Institute for Complex Systems (ICS)

Univ.-Prof. Dr.-Ing. Daniel Große
Web: jku.at/ics
Email: daniel.grosse@jku.at

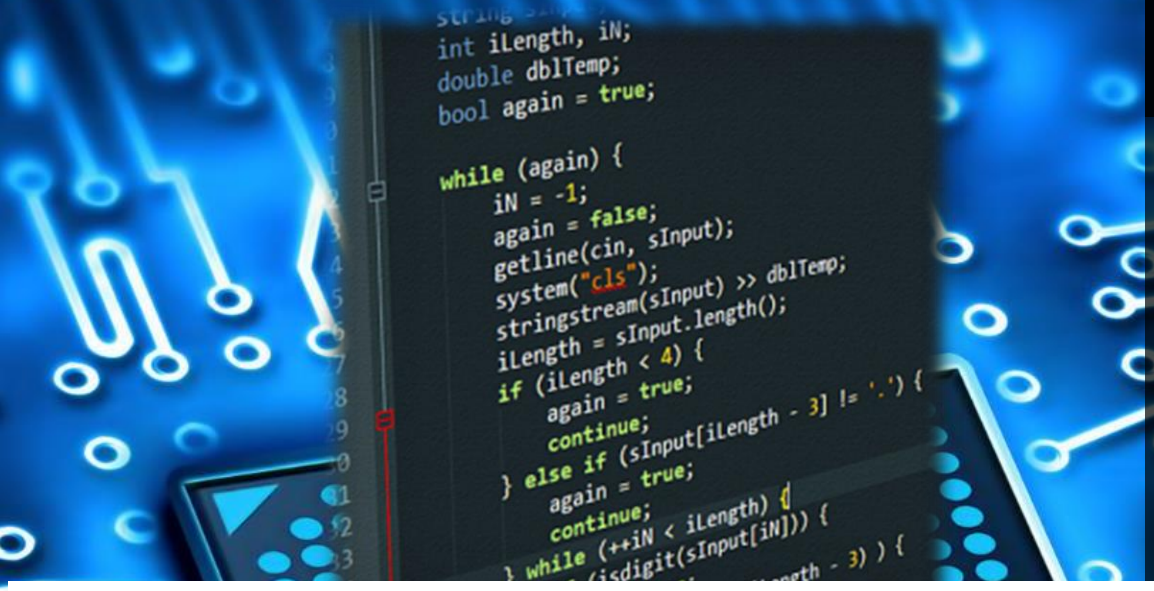
Follow @Daniel_Grosse

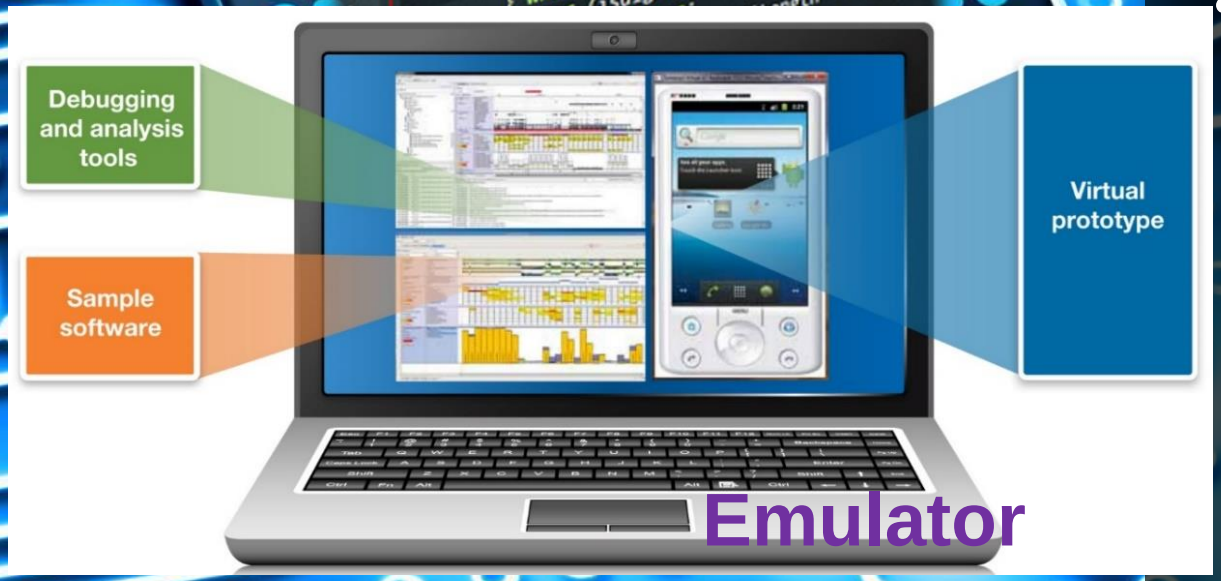
JOHANNES KEPLER UNIVERSITY LINZ

System-on-Chips (SoCs)

- Omnipresent
- Increasingly complex systems
- In particular SW importance/complexity rising rapidly:
  - ASIC era: SW was afterthought
  - SoC era: SW is integral part
    - Bootcode, device driver, firmware, OS, libraries (network stack), application
- **Problem:**
  Traditional design flow (first HW then SW) insufficient

… and Verification?





- Design already hard!
- **New approaches to verify <u>cooperation</u> of SW and HW** needed!
  - Application SW has to exploit <u>cores</u> & <u>IPs</u>
- **@ICS:**
  - Design & verification flow starting from **Virtual Prototype (VP)** down to **HW**
  - Core concept: abstraction
  - HW/SW no longer two worlds
  - Verification happens earlier, with less irrelevant detail

Open source and collaboration
are strategic to software and hardware across industries and geographies.

RISC-V®

Disruptive **Technology**

Barriers	**Legacy ISA**	**RISC-V ISA**
Complexity	1500+ base instructions Incremental ISA	47 base instructions Modular ISA
Design freedom	$$$ – Limited	Free – Unlimited
License and Royalty fees	$$$	Free
Design ecosystem	Moderate	Growing rapidly. Numerous extensions and cores. More design companies on RISC-V than any other architecture
Software ecosystem	Extensive	Growing rapidly. Easy to compile for RISC-V

