OSDForum
OPEN SOURCE DEVELOPER FORUM

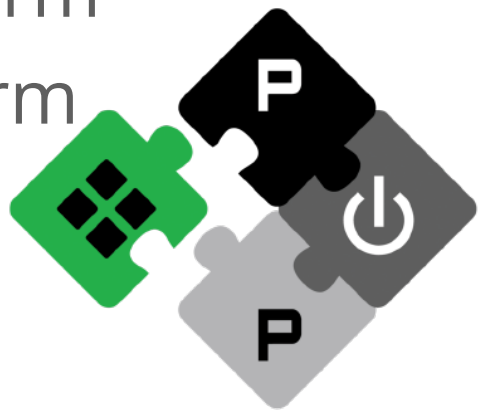
OPENHW™

Overview of CORE-V MCU & APU FPGA based platforms

Florian Zaruba florian@openhwgroup.org
T @be4web T @openhwgroup
www.openhwgroup.org

Outline

- Brief history
- Introduction to the CORE-V MCU Platform
- Introduction to the CORE-V APU Platform
- How debug works
- FPGA Platform
- Meet the team
- I want to contribute!

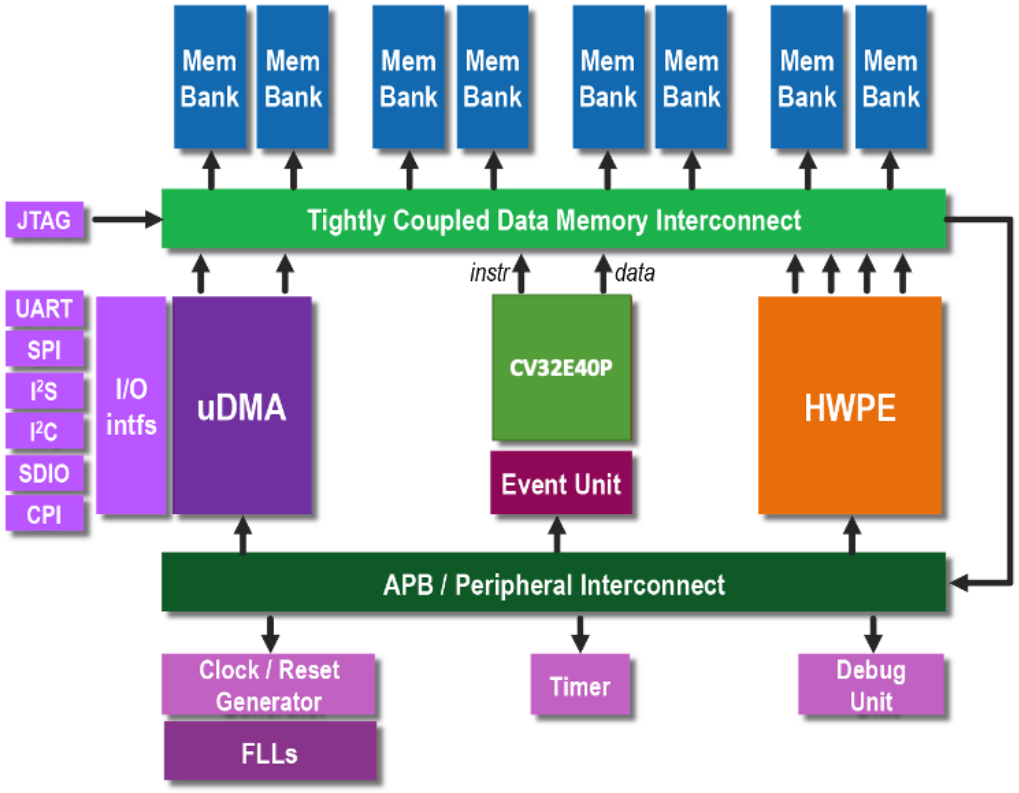


History

- Both platforms originate from the PULP Project
- CORE-V MCU is derived from PULPissimo
  - Efficient micro-controller
  - Improved CV32E40P
  - Standardized OBI protocol
  - APB for uDMA subsystem
- CORE-V APU is derived from Ariane's SoC
  - UNIX-capable system
  - Minimal infrastructure to demonstrate the core
  - AXI-based
  - Xilinx Peripherals (SPI, DW converter)

