Untrustworthy Hardware

And How to Fix It

Seeking Hardware Transparency

PRESENT DAY.

Thanks to Contributors:

- ##FPGA, ##crypto and #openRISC on Freenode

- Shorne and Olofk from #openRISC (hardware and cross-compilation help)

- PropellerGuy (Parallax Propeller open-source IO interface)

Greetz:

- Maitimo, International Finance, DC408

PRESENT TIME

Layer:01 Software

- core modern open source algorithms for strong cryptography have been heavy scrutinized, tested and are readily available

- weak (DES, WEP, etc) and “black box” privacy tools are becoming a thing of the past

- free and open source software has made it easier to trust the privacy of computer systems

Let’s assume the software (hypothetically) is 100% secure…

Linux

open source

GnuPG

Where do we go from here?

Layer:02 Firmware

- firmware is almost exclusively closed source and controls almost all hardware devices and functions

- due to their low-level nature, malicious firmware persists across OS reinstallations

- "SPI flash is a really nice place if you can get there" (DEF CON 22: Summary of Attacks Against BIOS and Secure Boot)

Layer:03 Hardware

- hardware is almost always absolutely trusted by the rest of the system, as it is not widely considered an attack surface (especially in the consumer space)

Layer:03 Hardware

- hardware is almost always absolutely trusted by the rest of the system, as it is not widely considered an attack surface (especially in the consumer space)
  - NSA has been caught hardware backdooring Cisco systems (Glenn Greenwald, *No Place to Hide*), and DoD, Apple suspect adversarial nation states may be doing this as well

Layer:03 Hardware

- hardware is almost always absolutely trusted by the rest of the system, as it is not widely considered an attack surface (especially in the consumer space)

- NSA has been caught hardware backdooring Cisco systems (Glenn Greenwald, *No Place to Hide*), and DoD, Apple suspect adversarial nation states may be doing this as well

- “if the hardware is compromised, then the whole machine is compromised”

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon

Glenn Greenwald - *No Place to Hide*

hardware backdooring is real

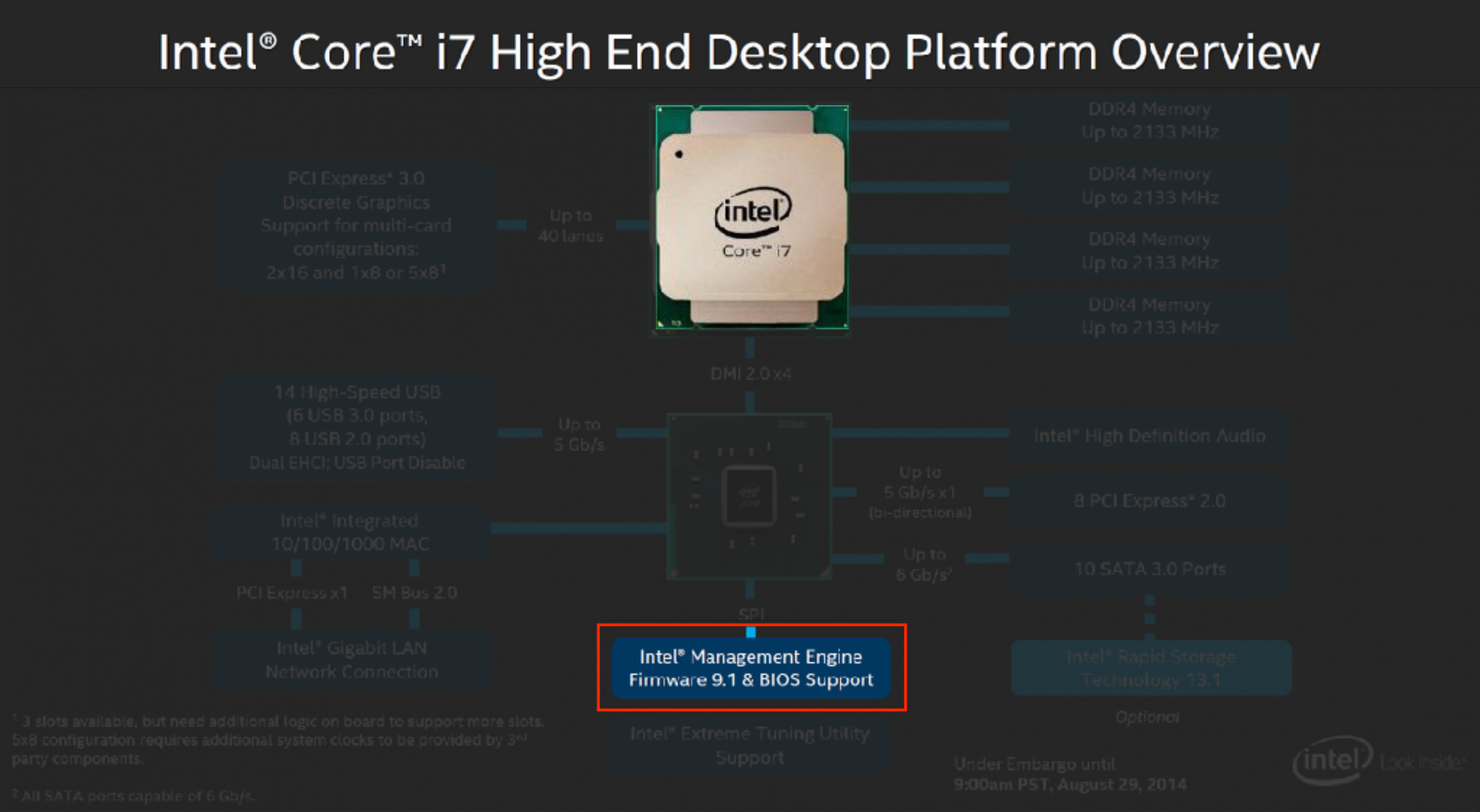
Layer:04 Intel Management Engine

- Management Engine runs on a dedicated logic device within the processor and runs proprietary firmware and OS

- Intel ME has full network device access with the ability to intercept network traffic without the CPU’s knowledge

- system access at the lowest level

- remains functional in the background even if the system is shut down but remains on standby power



Management Engine might sound like a feature reserved for enterprise or server applications, but it can be found everywhere

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

- Runs ThreadX real-time OS

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

- Runs ThreadX real-time OS (closed source, proprietary)

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

- Runs ThreadX real-time OS (closed source, proprietary)

- has its own MAC address and IP address for out-of-band features

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

- Runs ThreadX real-time OS (closed source, proprietary)

- has its own MAC address and IP address for out-of-band features

- some code hidden in an inaccessible on-chip ROM (decapping required to dump contents), other parts share space with the firmware ROM

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

- Runs ThreadX real-time OS (closed source, proprietary)

- has its own MAC address and IP address for out-of-band features

- some code hidden in an inaccessible on-chip ROM (decapping required to dump contents), other parts share space with the firmware ROM

- uses compression and encoding (LMZA, Huffman) to thwart reverse engineering

Intel ME Technical Overview

- most of our knowledge comes from Igor Skochinsky and a poorly secured company FTP server

- Runs ThreadX real-time OS (closed source, proprietary)

- has its own MAC address and IP address for out-of-band features

- some code hidden in an inaccessible on-chip ROM (decapping required to dump contents), other parts share space with the firmware ROM

