# Der PicoNut-Prozessor

## Ein freier, RISC-V-kompatibler Prozessor, entwickelt an der TH Augsburg

**Gundolf Kiefer, Johannes Hofmann und Daniel Dakhno**

*Efficient Embedded Systems* Group
Technische Hochschule Augsburg – University of Applied Sciences

https://ees.tha.de

**Partner:** IBV - Echtzeit- und Embedded GmbH & Co. KG

26. April 2025

(C) 2025 Gundolf Kiefer <gundolf.kiefer@tha.de>, CC BY-SA 4.0

# Übersicht

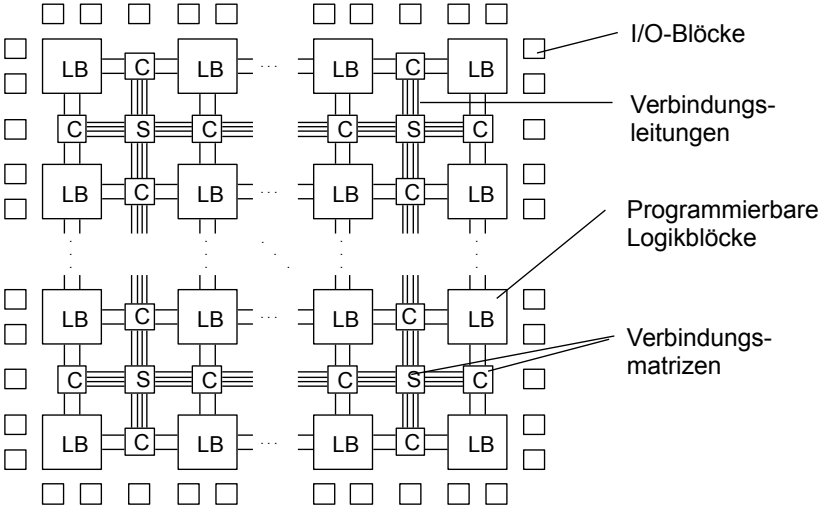
Kapitel 1

# Hintergrund und Projektüberblick

# Übersicht

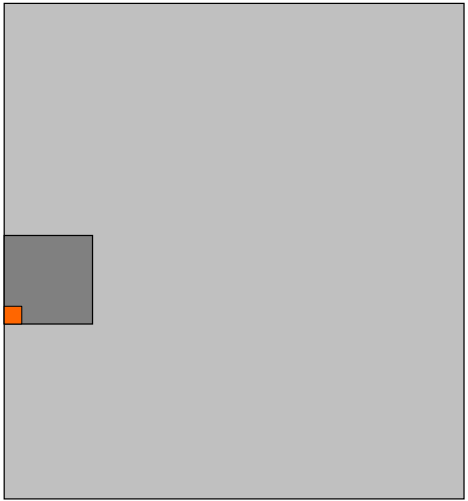
# Was ist ein FPGA?

## (Field-Programmable Gate Array)



# Kapazitäten gängiger FPGAs im Vergleich

- **High-End FPGA**
  - ~50.000.000 Gatter-Äquivalente
    (Xilinx Virtex UltraScale VU440: 2.532.960 LUT/FFpairs,
    1 LUT+2 FF = 20 GE, BRAM not counted)
- **Low-Cost FPGA**
  - ~2.000.000 Gatter-Äquivalente
    (Xilinx Spartan 6-LX150: 92152 LUT + FF pairs
    1 LUT+2 FF = 20 GE, BRAM not counted )
- **Intel 80386**
  - ~70.000 Gatter-Äquivalente
    (275.000 Transistoren)
  - Vollwertiger 32-Bit-Prozessor mit MMU, für Workstations entwickelt
  - *(Sehen Sie ihn?)*

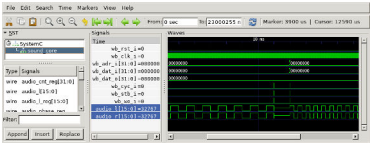


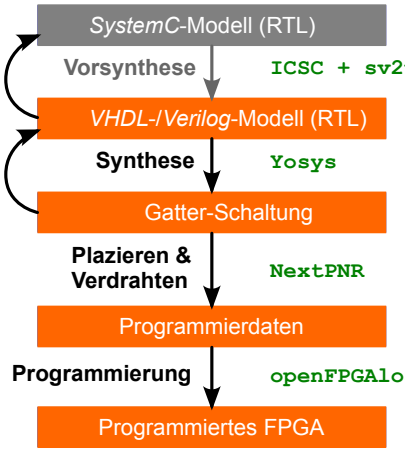
# FPGA-Entwicklung mit Open-Source-Tools

*OSS CAD Suite, Intel Compiler for SystemC (ICSC)*

**Simulation**

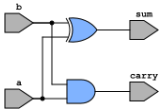
`GHDL/GCC, GtkWave`





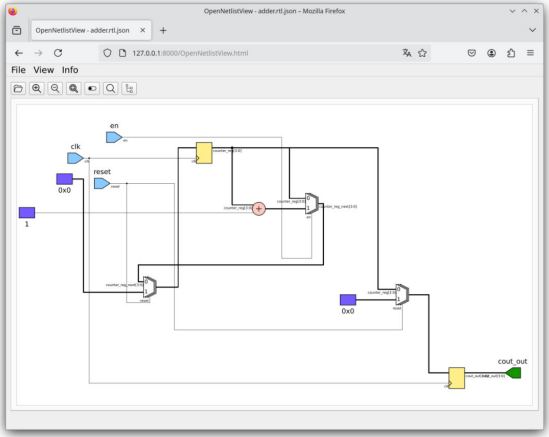

```
#include <systemc.h>

SC_MODULE (half_adder) {
  sc_in<bool> a, b;
  sc_out<bool> sum, carry;
```

```
entity HALF_ADDER is
  port (a, b: in std_logic;
        sum, carry: out std_logic);
end HALF_ADDER;
```