CORE-V MCU

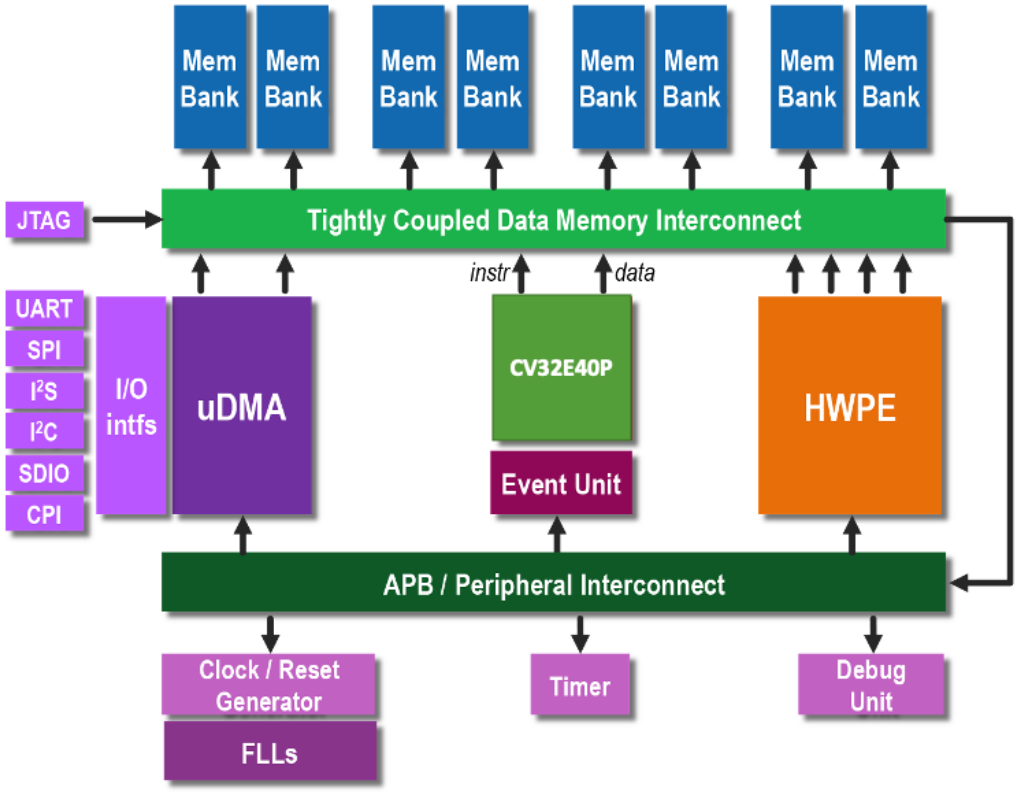
- Shared memory
  - Unified Data/Instruction Memory
- Support for Accelerators
  - Direct shared memory access
  - Programmed through APB bus
- uDMA for I/O subsystem
  - Can copy data directly from I/O to memory without involving the core
- Used as controller in larger systems



https://github.com/openhwgroup/core-v-mcu

CORE-V MCU

- Rich set of peripherals:
  - QSPI (up to 280 Mbps)
  - Camera Interface
  - I2C/I2S
  - JTAG, GPIOs
  - Interrupt controller
  - boot ROM
- Custom accelerators

