Writing a network-capable BootROM for RISC-V prototype bring-up

Nick Kossifidis <mick@ics.forth.gr>
Antony Chazapis <chazapis@ics.forth.gr>
CARV@ICS/FORTH

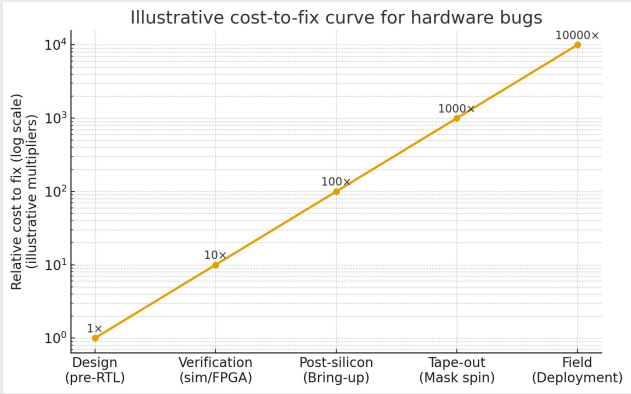
How it all started…

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

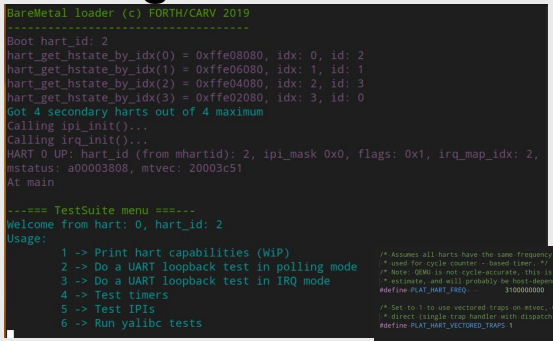
How it all started…

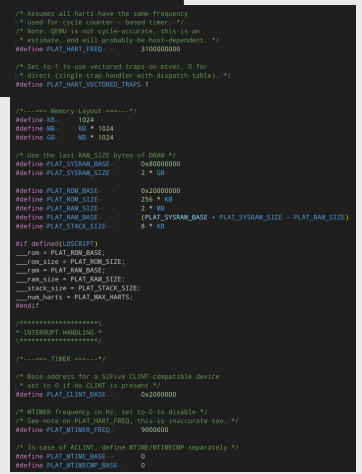
- All these issues survived the hw validation / verification process
- The longer they survive the development process, the higher the cost of fixing them (even worse than sw use cases, especially after tape out)
- We need more and better tests !
  - More flexible code that could work asap, even without 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



BareMetal SDK

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





BareMetal SDK - The pain

- Weird memory layouts from the hw team
  - E.g. ram is too far away from rom
    - PC-relative addressing won't work
    - GP-relative is limited to 4KB for .data/.bss
    - Large memory model not yet ratified
      - It increases binary size by storing lots of symbols
      - It stores them in .text instead of .rodata !
- Flexibility for prototypes that are still WiP
  - For example maybe not all atomic ops are available
  - Or maybe there is no PLIC/CLINT etc yet
  - Or we may have some quirks etc
- Size constrained since BootROM consumes chip area (also on FPGAs)
  - And you need to write everything from the hart's first instruction
  - Compiler optimization passes fighting each other
- ...

BareMetal SDK - The joy

- **FULL CONTROL!!!**
- **Much simpler/cleaner code than e.g. having to follow multiple abstraction layers on an OS environment designed to be multi-arch.**
- **RISC-V specs are not that complicated to follow (especially if you are involved in the process).**
  - **E.g. implementing bare metal drivers for (A)CLINT/(A)PLIC/IMSIC, or trap handling was straight forward.**
- **Great opportunity to stretch your skills, experiment, and learn new stuff.**
- **In case you get to work with students, it can be a great teaching tool.**

NetBOOT - A real-life production use case

- It’s easier to have network than storage early on
  - You can add a NIC in your FPGA design and go outside
  - You can’t add a persistent flash
  - No need for eMMC/SD/AHCI etc, just mmio/dma access
- There are very simple network interfaces to play with
  - E.g. Xilinx emaclite.
  - We have our own 1Gb one with the same programming interface as Xilinx AXI DMA and our own MAC.
- You can do everything !
  - Fetch boot images to get FSBL/Kernel/DTB etc.
  - Access system via ssh.
  - Mount rootfs over NFS and e.g. boot Redhat Enterprize / Ubuntu
  - Access the internet, or other hosts (in our case play with MPI etc).

NetBOOT - A real-life production use case

- **Networking stuff:**
  - Simple ethernet abstraction, without interrupts.
    - Xilinx emaclite
    - Our own AXI-DMA NIC
    - VirtIO-net (so that it works with QEMU virt machine)
  - Tiny network stack on top (IPv4/UDP/ARP)
  - DHCP with support for some useful options
  - TFTP with blocksize/windowsize/tsize negotiation and blocknum wraparound
- **Image parser:**
  - Image container format with support for multiple partitions/image types/system units.
  - Simple integrity checks for now, but signature/secure boot support is also part of the plan.
  - LZ4 decompressor

NetBOOT - A real-life production use case

- **Code size (initial public release):**
  - **Emaclite driver -> 250loc**
  - **Our NIC driver -> 320loc**
  - **VirtIO-net driver -> 413loc**
  - **Net -> 430loc**
  - **DHCP -> 868loc**
  - **TFTP -> 336loc**
  - **LZ4 -> 91loc**
  - **Image parser -> 282loc**
  - **Main (wrapper) -> 45loc**
- **Binary size:**
  - **32K with debugging + ANSI colors etc**
  - **Much less if I remove debugging / colors and add -Os in the mix (but -Os sometimes breaks things). Best case ~20-24K in previous tests.**

NetBOOT - A real-life production use case

- **What to do with the remaining ~8-10K…**
  - **Secure boot using Caliptra RTM**
    - **Image parser is ready for it, I'll need some more ram to store the public key to pass to Caliptra, and some code space if I decide to do the SHA384 myself instead of using Caliptra.**
  - **Add support for flash (e.g. over SPI/eMMC)**
- **We'll also spin up a version that does OpenDICE with built-in crypto (but that won't fit in 32K, I'll probably go for 64K target).**

Git repos...

We hope you find the whole thing fun and useful too:

https://github.com/CARV-ICS-FORTH/BareMetal

https://github.com/CARV-ICS-FORTH/NetBoot

Also let us know if you find any issues, or have any feedback to share.
Contributions are always welcome :-)

Questions ?

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."

European Commission
Horizon Europe
2021-2027


RISC-V
SUMMIT EUROPE 2026
BOLOGNA | 8-12 JUNE, 2026
Palazzo dei Congressi