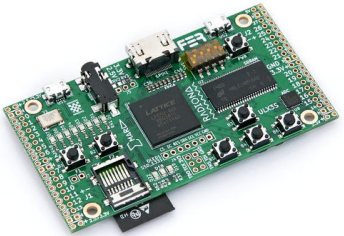

# Laufende Masterarbeit: *OpenNetlistView*

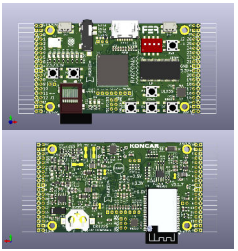
- **Bisher: *NetlistSVG***
  - generiert statische SVG-Dateien
  - kann nur ein Modul anzeigen
  - keine Interaktion mit Diagram
- ***OpenNetlistView*: Interaktive Anwendung (Qt6) mit Möglichkeit zum ...**
  - Markieren, Zoomen, ...
  - Navigieren durch Sub-Module
- **lauffähig im Web-Browser (WebAssembly)**
- **Ausblick:**
  - Verbesserung der Plazierung und Performance bei großen Schaltungen



# FPGA-Board: *Radiona ULX3S-03*

- **Open-Source-Hardware**
  - Projekt: `https://ulx3s.github.io/`
  - Open-Source-Toolkette: *Yosys*, *NextPnR*, ...
- **FPGA: *Lattice ECP5 85F* mit:**
  - Logik: 84K LUT/FF-Paare
  - 3,744 MBit Embedded RAM
  - 156 Multiplizierer (18 Bit)
- **32 MB SDRAM**
- **ESP32-Mikrocontroller**
- **Schnittstellen:**
  - 56 I/O-Pins (nutzerdefiniert, Pmod-kompatibel)
  - WLAN + Bluetooth
  - Video *(General Purpose Differential Interface, GPDI)*
  - USB-C
  - Audio, ADC, LEDs, Druckknöpfe, ...

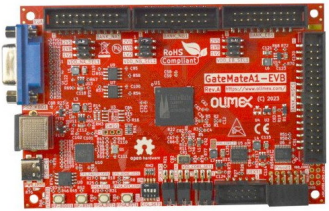


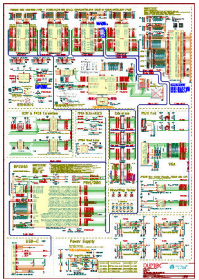


Quelle: https://ulx3s.github.io/

# FPGA-Board: *Olimex GateMateA1-EVB*

- **Open-Source-Hardware** (ca. EUR 50, April 2025)
  - `https://github.com/OLIMEX/GateMateA1-EVB`
- **FPGA: *Cologne Chip GateMate-A1* mit:**
  - Logik: 40K LUT4 + FF
  - 1,280 MBit Embedded RAM
  - → ***deutscher Hersteller mit Fertigung in Deutschland***
- **8 MB PSRAM**
- **RP2040** (zur Programmierung)
- **Schnittstellen:**
  - viele I/O-Pins (Pmod-kompatibel)
  - UEXT-Erweiterungsmodule (Olimex)
  - Video *(VGA)*
  - LEDs, Druckknöpfe, ...



*Quelle: https://github.com/OLIMEX/GateMateA1-EVB*

# Übersicht

# Was ist RISC-V?

- **Freie Befehlssatzarchitektur** (*Instruction Set Architecture, ISA*)
  - Basisbefehlssatz (***RV32I***, *RV64I*, *RV128I*, *RV32E*)
  - zahlreiche Extensions (z.B. *M, F, A, V, ...*)

- **Spezifikation:** `https://riscv.org/technical/specifications`

  ***The RISC-V Instruction Set Manual***
  - ***Volume 1: Unprivileged ISA***
  - *Volume 2: Privileged Architecture*

- **Software-Tools**: `https://riscv.org/exchange`
  - Compiler, Linker, Assembler, Debugger (*GCC, Binutils, GDB*)
  - Simulatoren
  - Bibliotheken (*newlib, glibc*)
  - Betriebssysteme (*Linux, FreeRTOS*) und Distributionen (*Yocto, Debian, FreeBSD, ...*)

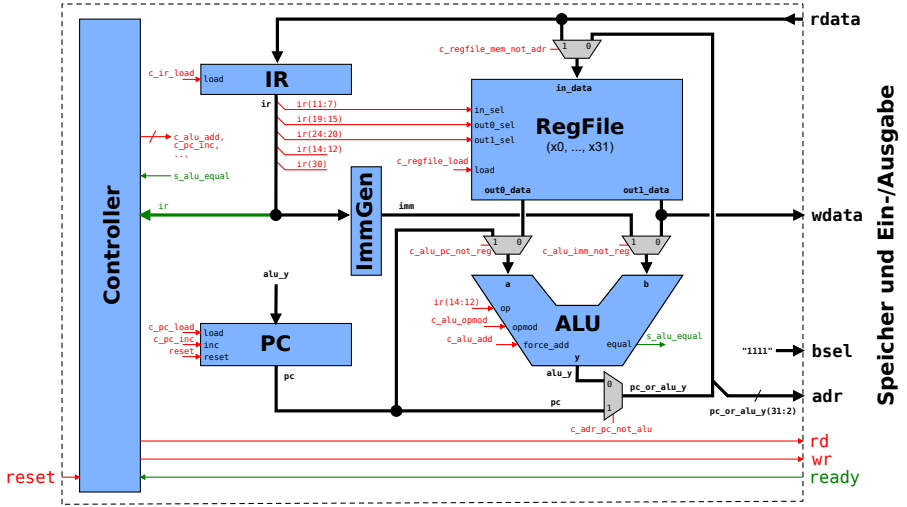
# Lehre an der THA: Entwurf digitaler Systeme