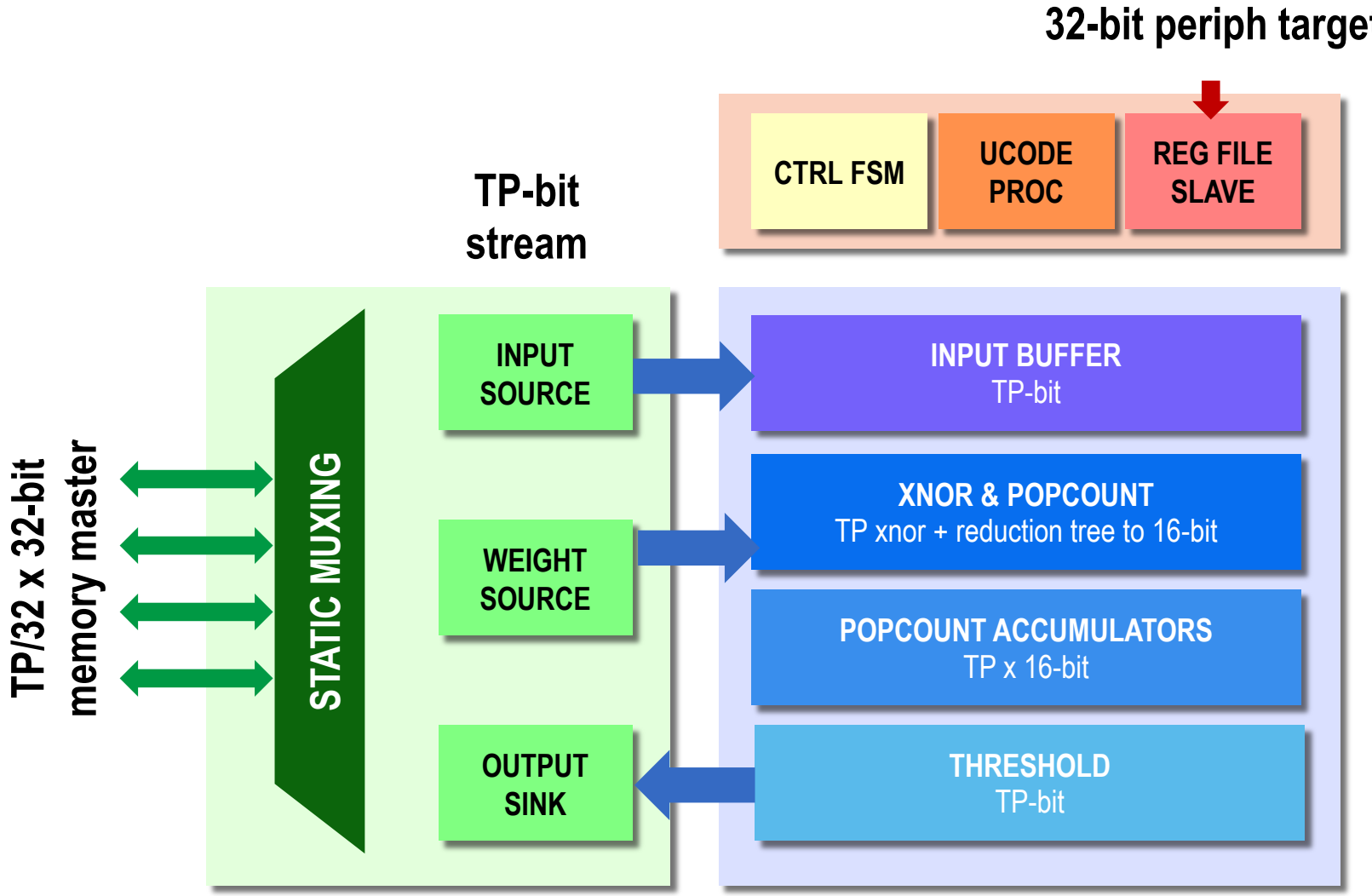


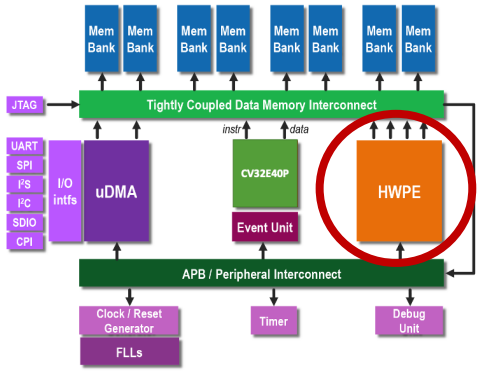
PULP Interrupt Controller

- Generates up to 32 requests
  - Events vs. interrupts
- Mapped on the APB bus
- Receives events in a FIFO from the SoC event generator
  - Unique interrupt ID
- Mask, pending, ack, event id registers
- Special set, clear, read, write operation registers

- Sources:
  - Timers
  - GPIO
  - Custom accelerator
  - Events from the uDMA
- CV32E4P
  - Support for standardized and “fast” RISC-V interrupt mechanisms

