What's on the Wire?

Physical Layer Tapping with Daisho

Dominic Spill
Mike Kershaw / Dragorn
Michael Ossmann

Black Hat USA 2013


HELLO
my name is
inigo montoya
you killed my father
prepare to die

Who we are

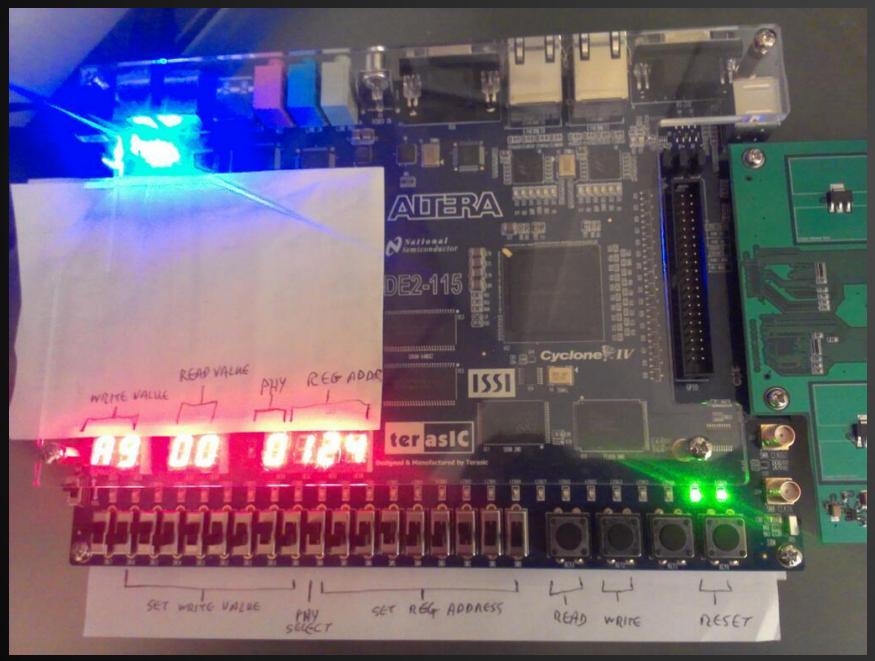
Michael Ossmann

Primary on Daisho CFT

Creator of multiple OSHW projects, Ubertooth, HackRF, YARDstick One

Founder of Great Scott Gadgets

Who we are



Dominic Spill

Dev on Ubertooth, BTBB and gr-bluetooth

Host code on Daisho

Other projects include BeagleDancer, PS/2 tap and fcc.io

Who we are

Mike Kershaw

Creator of Kismet

Front-end board designer for Daisho

Random other OSS/OSHW projects like Kisbee

Outline

Background

What?

Why?

How?

Progress

Disclaimer

The views expressed are the views of the authors and do not reflect the official policy or position of the Department of Defense or the United States Government.

Work In Progress

Built first devices in May

Hope to have them working by Autumn

Demos will use development platform; Final revisions will use Daisho mainboard

Team

Jared Boone
Marshall Hecht
Mike Kershaw
Michael Ossmann
Dominic Spill
Benjamin Vernoux

Others in #daisho on freenode

Outline

Background

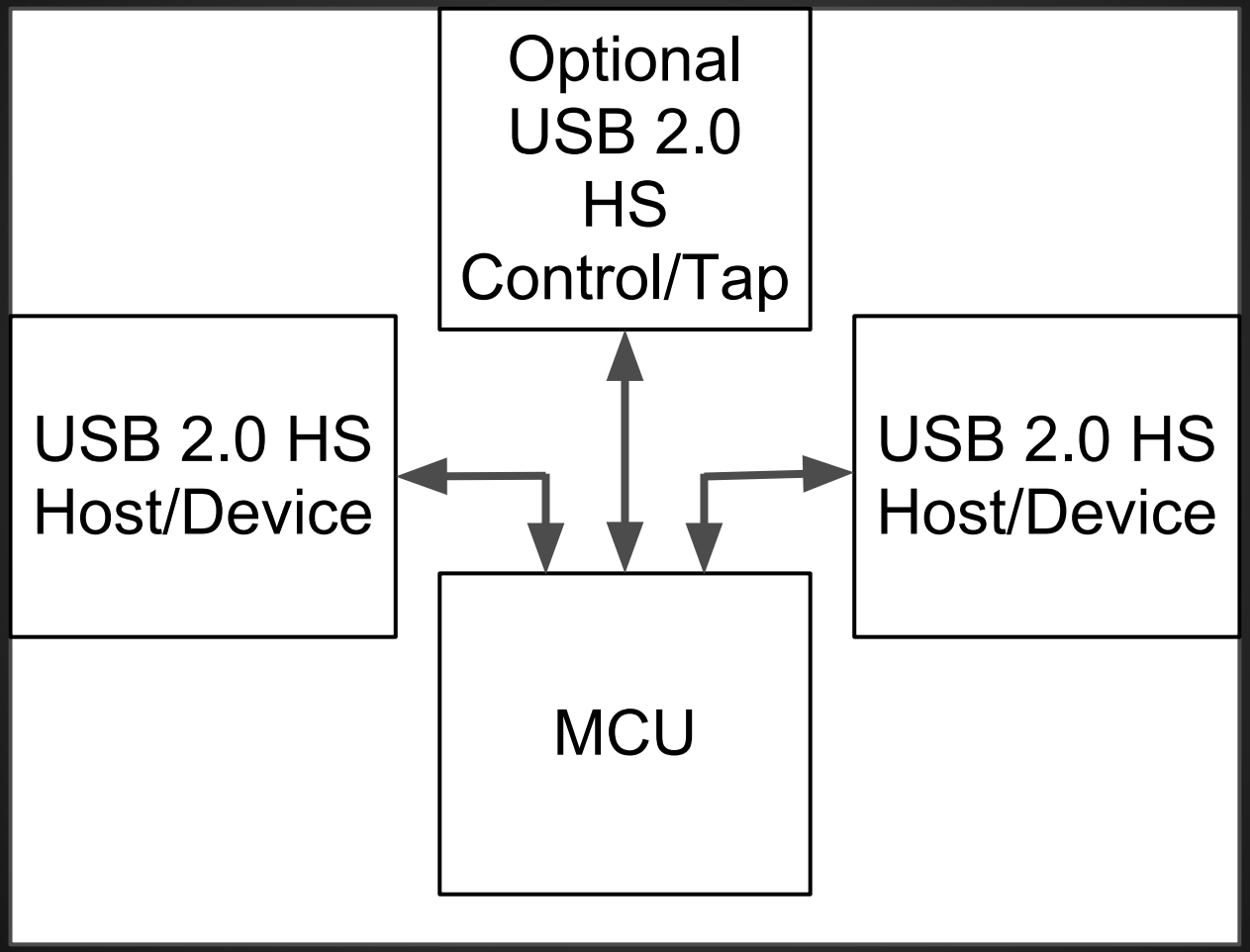
What?