**Technische Informatik, Bachelor, 3. Semester**

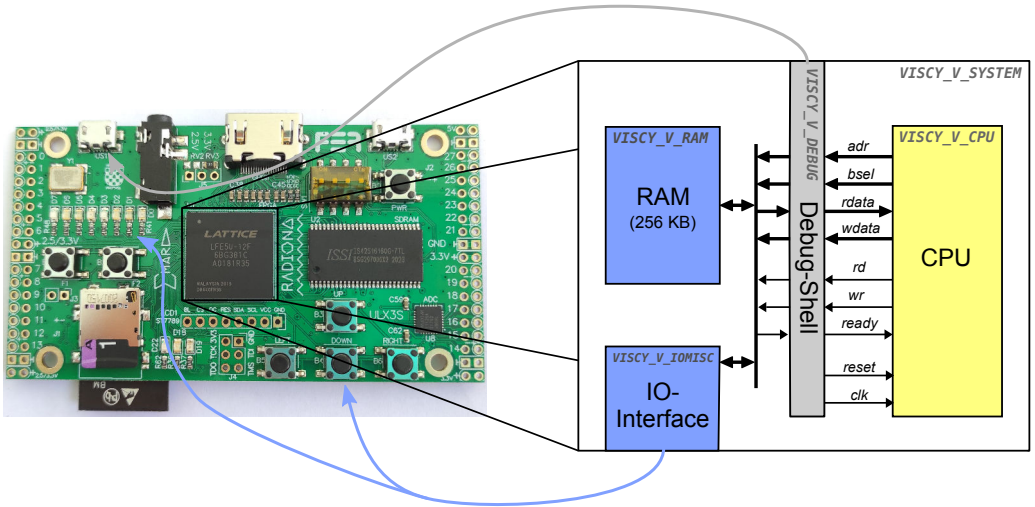
| Thema | Labor |
|---|---|
| Einführung in VHDL | Simulation *(GHDL)* |
| Schaltnetze | Entwurf und Simulation einer ALU *(GHDL)* |
| Schaltwerke | Entwurf und Simulation von Registern und Zählern (PC, IR, REGFILE) *(GHDL)* |
| Synthese | Synthese verschiedener Module *(Yosys, NextPNR, NetlistSVG)* |
| Automaten und Steuerwerke | Entwurf eines CPU-Steuerwerks |
| HW/SW-Schnittstelle | RISC-V-Programmierung *(GNU Assembler, Ripes)* |

# Struktur des VISCY-V-Prozessors

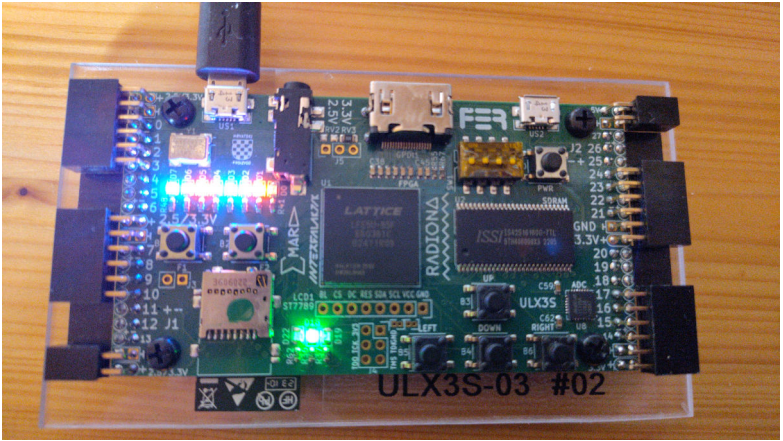
**Very Reduced Instruction Set Computer System based on RISC-V**



# Umgebung des VISCY-V-Prozessors



# Spiel: Schalte alle Lichter ein!

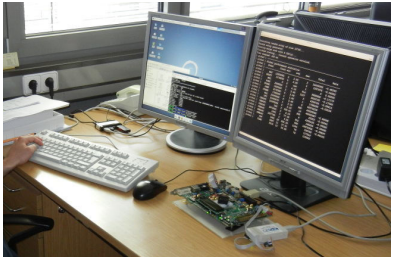



# Übersicht

# Rückblick: Die *ParaNut*-Architektur

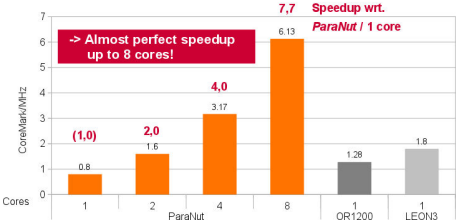
**2010, 2013:** **Entwurf der Architektur und Simulationsmodelle**

**2015:** **Erste funktionierende Hardware** (noch mit OpenRISC-Befehlssatz)



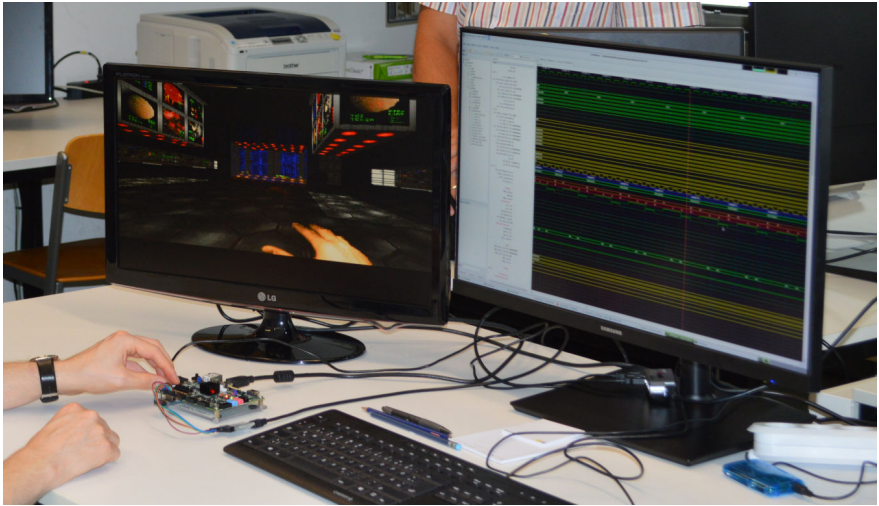
First Working Version (2015)