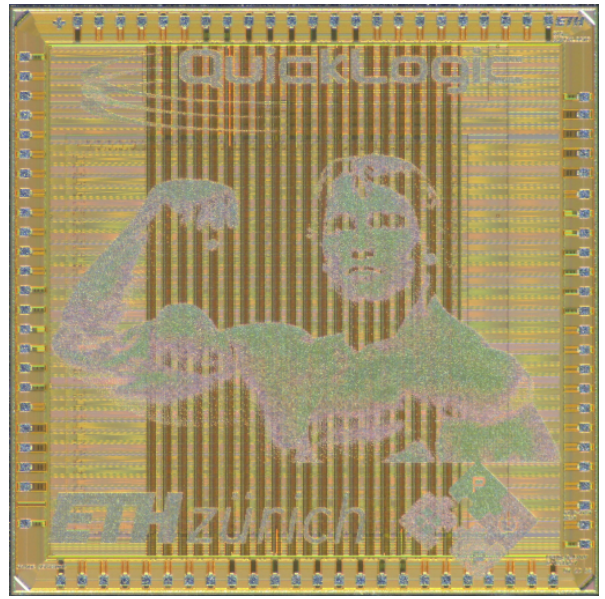
Example for a Custom Accelerator

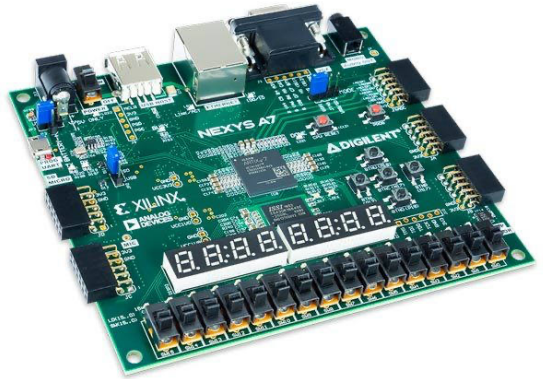




CORE-V MCU Physical Design

- Two realizations:
  - FPGA (Nexys A7)
  - ASIC (Globalfoundries 22FDX)
- ASIC coming early next year
  - Including eFPGA by Quicklogic
- Protoype your custom accelerator
  - FPGA
  - eFPGA

