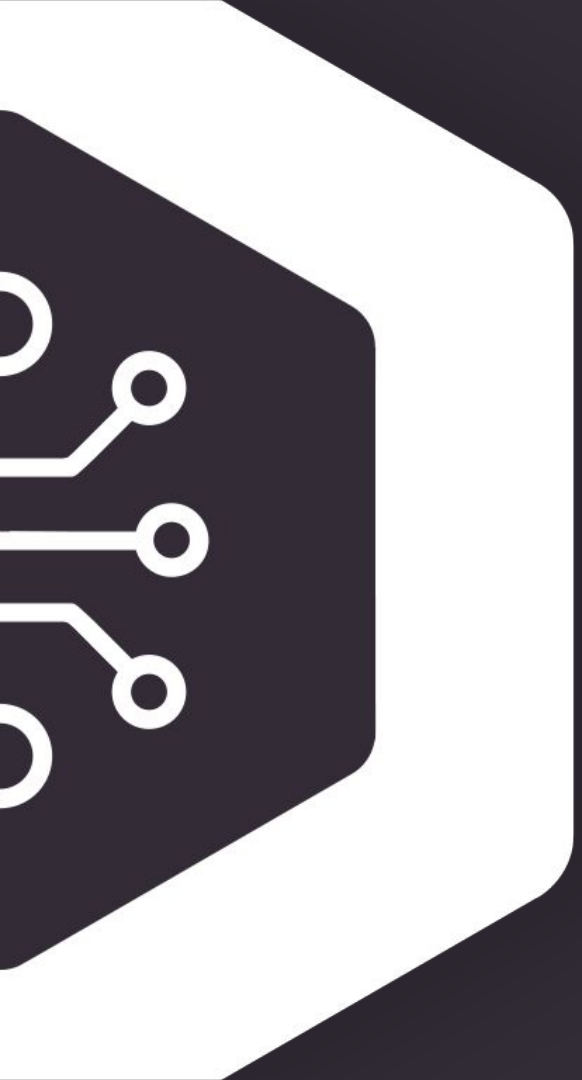


Running a Linux-Capable Open Source Soft SoC on the Avalanche Board with MicroSemi PolarFire FPGA

RISC-V Summit, Dec 3 2018

Karol Gugala, kgugala@antmicro.com
Al Kariminou, al.kariminou@futureelectronics.com

