OpenPiton+Ariane: The RISC-V Hardware Research Platform

Princeton University and ETH Zürich

http://openpiton.org
http://pulp-platform.org
Operating System and System Software
Privileged Specification 1.11 (WIP)

- Defines Control and Status Registers (CSR)
- Defines instructions to RMW CSR
- 3 Rings of operation
  - Machine-, Supervisor- and User-Mode
  - Hypervisor WIP
- Exceptions + IRQ support
  - IRQ/Exception stack
  - Wait for Interrupt (WFI) instruction
  - Specification for platform level interrupt controller
  - Instructions to enter/return from exceptions
- Virtual Memory
  - Page-based, 32-bit PA, 39-bit PA and/or 48-bit PA
- Platform Configuration String (DTS)
Open source system stack

• Applications run on Linux
• Linux manages HW, calls via SBI to BBL
• Modified BBL (Berkeley Bootloader)
  – Acts as a Firmware
• Open source hardware

• You can read, modify and recompile all of them!
Boot Process

1. Zero Stage Bootloader (ZSBL)
   – Core Starts fetching from ZSBL ROM
2. Reset Code
   – Clears registers and on-chip memories
3. First Stage Bootloader (BBL)
   – Sets up trap table, copies self to memory
   – Loads Linux kernel from SD card
4. Linux
   – Page table setup
   – Driver loading
   – Environment prepare
5. init (Busybox) – start shell
ZSBL

• Reset Code
• Bare metal driver for:
  – UART: Early console
  – SD Card/SPI
  – Device Tree (Open Firmware)
• Basic peripheral setup
• Copy FSBL image from SD into memory
• Only one core performs setup routines
FSBL

- Modified BBL
  - Remove atomic operations on peripherals
  - Zero data section
- Setup:
  - UART: Early console
  - PLIC (Platform Level Interrupt Controller)
  - CLIC (Core Local Interrupt Controller)
  - Filter Device Tree
- Activate secondary cores
- Start Linux Kernel boot process
Linux

- Detect Hardware via Open Firmware
- Setup Virtual Memory
- Load Kernel Modules
- Start `init`
Building Ariane SDK

• Clone our git repository from https://github.com/pulp-platform/ariane-sdk.git
• `git submodule update --init --recursive`
• If toolchain already installed set:
  – `export RISCV=/path/to/install/riscv/toolchain`
  – Otherwise Makefile will install the right toolchain
• `make all`
Components

• riscv-gnu-toolchain - GCC
• riscv-pk – Contains BBL (patched)
• Buildroot – Upstream buildroot system
  – Set of Makefiles and patches that automates building
    a bootable Linux environment
• rootfs – Overlay for rootfs
  – Initramfs in use
  – Directory structure which overlays rootfs
  – Use to include executables and other files into the
    image
• configs – Custom configuration
Customizing

• Buildroot automatizes most of the build process
  – Slightly patched Kernel set
  – Ethernet driver fixes and custom drivers.
  – Hope to eventually upstream
• Buildroot wraps around that:
  – cd buildroot
  – make linux-menuconfig
  – make linux-savedefconfig
  – Install permanently into SDK
  – cp output/build/linux-*/defconfig
    ../configs/linux-defconfig
Customizing

• (Most) Packages come from busybox
  – Lightweight re-writes of most GNU applications

• Buildroot wraps around that:
  – cd buildroot
  – make busybox-menuconfig
  – Install permanently into SDK
  – cp output/build/busybox-*/.config
    ../configs/busybox.config
Flashing the SD Card

• Generate FSBL with Linux payload:
  – make bbl.bin

• Preparing the SD Card
  • Only needs to be done once
  • Two partitions: 1. Bootloader + Kernel 2. Rootfs
    – sudo fdisk
    – sudo sgdisk --clear --new=1:2048:67583 --new=2 --typecode=1:3000 --typecode=2:8300 /dev/sdx

• Copy Disk Image
  – sudo dd if=bbl.bin of=/dev/sdb1 status=progress oflag=sync bs=1M