Hochschule Augsburg
University of Applied Sciences

Linux auf dem ParaNut/RISC-V-Prozessor

Open-Source-Software trifft auf Open-Source-Hardware

Christian Meyer
Felix Wagner

Forschungsgruppe Effiziente Eingebettete Systeme

Agenda

Teil A: Allgemeines

1. Das EES-Team
2. RISC-V
3. Der ParaNut-Prozessor
4. SystemC

Teil B: Linux

1. Speicherverwaltung
2. Linux auf dem ParaNut
3. ToDo‘s

Forschungsgruppe Effiziente Eingebettete Systeme

Michael Schäferling, Oleg Murashko, Siegfried Kienzle, Abdurrahman Celep, Prof. Dr. Gundolf Kiefer, Patrick Mihleisen, Elias Schuler, Marco Milenkowic, Lukas Bauer, (Daniel Bortkevyvh, Haris Vojic, Mahdi Mahdavi)

Sponsor

ibv
Realtime is **BLUE**

- Software-Dienstleister
  - Echtzeit
  - Embedded
- Finanzierung von
  - Abschlussarbeiten
  - Open Source Projekte

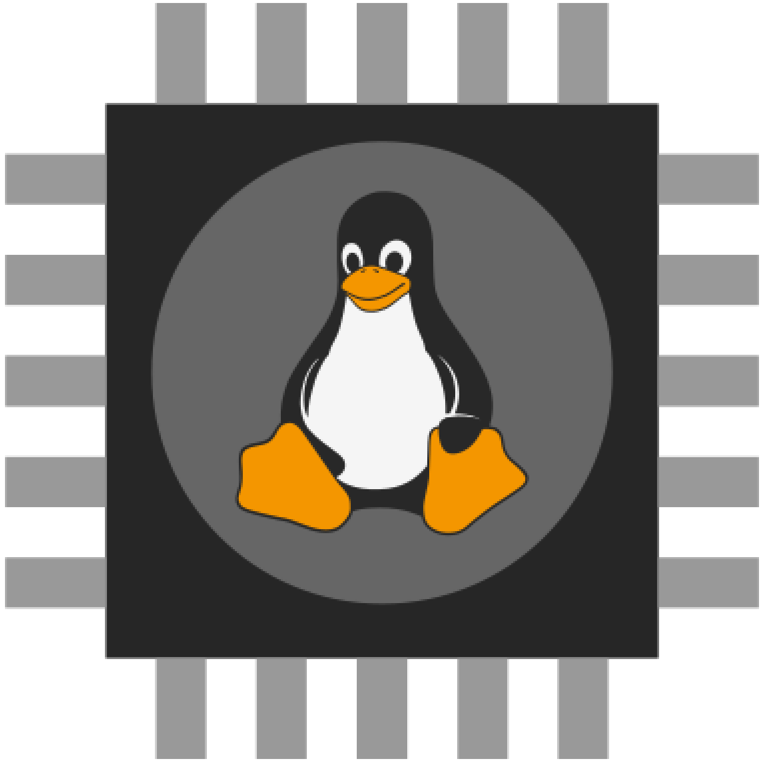
Christian Meyer

Master (Technische) Informatik

Interessen:
Linux, Betriebssysteme, Kommunikation

Aufgabe am ParaNut: Linux und MMU

Seit 2023: Software Ingenieur bei IBV



Felix Wagner

Bachelor Elektrotechnik

Interessen:
Embedded, Audiotechnik

Aufgabe am ParaNut:
Maintenance, Pthreads, AES

ParaNut

ParaNut

offen

ParaNut

offen, skalierbar

ParaNut

offen, skalierbar, RISC-V Prozessor

ParaNut

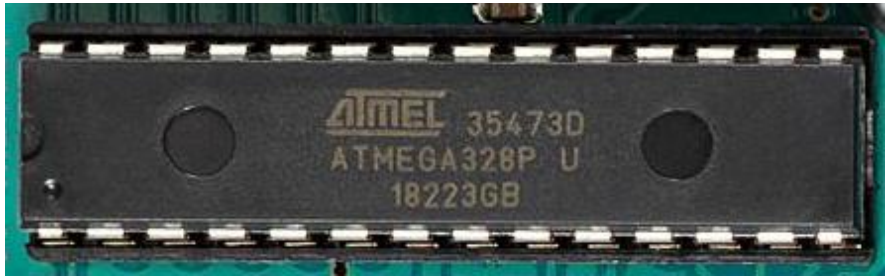
offen, skalierbar, RISC-V Prozessor für FPGAs

RISC-V

Befehlssatzarchitektur

Hochschule Augsburg
University of Applied Sciences

x86-64



AVR

RISC-V

RISC-V®

Cryptographic Extensions Task Group

Extensions

✓ RV32I - das Base Integer Instruction Set

✓ M - Standard Extension for Integer Multiplication and Division

(✓) A - Atomic

Der ParaNut-Prozessor

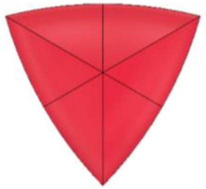
Das Parallelitäts-Konzept

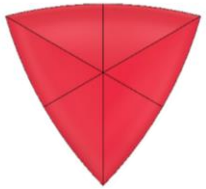
SMT & SIMD

SMT - Simultaneous Multi Threading


intel
CORE i7
X-series


AMD
RYZEN





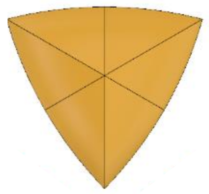
```
a[0] = a[0] * b[0]
c[0] = a[0] / 2
a[0] = a[0] ^ c[0]
b[0] = b[0] + 5
x[0] = x[0] ^ 2
y[0] = y[0] / 2
a[0] =    2 + 2
```

```
a[1] = b[1] - c[1]
b[1] = c[1] * 5
a[1] = a[1] ^ c[1]
x[1] = c[1] + 6
a[1] = x[1] + 1
y[1] = y[1] ^ 4
a[1] = y[1] - x[1]
```

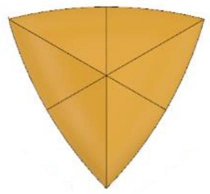
SIMD - Single Instruction Multiple Data

SIMD

SIMD

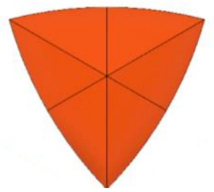


```
a[0] = a[0] + b[0]
c[0] = a[0] / 2
a[0] = a[0] ^ c[0]
b[0] = b[0] + 5
x[0] = x[0] ^ 2
y[0] = y[0] / 2
a[0] =    2 + 2
```



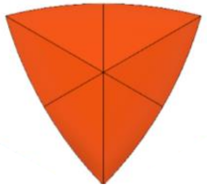
```
a[1] = a[1] + b[1]
c[1] = a[1] / 2
a[1] = a[1] ^ c[1]
b[1] = b[1] + 5
x[1] = x[1] ^ 2
y[1] = y[1] / 2
a[1] =    2 + 2
```

Capability Level

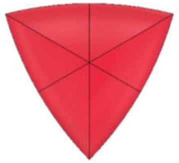


3 – Central Processing Unit (CePU)

Capability Level

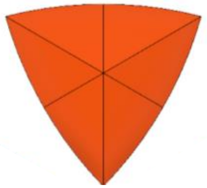


3 – Central Processing Unit (CePU)