Why?

How?

Progress

USB Multi-tool



I wanted to build something like this.

USB Man-in-the-Middle

PC
Phone
Tablet

Storage
HID
Ethernet
802.11
Bluetooth

Monitoring, injection, modification . . .

USB Device-to-Device

Storage
HID
Ethernet
802.11
Bluetooth

Storage
HID
Ethernet
802.11
Bluetooth

It would be cool to support unusual topologies.

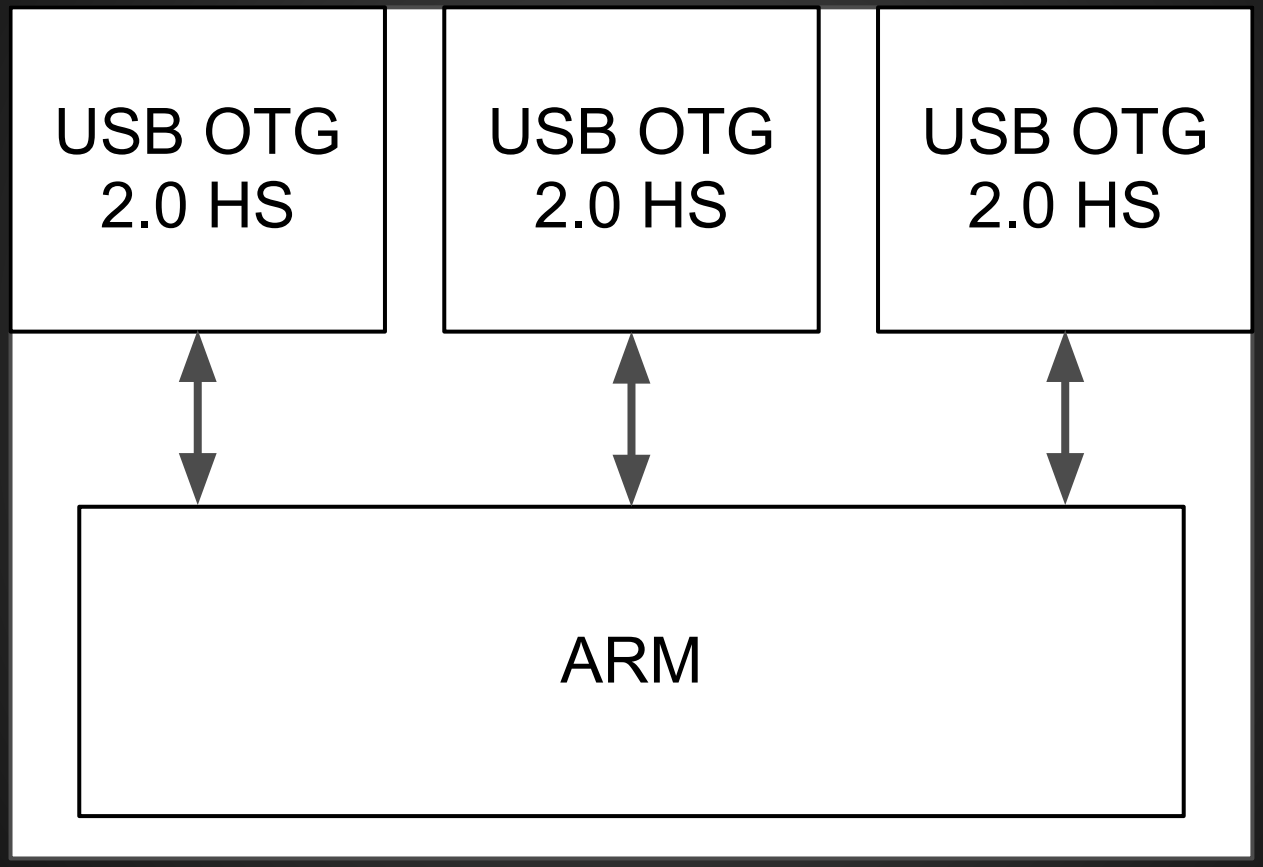
USB Host-to-Host

PC
Phone
Tablet

PC
Phone
Tablet

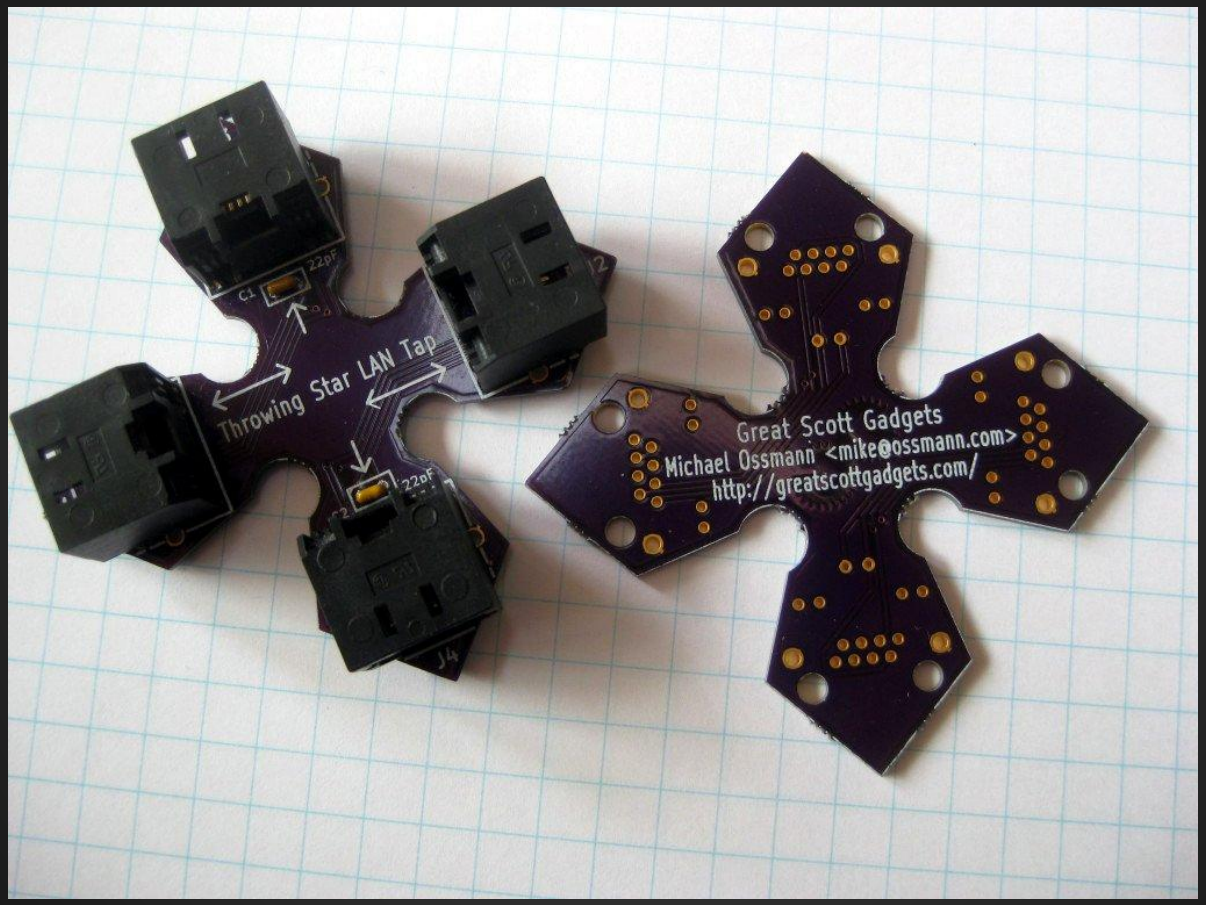
Stack fuzzing, file transfer, weird networks. . .

Microcontroller?



This didn't seem like it was going to happen.

Gigabit Ethernet Tap?



Only supports 10/100, not Gigabit Ethernet.

