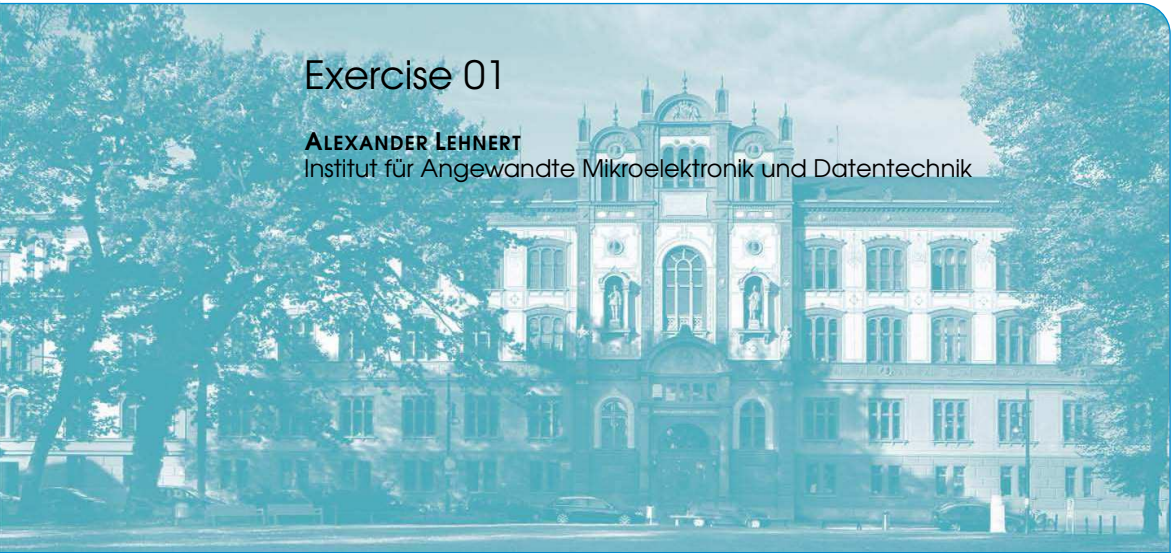


Exercise 01

Alexander Lehnert
Institut für Angewandte Mikroelektronik und Datentechnik

Who am I?

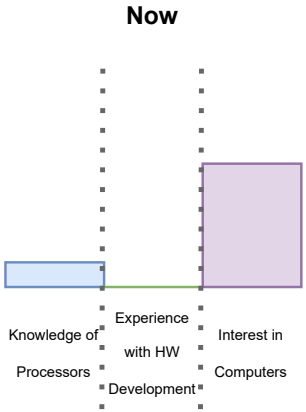
- Alexander Lehnert
- *Alex*
- Room AE26-126

← this person

Where am I from?
- Master of Science, Computer Science
- Friedrich-Alexander-Univerisität Erlangen-Nürnberg

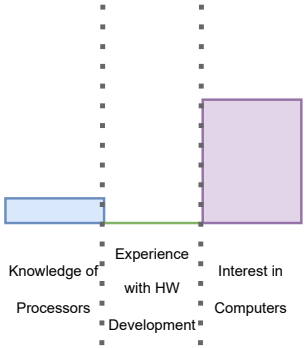
Research interests
- FPGA and ASIC design
- Hardware Accelerators for Inference in Neural Networks
- Matrix decomposition: *Computation Coding*

The Goal of this Project

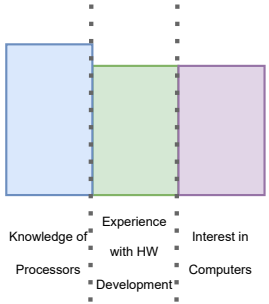


The Goal of this Project

Now



At the end of the Semester:



How we get there

- Hardware Description Languages
- Processor Architecture: RISC-V
- Technology: FPGA

- Step 1: CPU Building Blocks
- Step 2: Single Cycle RISC-V CPU
- Step 3: Pipelining
- Step 4: Individual Project

Format of this Project / Exercise

In the beginning: Weekly Exercises

- Thursday, 11:15 (:00, :30?)
- Interactive, Task-based

Later: Exercises morph to project

- Common starting point: Single Cycle CPU
- Project goals individually by difficulty level, group work is possible

Prerequisites

- Who knows VHDL?