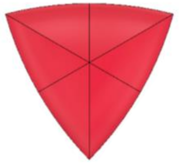


2 – CoPU: Threaded & Linked Mode

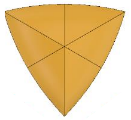
Capability Level



3 – Central Processing Unit (CePU)

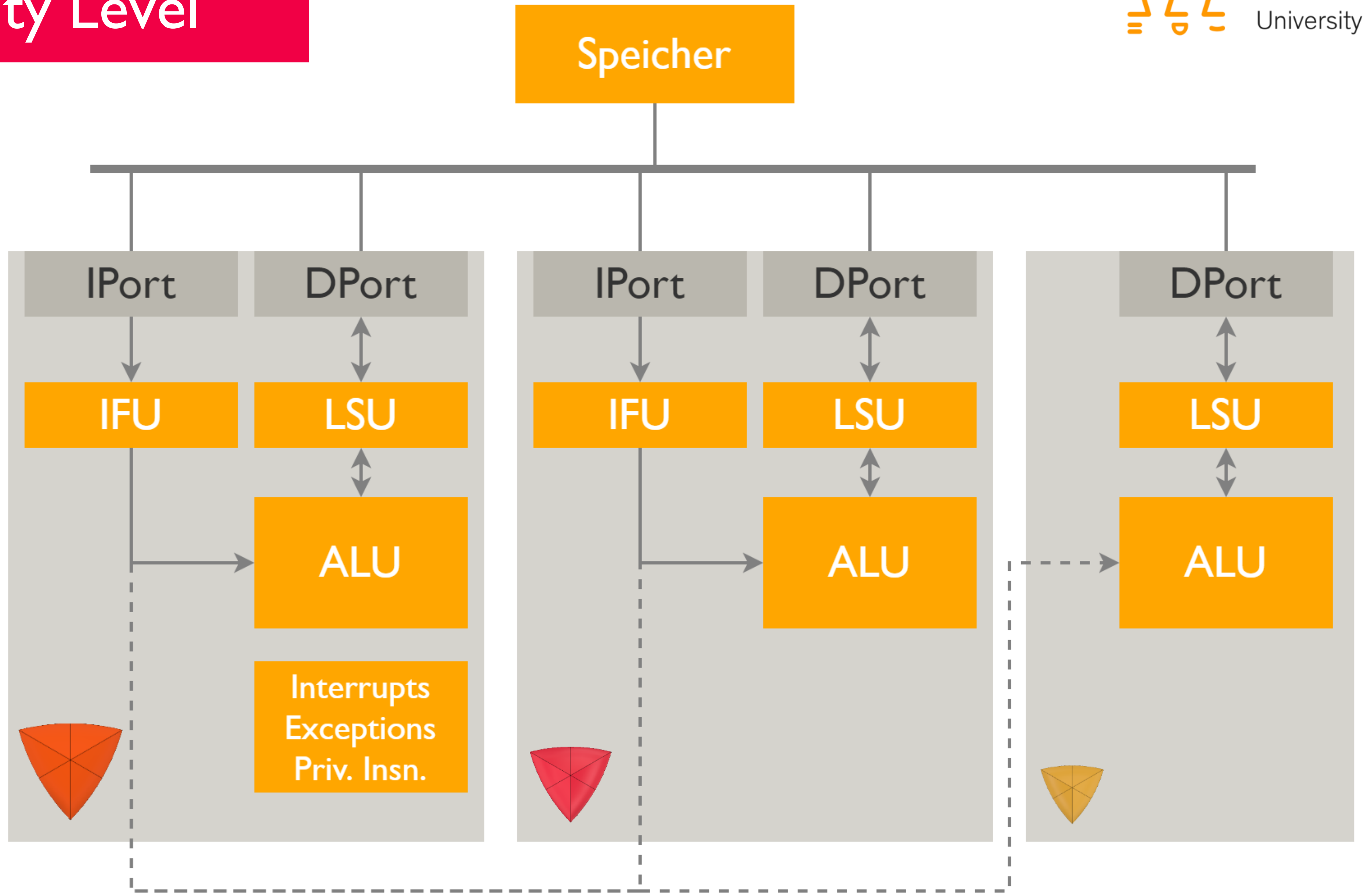


2 – CoPU: Threaded & Linked Mode



1 – CoPU: Linked Mode

Capability Level



Libparanut

Threaded Mode


```c
#include <libparanut.h>
int in[] = {1, 4, 5, 6, 23, 234, 5, 62, 4, 2, 11, 2, 32, 63, 7, 86};
int *out;


int main(){
  out = malloc(16*sizeof(int));

  uint id = PN_BEGIN_THREADED(4);
    for (uint n = id; n < 16; n += 4)
      if(in[n]%2 != 0)
        out[n] = in[n]*in[n];
  PN_END_THREADED();

  for(uint n = 0; n<16;n++){
    printf("%d: %d^2 = %d\n", n, in[n], out[n]);
  }
}
```

Linked Mode

Hochschule Augsburg
University of Applied Sciences

```c
#include <libparanut.h>
int in[] = {1, 4, 5, 6, 23, 234, 5, 62, 4, 2, 11, 2, 32, 63, 7, 86};
int *out;


int main(){
  out = malloc(16*sizeof(int));

  uint id = PN_BEGIN_LINKED(4);
    for (uint n = id; n < 16; n += 4)
      out[n] = ((in[n]/2)%2)*in[n]*in[n];
  PN_END_LINKED();

  for(uint n = 0; n<16;n++){
    printf("%d: %d^2 = %d\n", n, in[n], out[n]);
  }
}
```

FPGAs

Field Programmable Gate Array

Programmierbare Logik

Einsatzgebiete

- Digital Signal Processing
- Spezialisierte Hardware Algorithmen
- Kleinserien
- Softcore Prozessoren

Hardware-Beschreibungs-Sprachen

- Verilog
- VHDL

Verilog Beispiel

Hochschule Augsburg
University of Applied Sciences

```
library ieee;
use ieee.std_logic_1164.all;

entity simple_and is
  port (
    input_1    : in   std_logic;
    input_2    : in   std_logic;
    output : out std_logic
    );
end example_and;

architecture rtl of simple_and is
  signal and_gate : std_logic;
begin
  and_gate   <= input_1 and input_2;
  output <= and_gate;
end rtl;
```