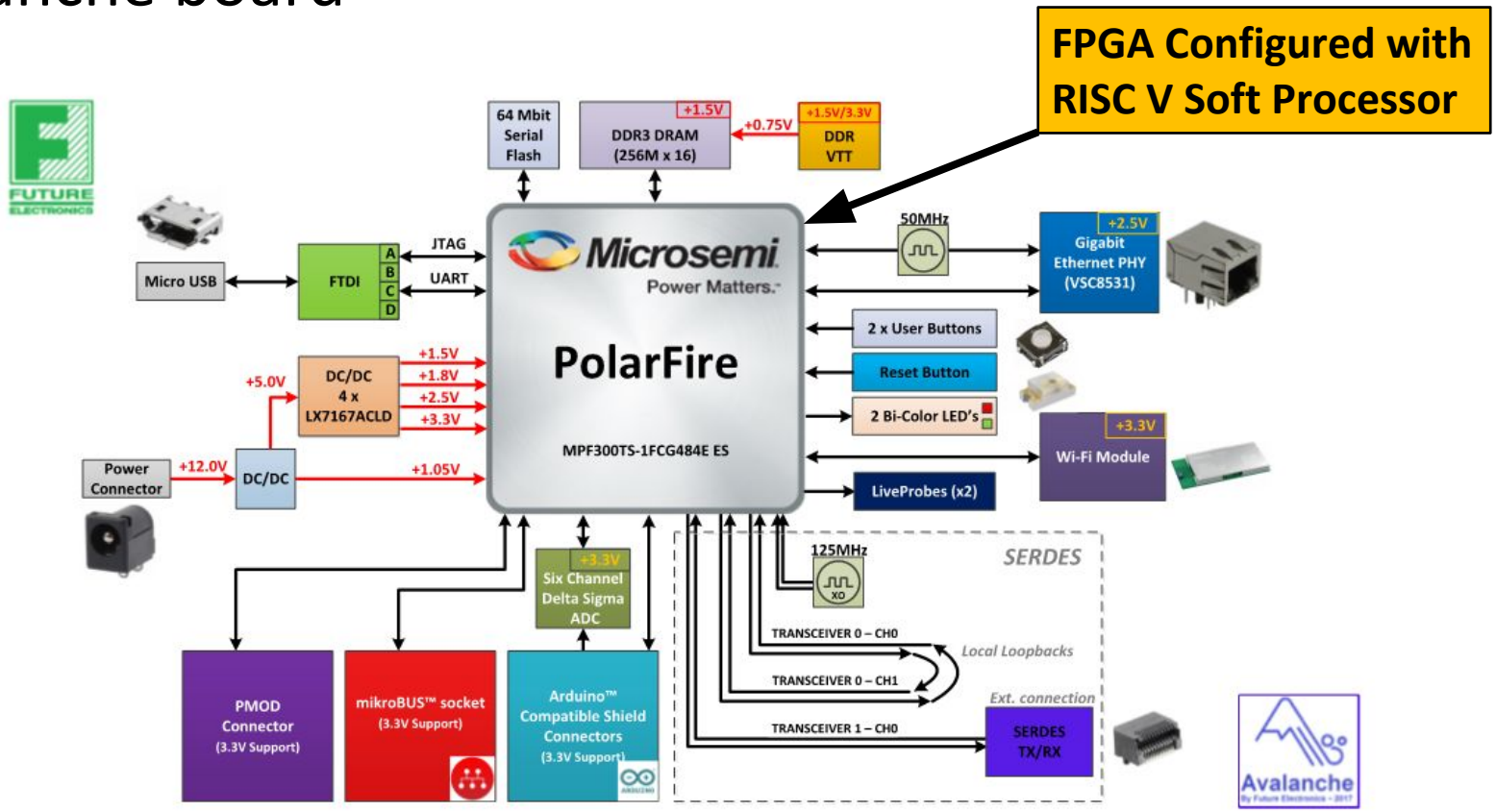
antmicro

antmicro

Today's tutorial

- ➔ platform: Future Electronics Avalanche board with Microsemi PolarFire FPGA
- ➔ using configurable, Python-based System on Chip builder based on MiGen - LiteX
- ➔ 32-bit RISC-V (rv32im) CPU + peripherals
- ➔ Linux!
- ➔ show-and-tell format (just too many of you!)
- ➔ **Thanks for Western Digital** for sponsoring parts of this effort as well as to **Future Electronics** and **Microsemi/Microchip** for helping to make this happen and providing licenses and boards.

Avalanche board



Avalanche board - Hardware

- 300K LUT PolarFire FPGA
- WiFi
  - Panasonic PAN9320
- 10/100/1000 Ethernet
  - Microsemi VSC8531 PHY
- High Speed SERDES Interface
  - SFP Cage
  - Testing of SERDES Interface
  - Potential add on cards
- DDR3 Memory
- Expansion I/F – PMOD, CLICK, Arduino