Prerequisites

- Who knows VHDL?
- Who has programmed an FPGA?

Prerequisites

- Who knows VHDL?
- Who has programmed an FPGA?
- Who can explain how a CPU works?

Prerequisites

- Who knows VHDL?
- Who has programmed an FPGA?
- Who can explain how a CPU works?
- Who has implemented a RISC-V CPU on an FPGA?

The ZEDboard

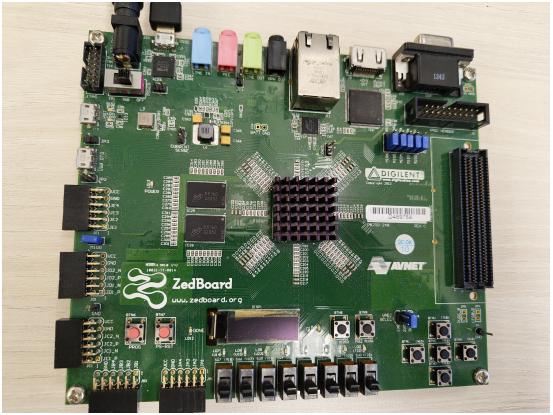
- One FPGA we use
- Also Nexys A7, Nexys4

FPGA:

- Zynq-7000
- 40600 LUTs
- 107 BRAMs

Some IO:

- LEDs
- Switches
- Buttons



Handling Advice

- Handle FPGA with care
- First, connect the power. Then turn them on.
- Watch out for Jumper Settings

Boot Mode	JP11	JP10	JP9	JP8	JP7
JTAG	0	0	0	0	0
Quad-SPI	0	1	0	0	0
SD Card	0	1	1	0	0

AMD/Xilinx Vivado

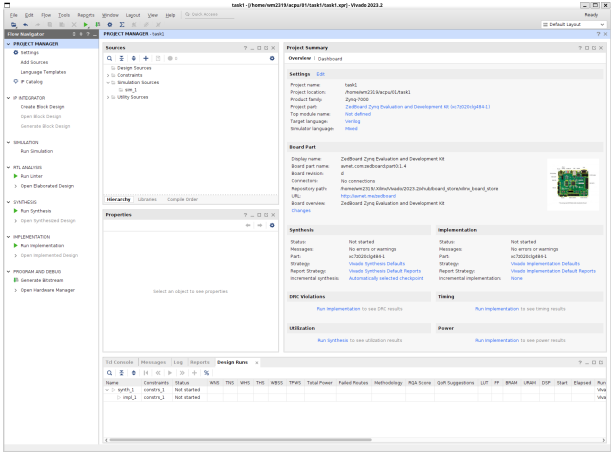
- Main tool for Xilinx FPGA Flow
- Synthesis and Implementation Tool
- Simulator
- Programming of the Device
- Text Editor

AMD/Xilinx Vivado

- Main tool for Xilinx FPGA Flow
- Synthesis and Implementation Tool
- Simulator
- Programming of the Device
- Text Editor

$\Rightarrow$ It does everything, but everything feels *eh*

AMD/Xilinx Vivado GUI



Working with Vivado

There are two ways to interact with Vivado

- Project Flow
- Manual Flow

There are two Vivado interfaces

- GUI (for now we use this)
- CLI via *Tool Command Language* (TCL) (we will learn this later)

Format of Tasks

- Core part of the exercises
- Tasks for you to do
- Here: I will be around
- You can finish the tasks by the next exercise
- Results will be uploaded to StudIP

The Linux Environment

- We will use a Linux system for this exercise
- Reboot, and boot into Linux: *RockyLinux*
- Login: IdM-login and password

Setting up Vivado

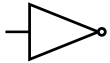
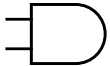
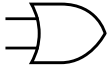
- Vivado is not installed locally
- Access via network share
- Execute once: setup script

→ StudIP - Files - Exercise - Scripts

- Run Vivado:
  - `vivado` *Starts the Vivado GUI*
  - `vivado path/to/script.tcl` *Starts the Vivado without GUI and executes the script*

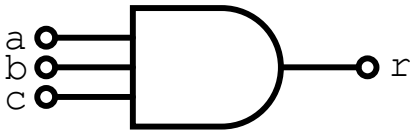
Some Basic Logical Functions

- Representation of logic gates
- Later also more complex functions
- Arithmetics?
- Custom functions?

