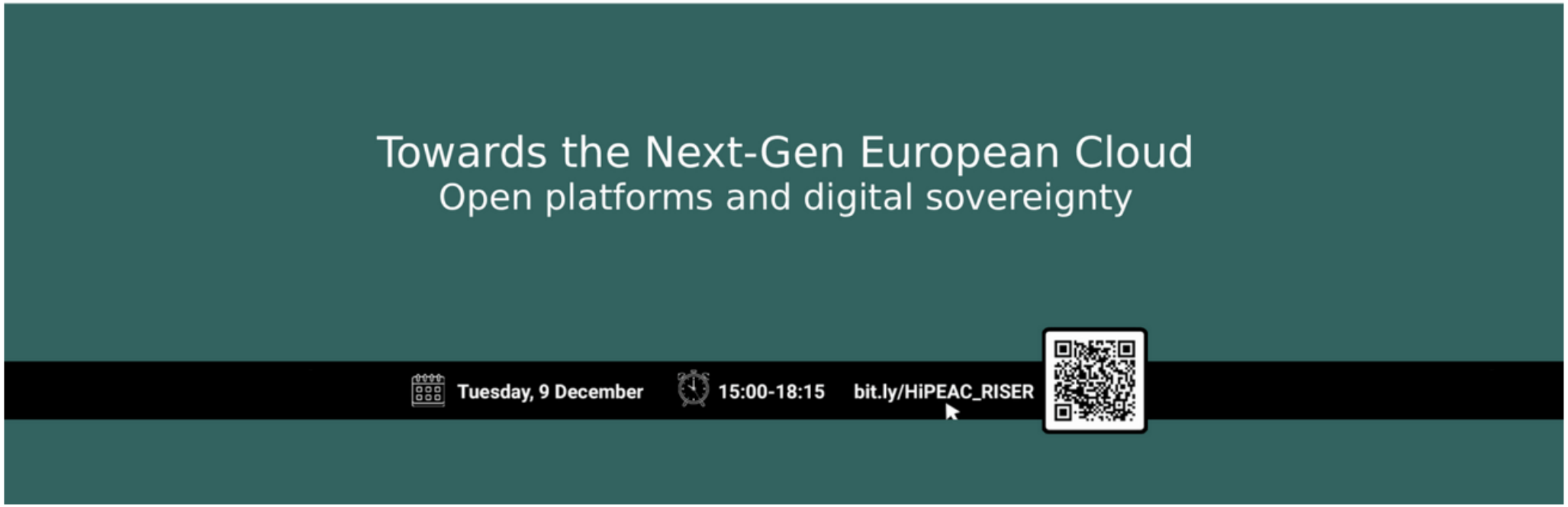
Riser Workshop

hosted by HiPEAC



[15:00 – 17:30 CET]

Organized by the RISER project [https://riser-project.eu]
Hosted by HiPEAC [https://www.hipeac.net]

Note: The workshop will be recorded.

Building RISC-V systems from the ground-up: It all starts with bare-metal

Nick Kossifidis – FORTH

December 9, 2025

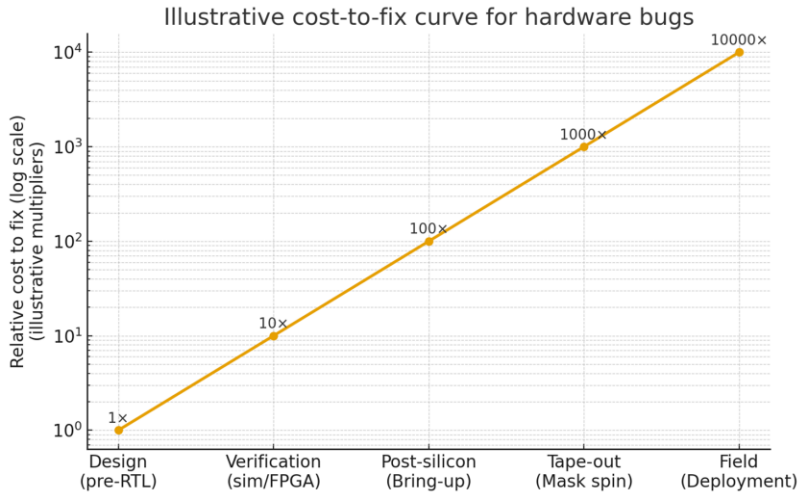
Grant agreement: 101092993
Start date: 1/1/2023 (duration: 3 years)

