Building Open Hardware with Open Software

Michael Tretter – m.tretter@pengutronix.de

About Me

- Michael Tretter
- Embedded Linux developer
- Pengutronix
- Graphics Team

Agenda

- Motivation
- How to build FPGA bitstreams
- Experience report
- Next steps

Expected Background Knowledge

- What is an FPGA?
- How do FPGAs work?
- What is RISC-V?
- Building blocks of SoCs?

FPGA Use Cases

- High data throughput
- Real time
- High parallelism

Motivation

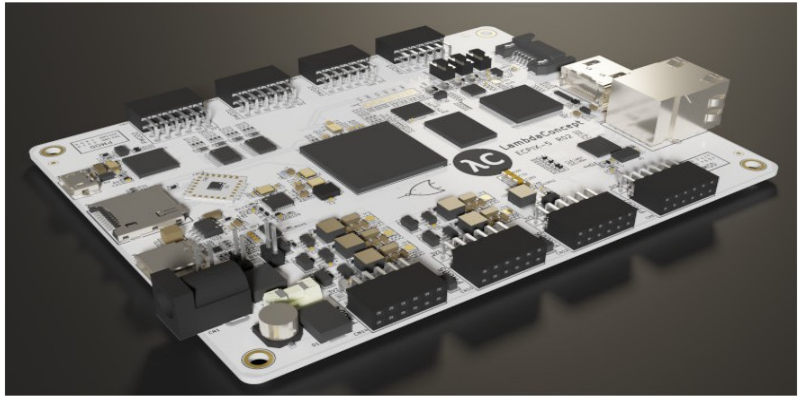
- Vendor tools proprietary, closed source, questionable licensing
- Open source toolchain: Yosys, nextpnr
- FPGAs are affordable
- Many open source IP cores due to RISC-V

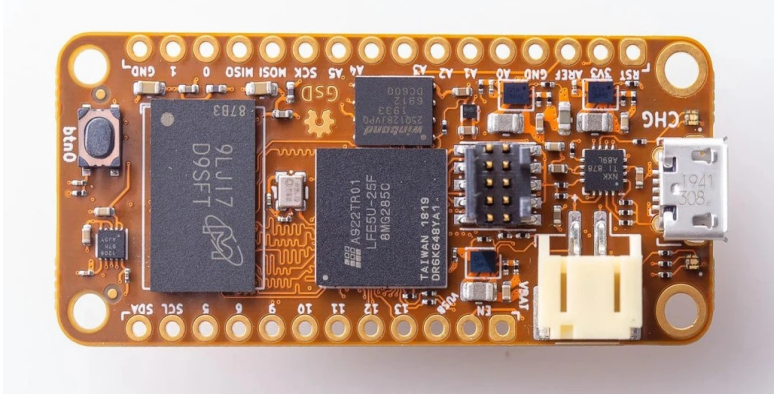
Starting Question

Can we use the open source toolchain

for (small) production systems?

Hardware





- Lambda Concept ECPIX-5
- Orange Crab

Does the toolchain work?

- “Hello World!” of hardware development
- orangecrab-examples: blink

```
yosys -p "synth_ecp5 -json blink.json" blink.v
nextpnr-ecp5 --json blink.json --textcfg blink_out.config \
                                --25k --package CSFBGA285 --lpf orangecrab_r0.2.pcf
ecppack --compress --freq 38.8 --input blink_out.config --bit blink.bit
```

Real World Example

- Implement CPU core in FPGA
- Run Linux as software on the CPU core

VexRiscV

- RV32I[M][A][F[D]][C] instruction set
- Implemented in SpinalHDL
- Optional MMU
- Optional SMP cluster
- Optional privilege levels
- Optional data and instruction cache

Rocket CPU

- RV64GC instruction set
- Implemented in Chisel
- MMU
- Floating point unit
- Machine, supervisor, user privilege levels