**Benchmark Results (2015):**




**2018:** **Umstieg auf RISC-V-Spezifikation**

# 2020: Auf einem *ParaNut* läuft ein 3D-Spieleklassiker und mehr ...

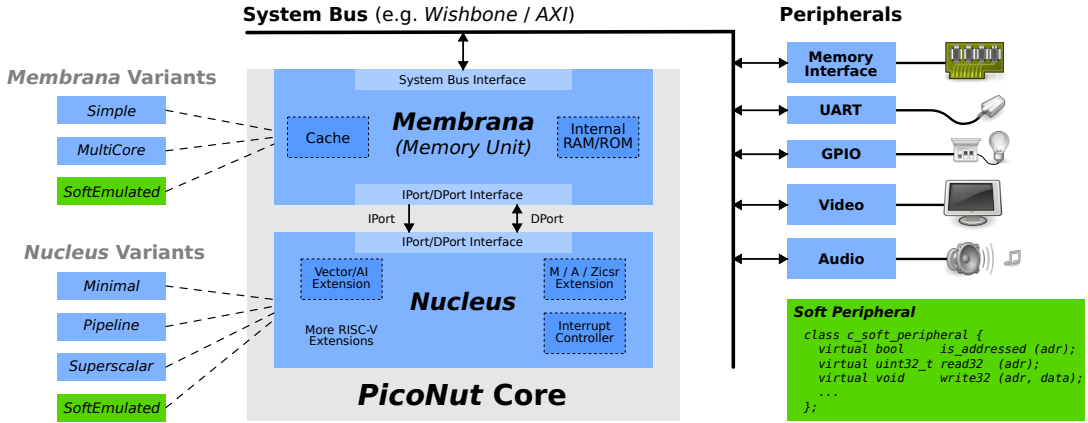


# 2024: Das *PicoNut*-Projekt

- **Minimaler, aber erweiterbarer RISC-V-Prozessor als Open-Source-Projekt**
- **Hardware-Entwicklung mit Open-Source-Tools und -Boards**
- **Modellierung in *SystemC* (C++)** (ermöglicht Simulation auf dem PC)
- **Gute Software-Unterstützung durch RISC-V-Kompatibilität:** GNU-Toolkette (GCC/GDB), newlib, FreeRTOS, Linux

**Philosophie: Von Studierenden für Studierende**

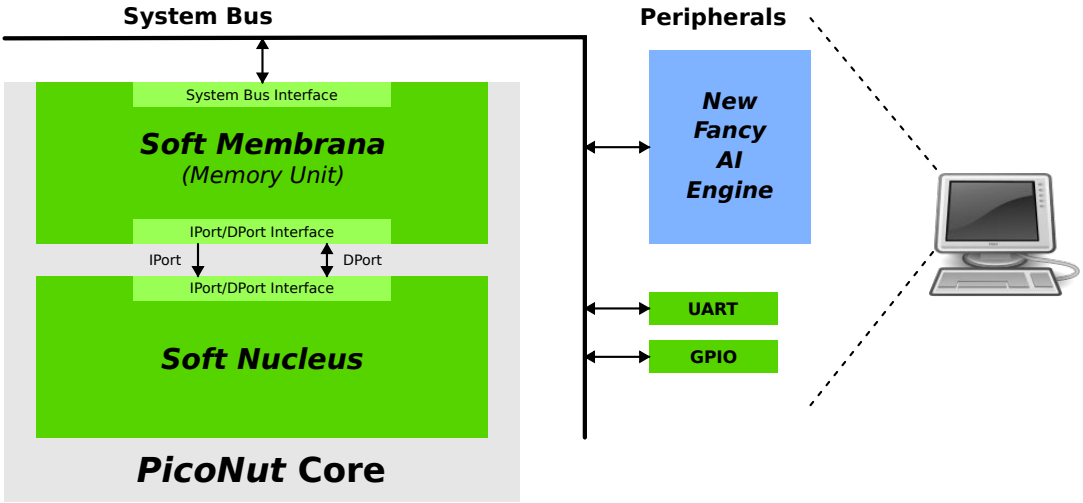
# *PicoNut*: Modulare Systemarchitektur



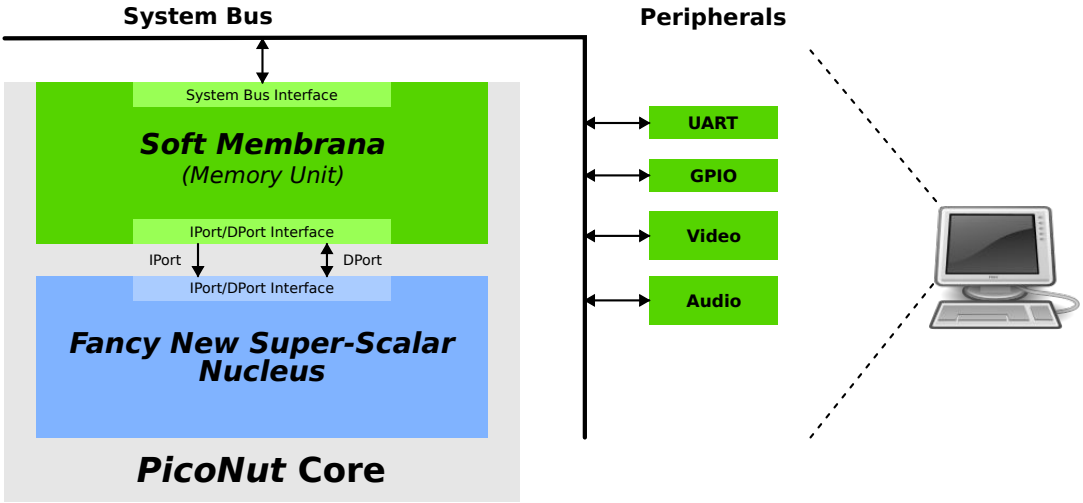
# *Soft Hardware:* Effiziente Simulationsmodelle

- **Simulation großer Hardware-Systeme ist sehr rechenaufwendig.**
- **Idee: Ersetze Module durch reine C++-Software-Modelle.**
  - ⇒ **schnelles Prototyping von neuer Peripherie**
  - ⇒ **Simulator/Emulator für die Software-Entwicklung**
  - ⇒ **hocheffiziente Hardware-Simulation**
    (nur relevant Module werden exakt simuliert).