MikroElektronika Click boards

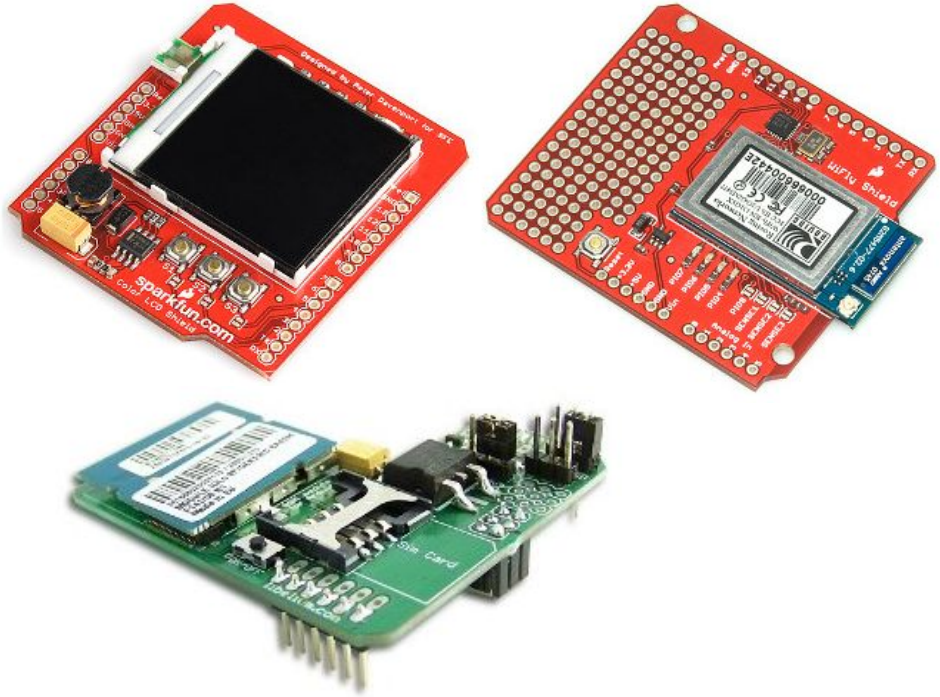
- 200+ Boards
- Interface Supported
  - I2C, SPI, UART, Analog, PWM, GPIO
- Category of Boards
  - Sensor
    - MEMs, Temp, Humidity, Touch, Hall, Magnetic, Optical, Proximity
  - Wireless
    - BLE, GPS/GNSS, GSM, NFC, RF Sub GHz, RF GHz
  - Interface
    - CAN, Ethernet, LIN, RS485, Fiber Optic, USB
  - Display
    - LCD, LED, OLED
  - Audio
    - FM Tuner, Audio AMP, MP3, Microphone
  - Mixed Signal
    - ADC/ DAC (12 – 22bit), Digital POT, AC Current, Voltmeter, Comparator
  - HMI
    - Joystick, Rotary, Fingerprint, Button

PMOD Boards

- 100+ Boards
- Interface Supported
  - I2C, SPI, UART, GPIO
- Category of Boards
  - Sensor
    - MEMs, Temp, Humidity, Touch, Hall, Magnetic, ALS
  - Wireless
    - BLE, WiFi, Bluetooth, 802.15, GPS, RF Sub GHz, RF GHz
  - Interface
    - CAN, Ethernet, RS485, USB, PS2, SD Card
  - Display
    - LCD(Char), LED, OLED
  - Audio
    - Audio AMP, Microphone
  - Mixed Signal
    - ADC/ DAC (24bit), Digital POT
  - HMI
    - Joystick, Rotary, Button

Arduino boards

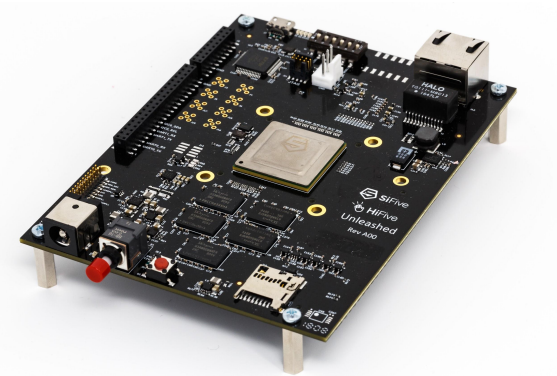
- 200+ Boards
- Interface Supported
  - I2C, SPI, UART, GPIO, Analog
- Category of Boards
- Too many to list
  - CPU Boards
  - Display Boards
  - Analog Boards
  - HIM Boards
  - Communication Boards
  - Sensor Boards
  - Etc….



antmicro

Why a 32-bit Linux-capable RISC-V SoC in FPGA?