How it all started…

- **Horror stories from prototype bring up at FORTH**
  - During integration / design phase (FPGA)
    - Unimplemented extensions (e.g. no FPU) on RISC-V cores
    - Buggy RISC-V implementations
      - Compressed instructions + alignment
      - Atomics + alignment
      - Misbehaving MMUs
      - Misbehaving / non-compliant VPU
      - Misbehaving branch predictor (on Ariane, submitted a bug fix)
      - Misbehaving / non-compliant wfi
      - … sky is the limit
    - Various integration issues / misbehaving IPs
      - Mixed up interrupt lines
      - … sky is the limit
  - Moving on to ASIC
    - A part of the toolchain “optimized out” a part of our NoC
    - Timing issues, esp. related to high speed links
  - At the PCB level
    - Misplacement of connectors
    - Reversed / mixed-up traces

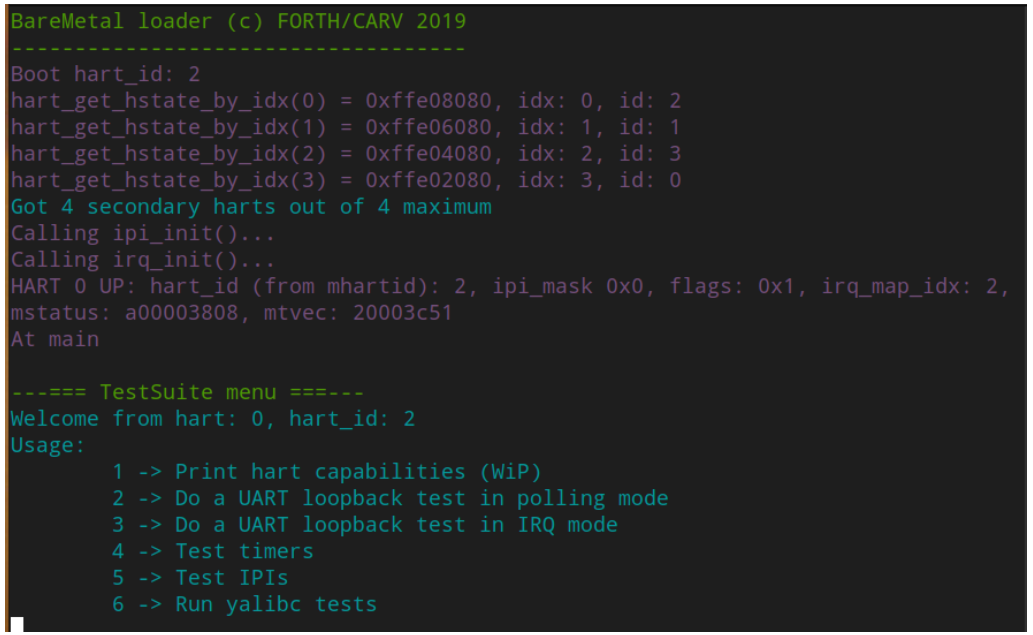
How it all started…

- **All these issues survived the hw validation / verification process**
- **The longer they survive the development process, the higher the cost of fixing them (even worse than sw use cases, especially after tape out)**
- **We need more and better tests !**
  - **More flexible code that could work asap, even without core extensions or IPs being present**
  - **Progressively more complex to increase test coverage**
  - **As efficient / small / focused as possible so that they can also be used under simulation during hw validation (e.g. verilator)**
  - **Able to run in an already broken setup (e.g. salvage what we can post-silicon, to provide as much feedback as possible to the hw team)**
  - **Able to run in a constrained environment**



How it all started…

- It made sense to create a common sw infrastructure for those tests
- As it evolved we ended up using it for other things too…
  - Benchmarks
  - Bootloaders (BootROM)
  - Education !