- uses compression and encoding (LMZA, Huffman) to thwart reverse engineering

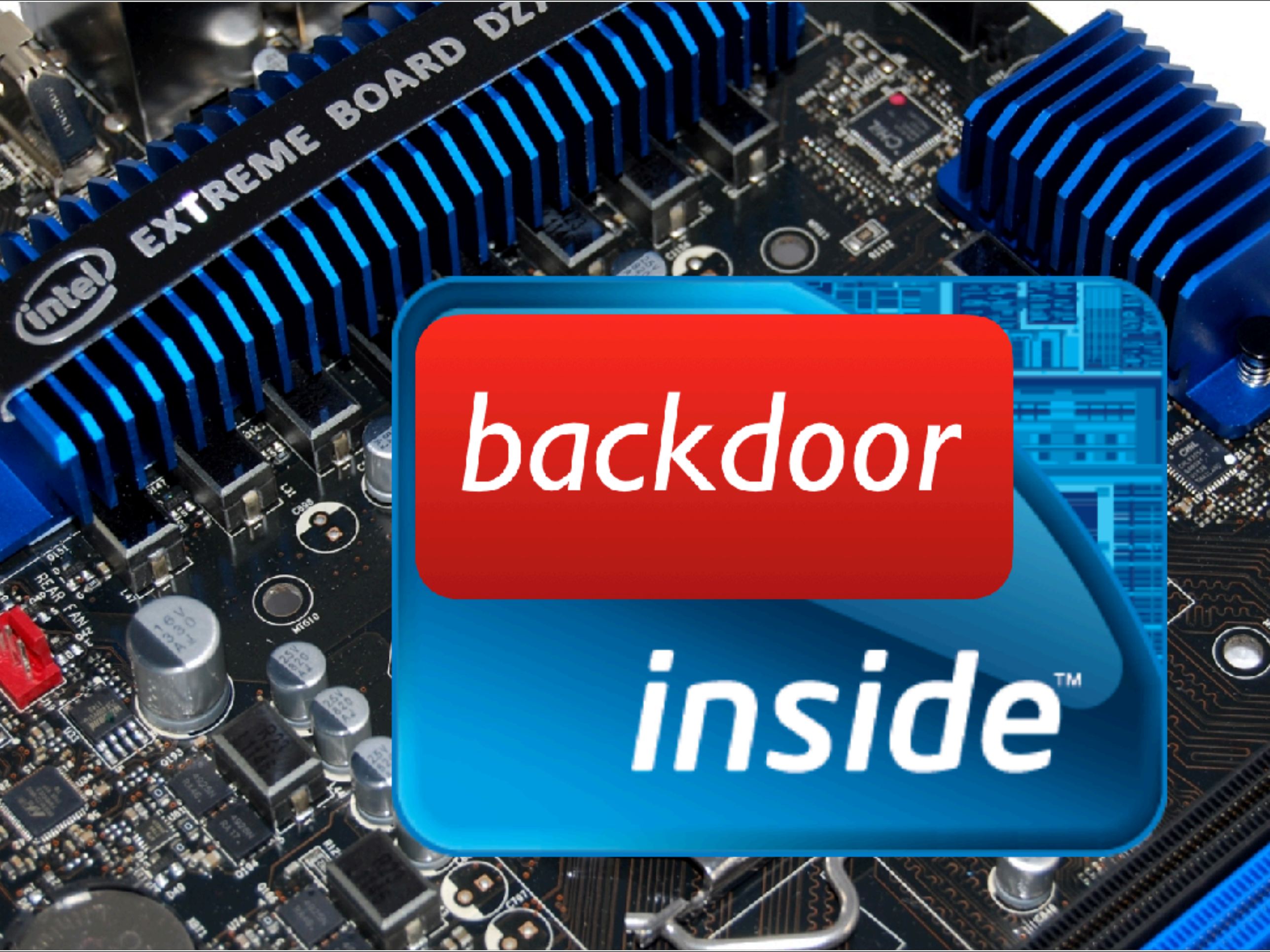
- multiple versions of IME exist using ARC, SPARC V8 and other instruction sets

Atmel Rad-hardened Sparc V8

ARC development platform

“In short, it’s a reverse-engineer’s worst nightmare.”

– hackaday.com


EXTREME BOARD
intel
backdoor
inside™

- effectively the perfect hardware backdoor, although ME is marketed as an IT out-of-band management tool

- present in all Intel systems since ~2008-2009, with no practical way to disable or audit

- handful of exploits exist for ME, with the number on the rise, requiring a firmware update from the manufacturer

Note: AMD also has a similar black-box platform, called TrustZone / PSP, but it has not been well documented / researched (they haven’t made new CPUs until recently)

Bonus Round: Speculation

If we’re discussing a worst case situation for hardware security, just how far can we go?

What About Nation States?

- Hardware backdooring has been documented as mentioned earlier is viewed as a viable threat by other state actors (DoD)

- nation states could backdoor product manufacturing with switched or additional components

- scarier yet, chips themselves (CPUs, chipsets, NICs, ROMs) could be backdoored at the fabrication center

(TS//SI//REL) FLUXBABBITT Hardware
Implant for PowerEdge 1950

(TS//SI//REL) FLUXBABBITT Hardware
Implant for PowerEdge 2950

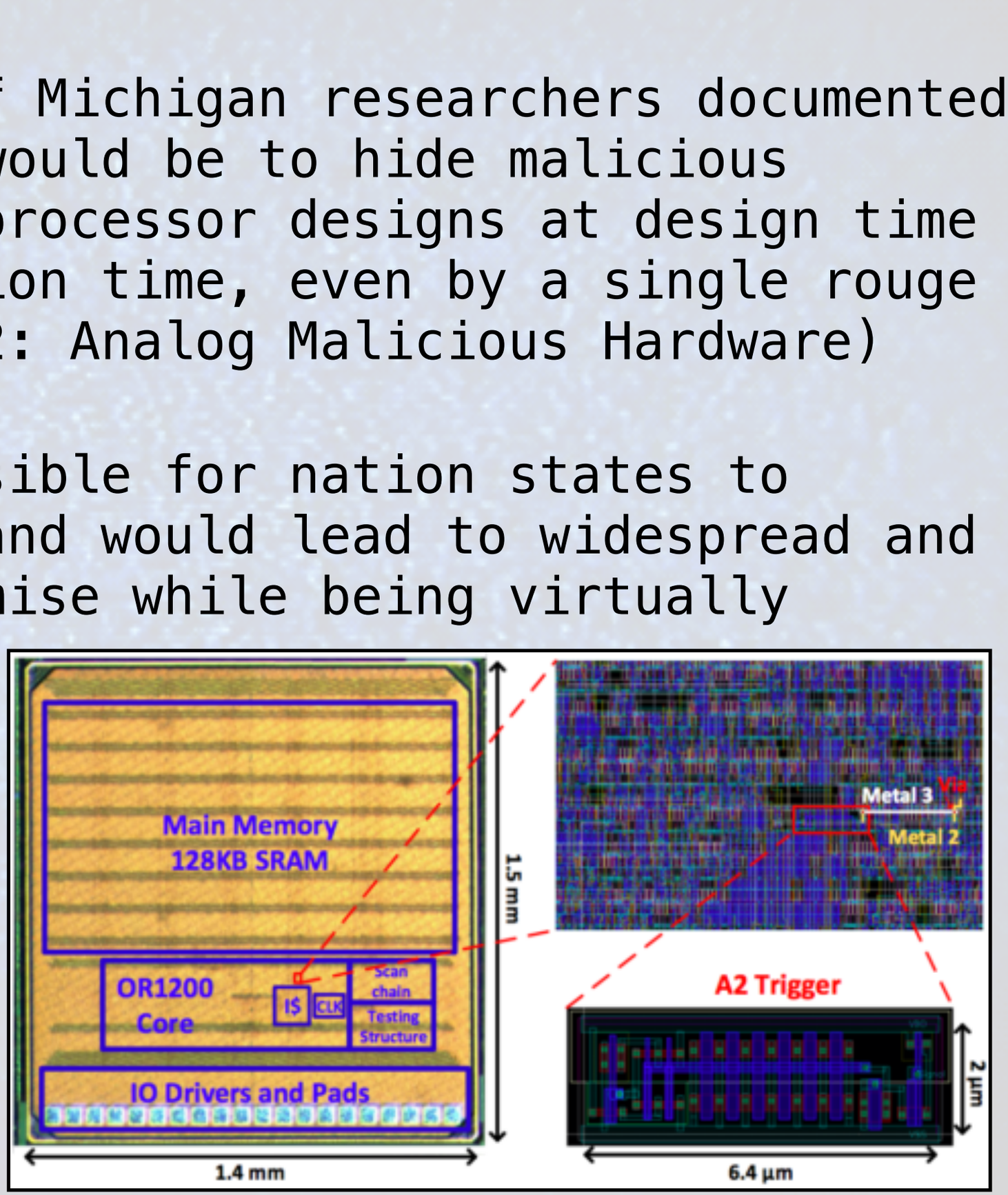
Der Spiegel / Jacob Appelbaum)

