RISC-V VP++: Unlocking the vast Linux ecosystem for Open Source RISC-V Virtual Prototypes - From Fast Bootup, VNC, Vector Extension to 3D-Games

Daniel Große
Institute for Complex Systems (ICS)
Web: jku.at/ics
Email: daniel.grosse@jku.at

SYSTEMC™ EVOLUTION DAY
NOV 16, 2023 | MUNICH | GERMANY

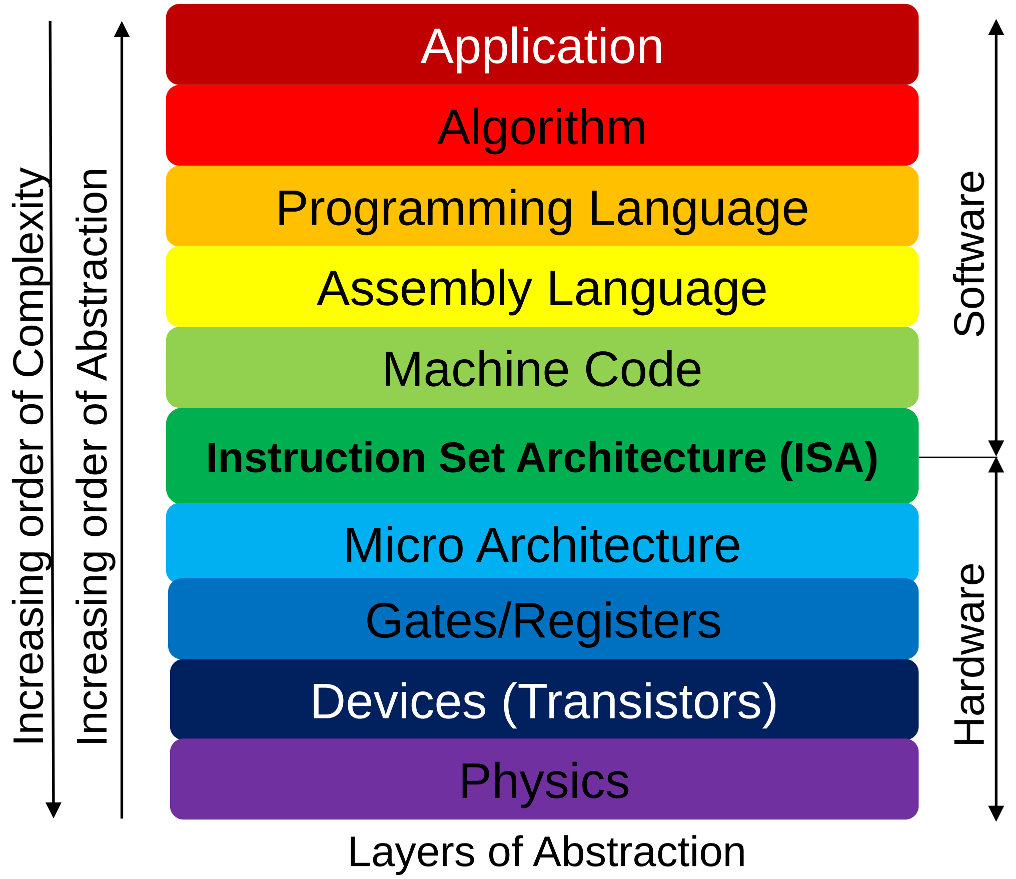
JOHANNES KEPLER UNIVERSITY LINZ

Outline

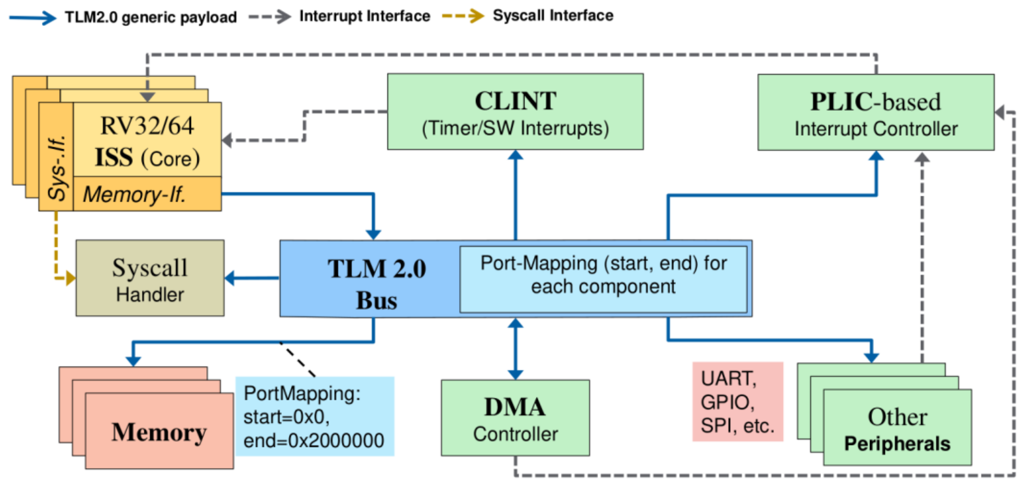
- RISC-V
- RISC-V VP++ Overview
  - Bare Matel Example
  - Linux: GUI-VP Kit / 3D-Games
  - Vector Extension
- Conclusions

RISC-V

- Open and royalty-free ISA
- Focus on simplicity and modularity
- Base Integer Instruction Set
  - Mandatory
  - 32, 64 and 128 bit configurations
  - ~40 Instructions
- Extensions:
  - M .. Multiply/Divide
  - A .. Atomic
  - F, D, Q .. Floating Point (Single, Double, Quad)
  - C .. Compressed
  - …



RISC-V VP++



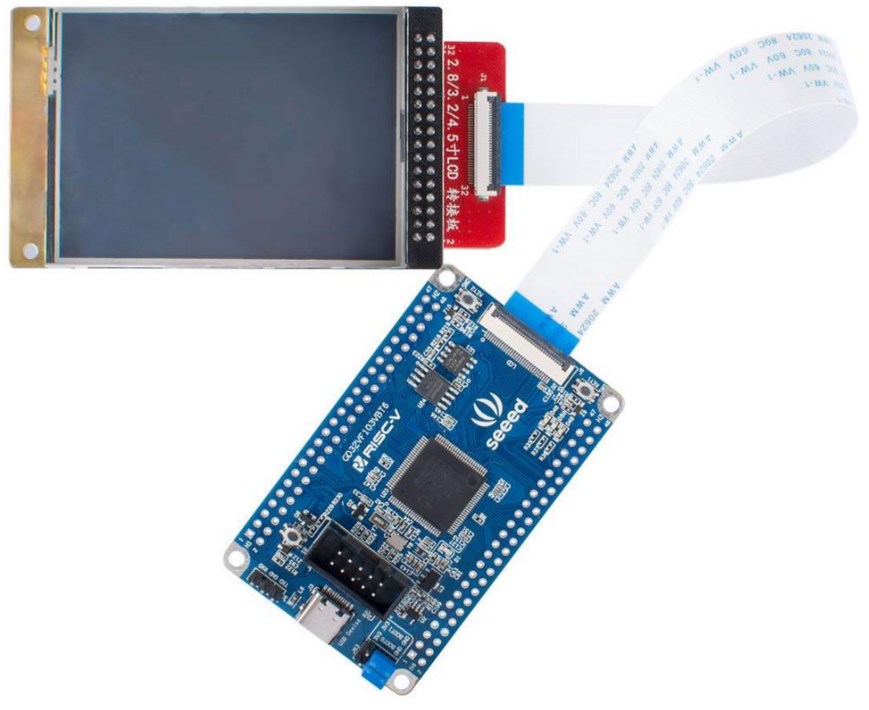
- Open source on GitHub
  - https://github.com/ics-jku/riscv-vp-plusplus
- Key features:
  - SystemC TLM-2.0
  - Bare metal/Small operating systems configurations, including:
    - SiFive HiFive1 - FE310
    - GD32VF103VBT6 microcontroller (Nuclei N205) including UI
  - Linux RV32 and RV64, single and quad-core VPs (SiFive FE540)
  - Full integration of GUI-VP, which enables simulation of interactive graphical Linux applications
  - Support for *RISC-V "V" Vector Extension* (RVV) version 1.0
  - Based on RISC-V VP introduced in 2018*
- More information: http://www.systemc-verification.org