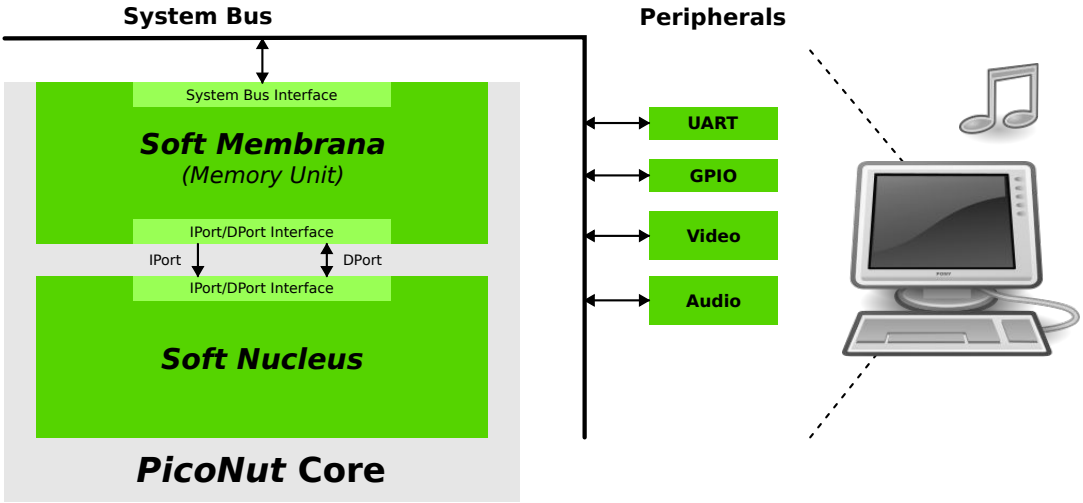
# Beispiel: Neuer Hardware-Beschleuniger



# Beispiel: Neue *Nucleus*-Variante



# Beispiel: Entwicklung von *PicoNut*-Software



# *PicoNut:* Projektstatus

- √ **Projektstart im Sommer 2024**
- √ **Basis-Architektur mit dokumentierten Schnittstellen**
- √ **"Hello World"-System funktioniert auf realer Hardware**
- √ **Professionelles Debugging (GDB, OpenOCD) im Simulationsmodell**
- **Professionelles Debugging (GDB, OpenOCD) mit realer Hardware**
- **Peripherie: Video, Audio, i2c, ...**

---

- ***PicoNut AI*: KI-Unterstützung (Vektor- und Matrix-Extensions)**
- **Betriebssystem-Unterstützung (FreeRTOS, MicroPython, Linux)**
- **Hardware-Erweiterungen** (z.B. Cache, MMU)
- **Alternative Architekturen** (z.B. Pipeline-Varianten)
- **...**

Kapitel 2

# Effizient entwickeln: Debug-Infrastruktur

Technische Hochschule Augsburg

# PICONUT

## EFFIZIENT ENTWICKELN: DEBUG-INFRASTRUKTUR

**Augsburger Linux-Infotag 2025**

**Johannes Hofmann – AUGSBURG 26.04.2025**

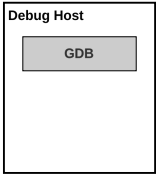
# MOTIVATION

- Essenziel für Entwicklung komplexerer Software auf dem PicoNut
  - Doom, FreeRTOS, Linux, etc …
- GDB erstmal nur für Desktop-Software
- OpenOCD ermöglicht eingebettetes Debugging mit GDB
- Hardwareunterstützung erforderlich

# ON-CHIP-DEBUGGING: OPENOCD UND GDB

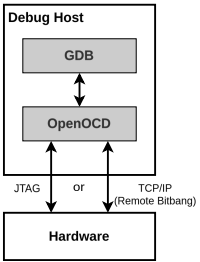
## ↗ GDB

- De-facto-Standard-Debugger von Linux-Systemen
- Programm anhalten / fortsetzen, Breakpioints setzen
- Speicher und Register auslesen / manipulieren



# ON-CHIP-DEBUGGING: OPENOCD UND GDB

- OpenOCD
  - Begonnen an der Hochschule Augsburg von Dominik Rath (IBV)
  - Schnittstelle zwischen GDB und Hardware
  - Hardwareschnittstellen: JTAG, SWD und Remote Bitbang

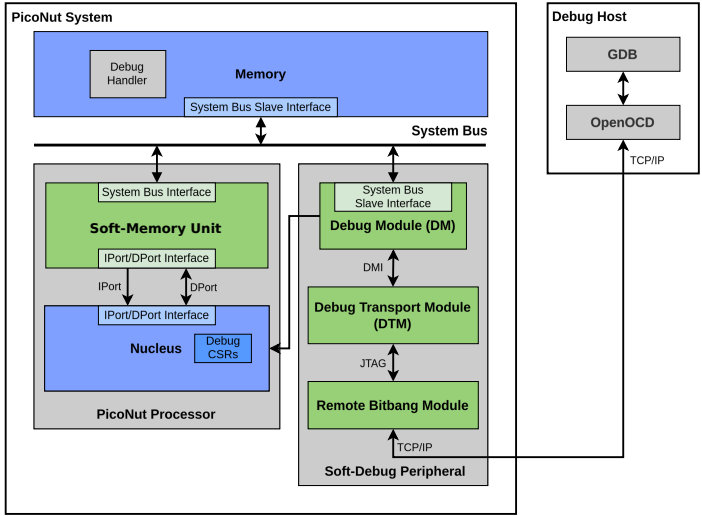



- Remote Bitbang
  - JTAG-Signale über TCP/IP Verbindung
  - Für Räumlich entfernte Hardware oder **Hardwaresimulation**