If they’re willing to go this far…

What’s to stop them from going further?

- University of Michigan researchers documented how easy it would be to hide malicious features in processor designs at design time and fabrication time, even by a single rouge employee! (A2: Analog Malicious Hardware)

- entirely possible for nation states to accomplish, and would lead to widespread and total compromise while being virtually undetectable



Credit: University of Michigan

Why can’t we do for hardware what we did for software?

- open source OS is a good start, open source firmware (Libreboot, etc) is better along with open source hardware, “no blob” system is ideal

Why can't we do for hardware what we did for software?

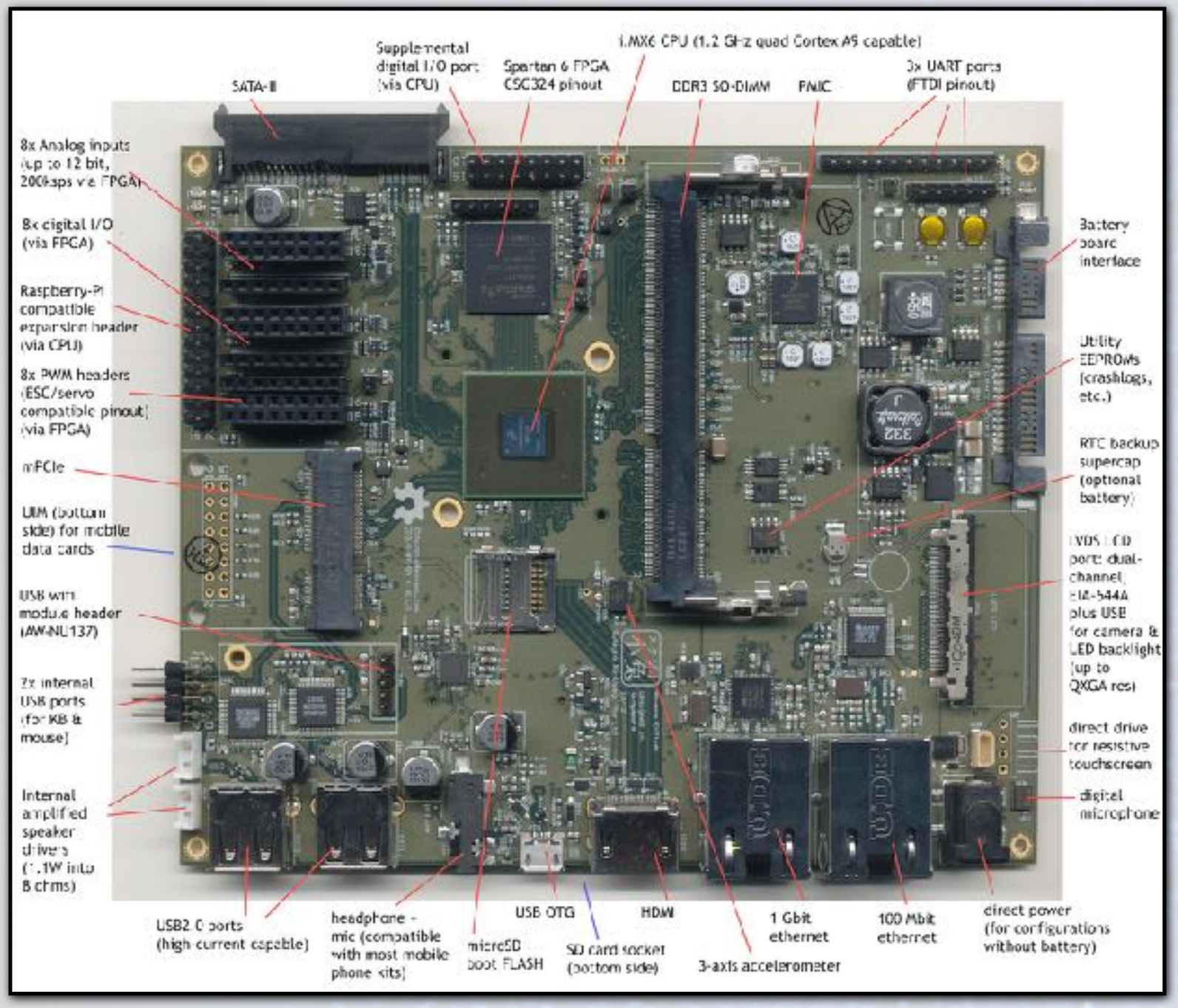
- open source OS is a good start, open source firmware (Libreboot, etc) is better along with open source hardware, "no blob" system is ideal
- some OSHW devices like Novena laptop are very close to this, but still require blobs for full functionality

Why can't we do for hardware what we did for software?

- open source OS is a good start, open source firmware (Libreboot, etc) is better along with open source hardware, "no blob" system is ideal

- some OSHW devices like Novena laptop are very close to this, but still require blobs for full functionality

- this also still leaves users trusting the chips



open source hardware

libreboot

What can be done for peace-of-mind private computation for critical situations and down-right paranoid users?

- Can we build a cost-effective low-level solution that offers maximum transparency?

- Can we build a cost-effective low-level solution that offers maximum transparency?