Using an Ethernet Switch IC

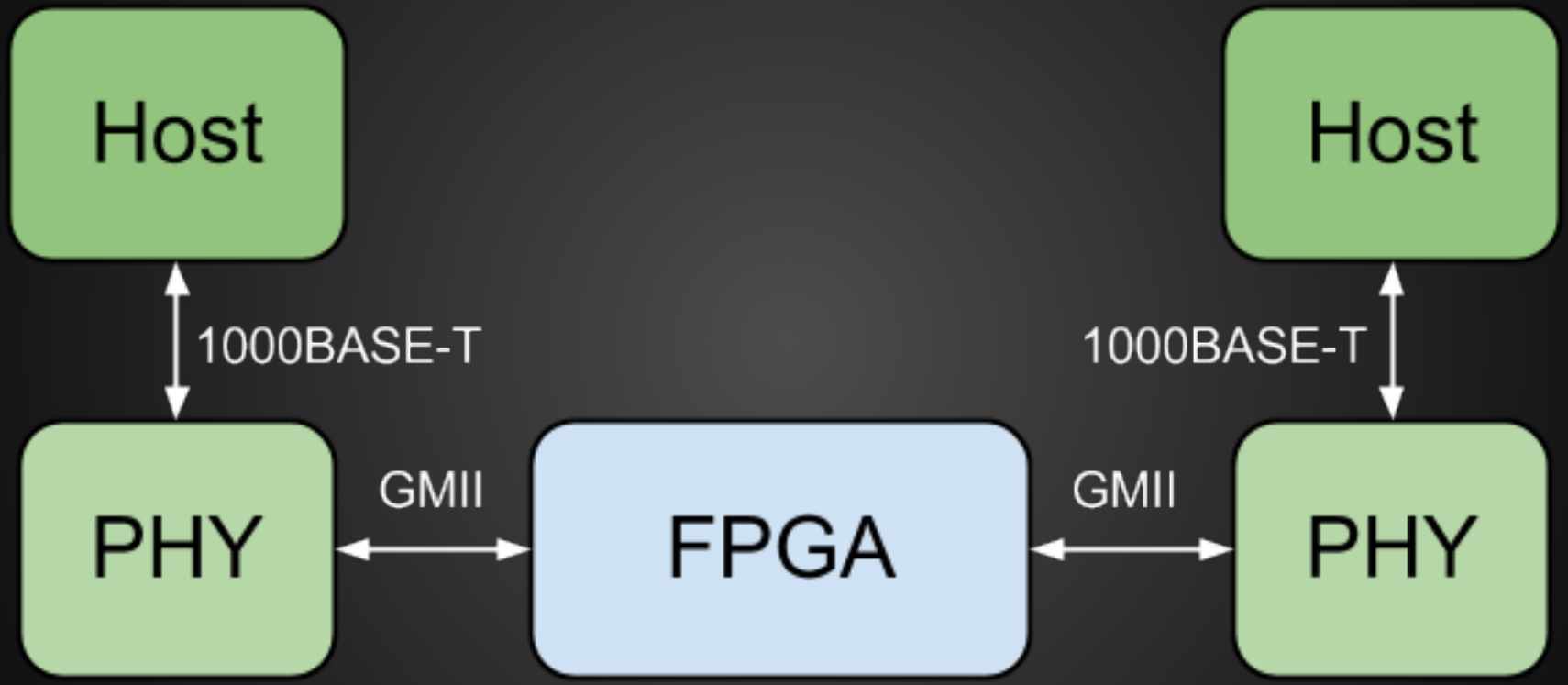
Several switch ICs are supported by Linux, popularly used in OpenWRT platforms.

None are open source hardware friendly.

Could build a specialized Ethernet switch platform that has a mirror port.

Products like this already exist.

Any Other Way?



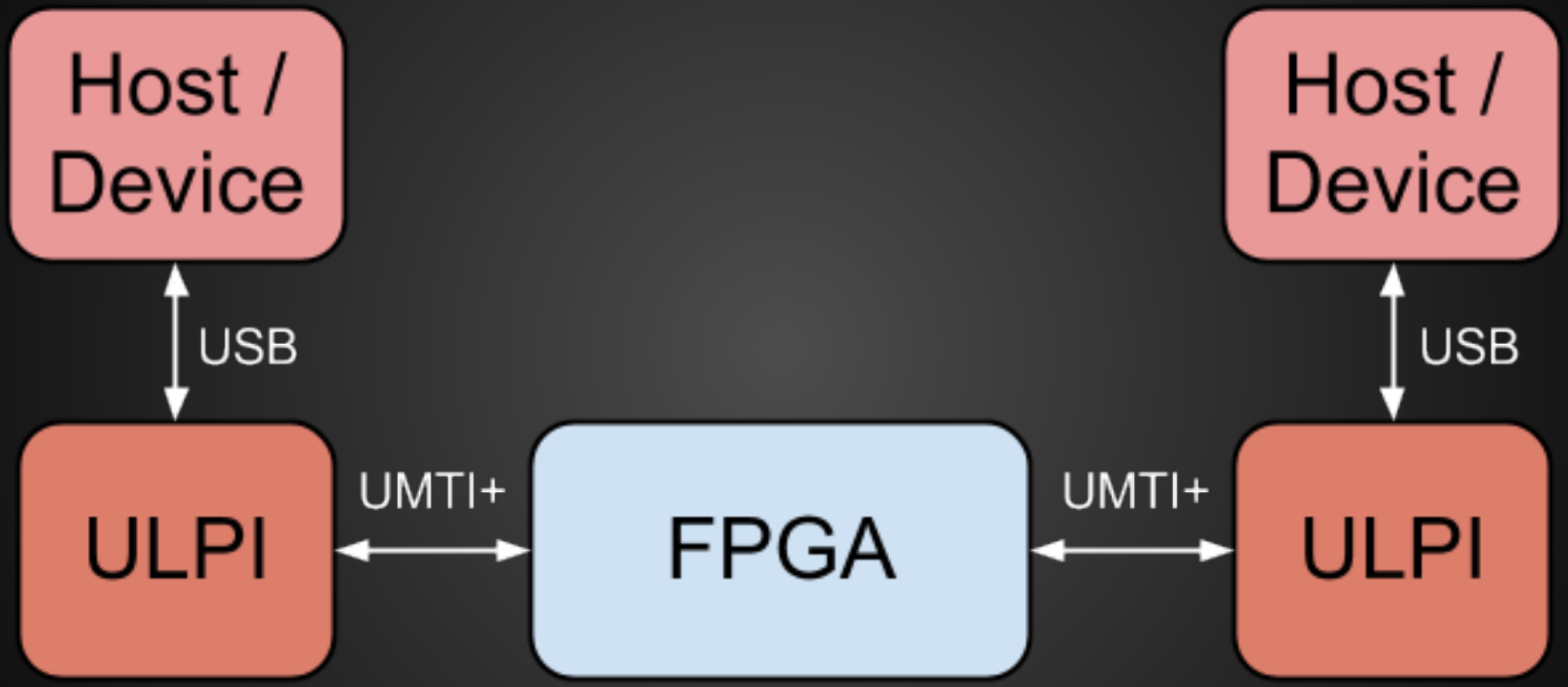
Connect PHY ICs together through an FPGA.

