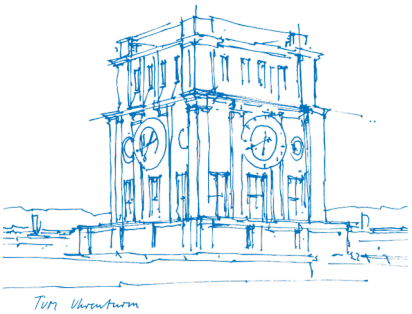


Table of contents

Table of contents

Formalities

What's the goal of the lab?

1. Provide *some* insight in the workflow RTL $\rightarrow$ Chip
2. Qualify to understand and extend a given RTL design
3. Understand the interaction between SW and HW
4. Learn to use manuals!

Formalities

Important Note

The lab is not designed in a step-by-step guided fashion! → You have to be pro-active

Tutor hours

There will be tutor hours twice per week

→ Room 2947, Tuesday 9:45am – 11:15am, Thursday 3:00pm – 4:30pm

→ Florian Gruber, florian2000.gruber@tum.de

Deliverables and Exam

Deliverables

Push your implementation, skripts etc. to Gitlab!

Your implementation can be in VHDL, Verilog or SystemVerilog

Exam

30 min oral exam - date will be announced later in semester

Questions will ask about your design, workflow, general understanding

Additional Note

There's a tutorial PDF giving Tasks, Hints, Tool explanations, additional resource etc.

Table of contents

Task Description

Task

- Implement a `shake128` accelerator
- Integrate the accelerator into a given RISC-V platform
- Design an ASIC layout

Task Description

Task

- Implement a `shake128` accelerator
- Integrate the accelerator into a given RISC-V platform
- Design an ASIC layout

You don’t have to start from scratch!

Task Description

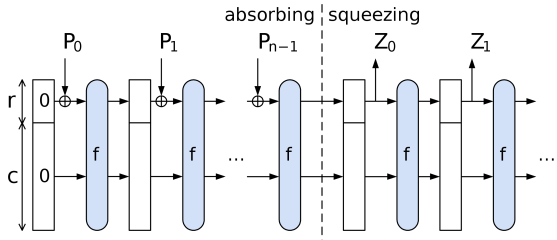
Task

- **Implement a `shake128` accelerator**
- Integrate the accelerator into a given RISC-V platform
- Design an ASIC layout

You don’t have to start from scratch!

Shake-128

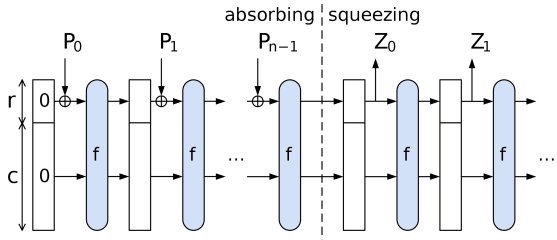
- `shake128` is an eXtendable Output Function
  - Expands seeds, hash data etc.
- Sponge construction based on the `Keccak` primitive
- Used in many Post-Quantum Cryptography schemes



By http://sponge.noekeon.org/, CC BY 3.0,
https://commons.wikimedia.org/w/index.php?curid=13463547

Shake-128

- Operates on 1600 bit state
  - Divided into capacity c and rate r
- Split message P into blocks P_i
  - Blockwise absorbtion (xor) and permutation
- Squeeze arbitrary size digest Z
  - Subsequent permutations



By http://sponge.noekeon.org/, CC BY 3.0,
https://commons.wikimedia.org/w/index.php?curid=13463547

- 24-round permutation function `keccak-f1600`

Round Function `Keccak-f1600`

$S' = \iota \circ \chi \circ \pi \circ \rho \circ \theta(S)$, ι adds round constant