Integration into System on a Chip

- CPU core without peripherals is useless
- SoC environment generators
- SoC should allow to use either CPU core

LiteX

- Framework for creating FPGA SoCs
- Support for mixed languages: VHDL, Verilog, SpinalHDL
- Build backends for open-source toolchains
- Ecosystem of cores: LiteDRAM, LitePCIe, LiteEth...
- And a lot more... :)

Linux on LiteX Rocket I

- Install tools on host system
- Install prebuilt RISC-V toolchain

```
wget https://raw.githubusercontent.com/[...]/litex_setup.py
```

Linux on LiteX Rocket II

```
litex-boards/litex_boards/targets/lambdaconcept_ecpix5.py \
            --build \
            --cpu-type rocket --cpu-variant linuxd \
            --sys-clk-freq 50e6 \
            --with-ethernet --with-sdcard
```

Linux on LiteX Rocket III

- Build BBL, Linux, Busybox-based initramfs
- Manually put everything together
- Load bitstream with OpenOCD

LiteX Internals

- Python description
- Python build tool
- Build first stage (ROM) bootloader
- Generates and calls script to build bitstream
- Configuration and build script are mixed

Going further

- Collaboration?
- Reproducibility?
- Multiple projects?
- Continuous integration?

Yocto

- Flexible tools to create tailored Linux images
- Package and version management
- Specified host tool versions

meta-hdl

- Yocto meta-layer for HDL tools
- https://github.com/nathanrossi/meta-hdl
- Recipes for synthesis, place and route, examples

meta-ptx-fpga

- Recipe for Linux on LiteX Rocket
- Fixes for LiteX, etc.
- Adjustments to Linux image

```
MACHINE=ecpix5-vexriscv bitbake core-image-minimal
MACHINE=ecpix5-rocket bitbake core-image-minimal
```

Continuous Integration

- Jenkins as CI infrastructure for Yocto image
- Building the bitstream is just another Yocto package
- Runtime tests on actual hardware planned

Open Issues

- Build bitstreams with latest LiteX
- Enable multi core for VexRiscV
- Replace Busybox on Rocket CPU
- Separate rootfs and kernel on Rocket CPU
- Add virtual package: “bitstream”

Pre-generated CPU Cores

- Pre-generated Rocket CPU and VexRiscV in LiteX
- Generated on demand in normal LiteX checkout
- Chisel, SpinalHDL not integrated in Yocto
- Reproducible CPU Cores?

Comparability of Results

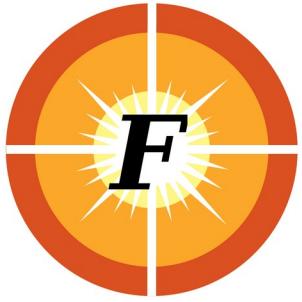
- Boards are identical
- Different SoC configuration in LiteX
- Significant differences under the hood
- Can we actually compare the configurations?

Alternatives to LiteX

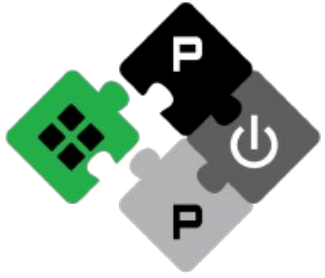
- VexRiscV: Briey, Murax

- Chipyard

- Fusesoc



- Pulp



Use Case for Linux on RISC-V in FPGA

- Evaluation of development environment and toolchain
- Education and experimentation of CPU features
- FPGA: high data throughput, real time, parallelism
- Any real world use cases for Linux on RISC-V in FPGA?

Next Steps

- Fix VexRiscV and Rocket CPU with latest LiteX
- Setup runtime tests with real hardware
- Fix multi core setup with VexRiscV
- Integrate custom IP cores into SoC

Thank You!

Michael Tretter – m.tretter@pengutronix.de

Further Reading

- https://github.com/SpinalHDL/VexRiscv
- https://github.com/chipsalliance/rocket-chip