Flexibility

An FPGA is more expensive than a Gigabit Ethernet switch IC, but it's worth it for the extreme flexibility.

It's the best choice for security research and development.

Hey!



We can do that with USB too!

Make it Modular

Let's support multiple front-end modules, each with PHYs and connectors for a particular target medium.

Daisho is born.

Daisho **n.** A matched pair of swords used by the Samurai class in feudal Japan

Image adapted from http://www.metmuseum.org/

Outline

Background

What?

Why?

How?

Progress

Daisho - A Physical Monitor

Physical layer monitor

Extensible - modular design for new hardware

Open source - hardware and software

Affordable - compared to existing offerings

Portable - bus powered for some applications

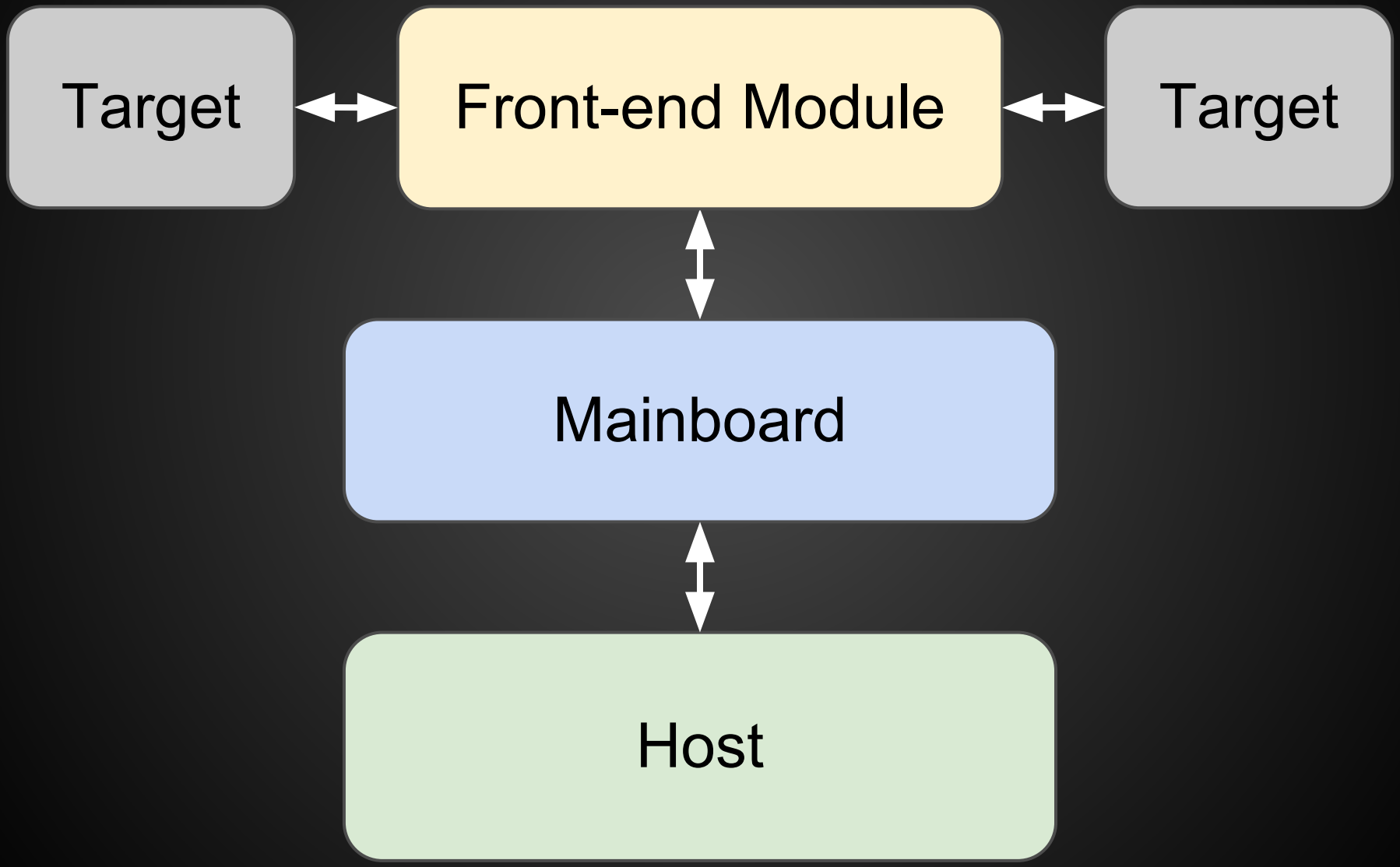
Daisho - Extensible

Current targets:

- 1000BASE-T
- HDMI
- USB 3.0
- RS-232

Easy to add future targets

Daisho - Extensible



Outline

Background

What?

Why?

How?

Progress

Wright's Law