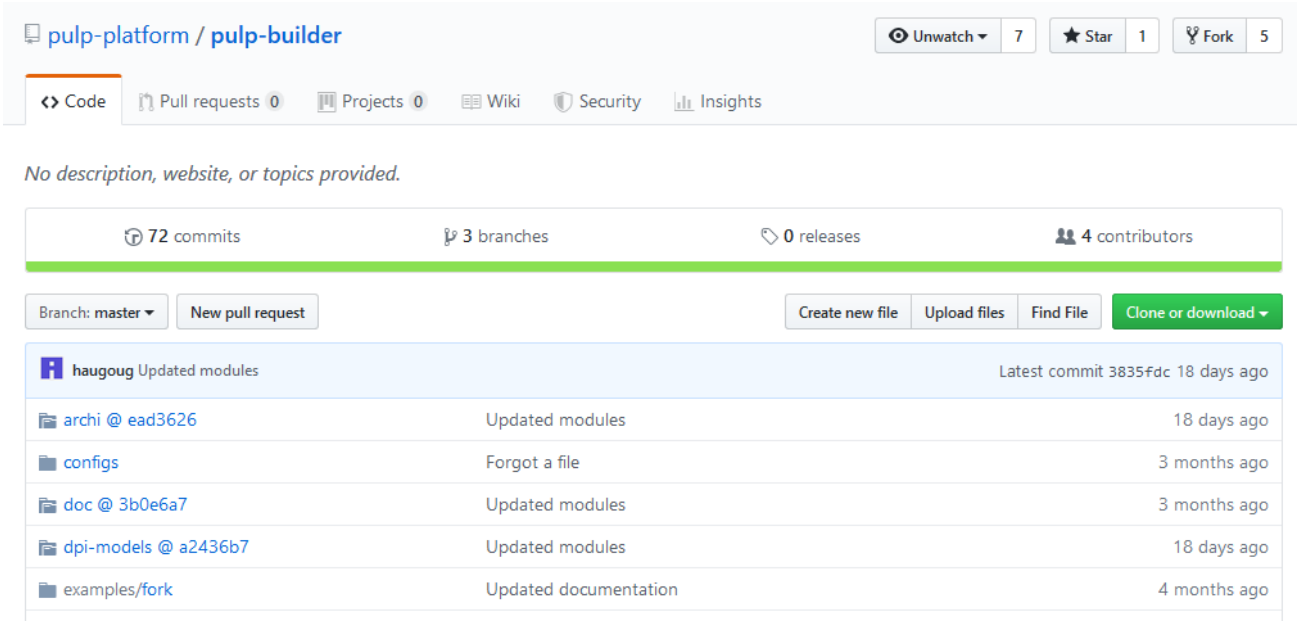




QuickLogic®

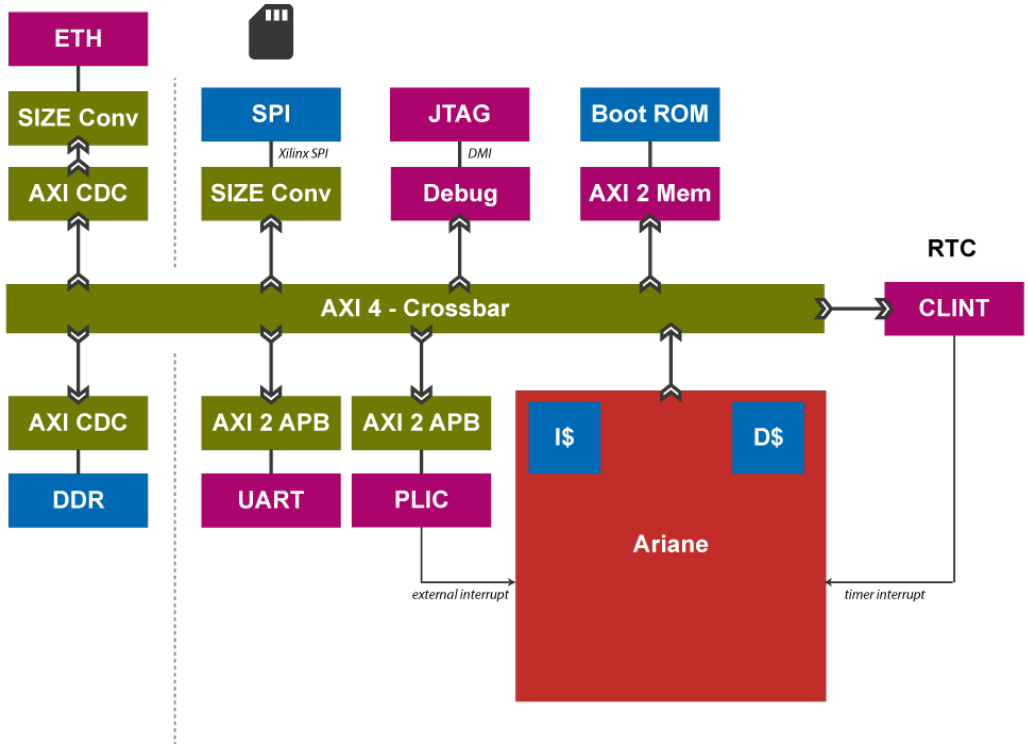
Software

- The SDK contains all the tools and runtime support for PULP based microcontrollers
- Provides:
  - HAL
  - crt0 and linker scripts
  - higher level functions (API)
- CORE-V IDE and GCC sessions in the afternoon!



CORE-V APU

- Minimum set of peripherals to boot Linux
- Code is on SD Card
- Zero-stage bootloader is in boot ROM
- UART as main UI
- Ethernet for network connectivity



https://github.com/openhwgroup/cva6

Absolute minimum necessary to boot Linux?

- Hardware
  - 64 or 32 bit Integer Extension
  - Atomic Extension
  - Privilege levels U, S and M
    - MMU (maybe NO-MMU port coming)
  - FD Extension or out-of-tree Kernel patch
  - 16 MB RAM
  - Interrupts
    - Core local interrupts (**CLINT**) like timer and inter processor interrupts
  - Serial
- Software
  - Zero Stage Bootloader
  - Device Tree Specification (DTS)
  - RAM preparation (zeroing)
  - Second stage bootloader
    - BBL
    - Uboot
    - …
  - Linux Kernel
  - User-space applications (e.g.: Busybox) or distro

Interrupt Handling

CLINT

- Core local interrupts
- Relatively fast to handle (short latency)
- More expensive HW tied to core

PLIC

- Platform level interrupt controller
- Part of UNIX platform specification
- Latency insensitive interrupts
- More advanced prioritization