Evolution and current status

- **Developed as a side project since 2019**
  - On an as-needed basis
  - Open Source using Apache 2.0 (not yet released)
- **Platform layer**
  - Supports typical RISC-V SoCs
    - Harts + 16650 UART + (A)CLINT + PLIC
  - Supports multiple harts
    - Sparse hart ids with / without boot lottery
  - Supports complex memory layouts
    - A hack to support rom/ram being far away without -mcmodel=large
  - Single header for hardware-specific configuration
    - From peripheral addresses and hart infos to linker script generation
  - Support for QEMU virt machine for reference
- **Yalibc**
  - An attempt for a freestanding libc
  - For now very minimal
  - Another side-project

```
/* Assumes all harts have the same frequency
 * used for cycle counter -- based timer. */
/* Note: QEMU is not cycle-accurate, this is an
 * estimate, and will probably be host-dependent. */
#define PLAT_HART_FREQ          3100000000

/* Set to 1 to use vectored traps on mtvec, 0 for
 * direct (single trap handler with dispatch table). */
#define PLAT_HART_VECTORED_TRAPS 1


/*------ Memory Layout ------*/
#define KB      1024
#define MB      KB * 1024
#define GB      MB * 1024

/* Use the last RAM_SIZE bytes of DRAM */
#define PLAT_SYSRAM_BASE        0x80000000
#define PLAT_SYSRAM_SIZE        2 * GB

#define PLAT_ROM_BASE           0x20000000
#define PLAT_ROM_SIZE           256 * KB
#define PLAT_RAM_SIZE           2 * MB
#define PLAT_RAM_BASE           (PLAT_SYSRAM_BASE + PLAT_SYSRAM_SIZE - PLAT_RAM_SIZE)
#define PLAT_STACK_SIZE         8 * KB

#if defined(LDSCRIPT)
___rom = PLAT_ROM_BASE;
___rom_size = PLAT_ROM_SIZE;
___ram = PLAT_RAM_BASE;
___ram_size = PLAT_RAM_SIZE;
___stack_size = PLAT_STACK_SIZE;
___num_harts = PLAT_MAX_HARTS;
#endif

/*******************\
* INTERRUPT HANDLING *
\*******************/

/*------ TIMER ------*/

/* Base address for a SiFive CLINT compatible device
 * set to 0 if no CLINT is present */
#define PLAT_CLINT_BASE         0x2000000

/* MTIMER frequency in Hz, set to 0 to disable */
/* See note on PLAT_HART_FREQ, this is inaccurate too. */
#define PLAT_MTIMER_FREQ        9000000

/* In case of ACLINT, define MTIME/MTIMECMP separately */
#define PLAT_MTIME_BASE         0
#define PLAT_MTIMECMP_BASE      0
```

Work in progress

- **Platform layer**
  - **AIA support**
    - **ACLINT + APLIC first**
    - **IMSIC + APLIC later on**
  - **(e)PMP support**
- **Yalibc**
  - **Switch from pre-processor macros to linker aliases (experimenting for now)**
  - **Create a comprehensive test suite targeting C11+ compliance**
  - **Keep adding functionality without hurting code quality or binary size**
- **Cleaning things up so that I can release this at some point**
  - **Having fun with compiler optimizations fighting each other**