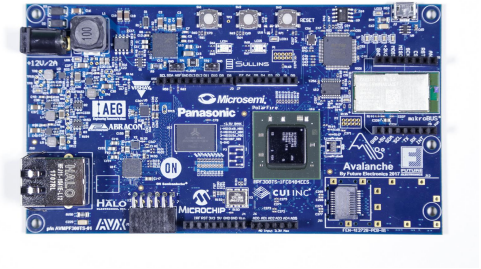
- ➔ there are the excellent HiFive RISC-V 64-bit Unleashed development and Expansion boards
- ➔ but these cost USD 1000 and 2000 respectively



antmicro

Why a 32-bit Linux-capable RISC-V SoC in FPGA? (cont.)

- ➔ the Avalanche board is $180 and thus available to a much broader audience (education, hobbyists)
- ➔ FPGAs are configurable, just like RISC-V - so you can experiment the core implementation itself
- ➔ you can also contribute to the peripheral IP ecosystem!
- ➔ we need 32-bit Linux for constrained systems
- ➔ because it can be done!



Why a 32-bit Linux-capable RISC-V SoC in FPGA? (cont.)

- ➔ for Linux on FPGA platforms, OpenRISC has been a popular choice - mainline since 3.X
- ➔ turns out it's actually a very productive endeavour to attempt the same for RISC-V
- ➔ shows where RISC-V Linux can be improved to be more portable and configurable
- ➔ the project showed and eliminated some of the missing pieces in the FPGA infrastructure needed

Why a 32-bit Linux-capable RISC-V SoC in FPGA? (cont.)

- ➔ for Linux on FPGA platforms, OpenRISC has been a popular choice - mainline since 3.X
- ➔ turns out it's actually a very productive endeavour to attempt the same for RISC-V
- ➔ shows where RISC-V Linux can be improved to be more portable and configurable
- ➔ the project showed and eliminated some of the missing pieces in the FPGA infrastructure needed