RISC-V VP++

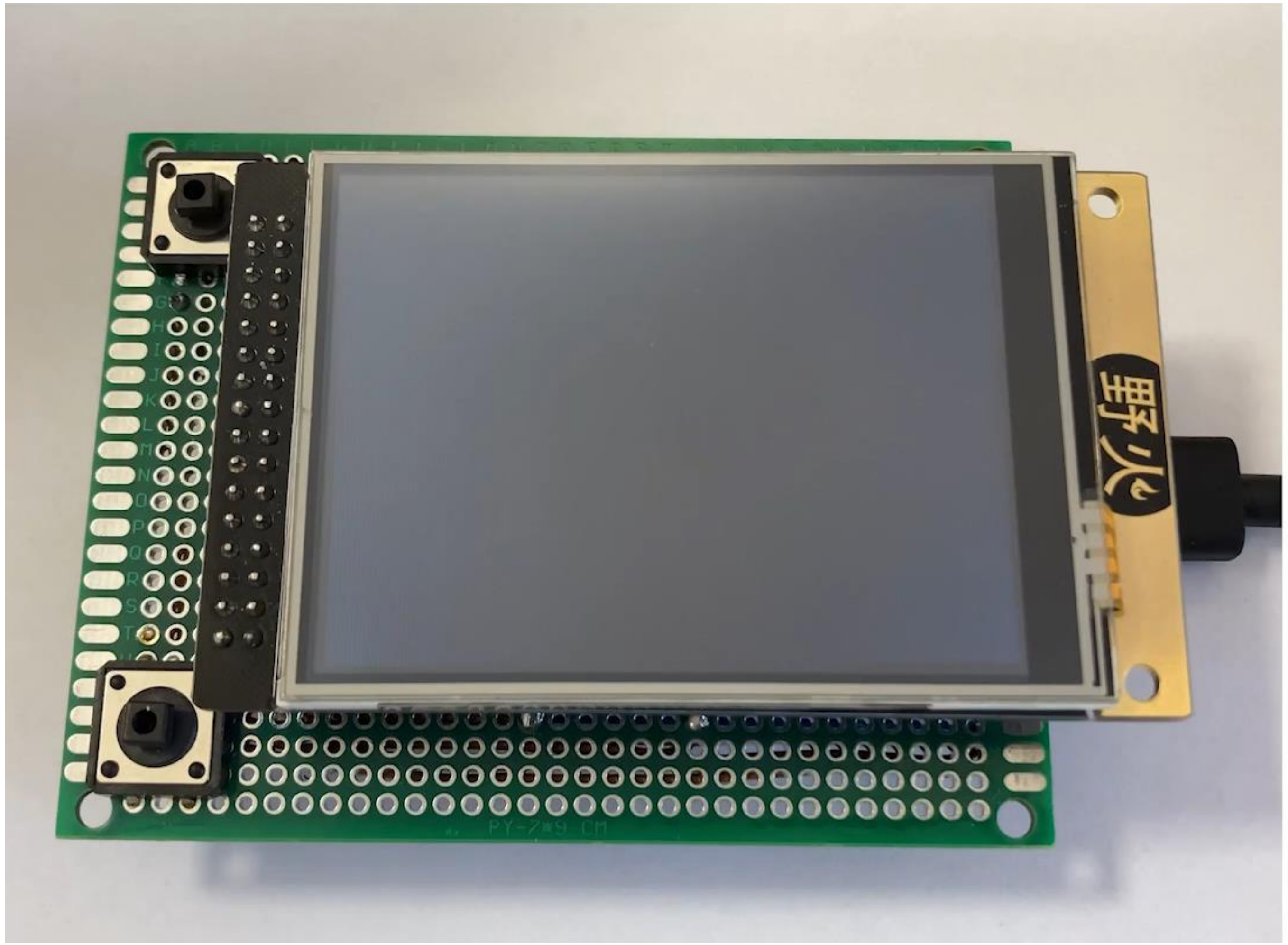
Bare Metal Example

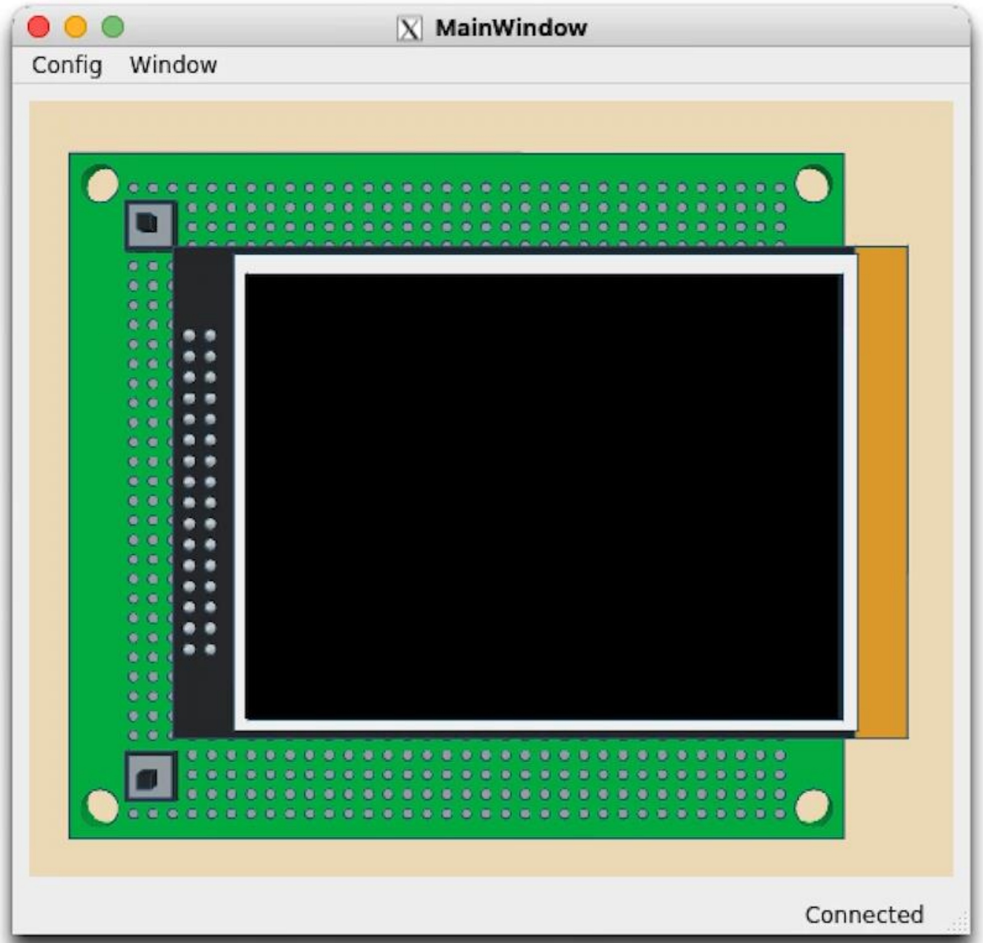
RISC-V VP++: GD32V

- GD32VF103VBT6 microcontroller (Nuclei N205)
- Implemented peripherals in RISC-V VP++
  - GPIO, AFIO, EXTI, SPI, EXMC, RCU
- UI
  - ILI9341 display w XPT2046 touch controller
  - TFT_eSPI
    - Widley used embedded graphics library
    - Adapted for RISC-V



RISC-V VP++: GD32V Demo





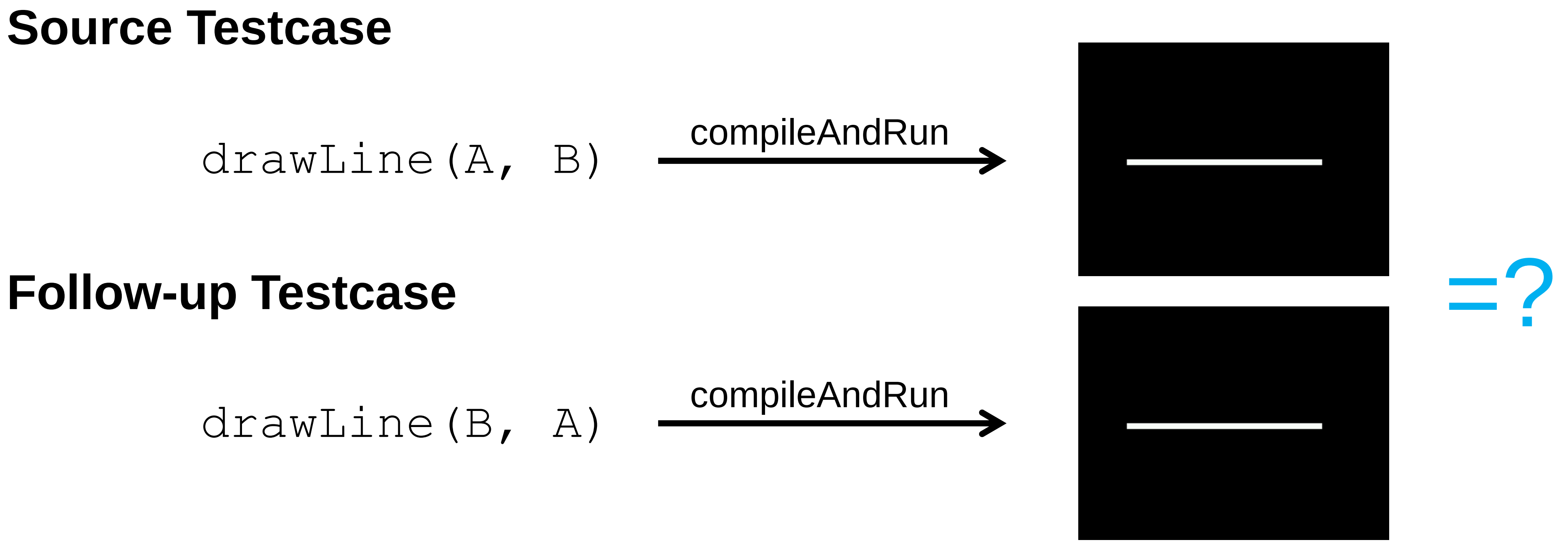
RISC-V VP++: TFT_eSPI Verification Challenge