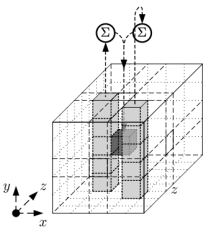


Figure: θ

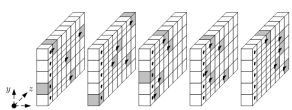


Figure: ρ

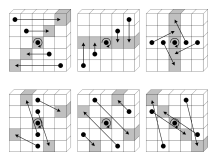


Figure: π

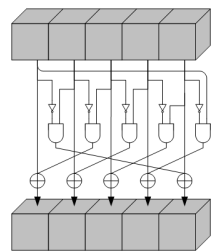


Figure: χ

Figures taken from https://keccak.team/figures.html

Task Description

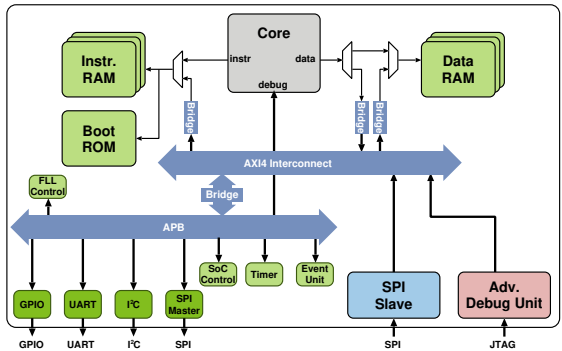
Task

- Implement a `shake128` accelerator
- **Integrate the accelerator into a given RISC-V platform**
- Design an ASIC layout

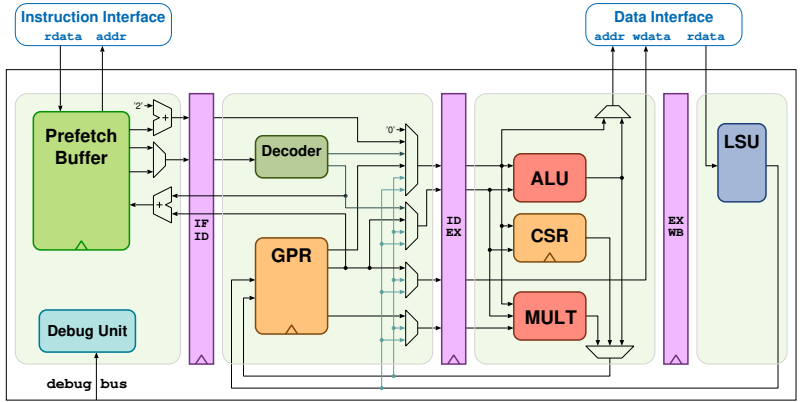
- Open-Source ISA
- Everyone can build his own processors, extend platforms etc.
- Suitable to integrate HW accelerators for cryptography!

PULPino MCU [1]

- 32 bit RISC-V core with several peripherals
- Compile C code → write into Instr. RAM → start fetching instructions
- Instr. writte via external SPI



RISC-V Core [1]


Instruction Interface
rdata addr
Data Interface
addr wdata rdata
Prefetch Buffer
'2'
Decoder
'0'
IF ID
GPR
ID EX
ALU
CSR
MULT
EX WB
LSU
Debug Unit
debug bus

Accelerator coupled to AXI4

You should integrate a `shake128` core into the PULPino!

- Most blocks given!
- You implement *Accelerator Logic*
- Template with dummy adder given

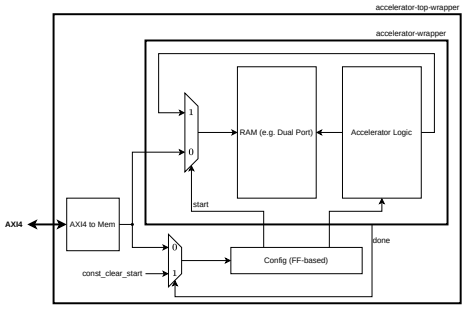
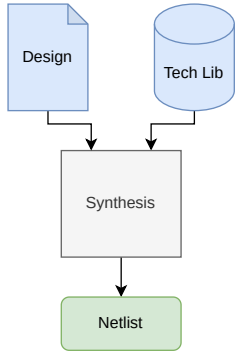