- on our platform, Linux and all programs run on the FPGA, so we know exactly what the CPU is doing

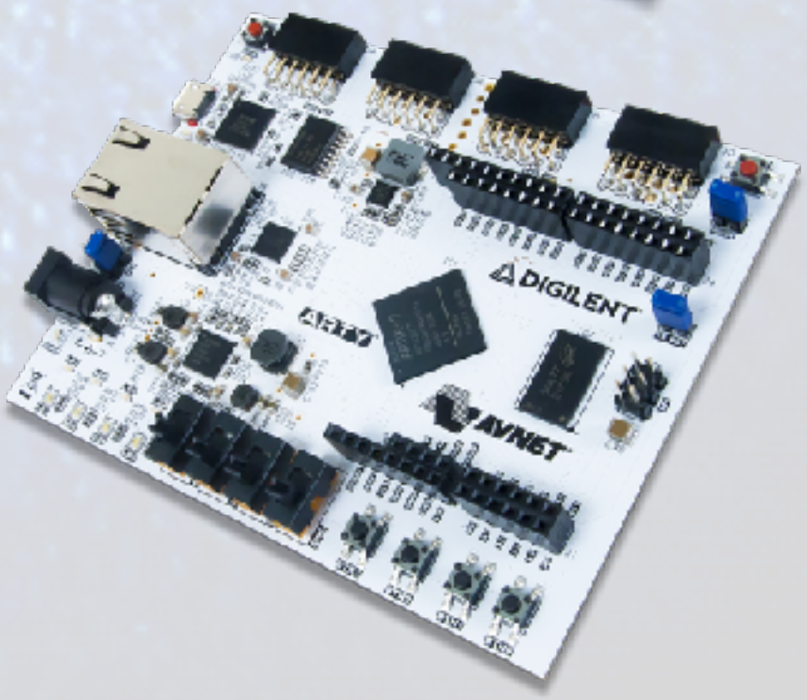
FPGA 101

- field programmable gate arrays are large blocks of configurable digital logic gates

- chips are designed in special languages called HDLs (hardware description languages)

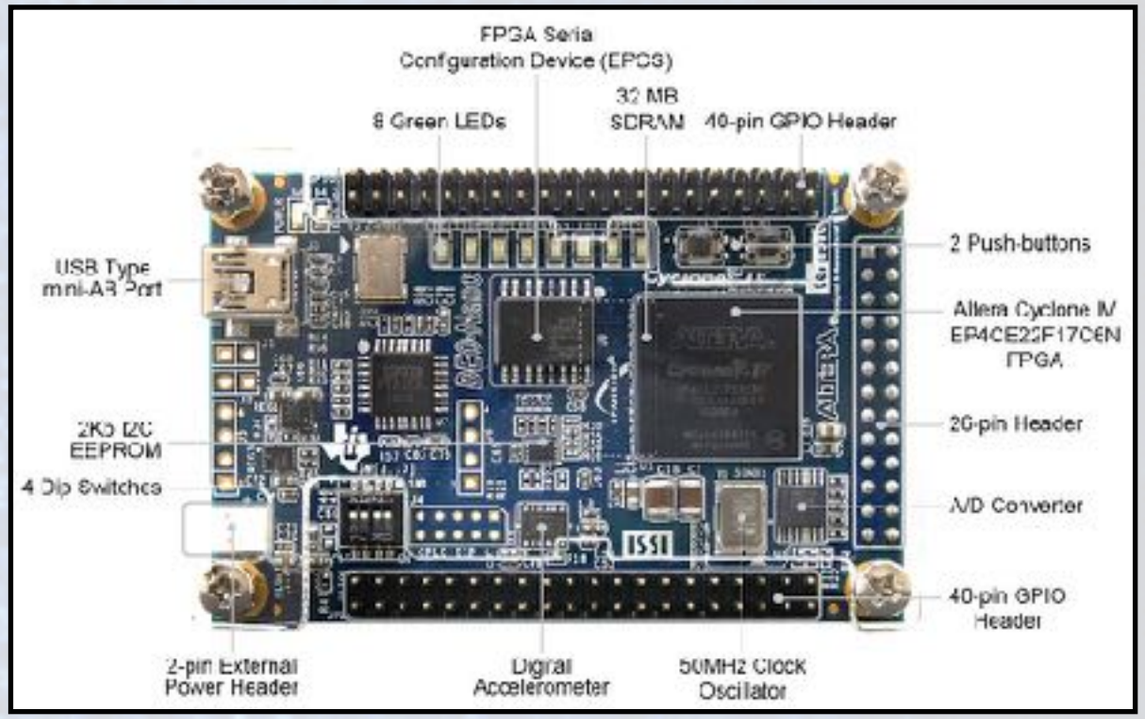
- bitstream generators read these files and program the gates within the FPGA to function as the HDL code dictates

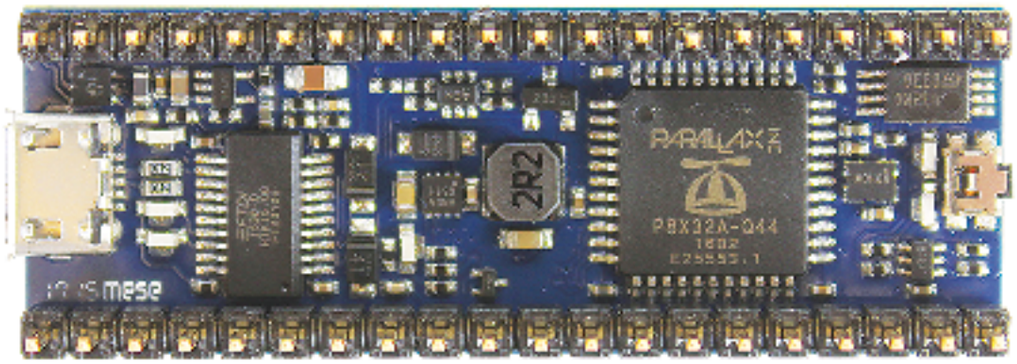
- most commonly used for chip design and testing, special hardware applications