- **Current verification:**
  Use TFT_eSPI and visually check the result

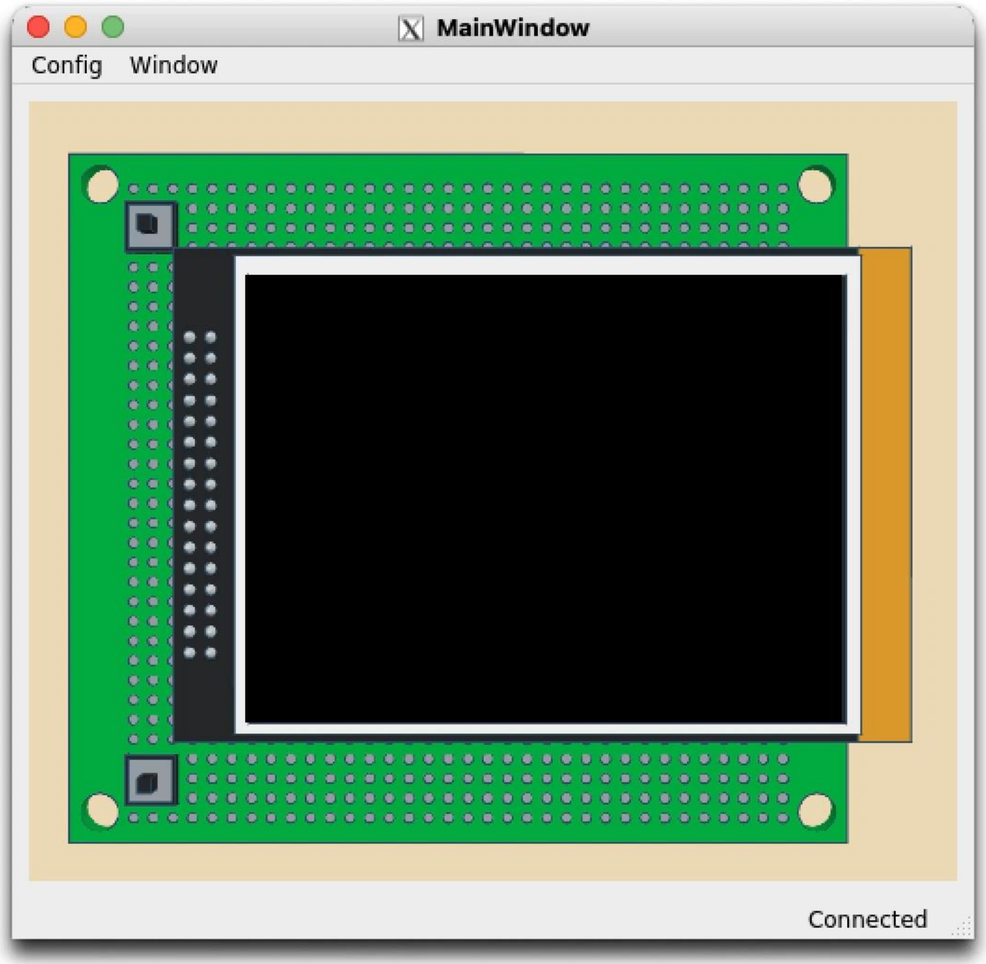
- **Our approach:**
  Overcome need of physical HW via *Metamorphic Testing*
  - Relates multiple program executions via Metamorphic Relations (MRs)
  - If relation is violated, bug is found

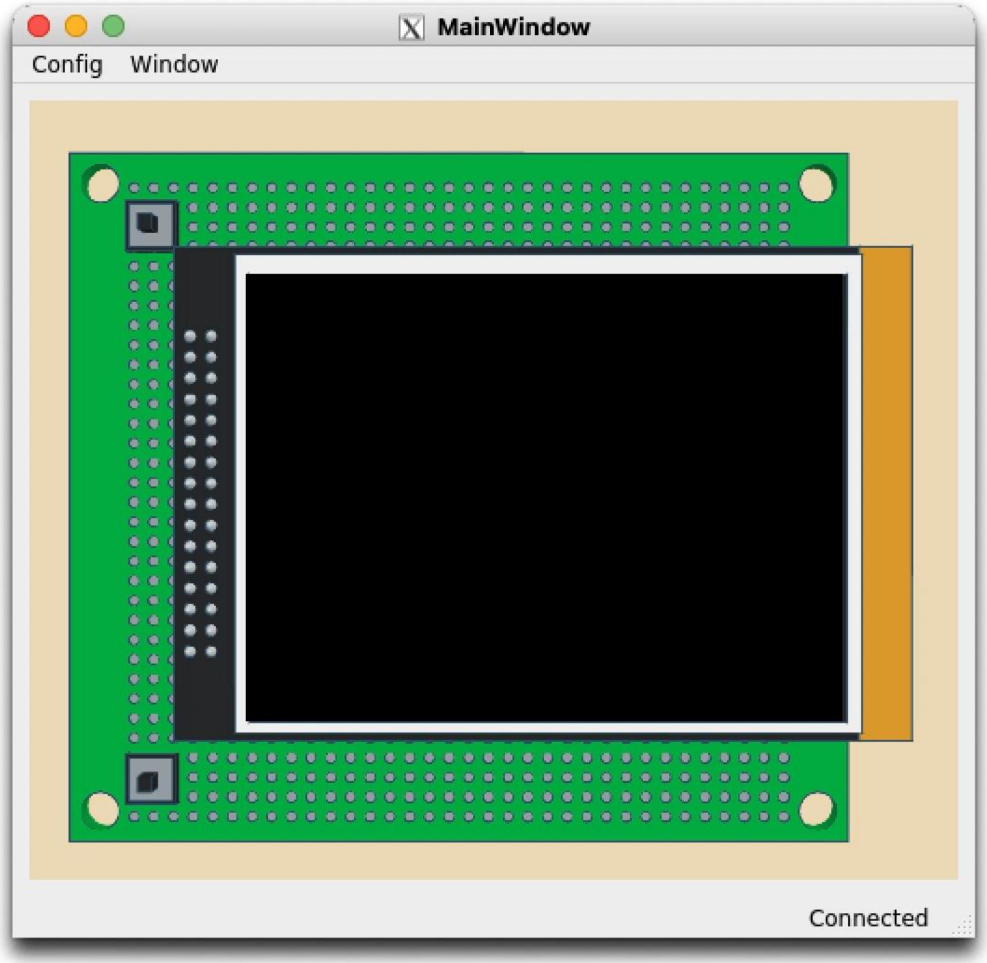
Metamorphic Testing for TFT_eSPI leveraging VPs

Relates multiple program executions



Metamorphic Testing Demo





Approach & Results

- Effective Firmware Testing Approach
  - No need for physical Hardware
  - No need for an Oracle
  - Highly automated

- Exposed 15 unknown bugs in TFT_eSPI library

Number of MRs	21
Ø Testcases per MR	4300
Failed MRs	11
Total Bugs	15

Christoph Hazott, Florian Stögmüller, and Daniel Große. Verifying embedded graphics libraries leveraging virtual prototypes and metamorphic testing. In *ASP-DAC*, 2024.
https://ics.jku.at/files/2024ASPDAC_Verifying_Embedded_Graphics_Libraries_leveraging_VPs_and_MT.pdf