Gate	VHDL
	`not`
	`and`
	`or`
	`xor`

Task 1: The AND3 Gate

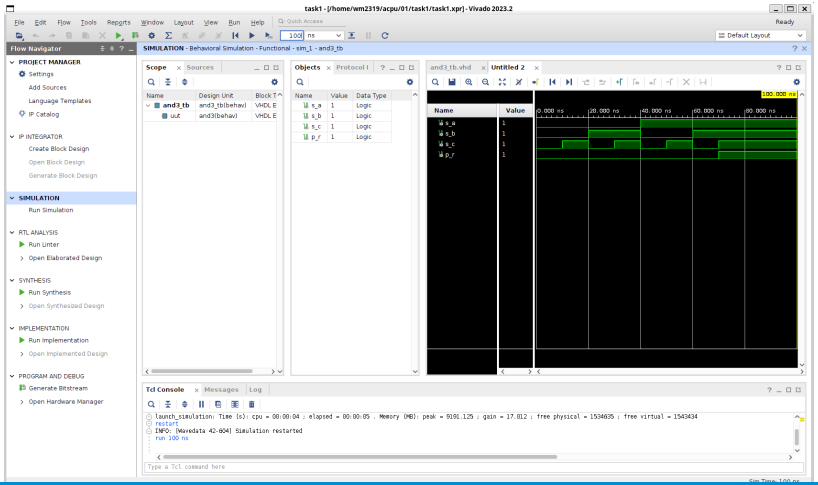
- 3 Inputs
- 1 Output
- Output = logical conjunction of all inputs



Task 1: The AND3 Gate

- Create a new project and set it up for the AND3 design.
- Add the file `and3.vhd` as a *Design File* to the project.
- Familiarize with the AND3 implementation by analyzing the VHDL.
- Elaborate the design. Compare the RTL layout to the VHDL implementation.
- Open the synthesized design and compare the Schematic to the previously explored RTL layout. What happened? How can you verify your observation?

Simulation and the Wave Viewer



Task 2: Simulation

- Now, we have a hardware design
- Next step: Ensure it is working correctly
- *Simulation* for *functional correctness*

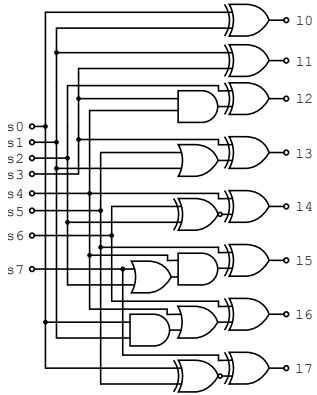
Task 2: Simulation

- Add the file `and3_wrapper.vhd` as a *Design File* to the project.
- Add the file `constraints_zedboard.vhd` as a *Constraint File* to the project.
- Implement the design and generate the bitstream.
- Setup the ZEDboard for USB-JTAG programming.
- Program the FPGA and verify the design by demonstration.

If you are using another FPGA (Nexys A7 or Nexys4), use the corresponding constraint file, no additonal power is needed.

Your first design

Implement this design:



Task Description

1. Implement the entity for the logic circuit. There are 8 inputs and 8 output. Name them according to the schematic.
2. Implement the architecture for the functional behavior of the logic circuit.
3. Check the functional correctness of your implementation via the provided testbench.
4. Using the provided `logic_wrapper.vhd`, synthesize and implement the design. Use the resulting bitstream to program an FPGA.
5. Use the switches (SW0 to SW7) to control the leds.
6. Can you solve the puzzle and manage that all 8 leds (LED0 to LED7) light up at once?