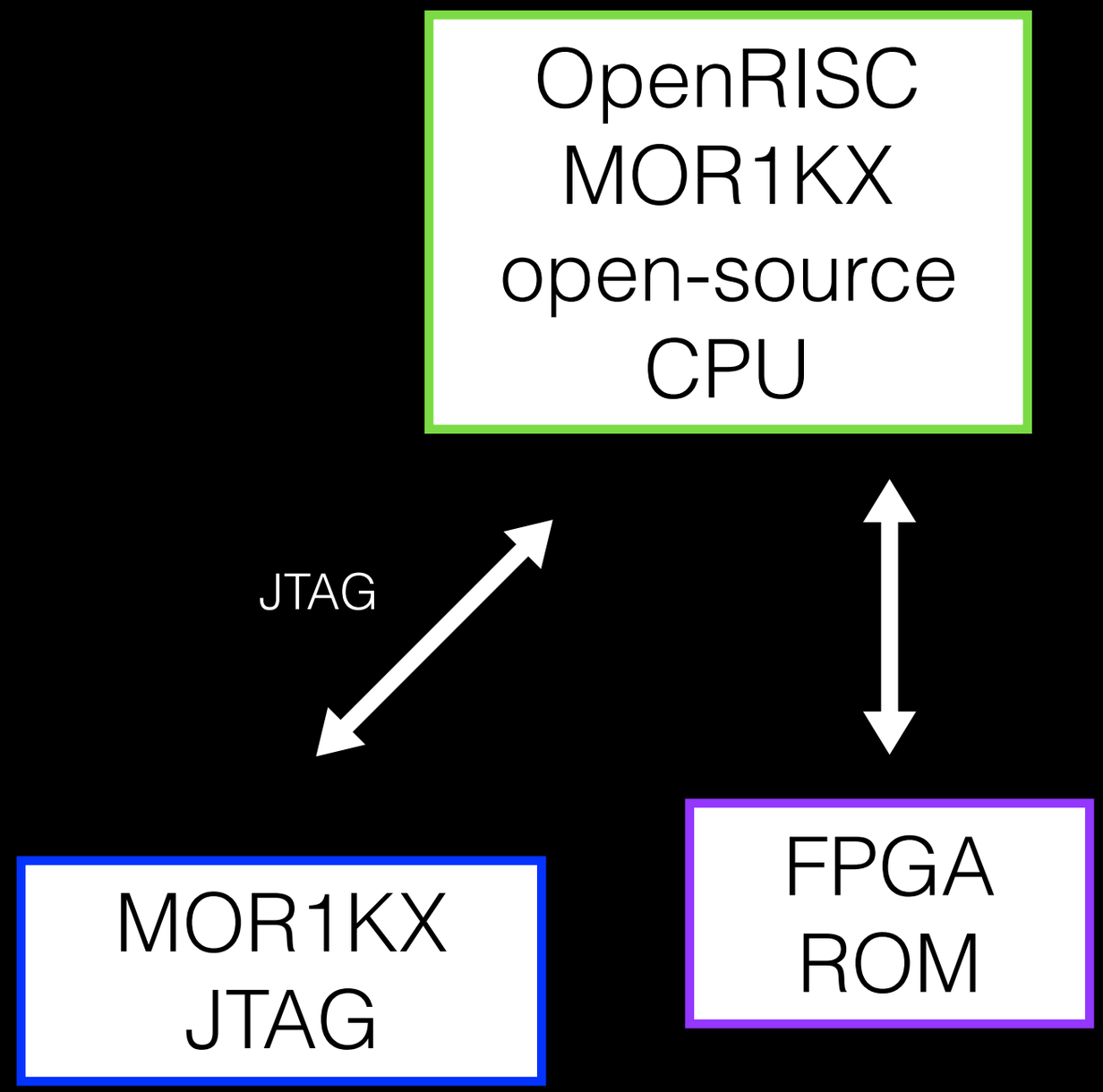
An Alternative

- Built around a cryptographic use case
- Runs GNU/Linux
- Fully open-source hardware and software down to the chip designs of both major components
- Parallax Propellor for IO, OpenRISC mor1kx 6-stage CPU running OS and tools