RISC-V®

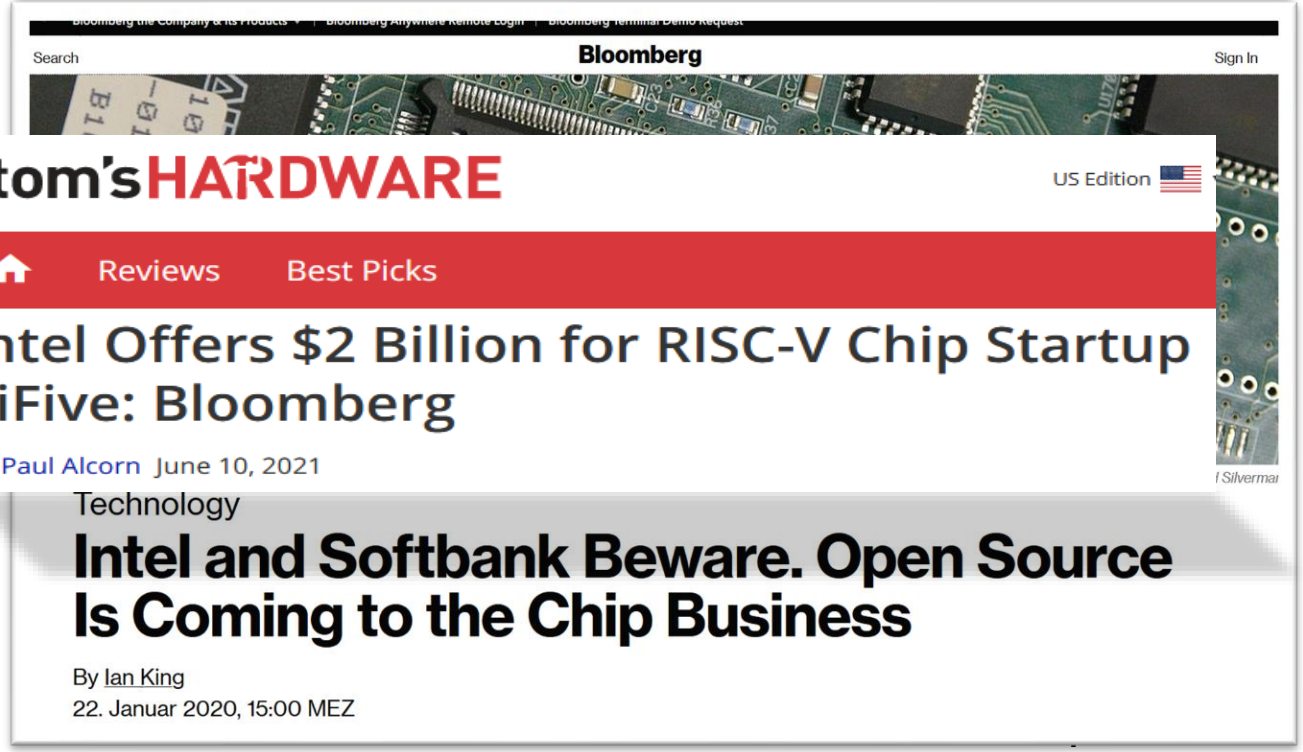
RISC-V Foundation Members (>1000 today)

RISC-V News

- **Prediction:** More than 64 billion