Register
Transfer
Level

SystemC

High Level Synthesis

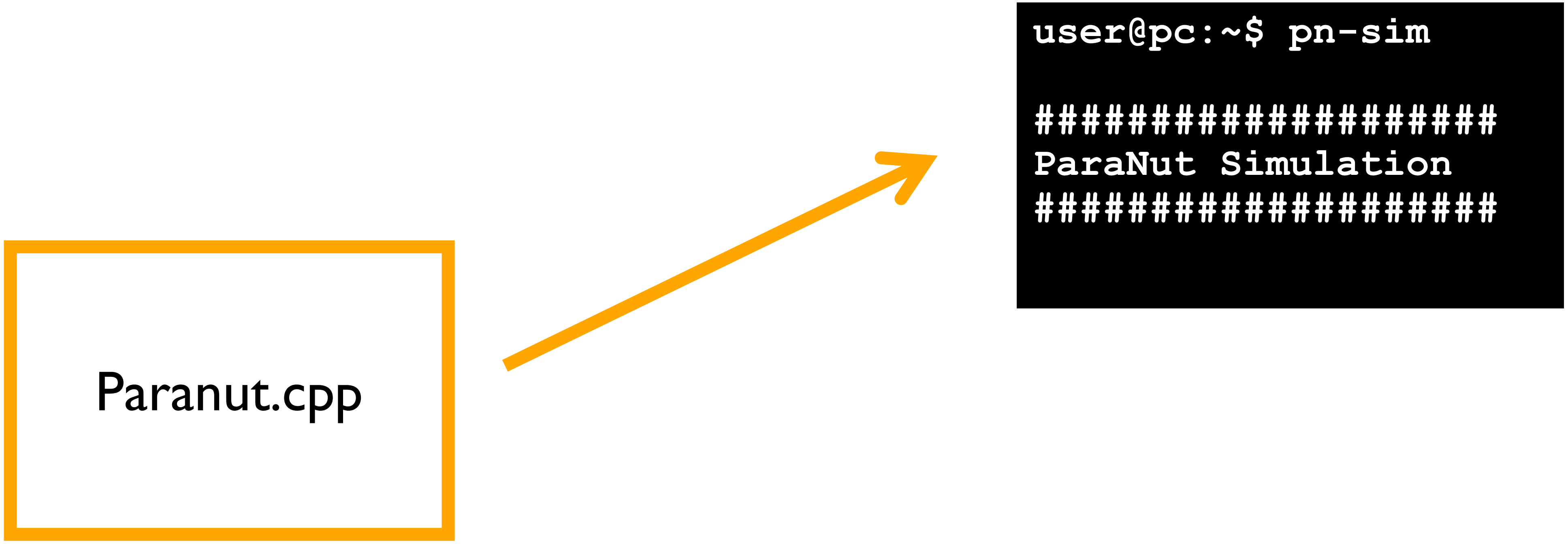
- **Bibliothek für C++**
- Definiertes C++ Subset
- Höhere Abstraktion als RTL
- Performante Simulation

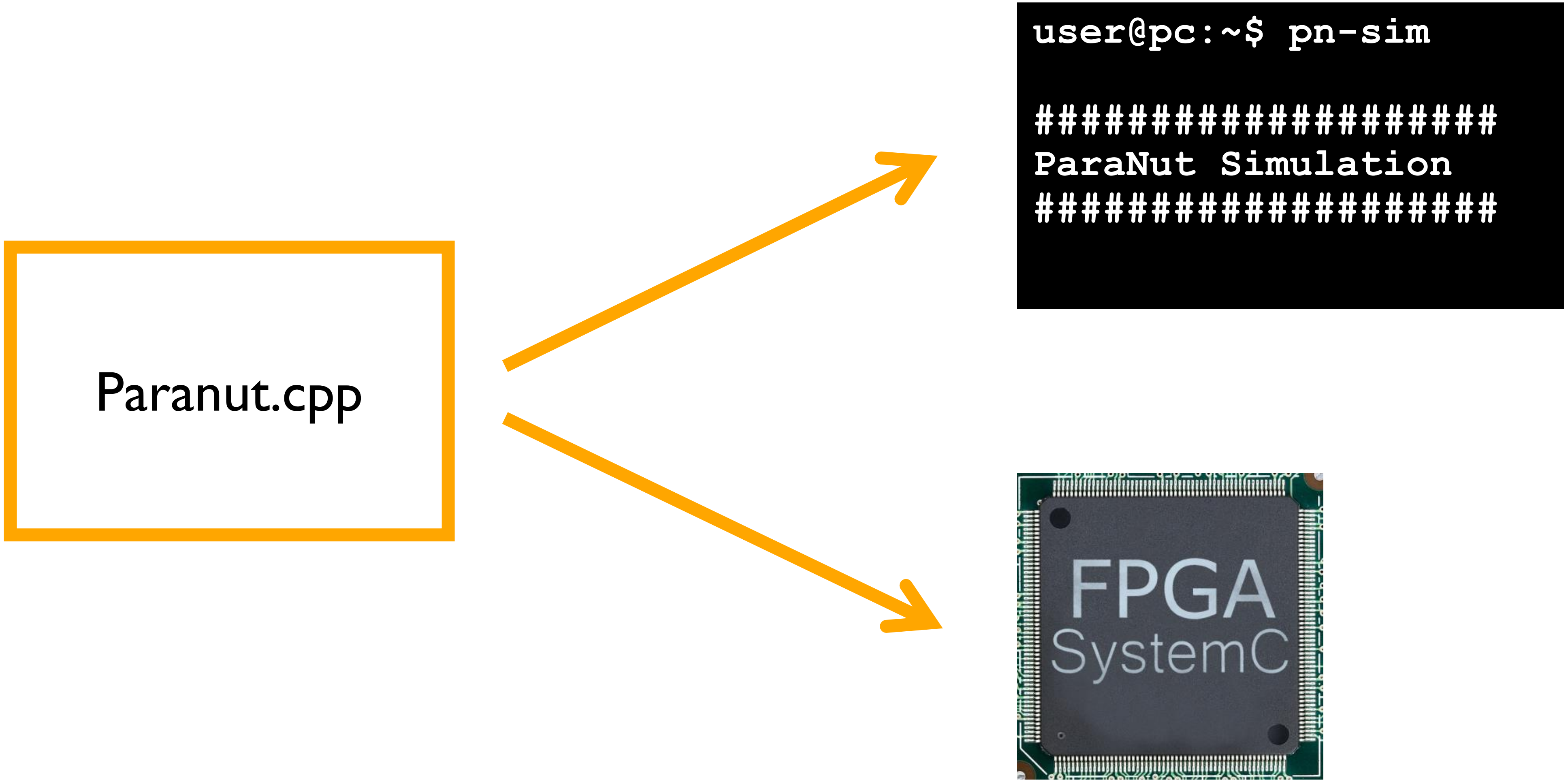
SystemC Beispiel

Hochschule Augsburg
University of Applied Sciences

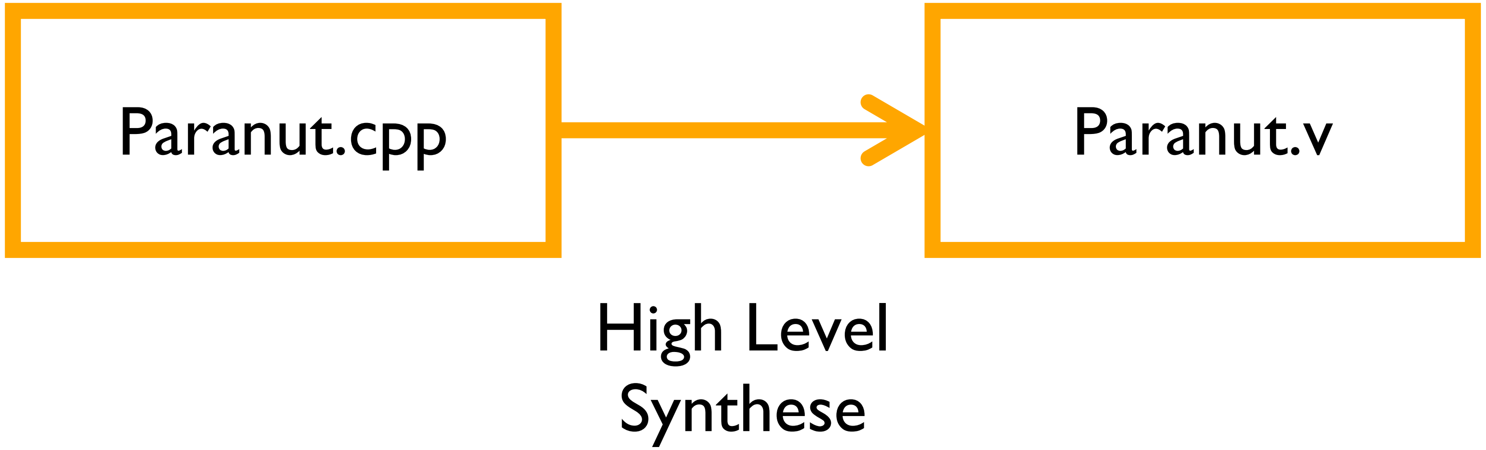
```
#include "systemc.h"
SC_MODULE("And")
{
  sc_in<bool> input_1, input_2;      // input signal ports
  sc_out<bool> output;               // output

  void do_and(){
    output.write( (input_1.read() && input_2.read()) );
  }

  SC_CTOR("And"){
    SC_METHOD(do_and);                // register method
    sensitive << input1 << input2;   // sensitivity list
  }
};
```