Fast interrupts as a hybrid

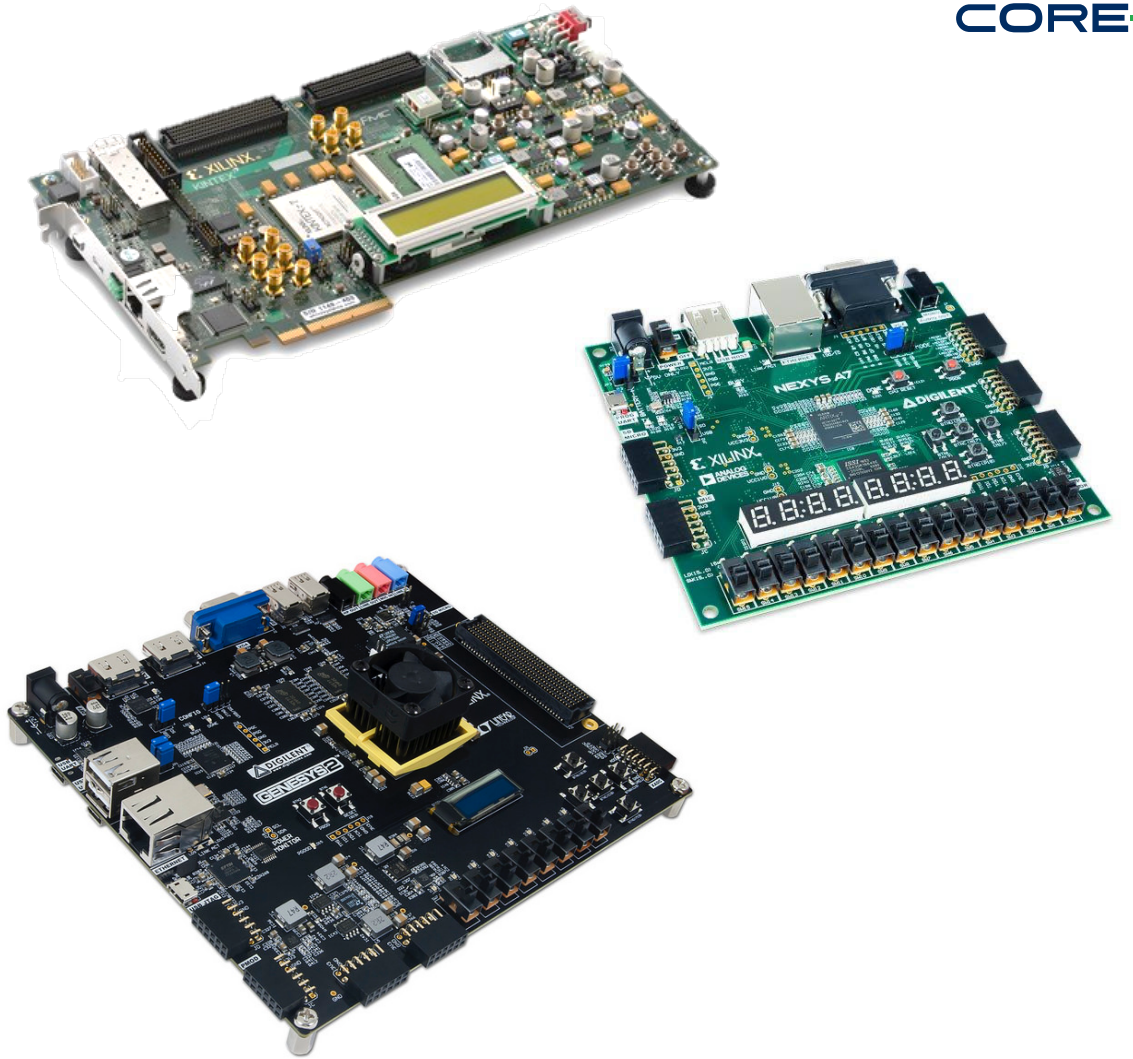
Device Tree

- How to detect capabilities?
- Open Firmware
  - Device Tree Format
  - Compiled to binary blob
- Passed to Bootloader in a0
- Parsed and filtered by each bootloader

```
/dts-v1/;
 / {
    #address-cells = <2>;
    #size-cells = <2>;
    cpus {
        CPU0: cpu@0  {
            clock-frequency = <50000000>; // 50 MHz
            device_type = "cpu";
            compatible = "eth, ariane",
            "riscv"; riscv,isa = "rv64imafdc";
             mmu-type = "riscv,sv39"; // HLIC - hart local interrupt controller
             CPU0_intc: interrupt-controller {
                interrupt-controller;
                compatible = "riscv,cpu-intc";
          };
       };
    };
    memory@80000000 {
        device_type = "memory";
        reg = <0x0 0x80000000 0x0 0x40000000>;
    };
};
```