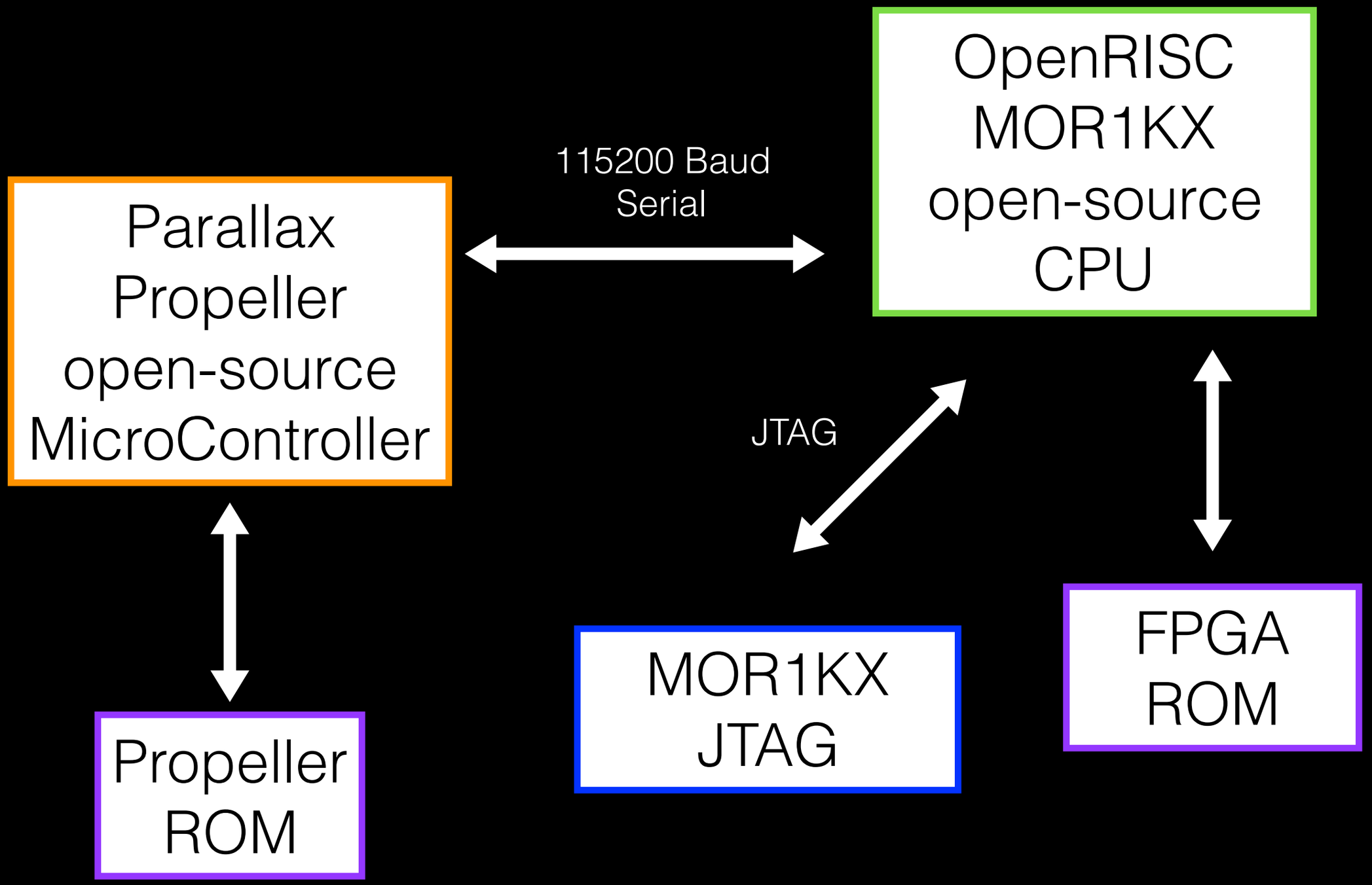




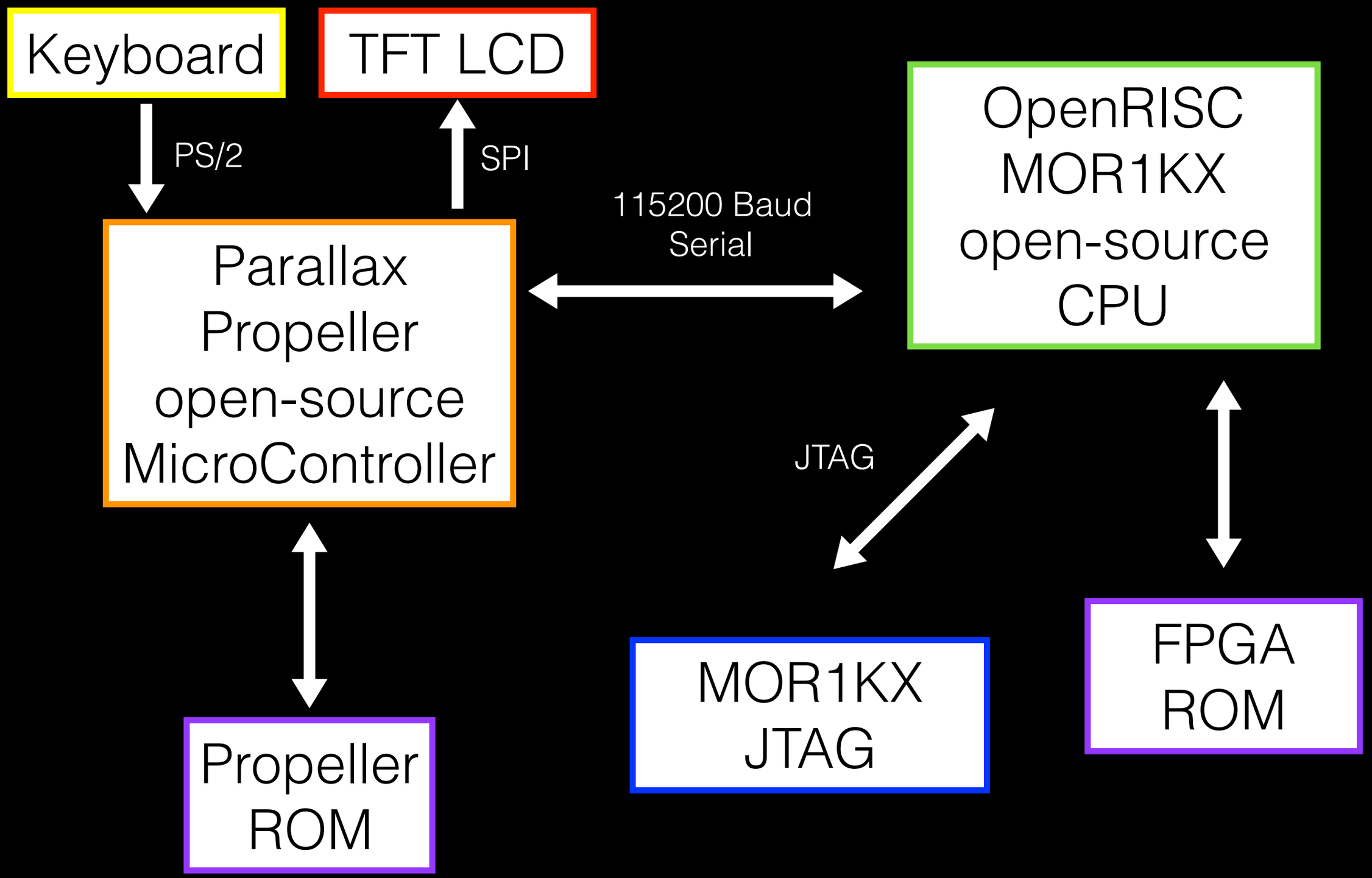
block diagram:



block diagram:



block diagram:



- Built linux image with MUSL, openADK tools; loading with OpenOCD and GDB

- Built linux image with MUSL, openADK tools; loading with OpenOCD and GDB

- Propeller programmed in SPIN and configured with Parallax tools

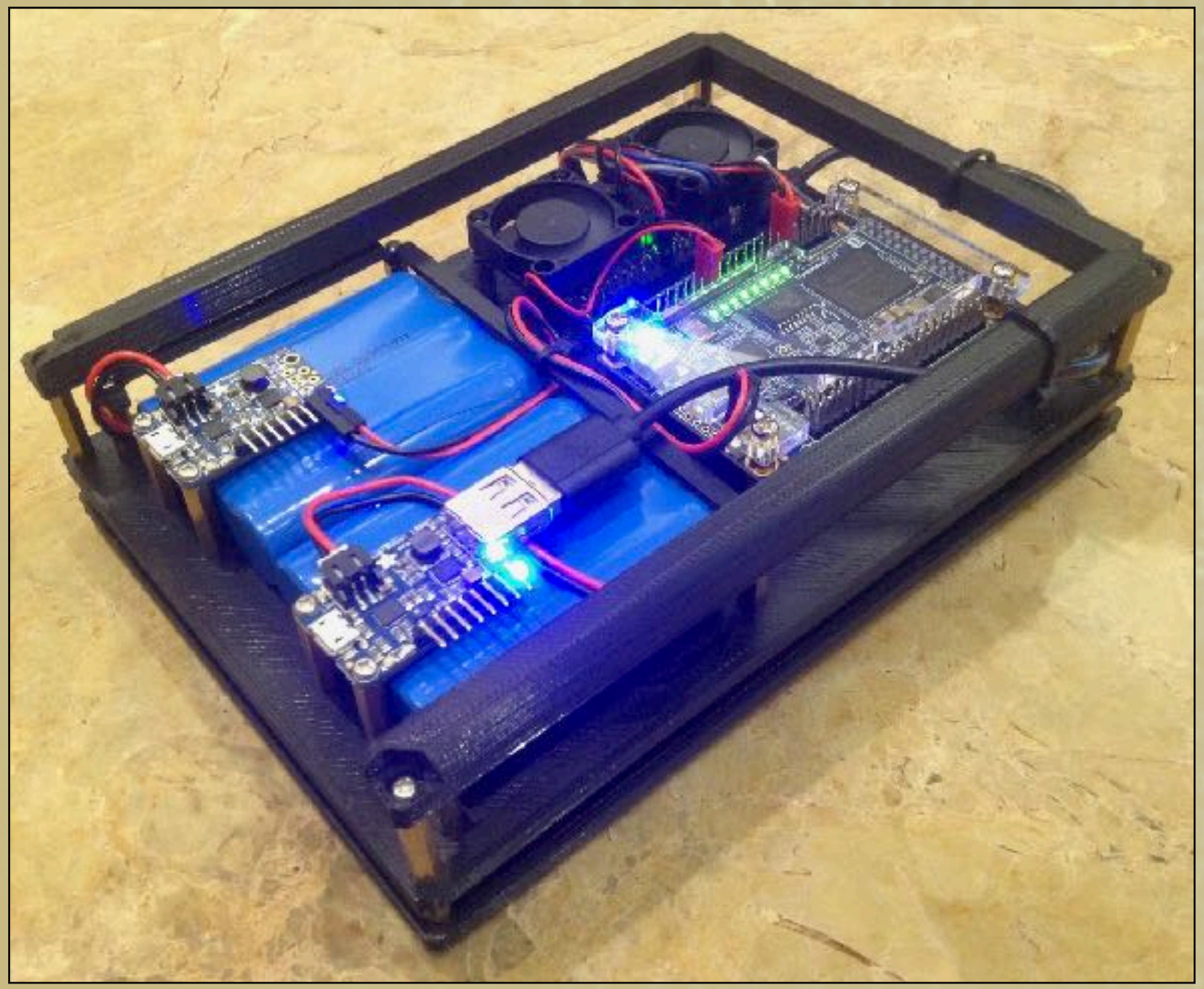
- Built Linux image with MUSL, openADK tools; loading with OpenOCD and GDB