RISC-V cores by 2025 (Semico Research Corp.)

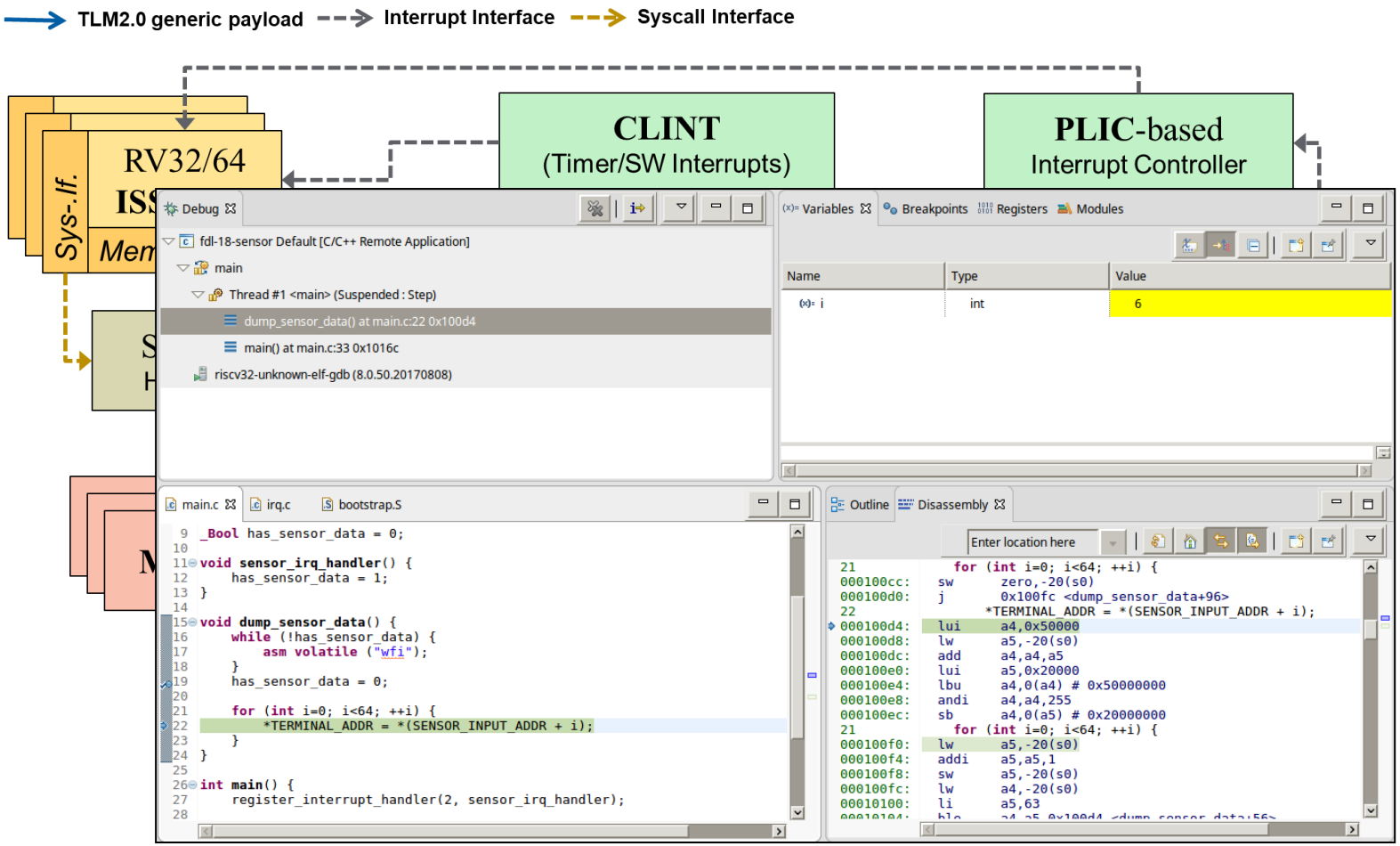
Worst-Case-Szenario als Chance für offene Alternative