SystemC

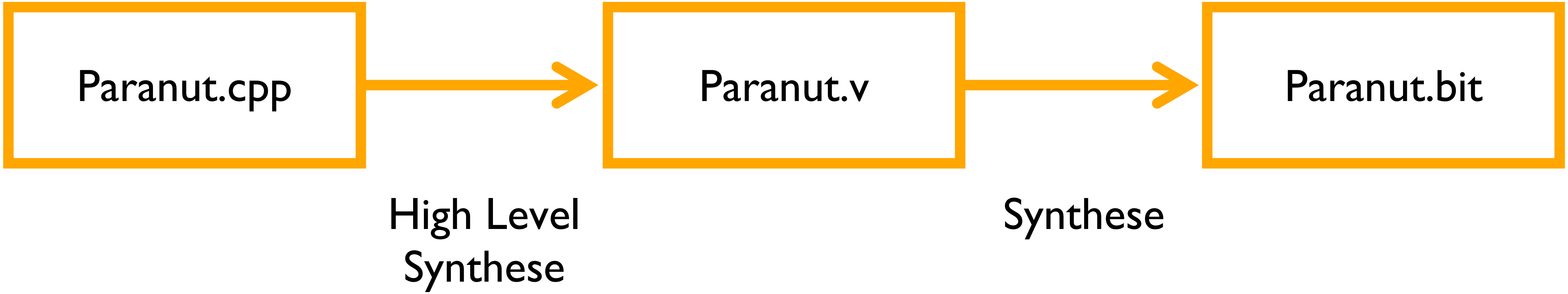



Paranut.cpp
user@pc:~$ pn-sim
###################
ParaNut Simulation
###################
FPGA
SystemC

Synthese

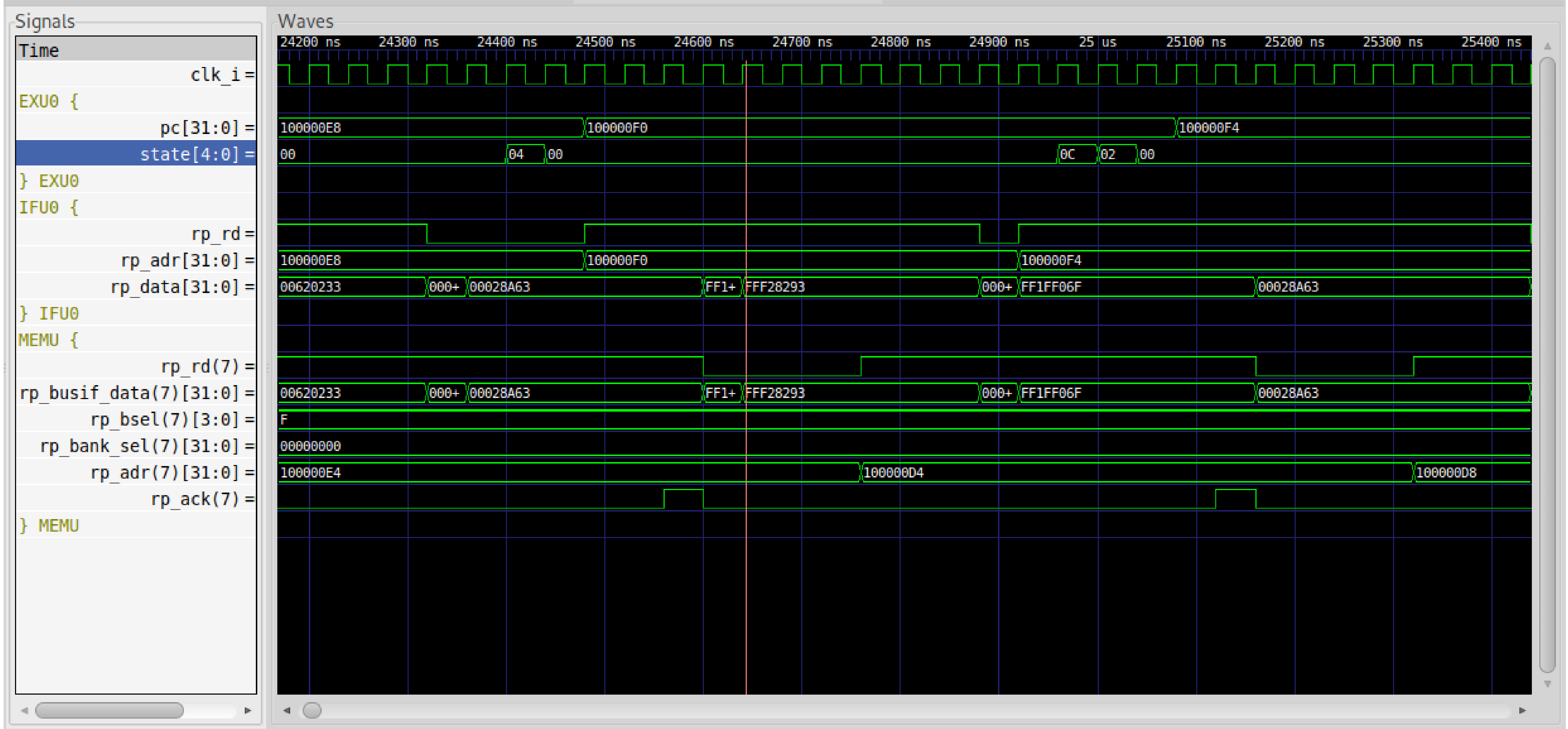


Synthese


Paranut.cpp
Paranut.v
Paranut.bit
High Level
Synthese
Synthese

Fehlersuche

Hardware Debugging



Hardware Debugging


```
user@pc:~$ gdb pn-sim

####################
ParaNut Simulation
####################
```