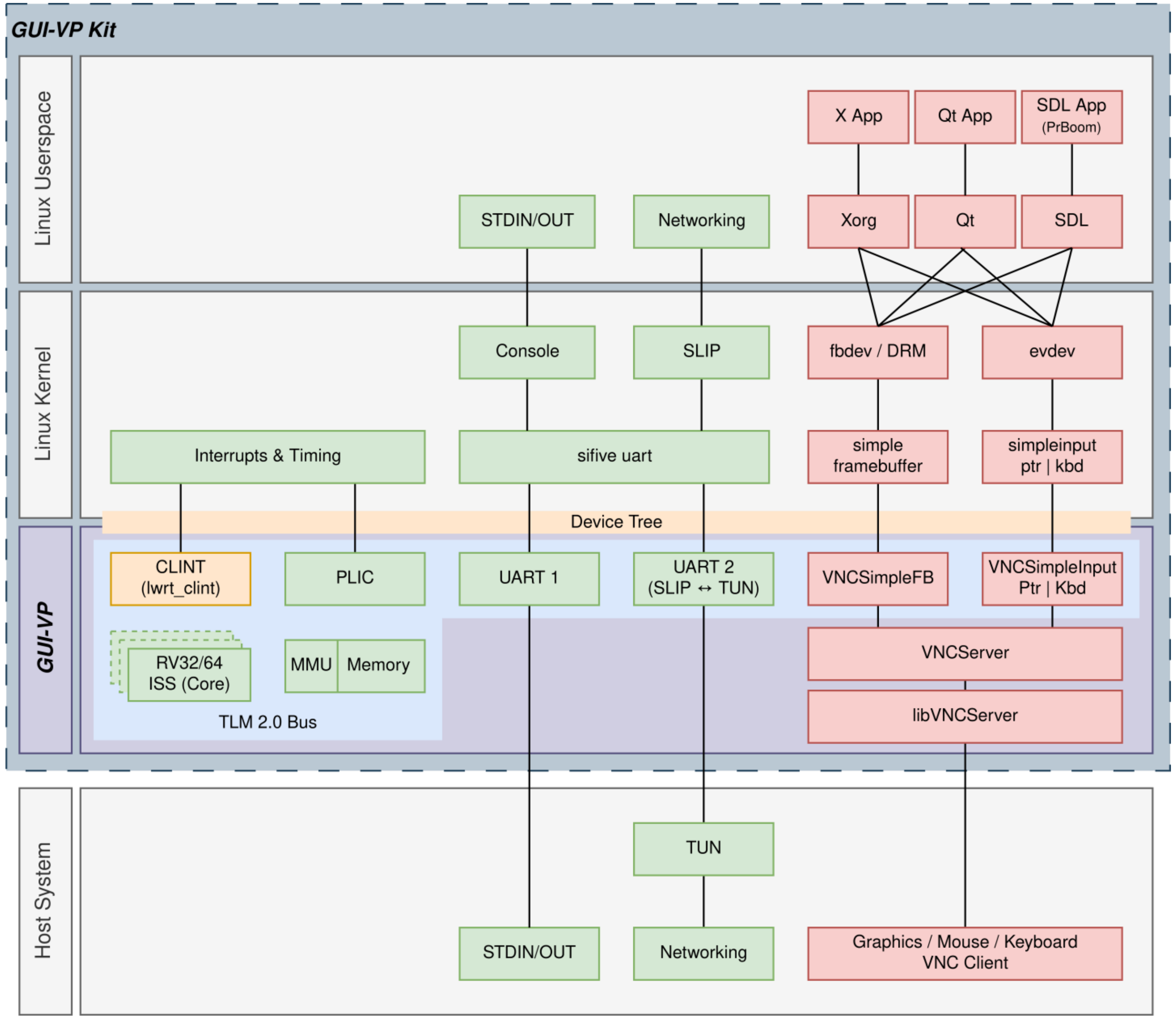
RISC-V VP++

Linux & GUI-VP Kit

GUI-VP Kit

Main Parts:

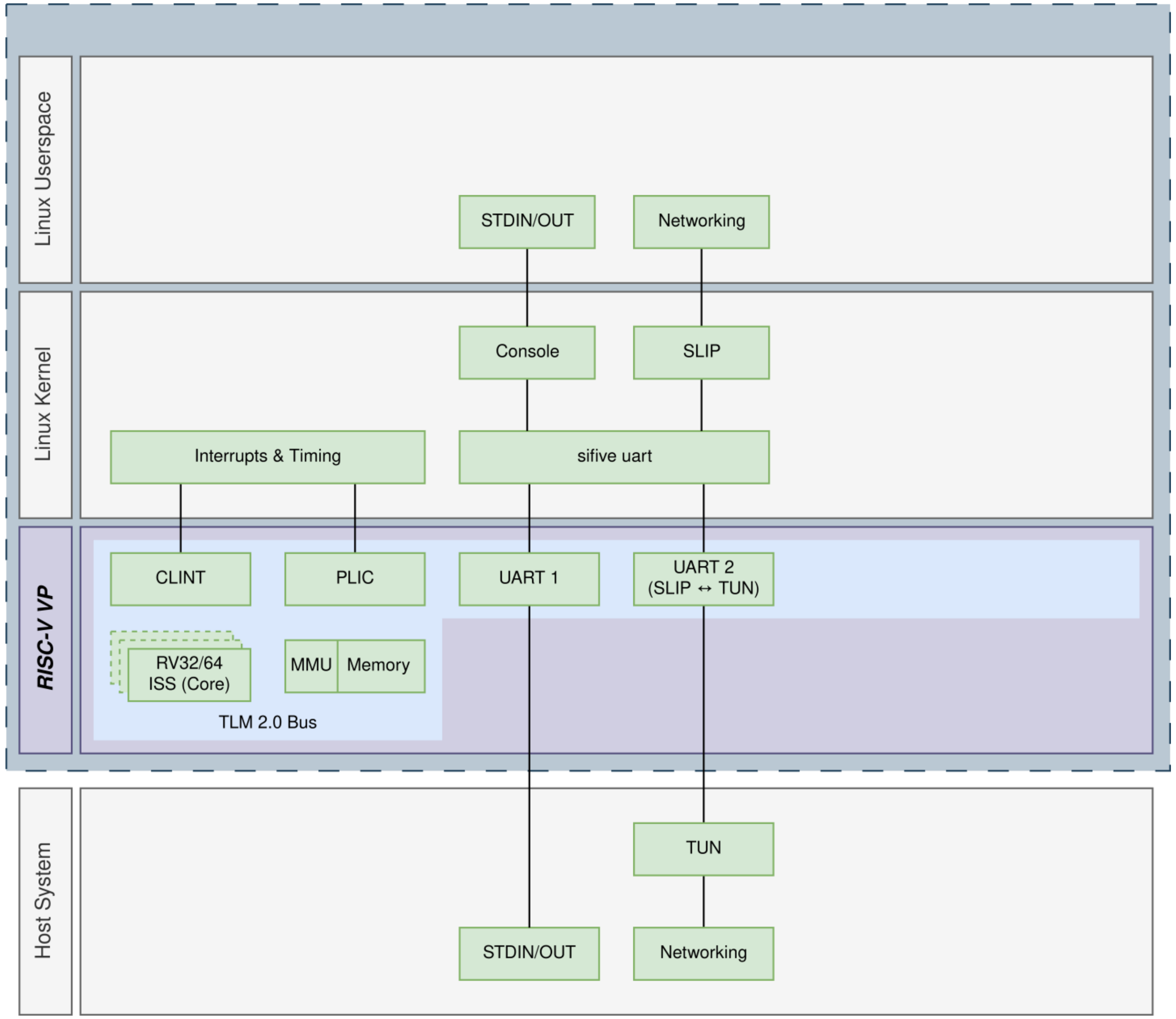
- **GUI-VP**
  - RISC-V VP
  - Real-Time behavior
  - Graphics output
  - Mouse/Keyboard input
  - VNC server
- **Linux Buildsystem**
  - Configurations
  - Linux drivers
  - Devicetree



Manfred Schlägl and Daniel Große. GUI-VP Kit: A RISC-V VP meets Linux graphics - enabling interactive graphical application development. In *GLSVLSI*, 2023. https://ics.jku.at/files/2023GLSVLSI_GUI-VP_Kit.pdf