Table of contents

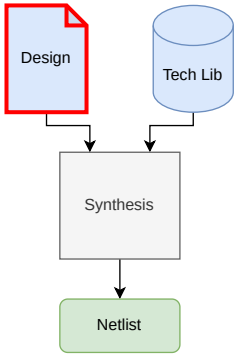
Frontend Design – Workflow

1. Circuit design (RTL description)
2. Choice of technology library
3. Map RTL description to technology



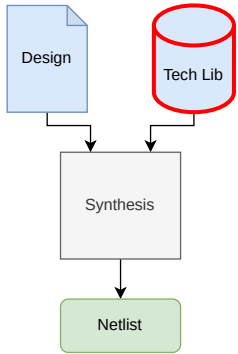
Frontend Design – Circuit Design

- Design circuit using Hardware Description Language
  - VHDL, Verilog, SystemVerilog etc.
- Sequential/combinatorial logic
  - Logic gates, FFs etc.
- Instantiation of macros
  - Memories, optimized multiplier etc.
- Constraining (timing, placement, routing, . . .)



Frontend Design – Technology Library

- Standard cell library provided by vendor
  - Type of standard cells defined
  - Technology definition (metal layers, via stacks, . . .)
  - Design rule definitions
- Library Exchange Format (.lef)
- Liberty Timing File (.lib)



Frontend Design – LEF File

- Includes abstract definitions/geometries for:
  - layers
  - vias
  - placement rules
  - macro definitions

- *Technology LEF*
  - Definitions of metal layers, interconnects, vias, routing direction

- *Cell LEF*
  - Geometry/Layout definition of each standard cell

```
LAYER met1
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;

  PITCH 0.37 ;
  MINENCLOSEDAREA 0.14 ;
  MINWIDTH 0.14 ;

  WIDTH 0.14 ;                    # Met1 1
  # SPACING 0.14 ;                # Met1 2
  # SPACING 0.28 RANGE 3.001 100 ; # Met1 3b
  SPACINGTABLE
     PARALLELRUNLENGTH 0
     WIDTH 0 0.14
     WIDTH 3 0.28 ;
  AREA 0.083 ;                    # Met1 6
  THICKNESS 0.35 ;

  ANTENNADIFFSIDEAREARATIO PWL ( ( 0 400 ) ( 0.0125 400 ) ( 0.0225 2609 ) ( 22.5 11600 ) ) ;

  EDGECAPACITANCE 40.567E-6 ;
  CAPACITANCE CPERSQDIST 25.7784E-6 ;
  DCCURRENTDENSITY AVERAGE 2.8 ; # mA/um Iavg_max at Tj = 90oC
  ACCURRENTDENSITY RMS 6.1 ; # mA/um Irms_max at Tj = 90oC

  RESISTANCE RPERSQ 0.125 ;
END met1
```

Figure: Tech LEF SKY130nm[1]

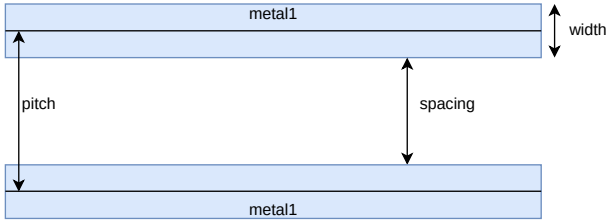
```
MACRO sky130_fd_sc_lp__nand2_0
  CLASS CORE ;
  SOURCE USER ;
  ORIGIN  0.000000  0.000000 ;
  SIZE  1.440000 BY  3.330000 ;
  SYMMETRY X Y R90 ;
  SITE unit ;
  PIN A
    ANTENNAGATEAREA  0.159000 ;
    DIRECTION INPUT ;
    USE SIGNAL ;
    PORT
      LAYER li1 ;
        RECT 0.985000 0.840000 1.355000 2.120000 ;
    END
  END A
```

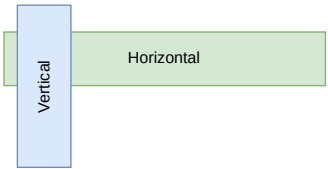
Figure: NAND2 LEF SKY130nm[2]