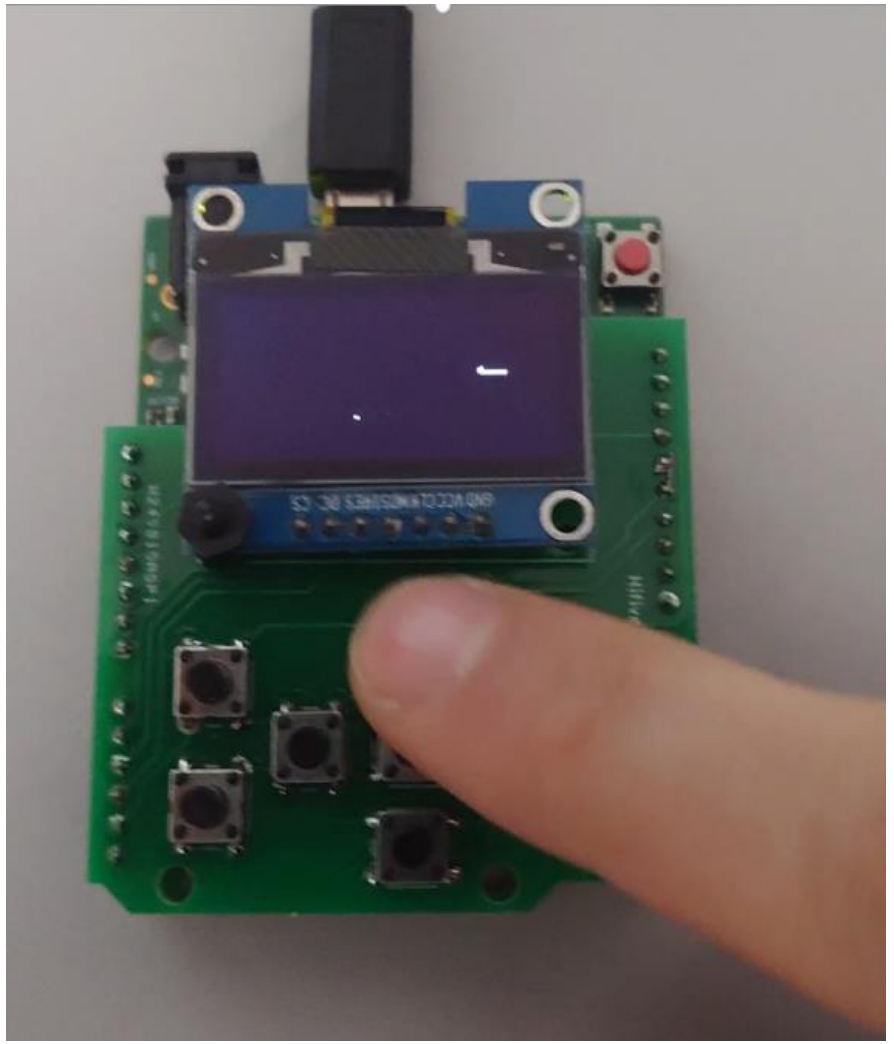
Sollte die Übernahme von Arm durch Nvidia tatsächlich Auswirkungen auf die weltweite Rechnerproduktion haben, könnte das - zumindest längerfristig - dazu führen, dass sich Hersteller nach Alternativen zu den Arm-Designs umsehen. Einen Kandidaten - bisher grober Außenseiter - gibt es dafür schon. Mit RISC-V existiert ein Konkurrenzstandard mit einem enormen Vorteil: Die Rechnerarchitektur ist offen verfügbar - das heißt, für die Verwendung in Chips fällt an sich keine Lizenzgebühr an.