- Propeller programmed in SPIN and configured with Parallax tools

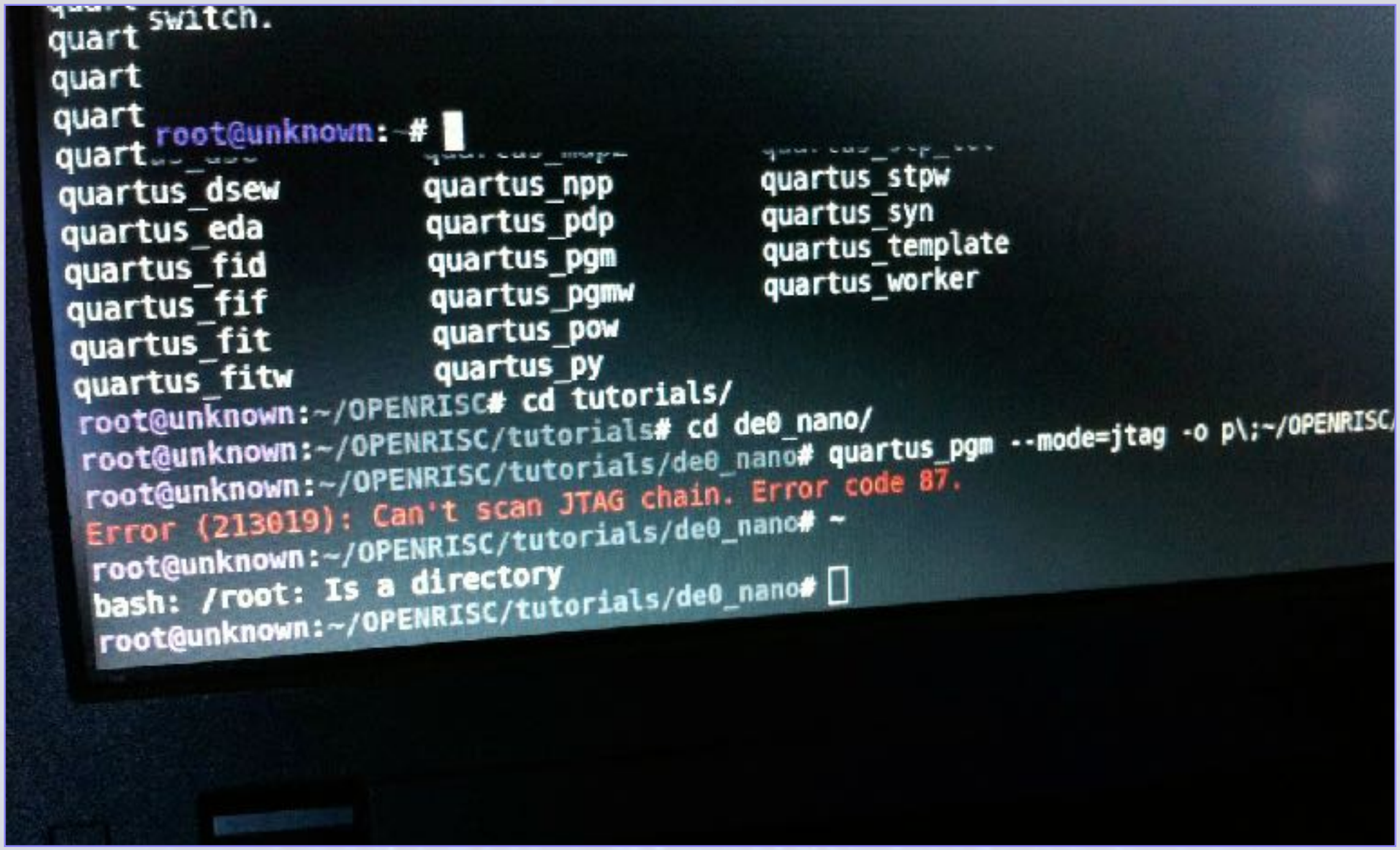
- FPGA programed with .sof openRISC image using Altera Quartus, FuseSoC

The Results
(it’s complicated)

The Final Product:



Works(ed) great!



until this

and this…

Device Family: Cyclone, Stratix
Type: Answers
Area: Component

Last Modified: September 11, 2012

Error: Can t access JTAG chain

Description

You may a receive this error message during auto-detect possibility due to a download cable problem, TDO is not connected properly, a noisy TCK signal, or TDI shorted to GND.

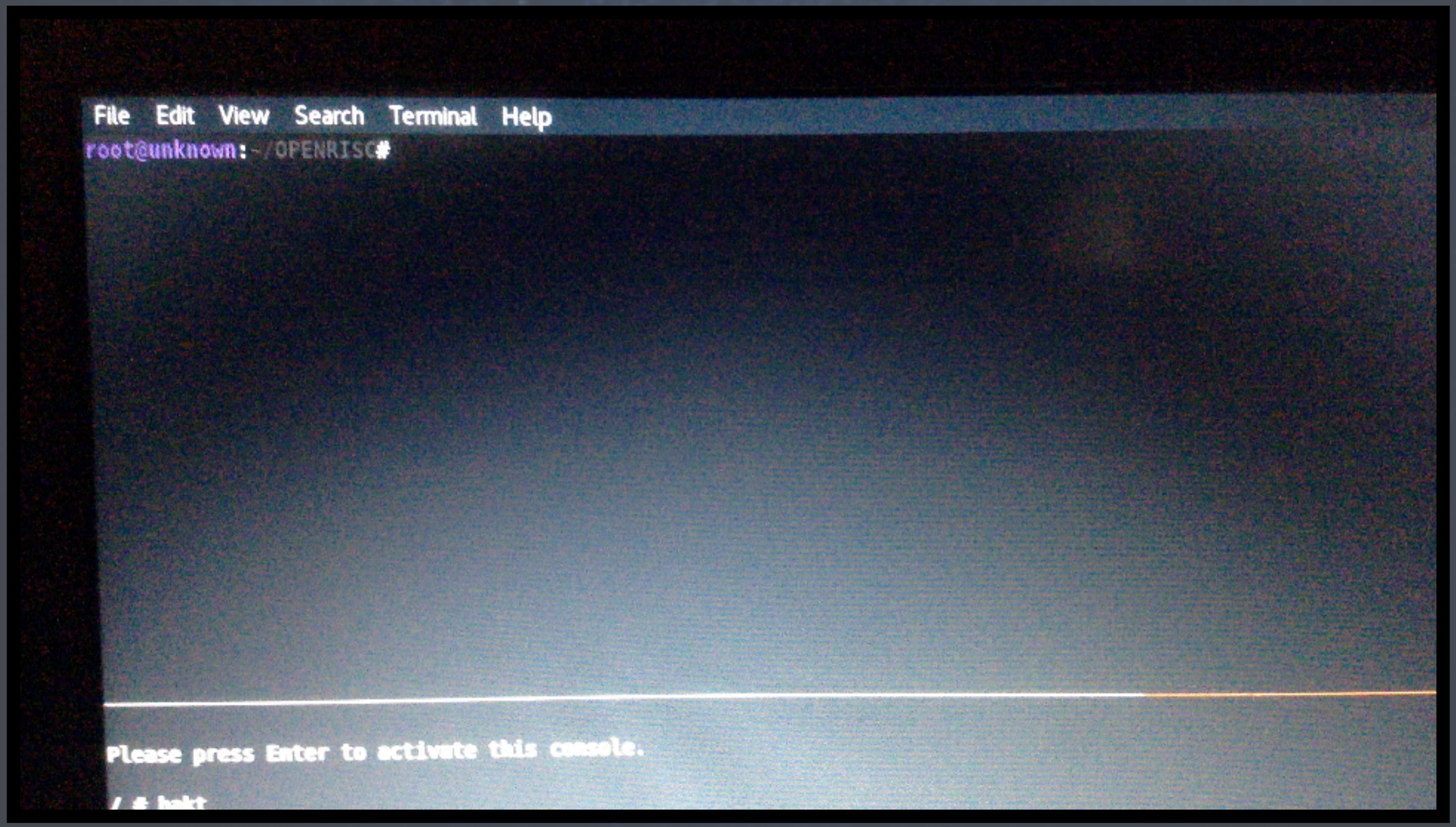
Here are some suggested steps for resolving the issue:

1. Make sure the pull-up and pull-down resistors connected to the JTAG pins are following the recommended values as stated in the respective device handbook.
2. Check the JTAG chain connectivity and the integrity of the TCK signal. Altera recommends pulling the TCK pin low through a pull-down resistor. Also make sure the TCK signal is clean to avoid any programming or JTAG configuration failure.
3. Use another USB-Blaster or other download cable. This is to ensure that the download cable is functioning.
4. Check the voltage seen on the pins of the JTAG header to see if they are at appropriate levels. The download cable is powered by the board, so ensure your board is properly powered.
5. Ensure all power supplies to the device are within recommended operating conditions. Refer to the respective device handbook for the correct voltage level.