"Security will not get better until tools for practical exploration of the attack surface are made available."

- Joshua Wright

Wright's Law

An example

WEP -> WPA -> WPA2

A counter example

Bluetooth PIN -> Secure Simple Pairing

If no tools exist...

... Then you can't really look for problems, can you?

When tools *do* exist, do they let you get low-level enough?

If tools are unaffordable, they might as well not exist

Daisho - Open Source

Software
Firmware
Hardware
Tools (where possible)

https://github.com/mossmann/daisho

Existing Solutions

Usually ***really*** expensive (USB analyzers, etc)

Not OSS/OSHW, limited in expandability

Don't play nice together - need a new tool for each target

Usually not designed to be portable

May have technical limitations

Enabling New Research

Fully Arbitrary 802.3 Packet Injection: Maximizing the Ethernet Attack Surface, Andrea Barisani and Daniele Bianco, Black Hat USA 2013

This is an excellent example of the kind of research we hope to enable.

Watch it.

Outline

Background

What?

Why?

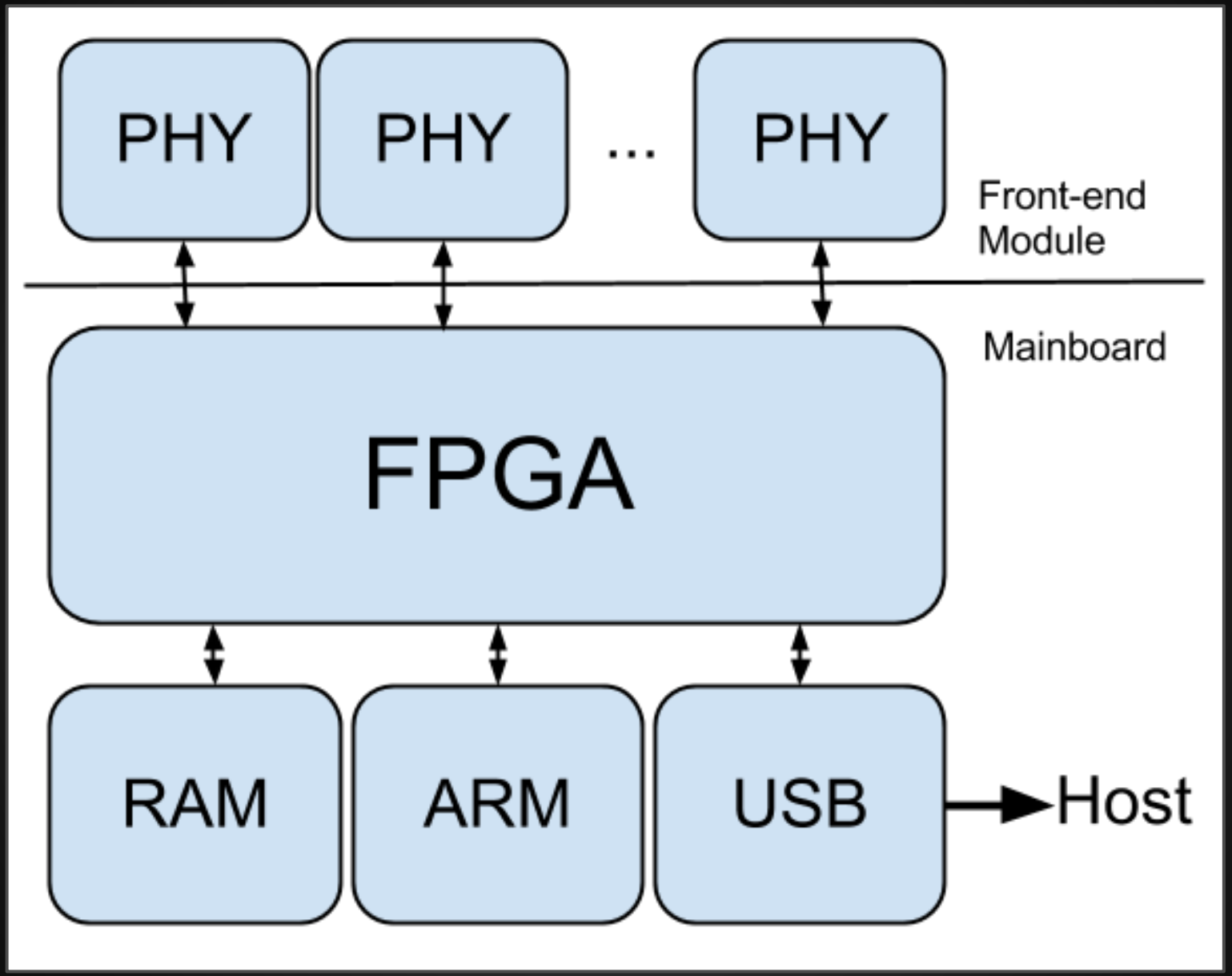
How?