→ gdb

Software Debugging

JTAG

OpenOC
D

gdb

Software Debugging

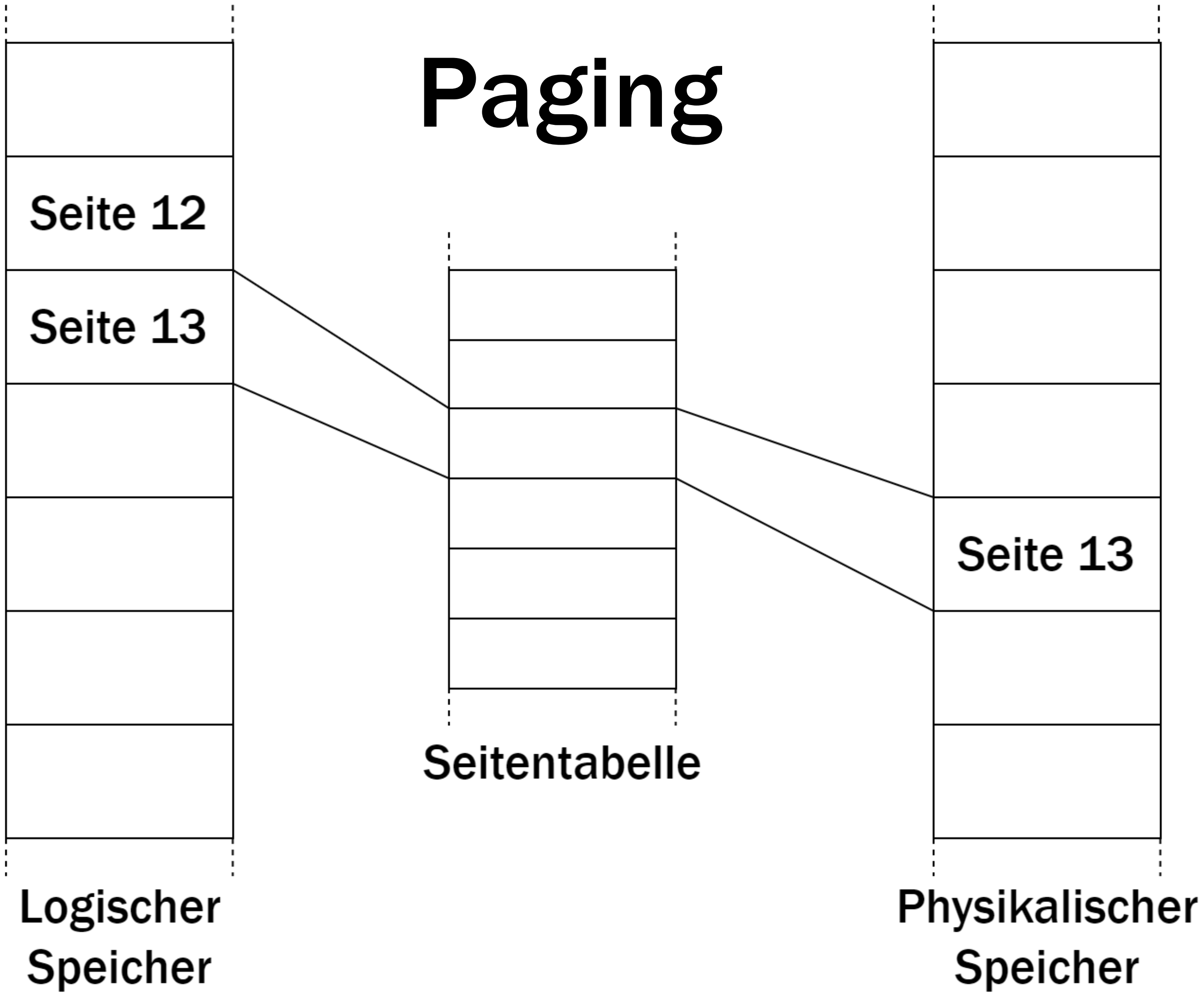
Teil B: Linux

1. Speicherverwaltung

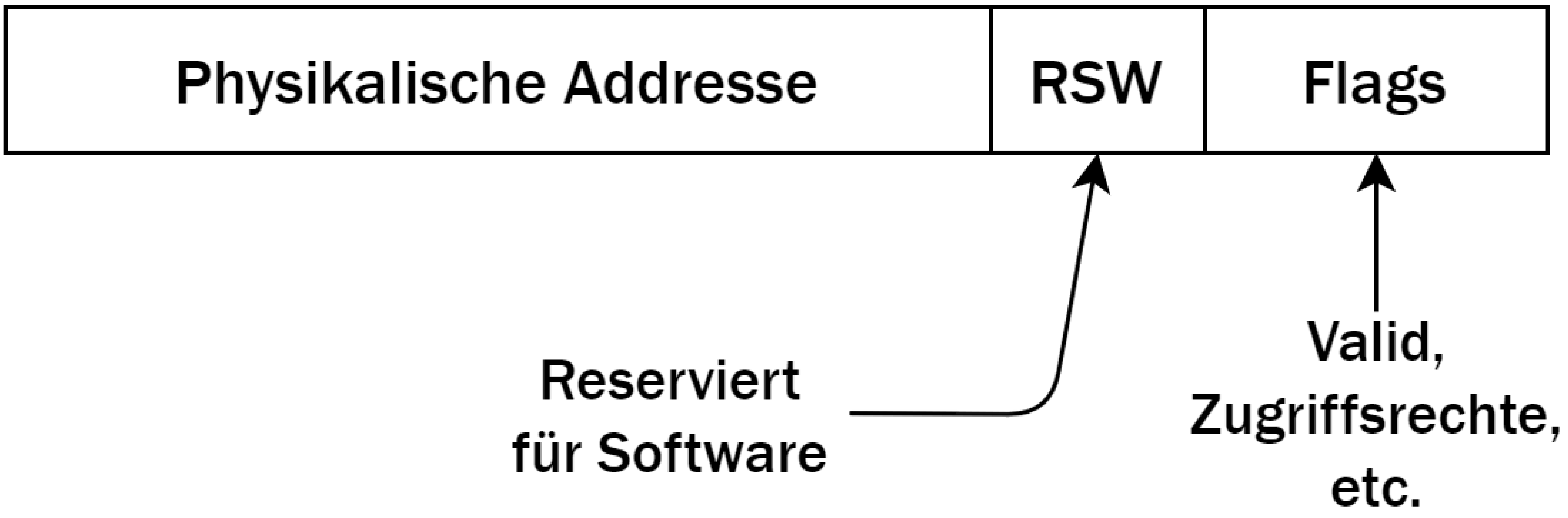
Wer kennt es?

```
user@host:~$ ./selbstzerstörung
Segmentation Fault (core dumped)
```

Paging


Seite 12
Seite 13
Seitentabelle
Seite 13
Logischer Speicher
Physikalischer Speicher