@ICS: Open Source RISC-V VP

VP Architecture





watchdog.cpp - marlin - Visual Studio Code

File Edit Selection View Go Run Terminal Help

uart.h U | uart.cpp U | MarlinSerial.cpp M | launch.json M | watchdog.cpp | PIC

Marlin > src > HAL > AVR > watchdog.cpp > ...

```
30  #include "../../MarlinCore.h"
31
32  // Initialize watchdog with 8s timeout, if possible. Otherwise, make it 4s.
33  void watchdog_init() {
34    #if ENABLED(WATCHDOG_DURATION_8S) && defined(WDTO_8S)
35      #define WDTO_NS WDTO_8S
36    #else
37      #define WDTO_NS WDTO_4S
38    #endif
39    #if ENABLED(WATCHDOG_RESET_MANUAL)
40      // Enable the watchdog timer, but only for the interrupt.
41      // Take care, as this requires the correct order of operation, with interrupts disa
42      // See the datasheet of any AVR chip for details.
43      wdt_reset();
44      cli();
45      _WD_CONTROL_REG = _BV(_WD_CHANGE_BIT) | _BV(WDE);
46      _WD_CONTROL_REG = _BV(WDIE) | (WDTO_NS & 0x07) | ((WDTO_NS & 0x08) << 2); // WDTO_N
47                                                                                // So wor
48      sei();
49      wdt_reset();
50    #else
51      wdt_enable(WDTO_NS); // The function handles the upper bit correct.
52    #endif
53    //delay(10000); // test it!
```