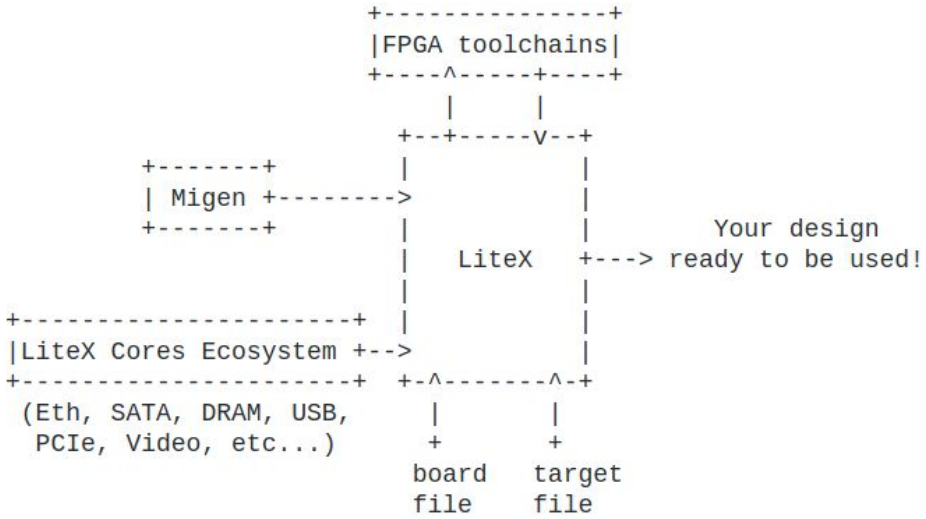
LiteX

- ➔ https://github.com/enjoy-digital/litex
- ➔ configurable SoC builder based on MiGen (Python based verilog generator) + MiSoC
- ➔ multiple CPU options possible
- ➔ allows Python programming of hardware (via MiGen)
- ➔ but still generates Verilog, so compatible with standard tooling

```
       __   _ __      _  __
      / /  (_) /____ | |/_/
     / /__/ / __/ -_)>  <
    /____/_/\__/\__/_/|_|
         Migen inside

  Build your hardware, easily!
Copyright 2012-2018 / EnjoyDigital
```

Typical LiteX design flow:



Why LiteX for this project?

- ➔ includes a RISC-V option - VexRiscv, which we use here (https://github.com/SpinalHDL/VexRiscv)
- ➔ earlier this year we ported the Zephyr RTOS to LiteX with VexRiscv, good experience
- ➔ VexRiscv is quite small but does have optional MMU - ideal target
- ➔ LiteX includes quite a few interfaces - Ethernet, USB, PCIex, which are a very good match for for Linux - lots of possibilities to develop further on top of this on the Avalanche board

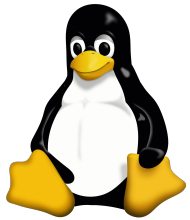
Toolchain

- ➔ RISC-V is supported now in mainline GCC now so you could try - but be aware there definitely are issues for 32-bit platforms we need to iron out, and we were already going quite experimental
- ➔ here we’re using the risc-v-tools toolchain that we knew to work on other platforms

antmicro

Linux support

- ➔ worked from mainline kernel
- ➔ due to platforms available currently, 32-bit support is not main focus
- ➔ there are dependencies which assume certain levels of functionality which you might not have or want on a smaller implementation
- ➔ secret aim: see how we can eliminate them!



RISC-V Linux dependencies (vs soft-cores)

- ➔ External DRAM
- ➔ Supervisor mode
- ➔ SMP / multicore
- ➔ BBL
- ➔ compressed instructions (“C” extension)
- ➔ Atomics (“A” extension)
- ➔ MMU

DRAM controller

- ➔ no open source choice - but Microsemi provides a “generated” controller, so adapted it to work with LiteX in collaboration with EnjoyDigital
- ➔ works like a charm!
- ➔ WIP, move to LiteDRAM in the future

Workflow - loading bitstream and binaries

- ➔ LiteX has a built-in rom-based bootloader (“BIOS”) - with tools to load binaries on UART, nice but slow (115200 vs megabytes of date needed for Linux)
- ➔ LiteX BIOS can support booting over ETH (but no such support for Avalanche yet)
- ➔ not suitable for development so we connected a debug access point, extended the CPU, connected openOCD - also did some fixes there
- ➔ 5m to 7s loading times - 40x improvement!
- ➔ no need to grab coffee in between runs

BBL, Memory Mappings

- ➔ Normally BBL maps the initial kernel space
- ➔ (as a side task, we took the Microsemi 64-bit U-Boot recently released on github, and added 32-bit support - really hacky version but it works on LiteX)
- ➔ the BBL did the mappings before the kernel run, so the kernel is run from a virtual address already
- ➔ BBL on RISC-V assumes multi-core
- ➔ someone had to make the mappings instead of the BBL - modified the MMU to support that out of the box in Supervisor mode

Supervisor mode

- ➔ needed for Linux
- ➔ Charles Papon did an initial implementation of Supervisor mode in VexRiscv
- ➔ CSRs, interrupt / exception handling, etc.
- ➔ delegates
- ➔ implement different interrupt source levels

antmicro

Compressed instructions

- ➔ not really mandatory but assumed to be on by default by many projects
- ➔ Precompiled toolchains with libc just assume “C” is there
- ➔ re-compiling libc without compressed works fine
- ➔ atomic instructions proved more of a challenge