GUI-VP Kit Simulation Stack

- Based on RISC-V VP

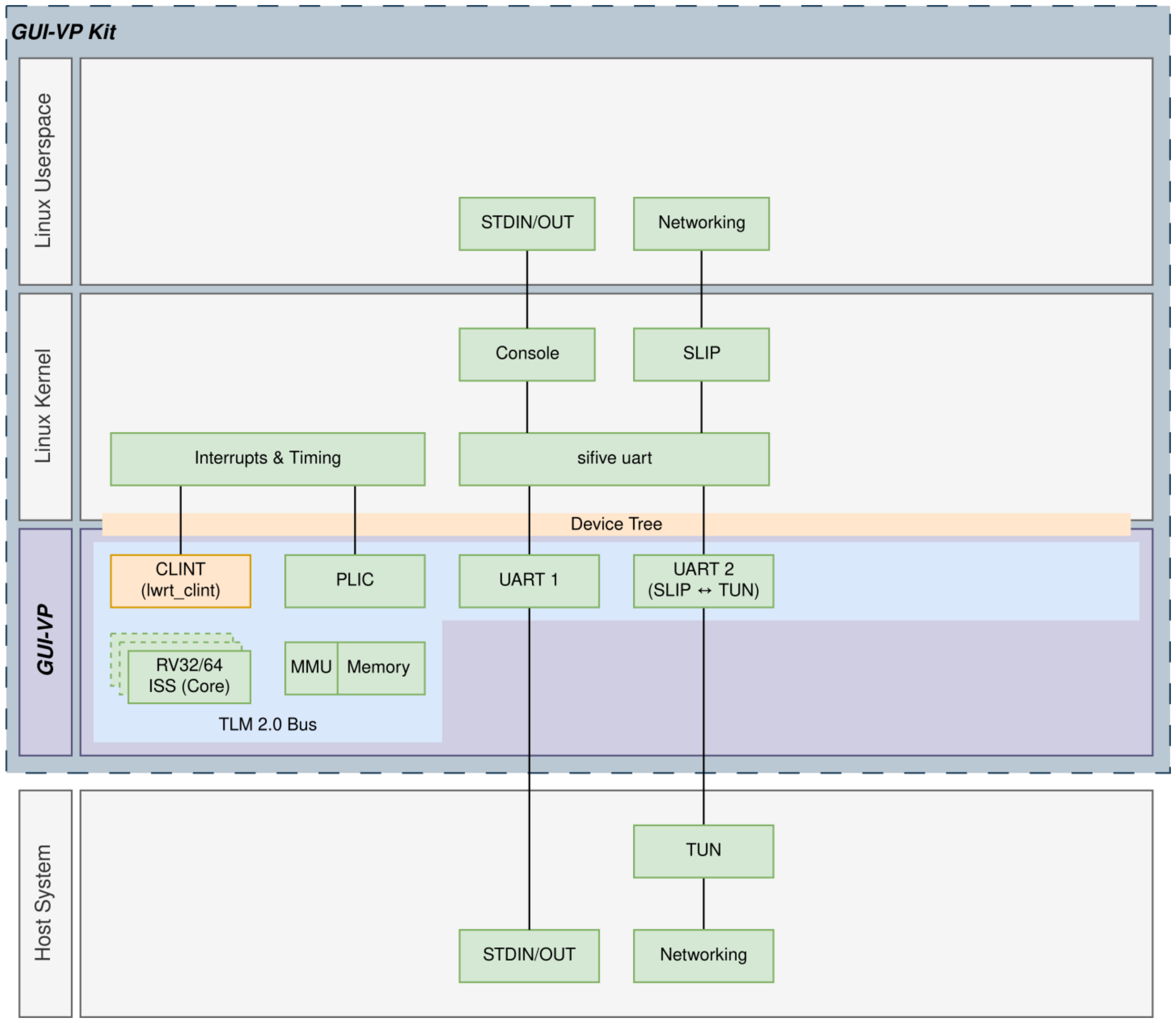


JOHANNES KEPLER UNIVERSITY LINZ

GUI-VP Kit Simulation Stack

- Based on RISC-V VP

- Modifications / Replacements
  - Linux Bring-Up ➞ Device Tree
  - Real-Time Behavior ➞ *lwrt_clint*