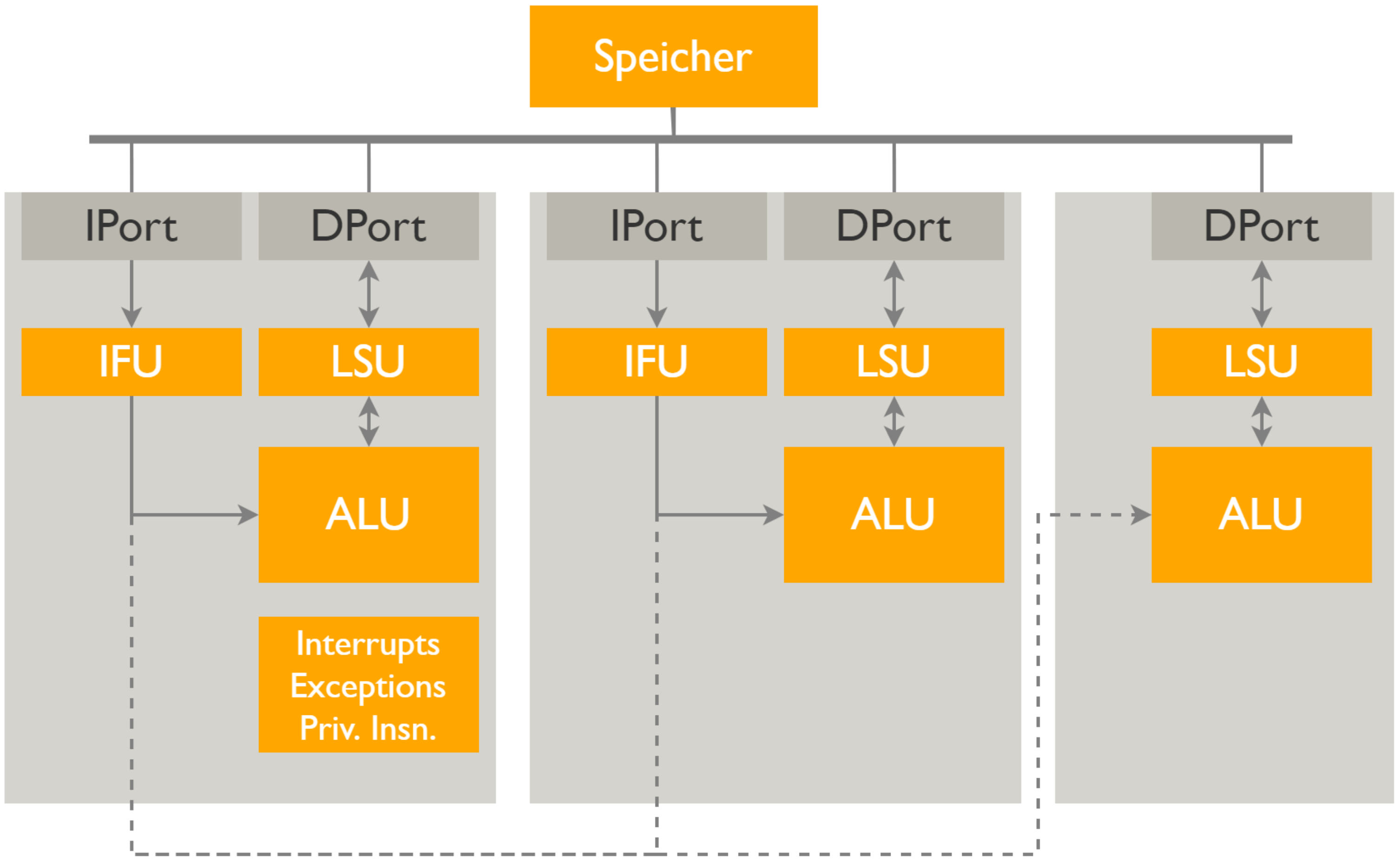
Seitentabelleneintrag



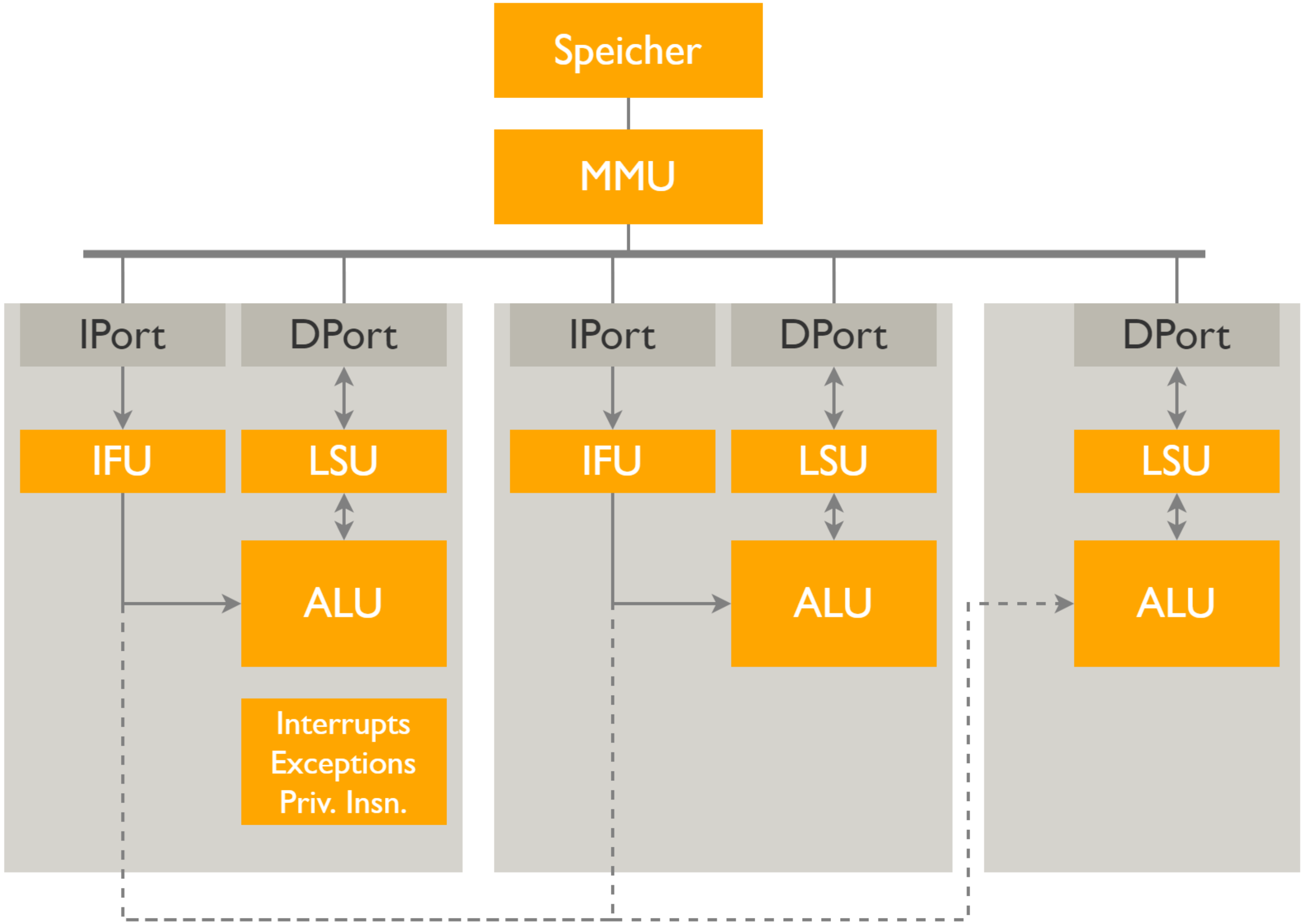
Aha!
Ungültiger Speicherzugriff!

```
user@host:~$ ./selbstzerstörung
Segmentation Fault (core dumped)
```

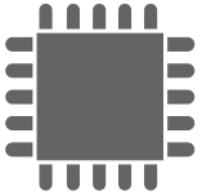
Memory Management Unit (MMU)



Memory Management Unit (MMU)



Privilegien-Modi gemäß RISC-V-Spezifikation

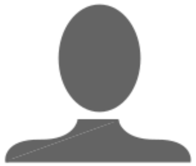


Machine (M)

MMU deaktiviert

Supervisor (S)

steuert (und nutzt MMU optional)



User (U)