[1] http://github.com/google/skywater-pdk-libs-sky130_fd_sc_lp/blob/e2c1e0646999163d35ea7b2521c3ec5c28633e63/tech/sky130_fd_sc_lp.tlef

[2] https://github.com/google/skywater-pdk-libs-sky130_fd_sc_lp/blob/e2c1e0646999163d35ea7b2521c3ec5c28633e63/cells/nand2/sky130_fd_sc_lp__nand2_0.lef

Examplary Geometric Notation





- Note: Usually only horizontal OR vertical routing allowed on the same layer
  - For change of directions, use vias or jogs

Frontend Design – LIB File

- Contains timing and power characteristics of cells
- Timing model
  - cell delay
  - setup/hold time characteristics
- Electrical characteristics
  - Power, voltage, resistance etc.
- Values of operating condiditions (typ, min, max)

Frontend Design – Synthesis

- RTL code, constraints, LEFs and LIBs $\rightarrow$ netlist
- Compilation (check syntax)
- Elaboration and Binding
  - Translates RTL into Boolean structure
  - Bind non-boolean modules, state encodings etc.
- Mapping and Optimization
  - Logic optimization
  - Mapping of library to cells
  - Optimizations to meet constraints

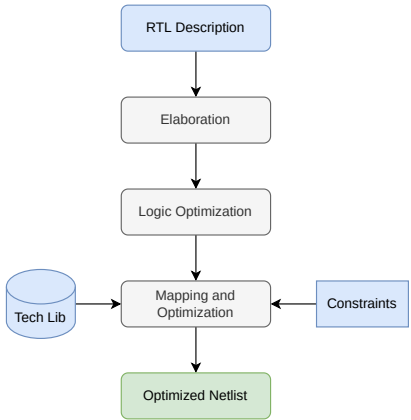


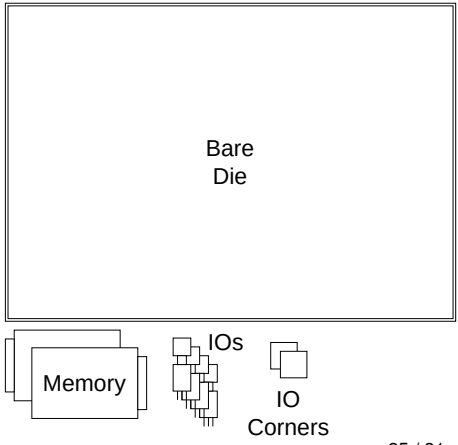
Table of contents

Backend

- Floorplanning
- Power planning, special route etc.
- Placement
- Clock-Tree Synthesis
- Global Routing

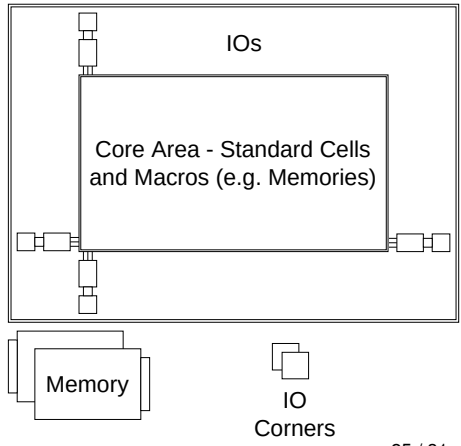
Backend - Floorplanning

- Determine necessary size of die based on:
  - Area required by std. cells and memories
  - Routing complexity
  - Number of IOs (Power and Signals)
- Place hard macros (memories, analog building blocks)
- Place IO-ports