Atomic instructions

- ➔ Linux for RISC-V was written as an SMP platform so kernel is enforcing the “A” extension
- ➔ but do we really absolutely need them? (no)
- ➔ implementing atomics in the CPU would take time and expand the implementation using valuable resources
- ➔ Linux does not need atomics by default on some of the other platforms

Further work on MMU

- ➔ the MMU itself was a challenge too; VexRiscv's MMU is much simpler than HiFive Unleashed and does not implement the MMU the "RISC-V" way (provides additional instructions instead).
- ➔ Added possibility to turn the MMU on or off in VexRiscv
- ➔ here Renode (simulator) was extremely useful - we run, compared executions, mappings, exceptions, faults
- ➔ working with busybox-based ramdisk

Userspace test apps / demos

- ➔ wrote simple static userspace apps (no libc or other deps) in assembly to test the fundamentals and be able to ‘fix’ bugs in the cpu/linux
- ➔ This allows us to do “unit tests” that we can use to test the CPU/linux port (at the same time) in practice.
- ➔ Going to do two demos

antmicro

Quick demo (booting linux on Avalanche)

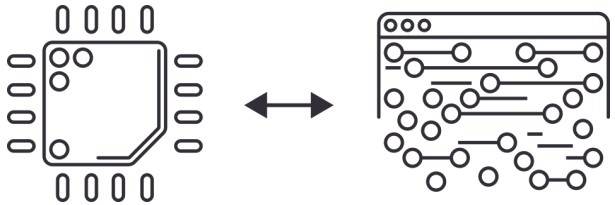
Renode to the rescue

- since the hardware was never tested with Linux, and we were developing both, we needed to divide problems into those with the Linux port and those with the HW itself
- could track down HW bugs by comparing execution results, register values, etc.
- this kind of HW/SW co-development is only possible because RISC-V has open source implementations!
- iterative work was much faster this way

antmicro

Co-development

- open hardware and open software provide a unique co-development possibility
- comparing execution in Renode and on hardware was really a life-saver



antmicro

Quick demo (booting linux on Litex/VexRiscv running on Renode)

antmicro

SUMMARY

antmicro

Summary

- ➔ as long as you use a a “proper” SMP HiFive-like RV64GC platform everything is fine, but when you go to a smaller platform, there is some work involved
- ➔ options: use HiFive Unleashed (definitely do!)
- ➔ and/or help us get this in shape so that we can be creative!
- ➔ RISC-V is awesome - it is extendible, configurable and open, so we could actually build this solution on both ends (we’re not extending the ISA but rather the tools to support the configurability of the ISA)

antmicro

Work done

- ➔ added initial LiteDRAM support on Avalanche board
- ➔ debug improvements (JTAG, OpenOCD)
- ➔ enhancements / fixes in VexRiscv MMU
- ➔ simplifications for single-core 32-bit platform in Linux
- ➔ SPI, I2C peripherals
- ➔ added drivers for shields (for the Hackaton tomorrow)

antmicro

TODO

- ➔ More work to clean this up
- ➔ release it all
- ➔ start tracking mainline with a minimal set of changes
- ➔ upstream VexRiscv and LiteX changes related to MMU and other
- ➔ upstream 32-bit related changes in RISC-V Linux kernel and toolchains
- ➔ describe the system in getting started guide!
- ➔ In other words lots of work

Potential of Avalanche for open source projects (for USD 180)

- ➔ 300K LUTs of FPGA space (that's a lot!)
- ➔ PolarFire is an interesting new architecture from Microsemi/Microchip
- ➔ 512 MB RAM
- ➔ Gigabit Ethernet
- ➔ 64 MB of Flash
- ➔ built-in wireless module
- ➔ plethora of modules, good for building PoCs

THANK YOU
FOR YOUR ATTENTION!