# RISC-V-DEBUG-SPEZIFIKATION

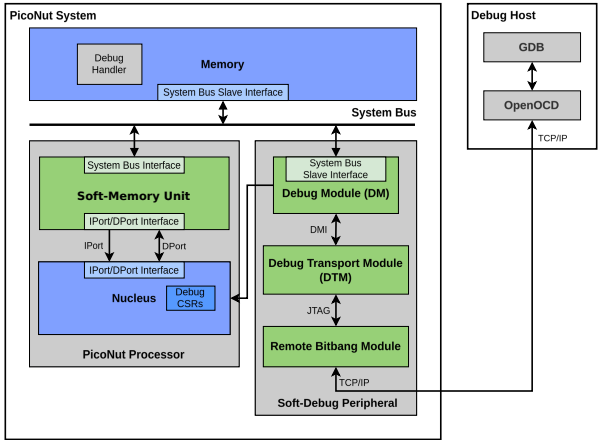
- Spezifikation: RISC-V External Debug Support v0.13.2
  - Seit Februar 2025 Release 1.0. Ist aber sehr ähnlich
- Ziel: Hardwareunabhängiges Debugging von RISC-V-Prozessoren
- Unterstützung durch OpenOCD

# DEBUG-INFRASTRUKTUR IM PICONUT



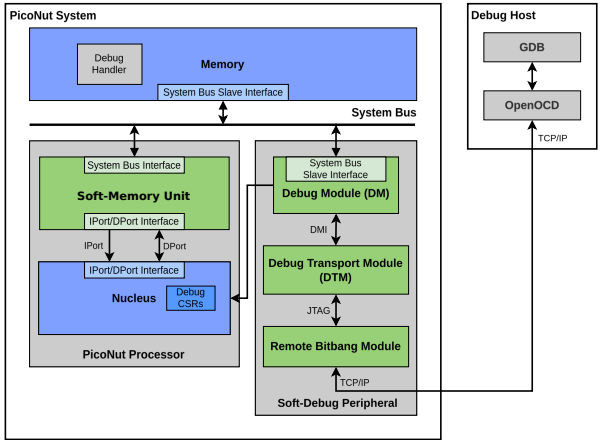
# DEBUG-INFRASTRUKTUR IM PICONUT

- Remote Bitbang Module
  - Übersetzt Bitbang in JTAG
- Debug Transport Module
  - JTAG Standard + Erweiterung
  - Schnittstelle zu Debug Module



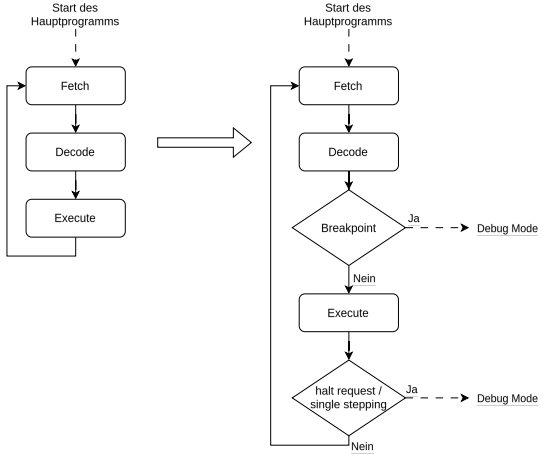
# DEBUG-INFRASTRUKTUR IM PICONUT

- Debug Module
  - Steuert Nucleus
  - Kommunikation mit Nucleus
- Debug Handler
  - Routine im Debug Mode
  - Vergleichbar mit einem Exception Handler

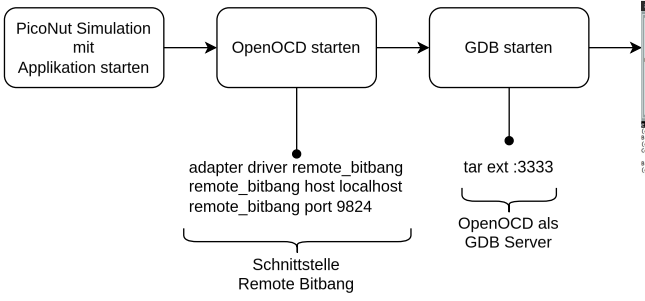


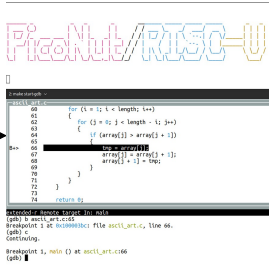
# DEBUG-INFRASTRUKTUR IM PICONUT:
## WIE KOMME ICH IN DEN DEBUG MODE?

- Breakpoint
- Halt Request
- Single Stepping



# DEBUGGING IM SIMULATOR:
## BENUTZUNG



# DEMONSTRATION

Hier kommt die Demonstration

# DEBUGGING IM SIMULATOR:
## VORTEILE

- Keine FPGA-Hardware notwendig

  → Einfaches Debuggen von Software für RISC-V Architektur

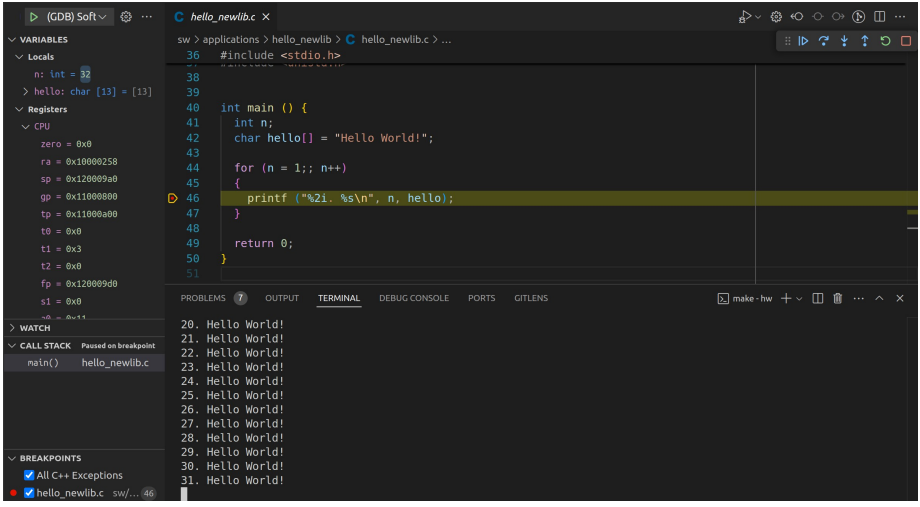
- Extra Hardware für Debugging muss nicht simuliert werden