Future work

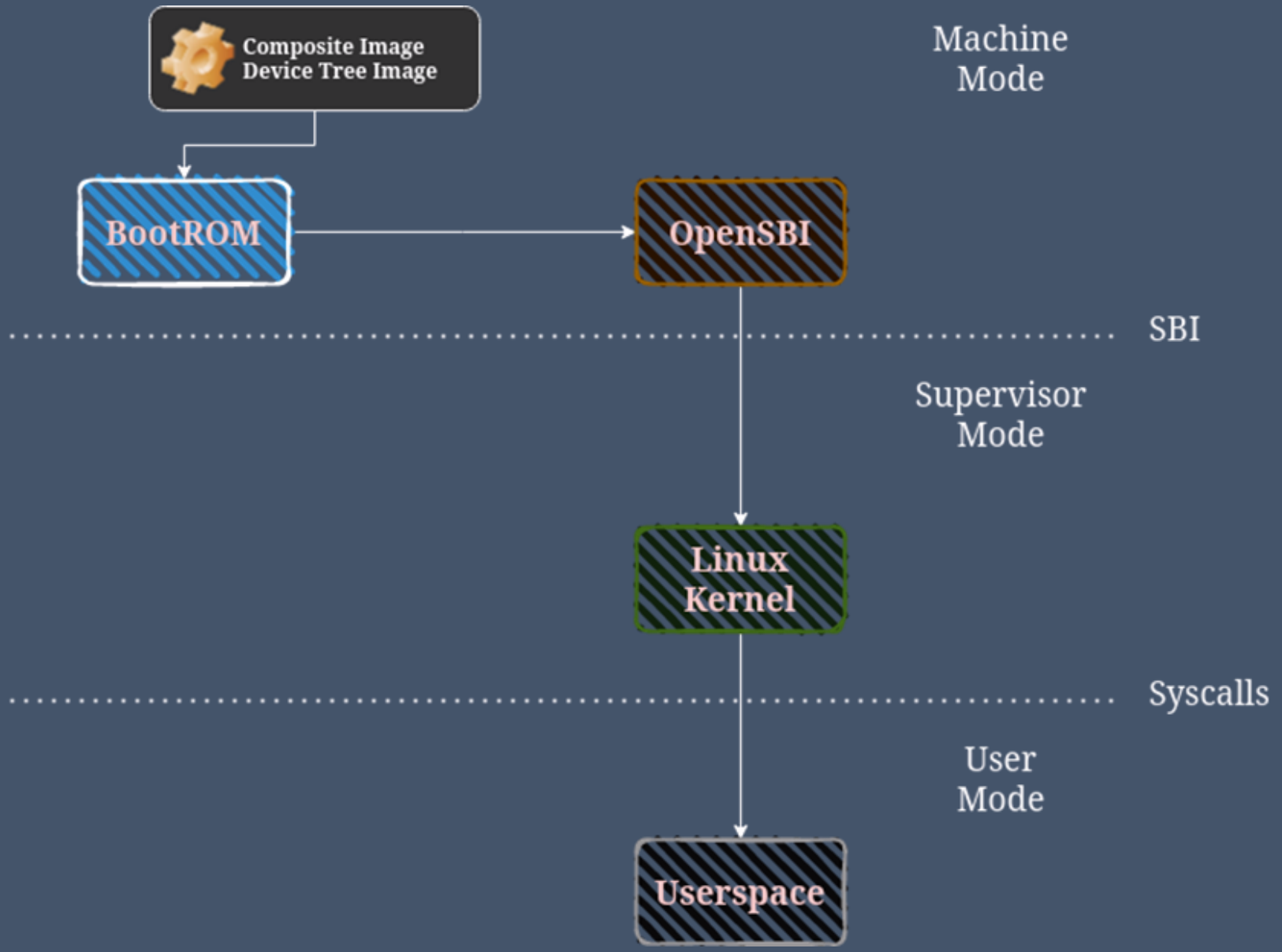
- Infrastructure for running on S-mode with virtual memory
  - With IOMMU support
- Comprehensive test suite for the platform layer (based on the various tests I have lying around + riscv test suite)
- Hart probing / profiling suite
- Support for more RISC-V extensions and non-ISA specs (e.g. IOPMP/IOMPT)

An example use case: NetBOOT

- A zero-stage boot loader (aka BootROM) that can fetch boot images from the network
- Used for prototype bring up when no storage is available
- Can also be used in production, also on a board controller or a security controller
- Includes:
  - Ethernet driver (supported: virtio-net, emaclite, carv-ethdma)
  - Tiny network stack
  - DHCP client (a decent one)
  - TFTP client (includes blocksize and windowsize options)
- All in ~19K (-Os, no debug symbols, no ANSI colors) but I believe I can optimize it further

Boot flow for Linux (full)

Boot flow for Linux (simplified)



Future work on NetBOOT

- Unified image format
  - Including both composite image + device tree
- Secure boot
  - Signature verification of unified image
    - Using an external TPM / RTM (e.g. Caliptra)
    - Using a software RTM as fallback
- Add support for SPI flash storage (JEDEC compliant)
- Add support for eMMC/SD storage
- Target binary size: 32K max
  - For the external RTM case

Thank you for your attention.

Disclaimer:

"Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them."