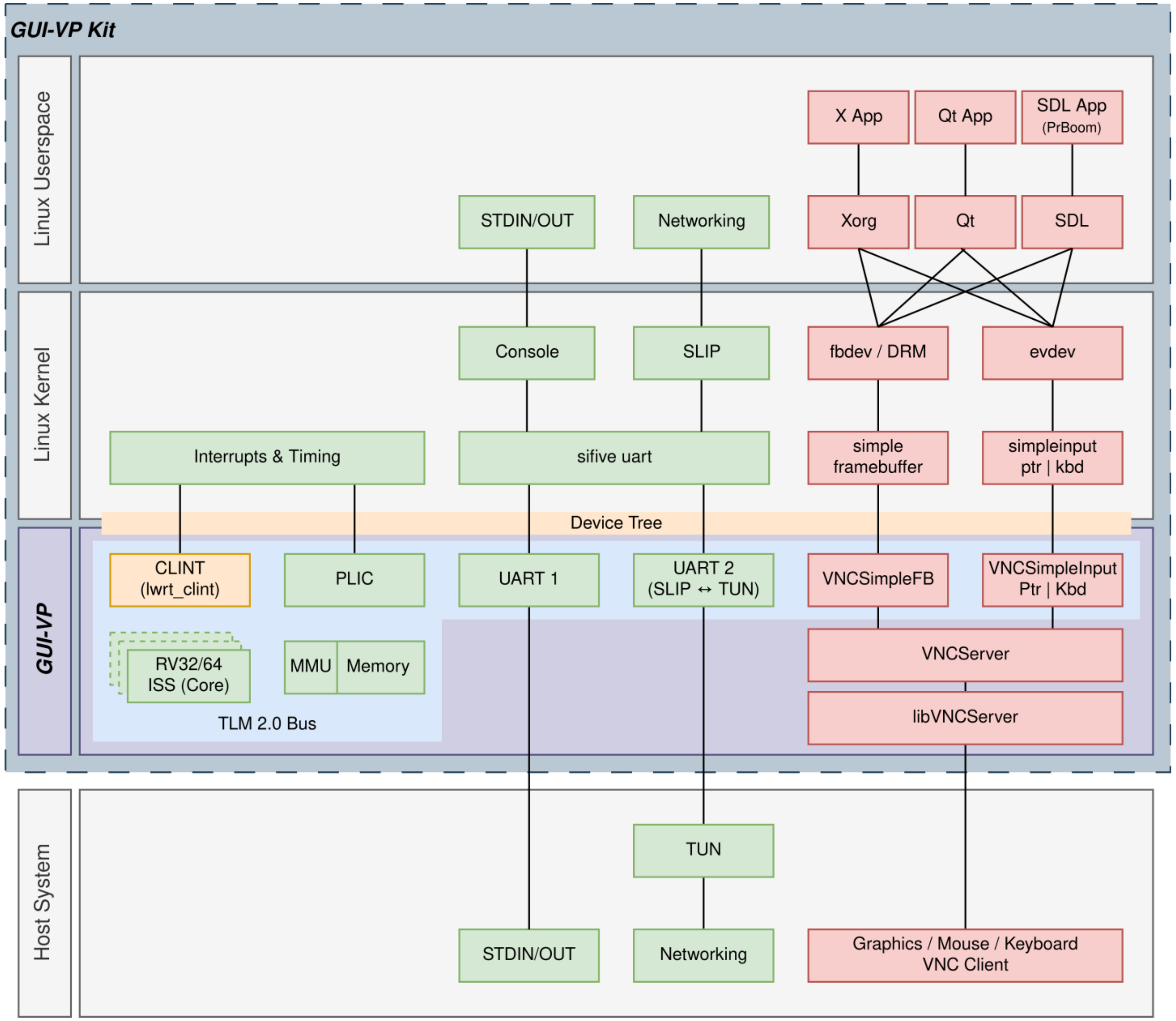
JOHANNES KEPLER UNIVERSITY LINZ

GUI-VP Kit Simulation Stack

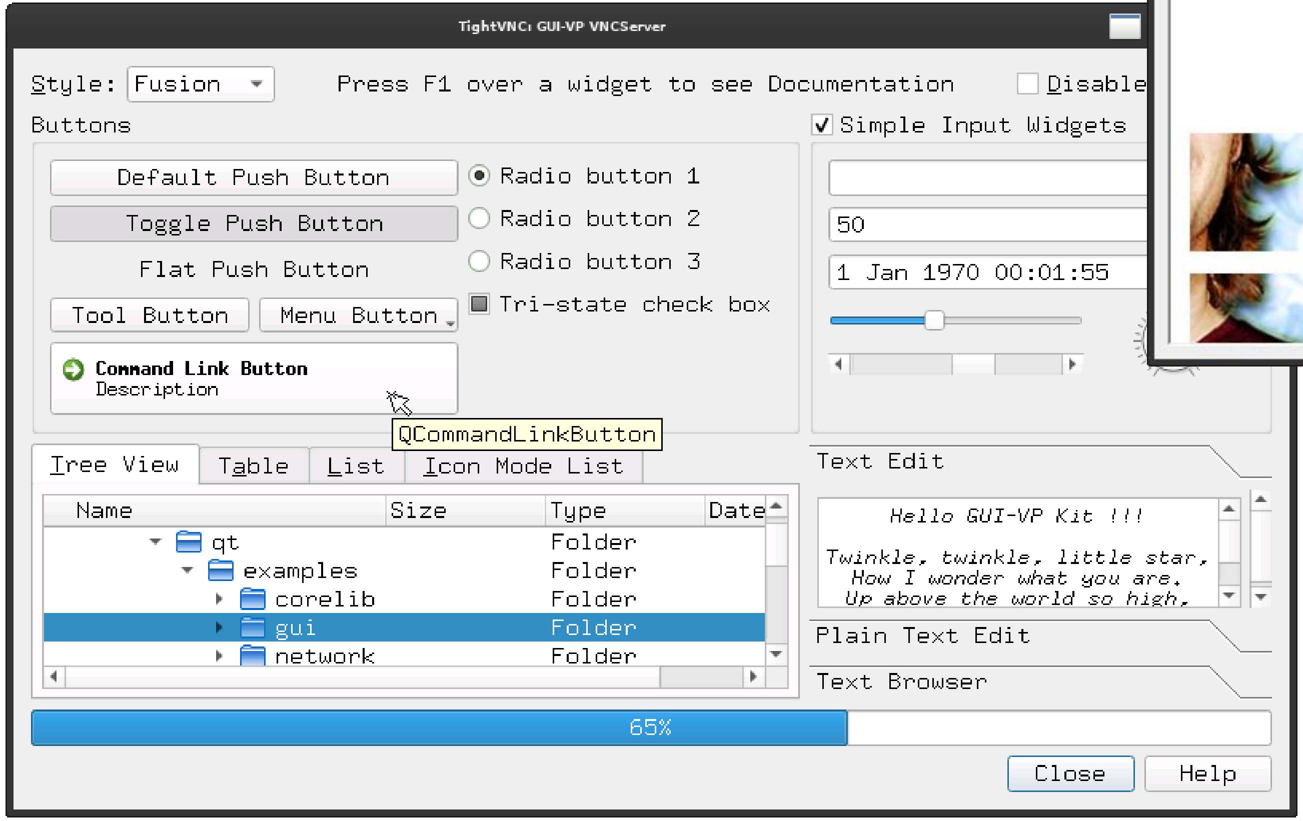
- Based on RISC-V VP

- Modifications / Replacements
  - Linux Bring-Up ➞ Device Tree
  - Real-Time Behavior ➞ *lwrt_clint*

- Extensions
  - VNC Server
  - Graphics Output
  - Mouse Input (Ptr)
  - Keyboard Input (Kbd)



Graphics Development: Qt5



Style: Fusion

Press F1 over a widget to see Documentation

Disable

50

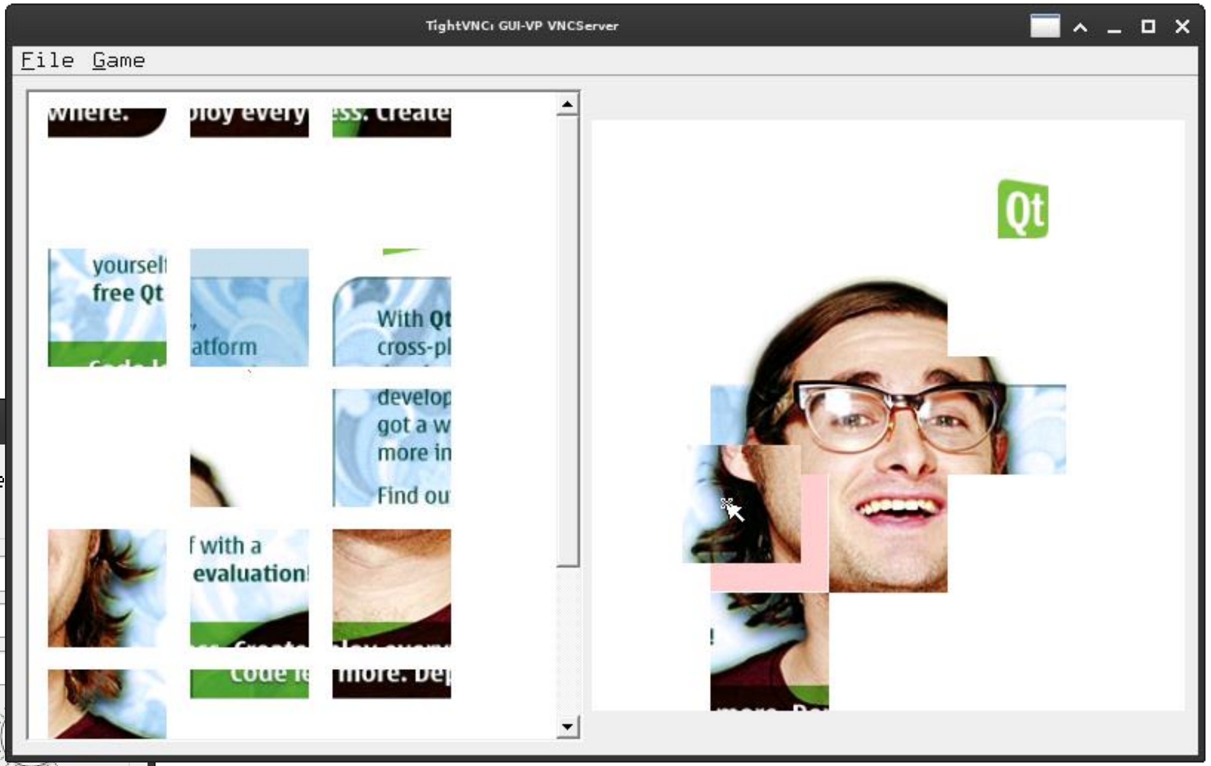
1 Jan 1970 00:01:55

Name	Size	Type
qt		Folder
examples		Folder
corelib		Folder
gui		Folder
network		Folder

Hello GUI-VP Kit !!!

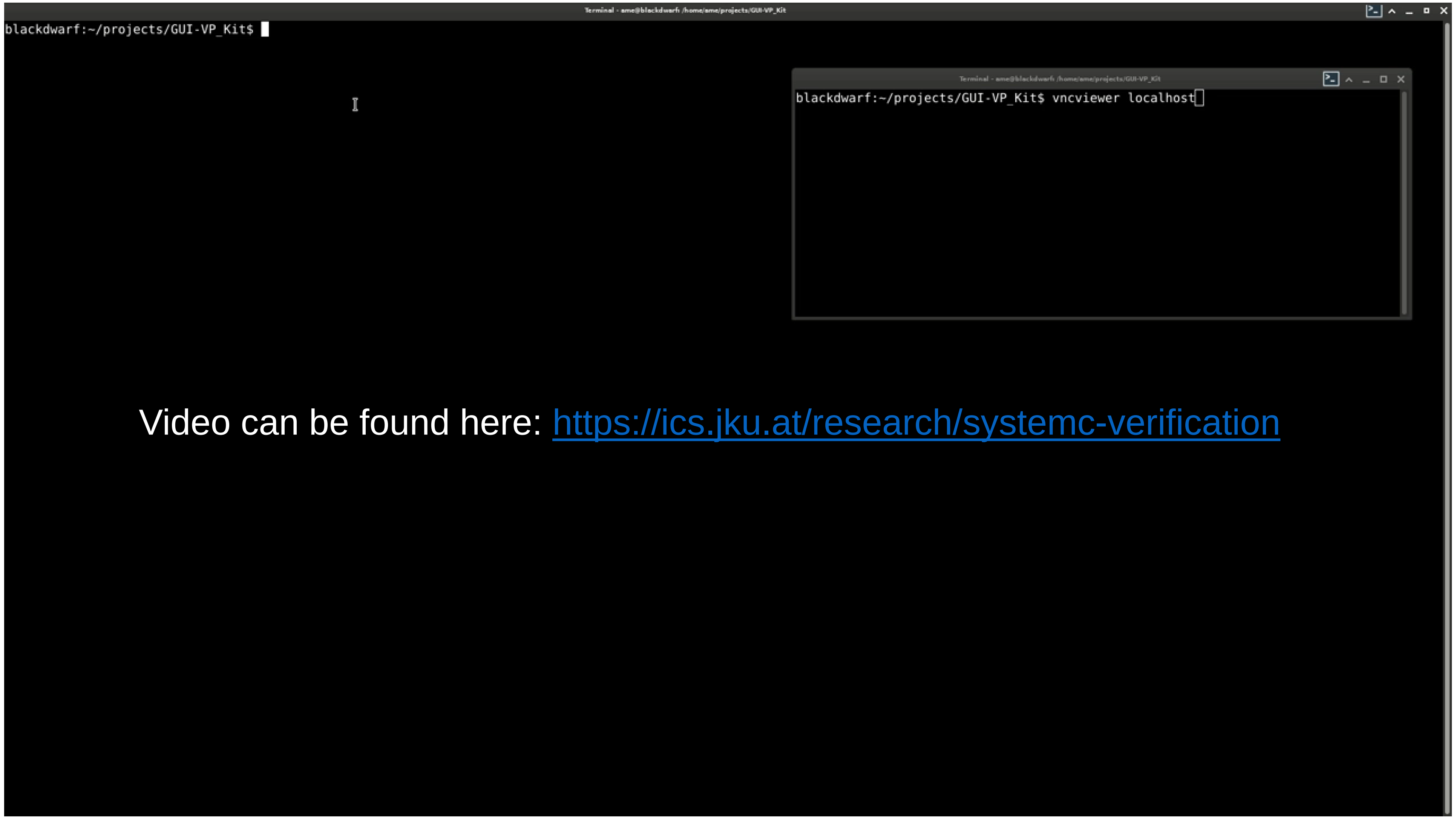
Twinkle, twinkle, little star,
How I wonder what you are,
Up above the world so high,