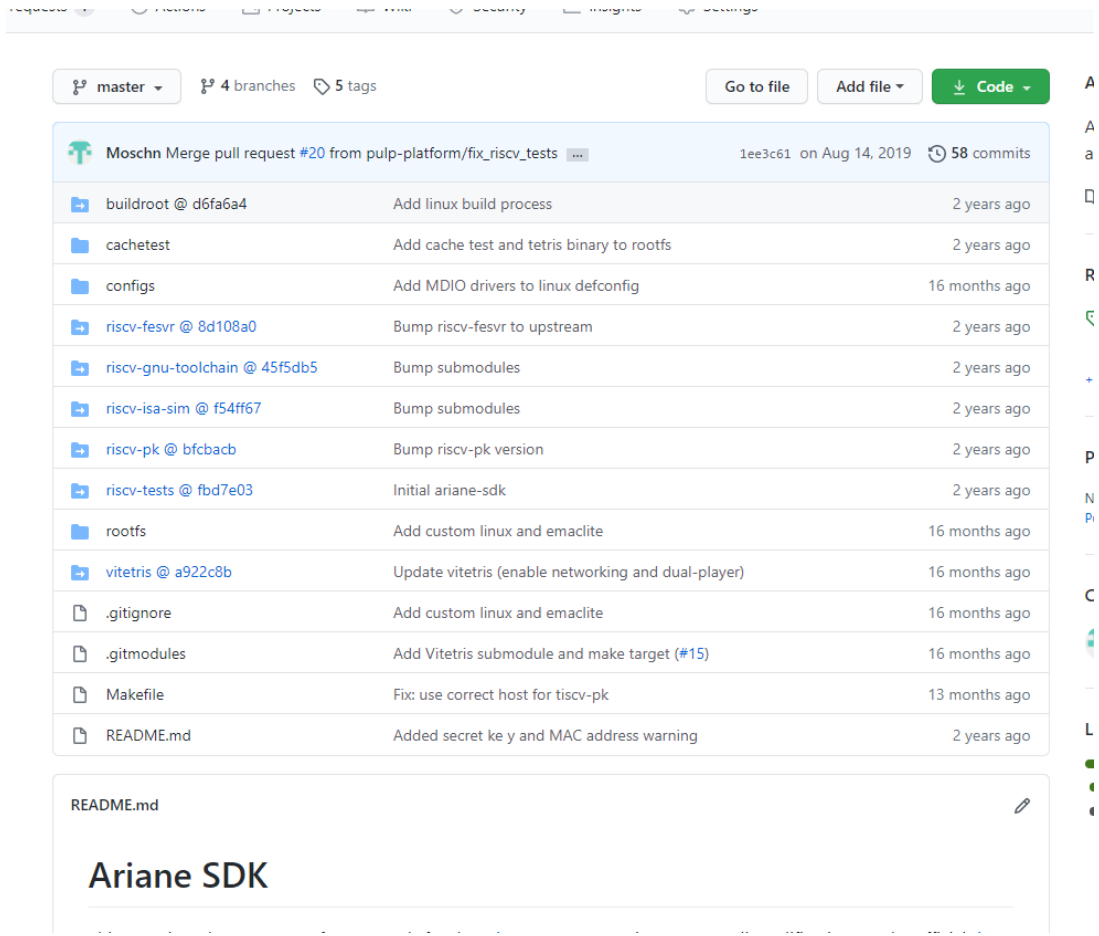
FPGA

- Default Board: **Genesys II**
  - Wide-spread adoption
  - Rich set of board peripherals
  - Reasonably cheap (discounts through OpenHW, academics)
- Alternative: **Nexys A7**
  - Cheaper
  - Higher availability
- “User ports”:
  - VC707, KC705
  - Ultrascale Boards