EXPLORER

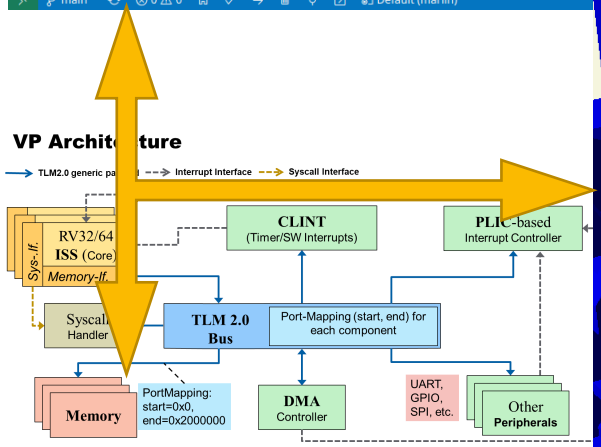
MARLIN

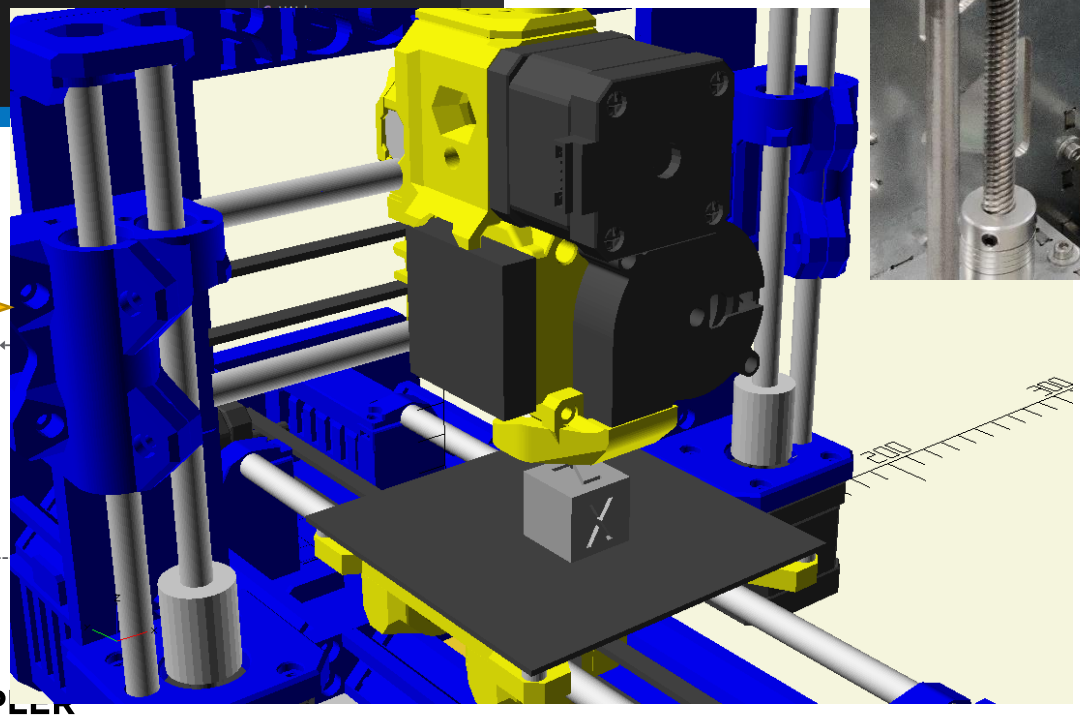
- ESP32
- LINUX
- LPC1768
- NATIVE_SIM
- RISCV
  - inc
  - lib
  - HAL.cpp
  - HAL.h M
  - main.cpp M
  - MarlinSerial.cpp M
  - MarlinSerial.h M
  - spi_pins.h
  - timers.cpp
  - timers.h
  - watchdog.cpp
  - watchdog.h
- SAMD51
- shared
- STM32
- STM32F1
- TEENSY31_32
- TEENSY35_36
- TEENSY40_41