65%



- < 2 Minutes
- ~ 2.8 billion instructions
- Smooth & Responsive
- Clean Drag & Drop

Demo: X.Org / Networking / Web



Terminal - ame@blackdwarf: /home/ame/projects/GUI-VP_Kit

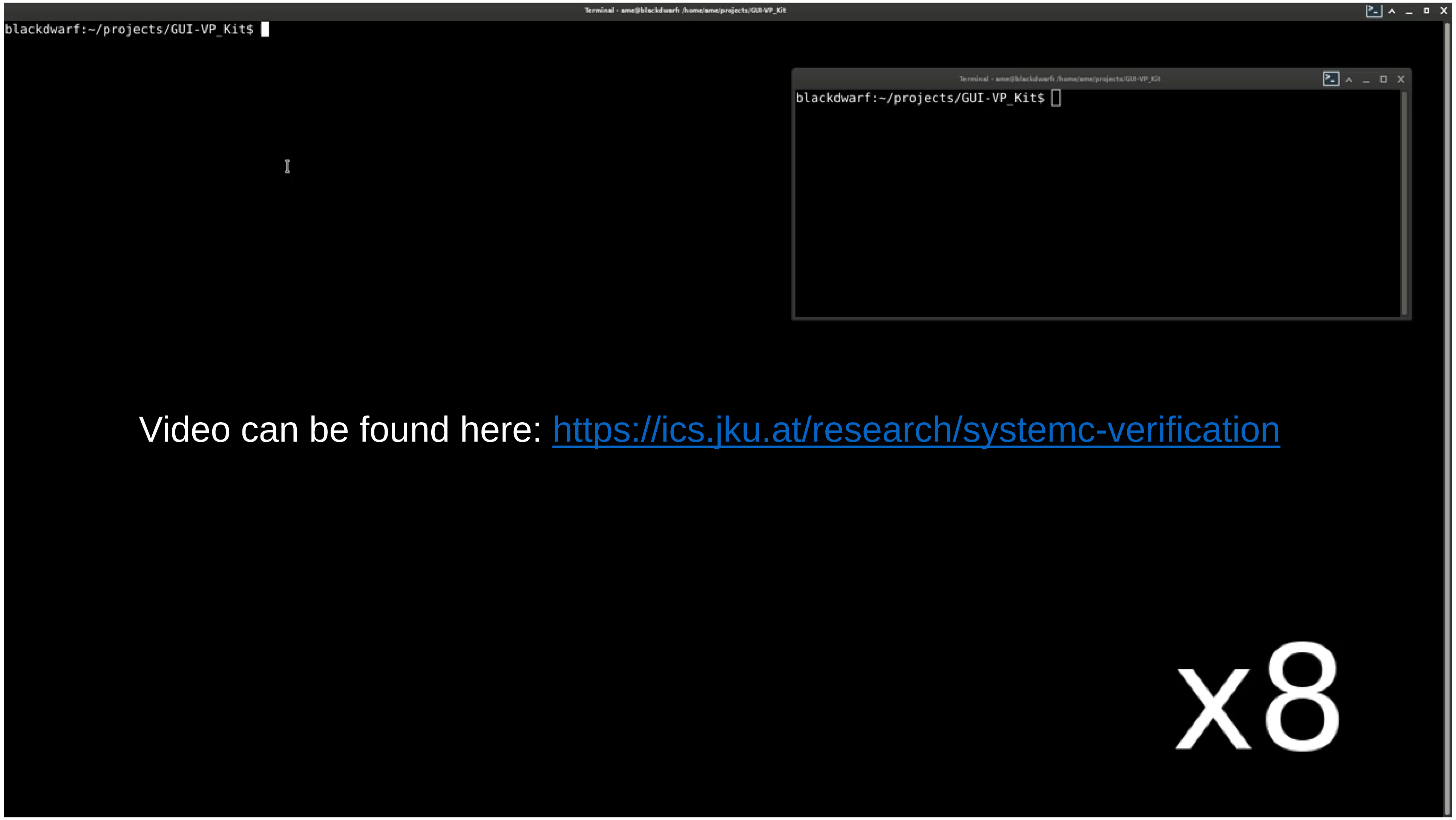
```
blackdwarf:~/projects/GUI-VP_Kit$
```

Terminal - ame@blackdwarf: /home/ame/projects/GUI-VP_Kit

```
blackdwarf:~/projects/GUI-VP_Kit$ vncviewer localhost
```

Video can be found here: https://ics.jku.at/research/systemc-verification

Demo: PrBoom


Terminal - ame@blackdwarf: /home/ame/projects/GUI-VP_Kit
Terminal - ame@blackdwarf: /home/ame/projects/GUI-VP_Kit
x8

```
blackdwarf:~/projects/GUI-VP_Kit$
```

```
blackdwarf:~/projects/GUI-VP_Kit$
```

Video can be found here: https://ics.jku.at/research/systemc-verification

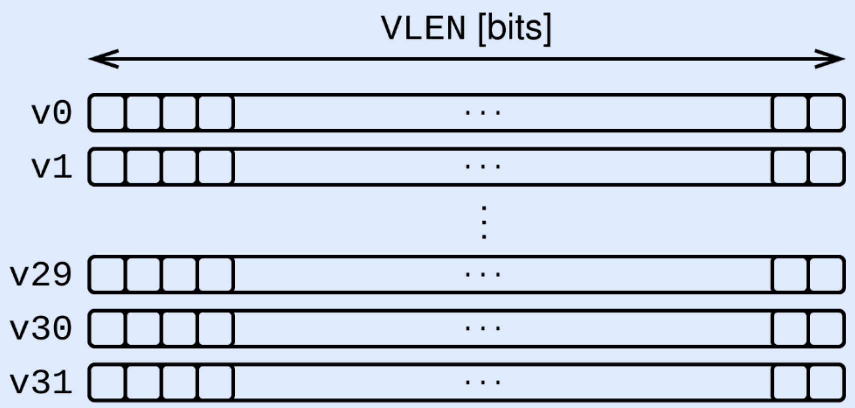
- Up to 8.8 FPS ➞ Reasonable (up to 13.7 million instructions per frame)

RISC-V VP++

Vector Extension (RVV)

RVV

32 Registers



7 CSRs

`vlenb`	VLEN/8 (vector register length in bytes)
`vtype`	Vector data type register
`vl`	Vector length
`vxrm`	Fixed-Point Rounding Mode
`vxsat`	Fixed-Point Saturate Flag
`vcsr`	Vector control and status register
`vstart`	Vector start position

624 Instructions

Configuration

Load/Store
(Strided, Indexed, etc)

Integer

Fixed-Point

Floating-Point

Reductions, Permutations

Widening / Narrowing

Masking

VLEN can be set by the designer (not fixed in ISA)

JOHANNES KEPLER UNIVERSITY LINZ

Classic SIMD vs. Vector

Example: Calculate the sum of two vectors with 80 integers
(assume integer is 32 bit)

On Classic SIMD:

```
SW> I'll take some of your vectors with space for 4 integers
    because ISA says 4 integers fits in 128 bit
HW> OK

SW> Calculate the sum of two such vectors
HW> OK

SW> Repeat this 80 divided by 4 times
HW> OK
```

- Available vector sizes are specified in the ISA
- Software must know vector sizes in advance

→ Software cannot automatically adapt to capacities of hardware

```
void v_add(int8_t sum[], int8_t addend1[],
           int8_t addend2[], int len)
{
  /* ARM Neon Pseudocode
   * (len must be a multiple of 16)
   */

  while (len > 0) {

    /* load addend1 (16) */
    vld1.8    {q0}, [addend1:128]
    /* load addend2 (16) */
    vld1.8    {q1}, [addend2:128]
    /* add (16) */
    vadd.i8   q2, q0, q1
    /* store sum (16) */
    vst1.8    {q2}, [sum:128]

    /* update ptrs and len (16) */
    addend1 += 16; addend2 += 16;
    sum += 16;
    len -= 16;
  }
}
```