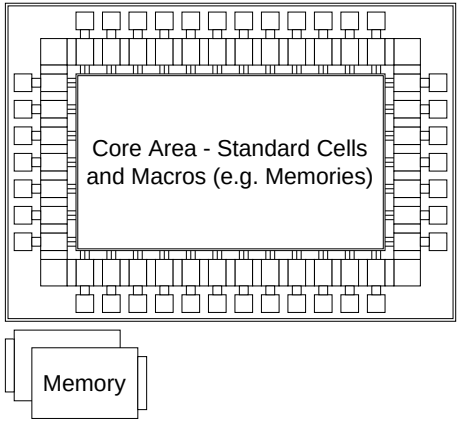
Backend - Floorplanning

- Determine necessary size of die based on:
  - Area required by std. cells and memories
  - Routing complexity
  - Number of IOs (Power and Signals)
- Place hard macros (memories, analog building blocks)
- Place IO-ports



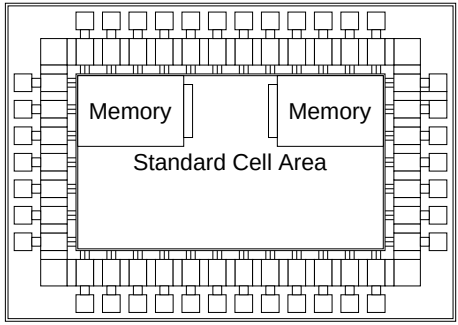
Backend - Floorplanning

- Determine necessary size of die based on:
  - Area required by std. cells and memories
  - Routing complexity
  - Number of IOs (Power and Signals)
- Place hard macros (memories, analog building blocks)
- Place IO-ports



Backend - Floorplanning

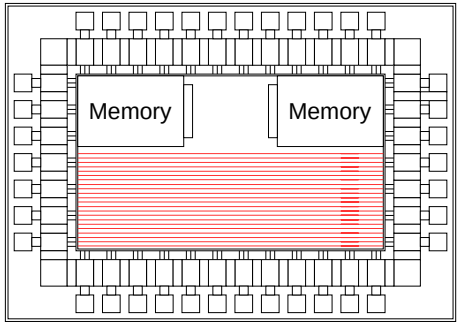
- Determine necessary size of die based on:
  - Area required by std. cells and memories
  - Routing complexity
  - Number of IOs (Power and Signals)
- Place hard macros (memories, analog building blocks)
- Place IO-ports



Digital ASIC Power Distribution and Standard Cell Grid

Components that need connection:

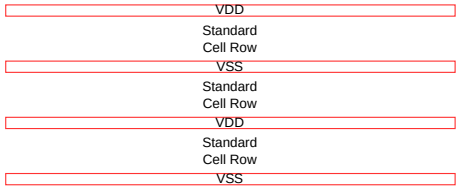
- Alternating VDD and VSS on lowest metal layer for standard cells
- Power rings or pins on macros



Digital ASIC Power Distribution and Standard Cell Grid

Components that need connection:

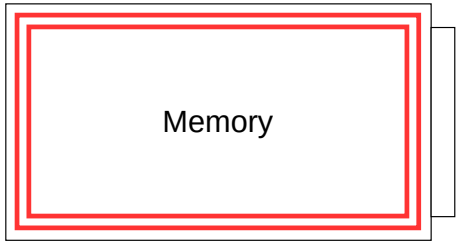
- Alternating VDD and VSS on lowest metal layer for standard cells
- Power rings or pins on macros



Digital ASIC Power Distribution and Standard Cell Grid

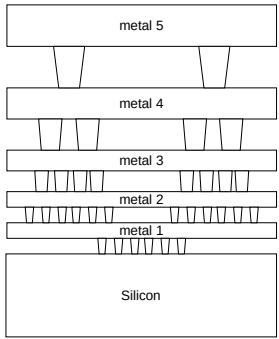
Components that need connection:

- Alternating VDD and VSS on lowest metal layer for standard cells
- Power rings or pins on macros



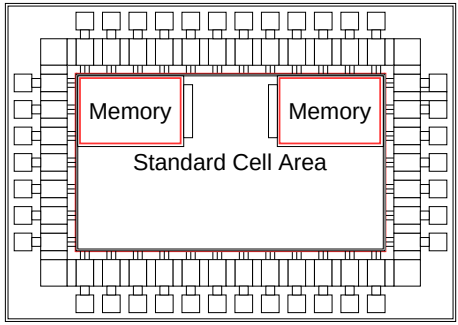
Backend - Power planning

- Choose suitable layer for power routing
  - Highest layer possible, as tracks can be wider and lower layers are needed for signal routing
  - Via dimensions of top layer need to be in reasonable distance to via dimensions of lower layers
- Create Rings around core area and around macros (if necessary)
- Create Power grid