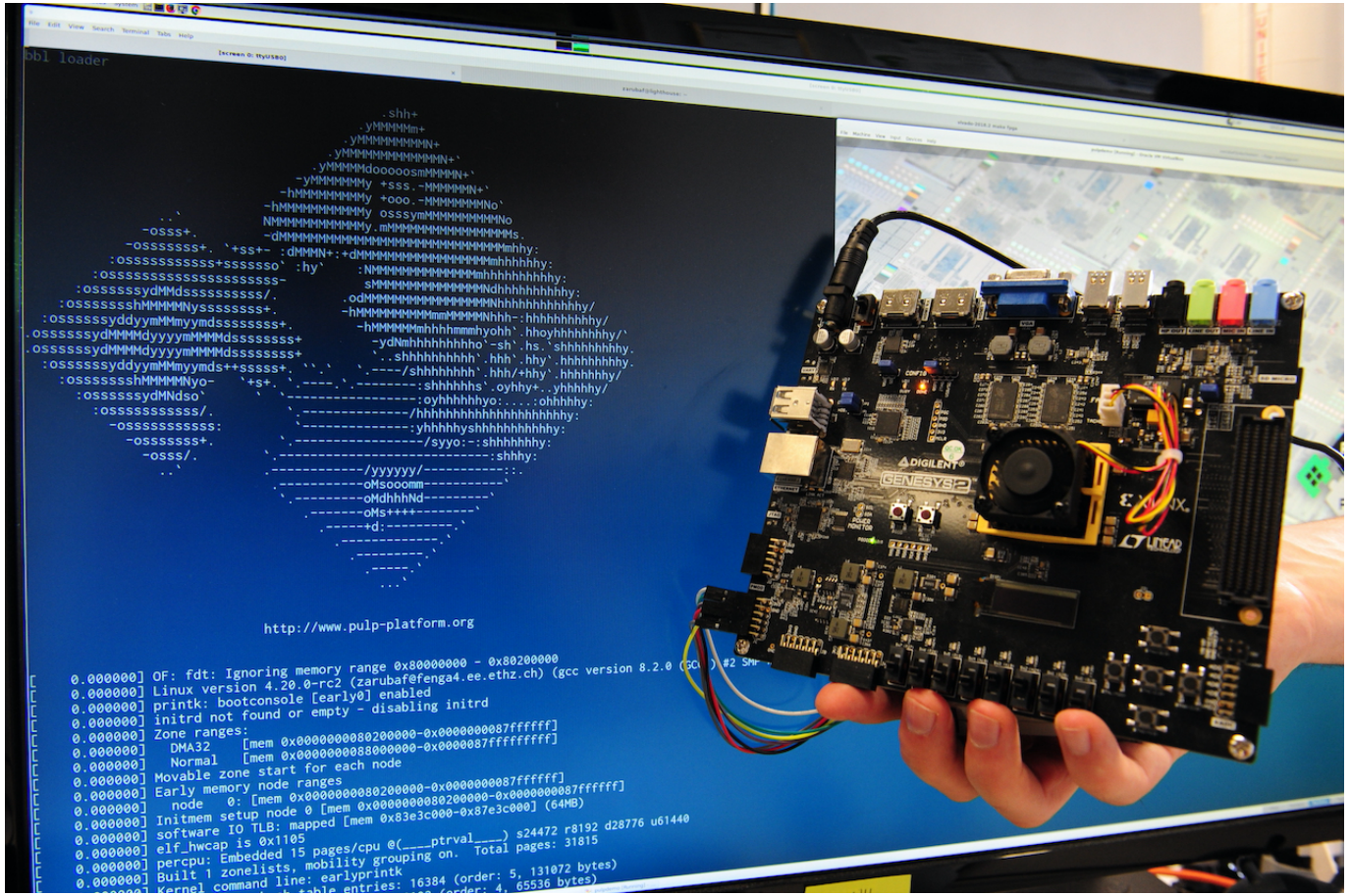


Software

- We provide a basic SDK
  - Toolchain
  - Buildroot
  - Linux
- Linux Applications
- Pre-built images
- SW task group's charter:
  - Simplify
  - Provide images



Demo: Booting Linux



RISC-V Debug

- Draft specification 0.13
- Defines debug registers for
  - run/halt/single-step
  - reading/writing GPR, FPR and CSRs
  - Querying hart status
- JTAG interface
- OpenOCD support
- Our choice: **Execution Based**

- Standard allows for different debug probes to be used
- IDE integration

ASHLING

Debug (Detailed)

- Special Debug Mode
  - Less intrusive, leverage existing pipeline
  - DPC, debug-interrupt, `dret` and `ebreak`
- Halt, Resume, (Single-)step
  - Only required command: Abstract read (read floating-point register, register and CSR)
  - Debug Transport Protocol (orthogonal)
    - Currently only JTAG specified
- SystemVerilog reference implementation available

Meet the Team: HW and SW Task Groups

- Hardware TG
  - Hugh Pollitt-Smith (CMC)
  - Tim Saxe (QuickLogic)
- Software TG
  - Jeremy Bennet (Embecosm)
  - Yunhai Shang (Alibaba T-Head)





OpenHW Director of Engineering: Florian Zaruba

Get on Board!

- Development is in the open!
- CORE-V MCU
  - https://github.com/openhwgroup/core-v-mcu
- CORE-V APU
  - Directly in the core's repository
  - https://github.com/openhwgroup/cva6