JOHANNES KEPLER
UNIVERSITY LINZ

Classic SIMD vs. Vector

Example: Calculate the sum of two vectors with 80 integers
(assume integer is 32 bit)

On RVV (Vector Architecture):

```
SW> Give me some vectors with space for 80 integers
HW> I can give you only vectors with space for X integers
SW> OK

SW> Calculate the sum of two such vectors
HW> OK

SW> Repeat this 80 divided by X times
HW> OK
```

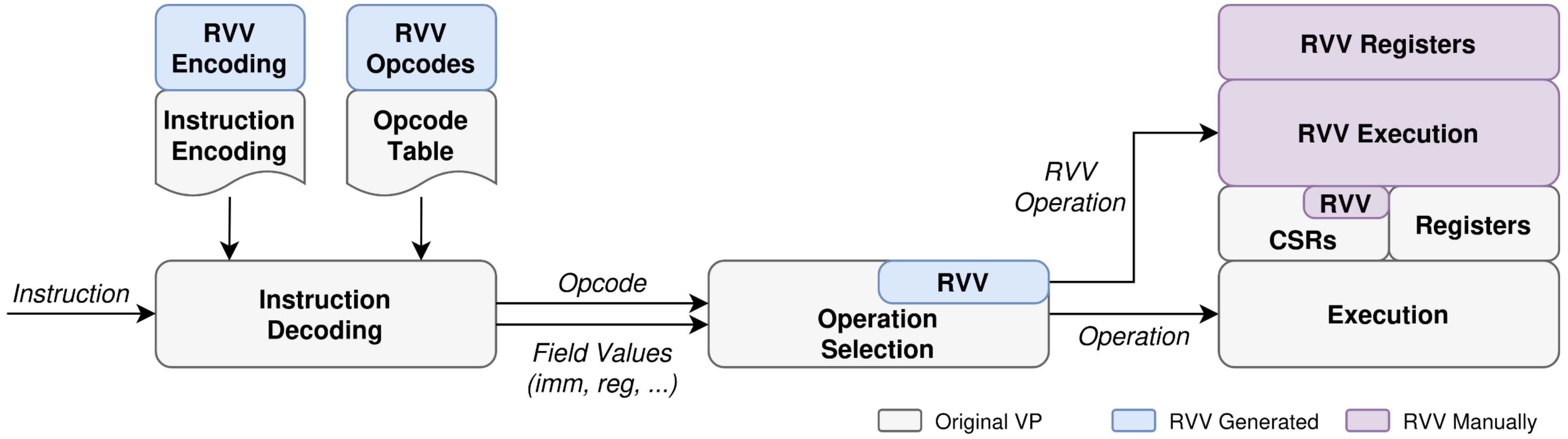
- Vectors are requested on demand
- Software does not need to know vector sizes (VLEN) of HW in advance

→ Software can automatically adapt (at runtime) to capabilities of hardware

```
void v_add(int8_t sum[], int8_t addend1[],
                int8_t addend2[], int len)
{
  /* RVV Pseudocode */

  /* #elem processed in iteration */
  unsigned int vl = 0;

  while (len > 0) {

    /* request processing of len
     * vl = min(len, VLEN/8)
     */
    vsetvli    vl, len, e8, m1

    /* load addend1 (vl) */
    vle8.v     v0, (addend1)
    /* load addend2 (vl) */
    vle8.v     v1, (addend2)
    /* add (vl) */
    vadd.vv    v2, v0, v1
    /* store sum (vl) */
    vse8.v     v2, (sum)

    /* update ptrs and len (vl) */
    addend1 += vl; addend2 += vl;
    sum += vl;
    len -= vl;
  }
}
```

RVV in RISC-V VP++ (1)

- 32 Vector Registers and 7 Control/Status Registers added
- Generic Implementation of 624 Instructions
- Integration in RV32 and RV64 ISS → Code Generator



JOHANNES KEPLER UNIVERSITY LINZ

RVV in RISC-V VP++ (2)
Progress and Outlook

Verification:

- Test Generation: *FORCE-RISCV* Framework (Instruction Sequence Generator)
- Trace Comparison: *Handcar*(*Spike*) vs. VP
- Coverage: 26936/33076 *riscvOVPsim* basic Coverage points → 81.44%

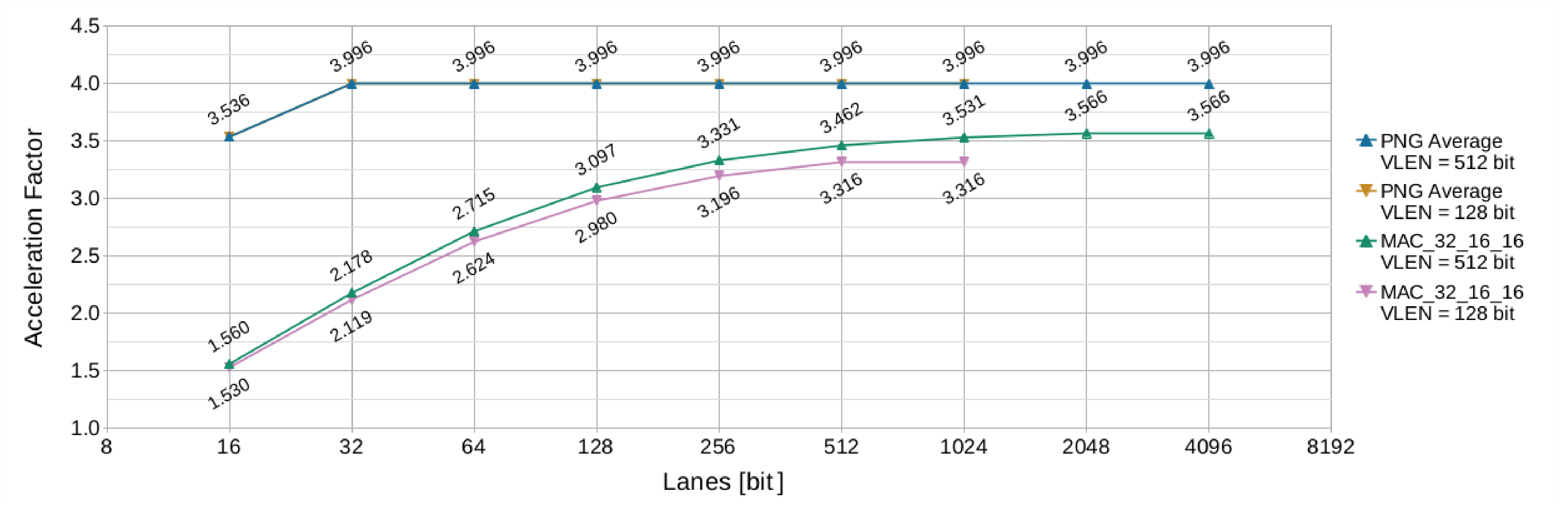
Current State: Released on November 10th 2023

- First successful bare-metal Case Studies (simple execution cycle model)
- GUI-VP Kit migrated to RISC-V VP++:
  - Quick and easy-to-use Linux experimentation environment (linux-6.6, gcc-13)
  - First Experiments: Linux Mainline with RVV, RVVRadar[1]

[1] Lucas Klemmer, Manfred Schlägl, and Daniel Große. RVVRadar: a framework for supporting the programmer in vectorization for RISC-V. In *GLSVLSI, 2022.* (https://ics.jku.at/files/2022GLSVLSI_RVVRadar.pdf)

Case Study: RVV VP for System Level Evaluation

- Simple Execution Cycle Model
- Varying two parameters corresponding to different micro-architectural RVV implementations
- Compare acceleration of vectorized algorithms to non-vectorized implementations



→ **Valuable assessments for designs (e.g. Cost/Performance non-linear)**

JOHANNES KEPLER UNIVERSITY LINZ

Manfred Schlaegl, Moritz Stockinger and Daniel Große. A RISC-V "V" VP: Unlocking Vector Processing for Evaluation at the System Level. In *DATE*, 2024 (to appear).

Conclusions

RISC-V VP++

- https://github.com/ics-jku/riscv-vp-plusplus
- New bare metal configurations incl. virtual displays
- Linux / GUI-VP Kit
  - VP-based experimentation environment for RISC-V, Linux and Graphics
  - Development environment for interactive graphical Linux applications
- Vector extension
  - 624 vector instructions using a code generator
  - First very promising results

RISC-V VP++: Unlocking the vast Linux ecosystem for Open Source RISC-V Virtual Prototypes - From Fast Bootup, VNC, Vector Extension to 3D-Games

Daniel Große
Institute for Complex Systems (ICS)
Web: jku.at/ics
Email: daniel.grosse@jku.at

SYSTEM C™ EVOLUTION DAY
NOV 16, 2023 | MUNICH | GERMANY

JOHANNES KEPLER UNIVERSITY LINZ