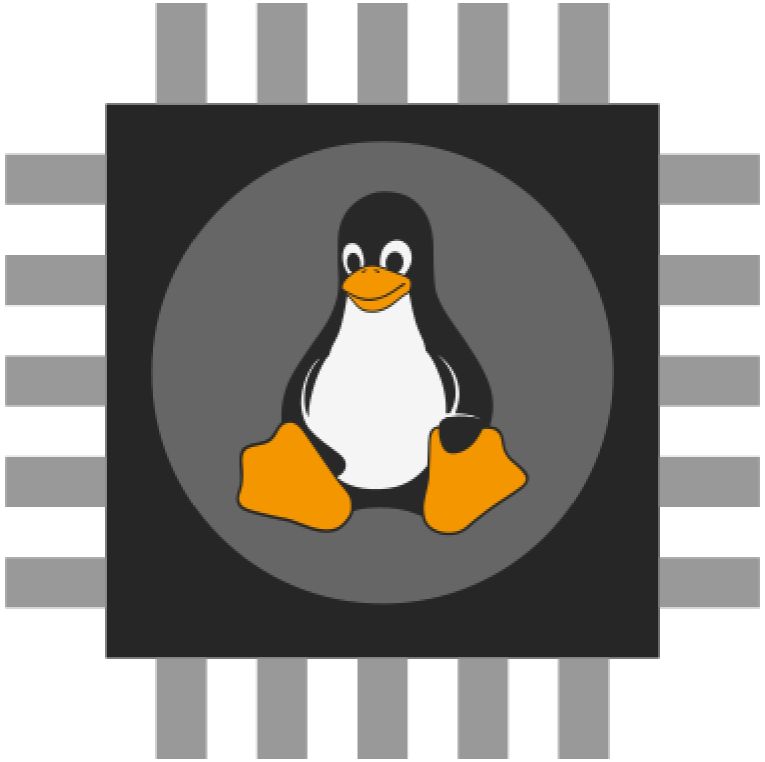
nutzt MMU wie von S konfiguriert

Teil B: Linux

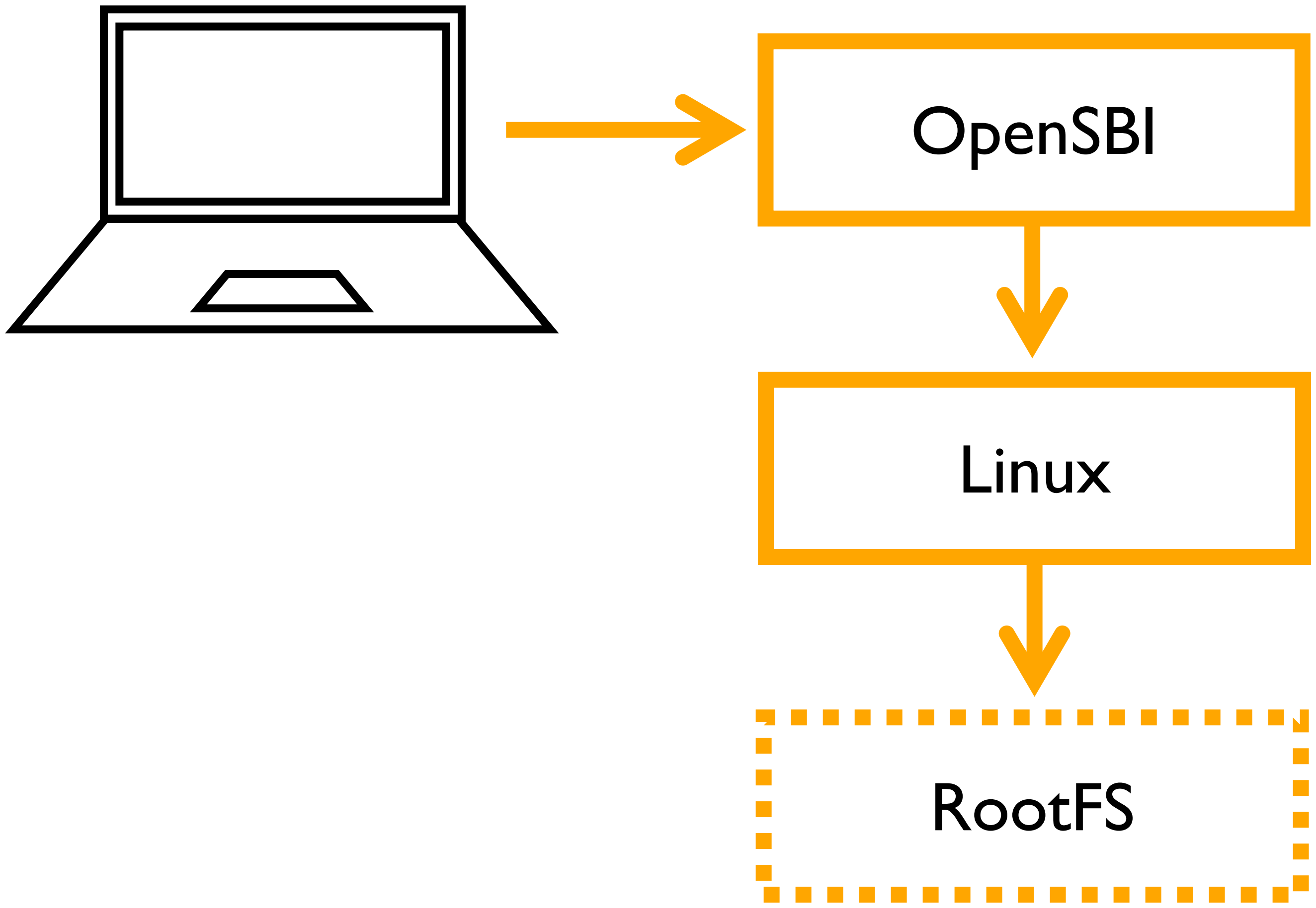
2. Linux auf dem ParaNut

Bisherige Anpassungen

- MMU + TLB
- Privilegienmodi
- Debugging-Anpassungen
- Bootloader (OpenSBI)
- Kernel konfiguriert
- Timer