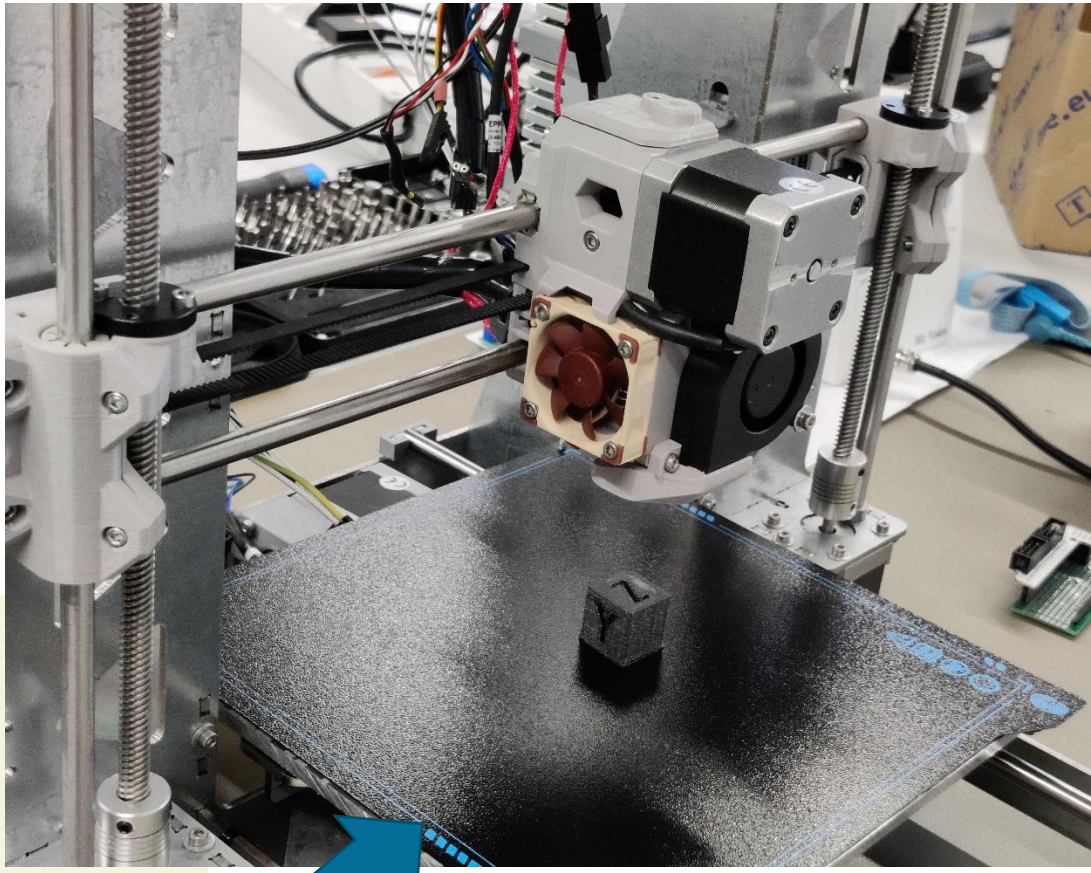
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL bash

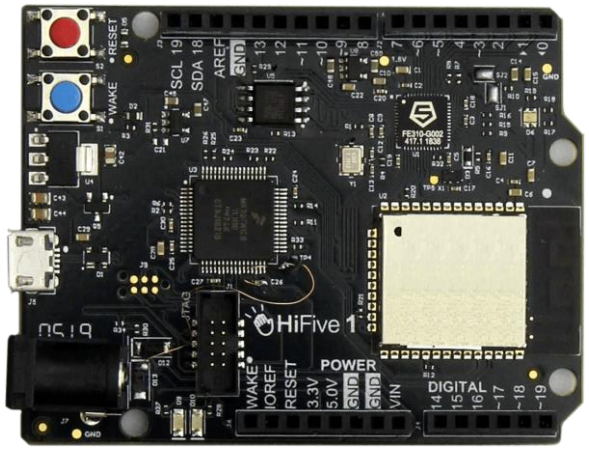
```
hazott@hazott-ThinkPad-T14-Gen-1:~/Projects/riscv3dp/software/sw/marlin$
```

main* 0 0 Default (marlin)





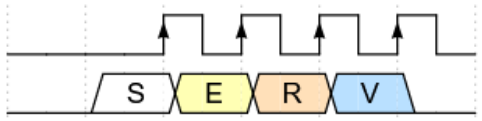




JOHANNES KEPLER UNIVERSITY LINZ

WAL: Waveform Analysis Language

- Waveform debugging
  - Tedious and slow
  - Manual
  - Can not be automated
    - Or can it?
- WAL is *Domain Specific Language* (DSL) to express waveform analysis problems