  → Schnelle Simulationen möglich

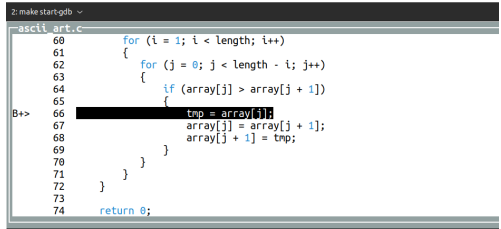
- Effizient durch einfache Benutzung

  → Viele verschiede Frontends möglich (TUI, VS Code, ...)

# DEMONSTRATION



# DEMONSTRATION



# QUELLENVERZEICHNIS

[1] OpenOCD User's Guide https://openocd.org/doc-release/pdf/openocd.pdf (zuletzt besucht am 14.04.2025)

[1] GDB: GNU Debugger https://www.sourceware.org/gdb/ (zuletzt besucht am 14.04.2025)

[1] RISC-V Debug Spezifikation https://drive.google.com/file/d/1h_f9NgB_8m2fS6uCnKP1Oho-3x1MpBEl (zuletzt besucht am 14.04.2025)

[1] RISC-V Logo https://lf-riscv.atlassian.net/wiki/download/attachments/65546/atl.site.logo (zuletzt besucht 14.04.2025)

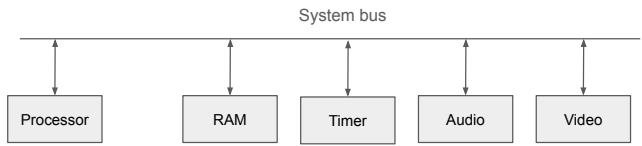
Kapitel 3

# Software-Modelle für Video- und Audio-Peripherie

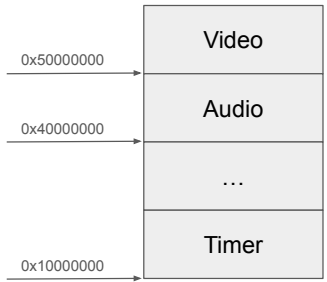
# Software Peripherie

- Audio
- Video

# Peripheral interface



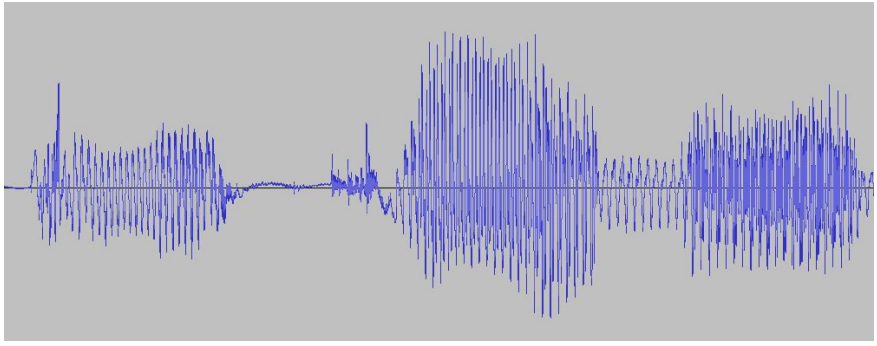
# Adressraum



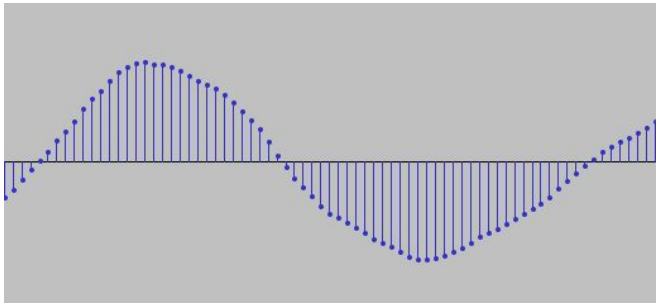
# Software interface

```
class c_soft_peripheral
{
public:
    uint64_t base_address;

    virtual uint32_t read32(uint64_t adr);
    virtual void write32(uint64_t adr, uint32_t data);
    virtual void on_reset();
};
```

“PicoNut”



44100 Samples per second



# Variations

- Bitrate
  - 44100
  - 10000
- Mono / Stereo
- Format
  - 8 Bit
  - 24 Bit
  - Float

# Example

```
65  float sin_datapoints[441] = {
66      0.0, 0.014246, 0.028493, 0.042740, 0.056986, 0.071233, 0.085480, 0.099726,
67  };
```

- 44100 Samples per second
- 441 Sample per sine wav
- = 100Hz

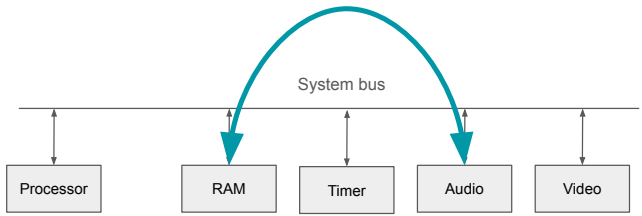
# Register

| Offset | Name |
|---|---|
| 0x00 | Control register |
| 0x04 | Status register |
| 0x08 | Cycles per sample |
| 0x0C | Read head position |
| 0x10 | Current buffer start |
| 0x14 | Current buffer length |
| 0x18 | Next buffer start |
| 0x1C | Next buffer length |