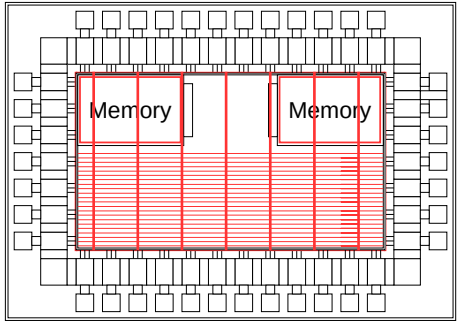
Backend - Power planning

- Choose suitable layer for power routing
  - Highest layer possible, as tracks can be wider and lower layers are needed for signal routing
  - Via dimensions of top layer need to be in reasonable distance to via dimensions of lower layers
- Create Rings around core area and around macros (if necessary)
- Create Power grid



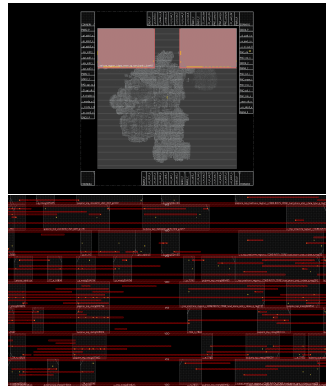
Backend - Power planning

- Choose suitable layer for power routing
  - Highest layer possible, as tracks can be wider and lower layers are needed for signal routing
  - Via dimensions of top layer need to be in reasonable distance to via dimensions of lower layers
- Create Rings around core area and around macros (if necessary)
- Create Power grid



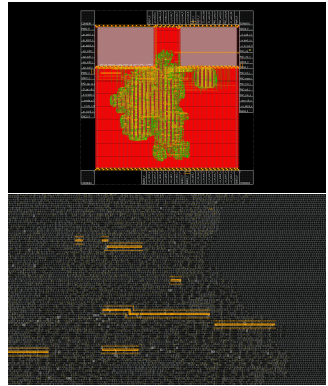
Backend - Placement of Std. Cells

- Std. Cells needs to placed on grid
- Control density
- Control placement of submodules in specific parts of the design
- Honor rules for adjacent cell placement



Backend - Clock Tree Synthesis

- Clock signals integrity is critical for correct functionality
- Clock signal should be routed with special care
  - Use higher layers due to better signal integrity
  - Use wider tracks to reduce resistance
  - Shield clock network



Backend - Global Routing

- Connect all ports of all cells
- Modern design density usually limited by routing
- Yield optimization vs. maximum possible density
  - Wider tracks
  - Wider spacing
  - Redundant vias

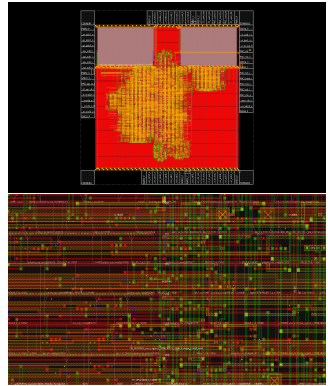


Table of contents

To start the lab...

1. Log into Gitlab such that we can assign you a repository
2. Sign the NDA and and write down your LRZ-ID
3. Read through the tutorial, tasks, hints and tool explanations are given there

To start the lab...

1. Log into Gitlab such that we can assign you a repository
2. Sign the NDA and and write down your LRZ-ID
3. Read through the tutorial, tasks, hints and tool explanations are given there

Questions?

Thank you for your attention!

References

[1] Michael Gautschi et al. “Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices”. In: *IEEE Trans. Very Large Scale Integr. Syst.* 25.10 (2017), pp. 2700–2713.