3. Use another USB-Blaster or other download cable. This is to ensure that the download cable is functioning.

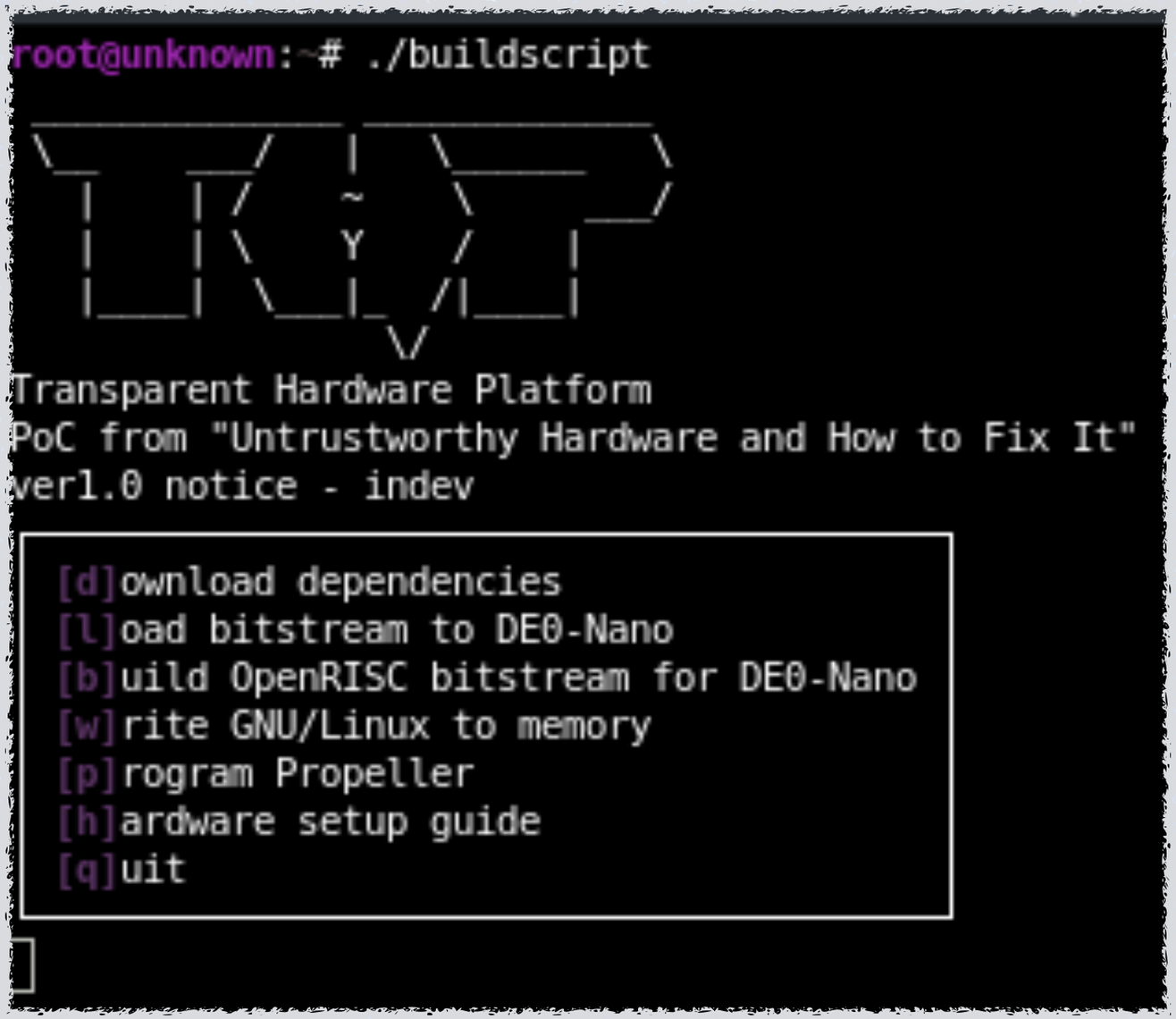
except the USB-Blaster is on the board…

The Backup:


File Edit View Search Terminal Help
root@unknown:~/OPENRISC#
Please press Enter to activate this console.
/ # halt

FPGAs Suck

but recreating this project is easy



all code, 3d models and guides will be available shortly
at https://github.com/UntrustworthyHardware-DC25/THP

One more thing(s)

- in a recent AMA on Reddit, AMD has publicly stated that they are “strongly considering” making the source code for their IME-equivalent, PSP / Trustzone **OPEN SOURCE**

- This is an ongoing project! I will continue to improve the system I have built here with plans for RF side channel hardening and increased system independence.

Further Reading and Additional Resources

- DEF CON 22: Summary of Attacks Against BIOS and Secure Boot: https://www.youtube.com/watch?v=QDSlWa9xQuA

- DEF CON XX: Hardware Backdooring is Practical: https://www.youtube.com/watch?v=8Mb4AiZ51Yk

- NSA shipment hijacking: http://www.theverge.com/2013/12/29/5253226/nsa-cia-fbi-laptop-usb-plant-spy

- Windows “golden keys” leaked: https://arstechnica.com/security/2016/08/microsoft-secure-boot-firmware-snafu-leaks-golden-key/

- NSA Cisco implant: https://arstechnica.com/tech-policy/2014/05/photos-of-an-nsa-upgrade-factory-show-cisco-router-getting-implant/

- Apple suspects hardware backdoors by state actors in server shipments: https://www.extremetech.com/extreme/225524-apple-may-design-its-own-servers-to-avoid-government-snooping

- NSA deploys low level / hardware backdoors against intercepted consumer devices: http://www.extremetech.com/computing/173721-the-nsa-regularly-intercepts-laptop-shipments-to-implant-malware-report-says

- Summary of Intel ME: https://boingboing.net/2016/06/15/intel-x86-processors-ship-with.html

- Detailed IME breakdown by Libreboot team: https://libreboot.org/faq/#intel

- REcon 2014: Intel Management Engine Secrets (Igor Skochinsky): https://www.youtube.com/watch?v=4kCICUPc9_8

- Hackaday: The Trouble with Intel’s Management Engine: http://hackaday.com/2016/01/22/the-trouble-with-intels-management-engine/

- Hackaday IME workarounds: https://hackaday.com/2016/11/28/neutralizing-intels-management-engine/

- A2: Malicious Analog Hardware: https://www.ieee-security.org/TC/SP2016/papers/0824a018.pdf

- Wired Summary of silicon backdooring: https://www.wired.com/2016/06/demonically-clever-backdoor-hides-inside-computer-chip/

- Power-based side channel attacks: https://www.rsaconference.com/writable/presentations/file_upload/br-w03-watt-me-worry-analyzing-ac-power-to-find-malware.pdf

- openRISC homepage: http://openrisc.io/

- getting started with openRISC: https://github.com/openrisc/tutorials

- open source Propeller terminal using Adafruit 2.8” TFT: https://github.com/tdoylend/tgi2