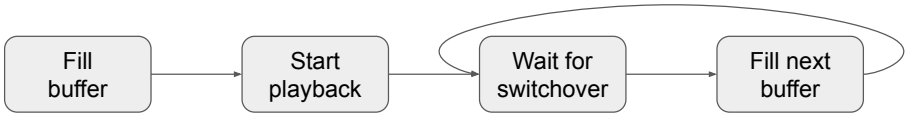
Playback, Stereo, …

Data information

# DMA

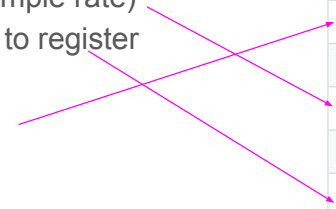


# Double buffering



# Usage

1. Fill buffer
2. Set bitrate (CPU freq / sample rate)
3. Write buffer address/size to register
4. Enable/disable features
    a. Sample format
    b. Mono/stereo
    c. Double buffer
5. Start playback
6. Wait for buffer interrupt
7. Fill next buffer



| Offset | Name |
|---|---|
| 0x00 | Control register |
| 0x04 | Status register |
| 0x08 | Cycles per sample |
| 0x0C | Read head position |
| 0x10 | Current buffer start |
| 0x14 | Current buffer length |
| 0x18 | Next buffer start |
| 0x1C | Next buffer length |

# Video

Picture:

- Picture width/height
- Pixel format
  - Single/multi color
  - Color depth
  - Color sequence (RGB, BGR, …)

# Video registers

| Offset | Name |
|---|---|
| 0x000 | CONTROL |
| 0x004 | STATUS |
| 0x008 | RESOLUTION_MODE |
| 0x00C | RESOLUTION_MODE_SUPPORT |
| 0x010 | COLOR_MODE |
| 0x014 | COLOR_MODE_SUPPORT |
| 0x018 | FRAMEBUFFER |
| 0x01C | SCANLINE |
| 0x100 | COLOR_MAP |

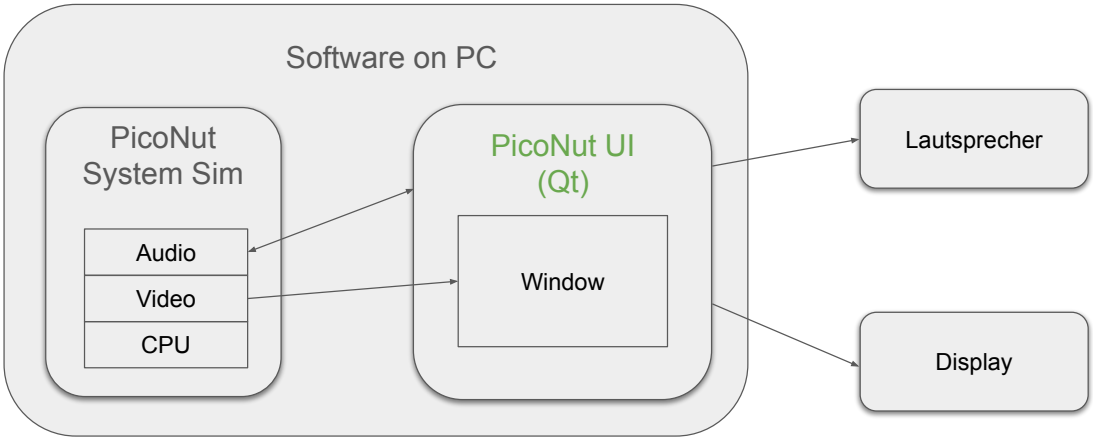
640x480, 800x600 , 1024x768

1 Bit Monochrome, 2-Bit Mapped, 3-Bit RGB

# Usage example

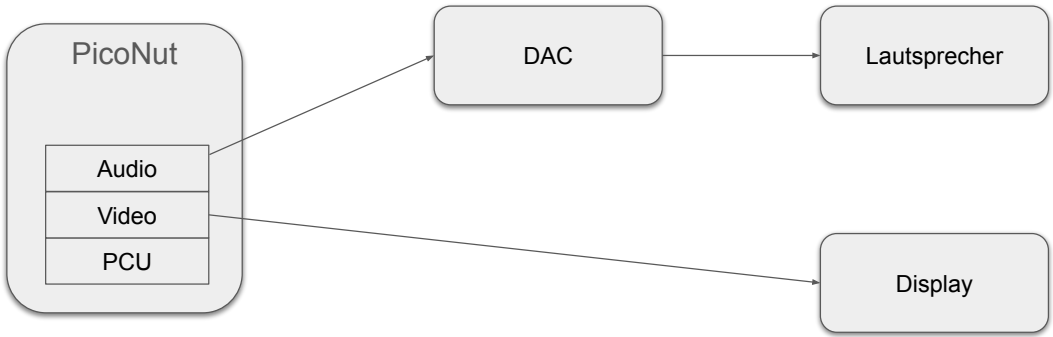
1. Check resolution, color mode
2. Select resolution, color mode
3. Enable image output
4. Fill framebuffer
5. Await image output
6. Re-fill framebuffer

# Realisation via QT



Example

# Hardware



# Vorteile

- Peripheral simulieren
- Audio / Video in Sim anzeigen
- Prototypen
- Performance / besser als VHDL

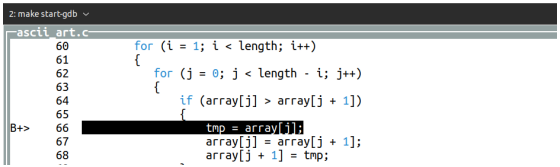
Kapitel 4

# Fazit

# Fazit

- **RISC-V ist eine freie Befehlssatzarchitektur für Prozessoren und damit ein Schlüsselelement für digitale Souveränität in Deutschland und Europa.**
- ***PicoNut/RISC-V* ist ein freies Projekt an der TH Augsburg, von Studierenden für Studierende.**
- **Open-Source-Tools und -Hardware zur FPGA-Entwicklung werden im Projekt und in der Lehre eingesetzt.**
- **Jede(r) Interessierte kann mitmachen und -lernen.**

# Vielen Dank! – Fragen?




https://ees.tha.de/piconut