Boot-Prozess



Bisherige Erfolge

=>

~15 Sekunden Boot

15s • 25MHz = 375 Mio. Takte

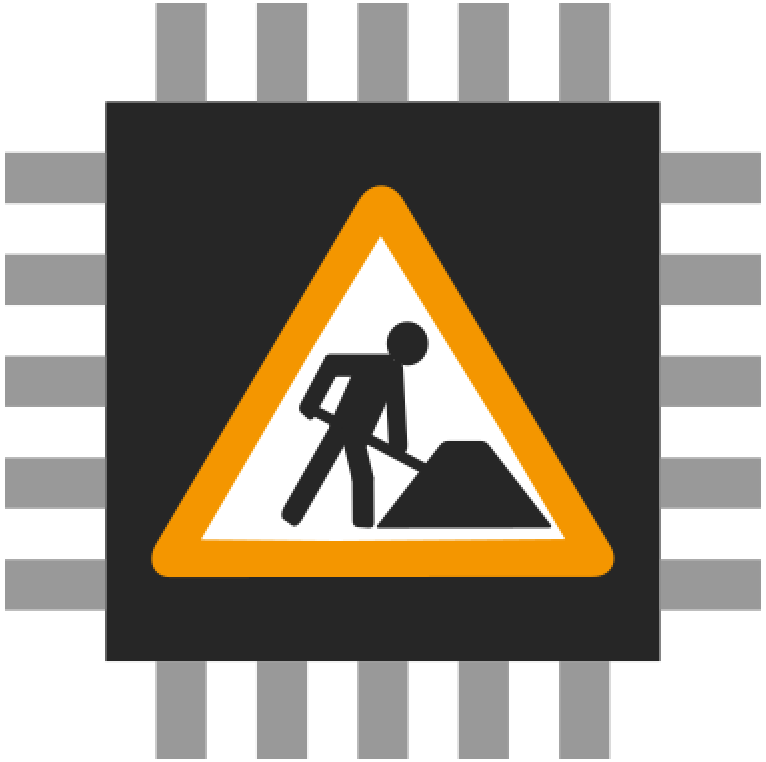
~30 Boot-Zeilen
von Linux

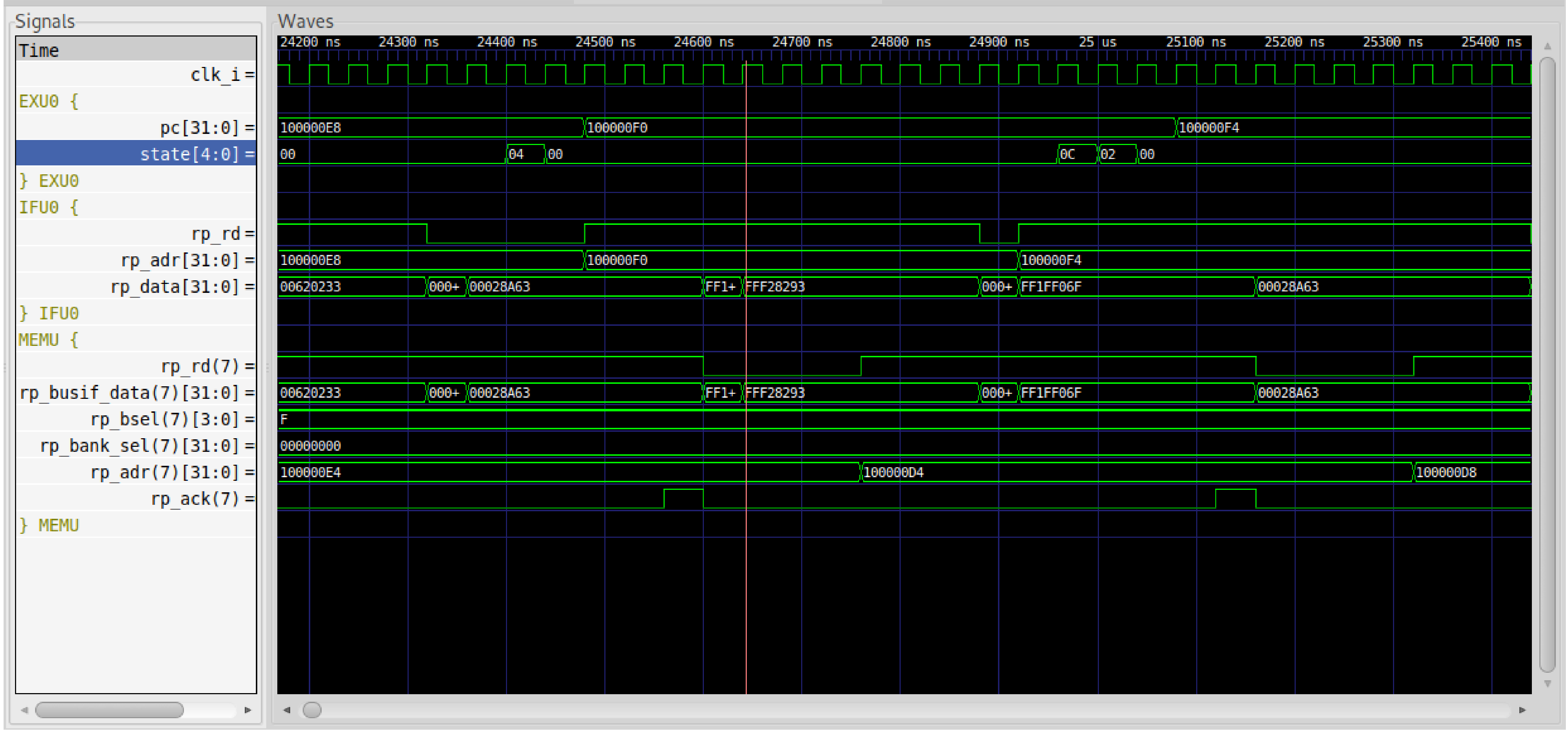
Teil B: Linux

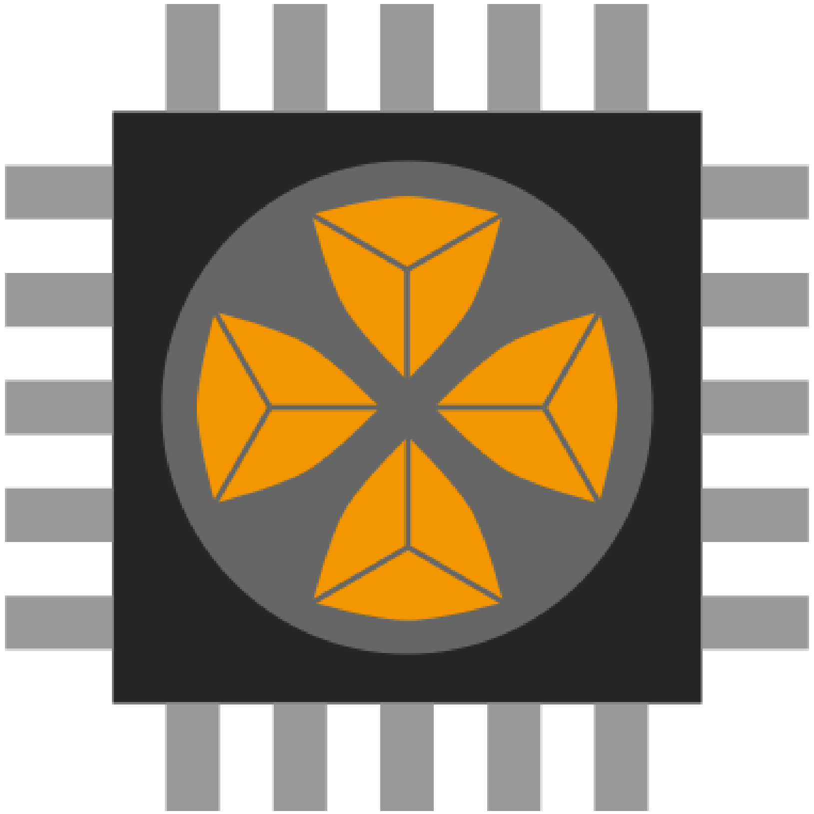
3. Fazit

Weitere Baustellen

- Vollständig booten
- RootFS
- Peripherien
- Weitere Debugging-Anpassungen




Signals
Waves
Time
clk_i =
EXU0 {
pc[31:0] =
state[4:0] =
} EXU0
IFU0 {
rp_rd =
rp_adr[31:0] =
rp_data[31:0] =
} IFU0
MEMU {
rp_rd(7) =
rp_busif_data(7)[31:0] =
rp_bsel(7)[3:0] =
rp_bank_sel(7)[31:0] =
rp_adr(7)[31:0] =
rp_ack(7) =
} MEMU
24200 ns
24300 ns
24400 ns
24500 ns
24600 ns
24700 ns
24800 ns
24900 ns
25 us
25100 ns
25200 ns
25300 ns
25400 ns
100000E8
100000F0
100000F4
00
04
00
0C
02
00
100000E8
100000F0
100000F4
00620233
000+
00028A63
FF1+
FFF28293
000+
FF1FF06F
00028A63
00620233
000+
00028A63
FF1+
FFF28293
000+
FF1FF06F
00028A63
F
00000000
100000E4
100000D4
100000D8



BSD-Lizenz

https://github.com/hsa-ees/paranut

Fragen!