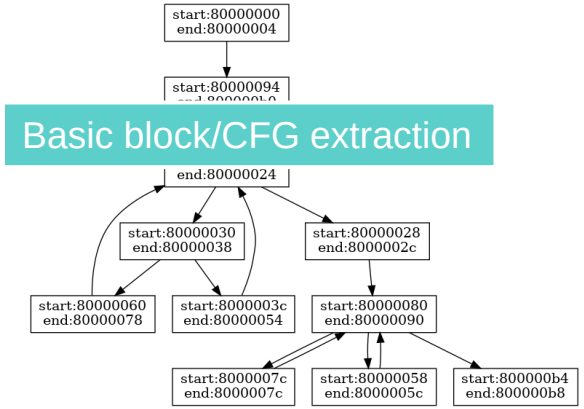


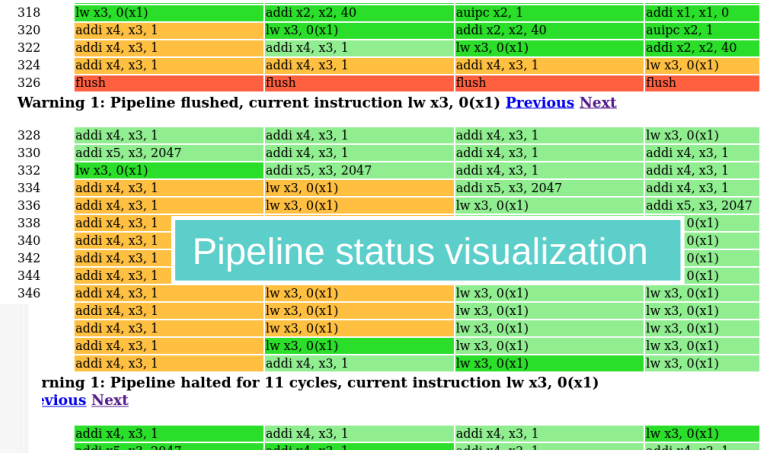
Du hast retweetet

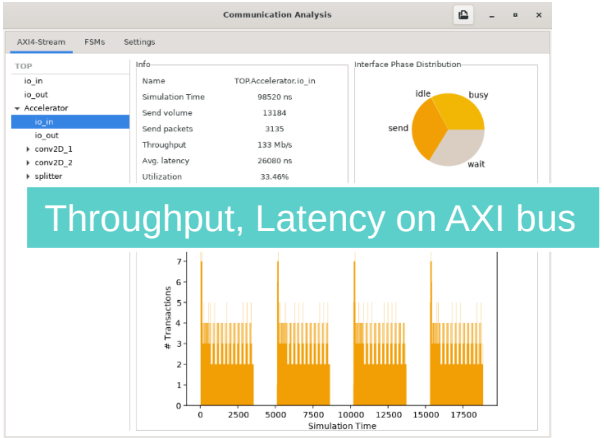
Olof Kindgren @OlofKindgren · 21. Jan.

So, @Icsklmmr and @Daniel_Grosse made a DSL called WAL for analysing waveforms. Now what can you do with such a thing? Well, for one you can calculate how many cycles it takes for SERV to execute each instruction. Check it out here

LucasKI/**serv-cpi**







JOHANN UNIVER

Areas

- Modelling & Implementation of FPGAs
  - RISC-V cores
  - Accelerators (NNs, Approximate Computing)
  - …

- Verification
  - Fuzzing
  - Metamorphic Testing
  - Symbolic execution of SW interacting w HW
  - Symbolic Computer Algebra (SCA) for hard arithmetic (e.g. multipliers) at gate level
  - …

Institute for Complex Systems (ICS)

Univ.-Prof. Dr.-Ing. Daniel Große
Web: jku.at/ics
Email: daniel.grosse@jku.at

Follow @Daniel_Grosse

JOHANNES KEPLER UNIVERSITY LINZ