Progress

Two-part design

FPGA mainboard connected via USB3; HW-assisted signal handling with extremely fast pipe to host OS

Multiple modular front-end boards for interfacing with various physical layers


PHY
PHY
...
PHY
Front-end
Module
Mainboard
FPGA
RAM
ARM
USB
Host

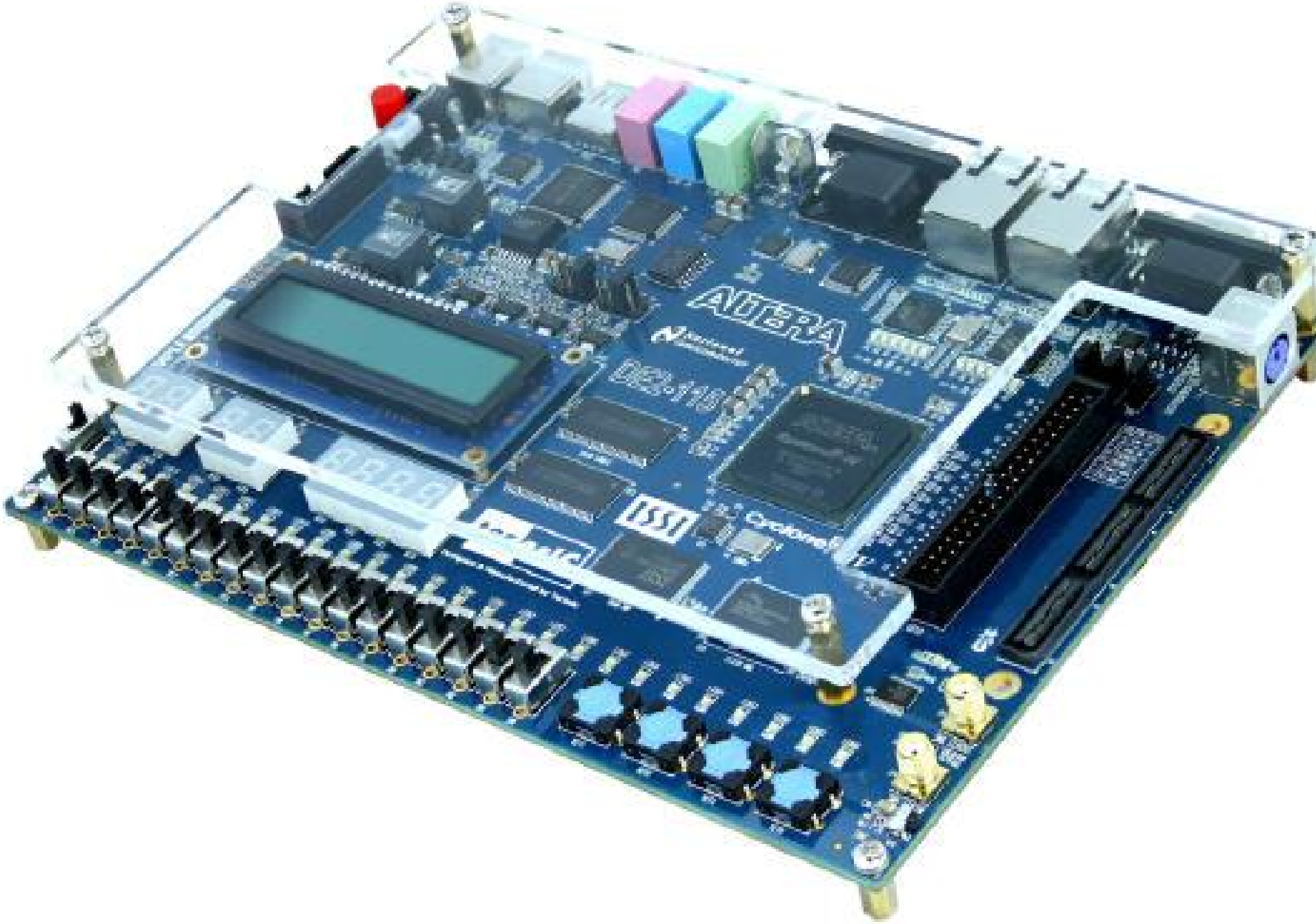
Development Hardware

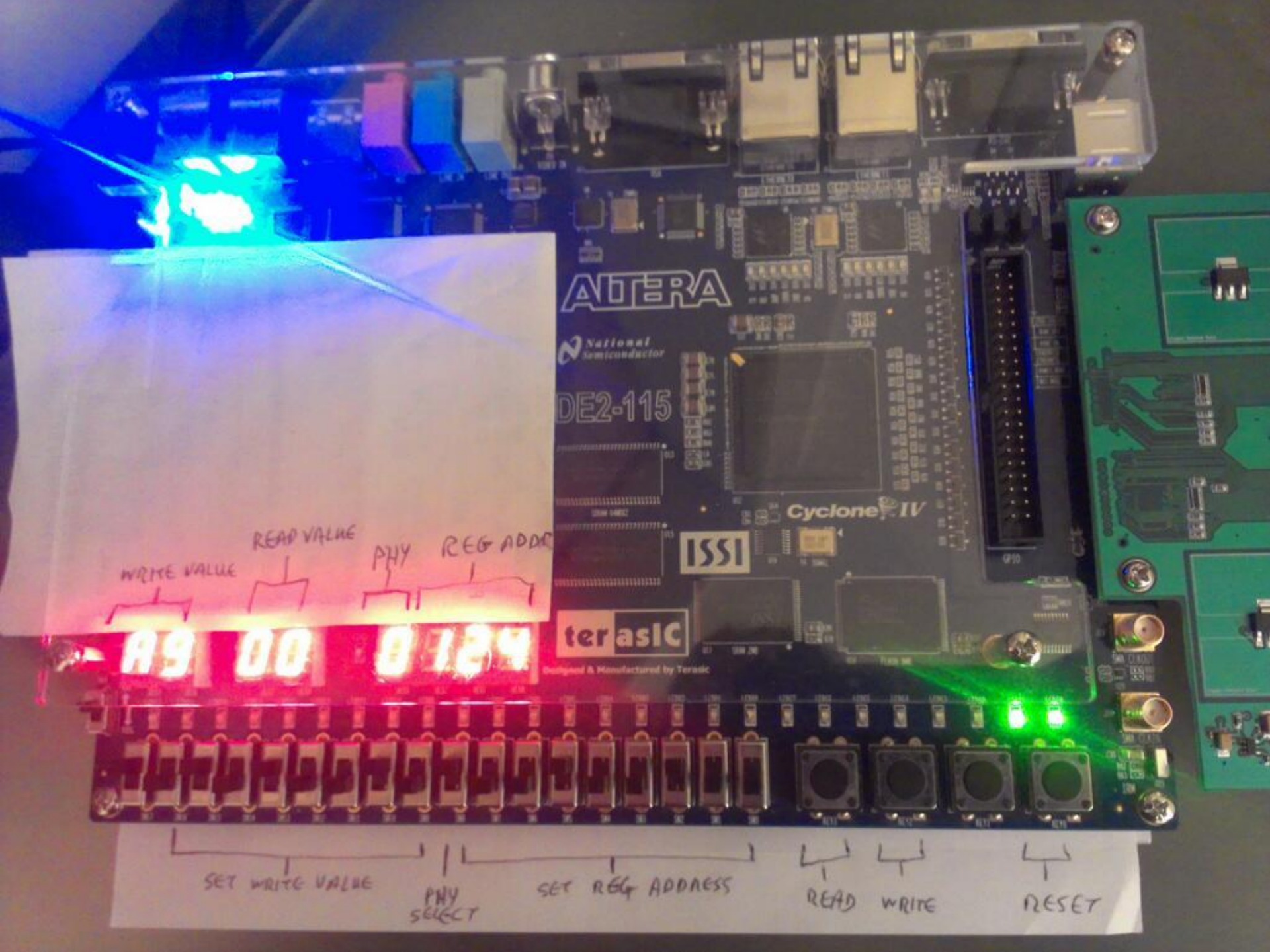
Terasic DE2-115

Altera FPGA

Low-speed 2x20 parallel header, high speed mezzanine connector

Well supported by dev tools, bootstraps front-end dev


ALTERA
ISSI
Cyclone


ALTERA
National Semiconductor
DE2-115
Cyclone IV
ISSI
terasIC
Designed & Manufactured by Terasic
WRITE VALUE
READ VALUE
PHY
REG ADDR
A9 00 0 124
SET WRITE VALUE
PHY SELECT
SET REG ADDRESS
READ
WRITE
RESET

Development Downsides

DE2 is great, but...

... Mezzanine connector can't handle high ENOUGH speed comms for USB3 or full-rate HDMI front-ends

Not particularly portable

Expensive / Closed design

Outline

Background

What?

Why?

How?

Progress

Hardware: RS-232

Simplest front-end board

2 pairs of RS-232 ports (DTE and DCE)

Low speed (compared to the others anyhow)

2-layer PCB

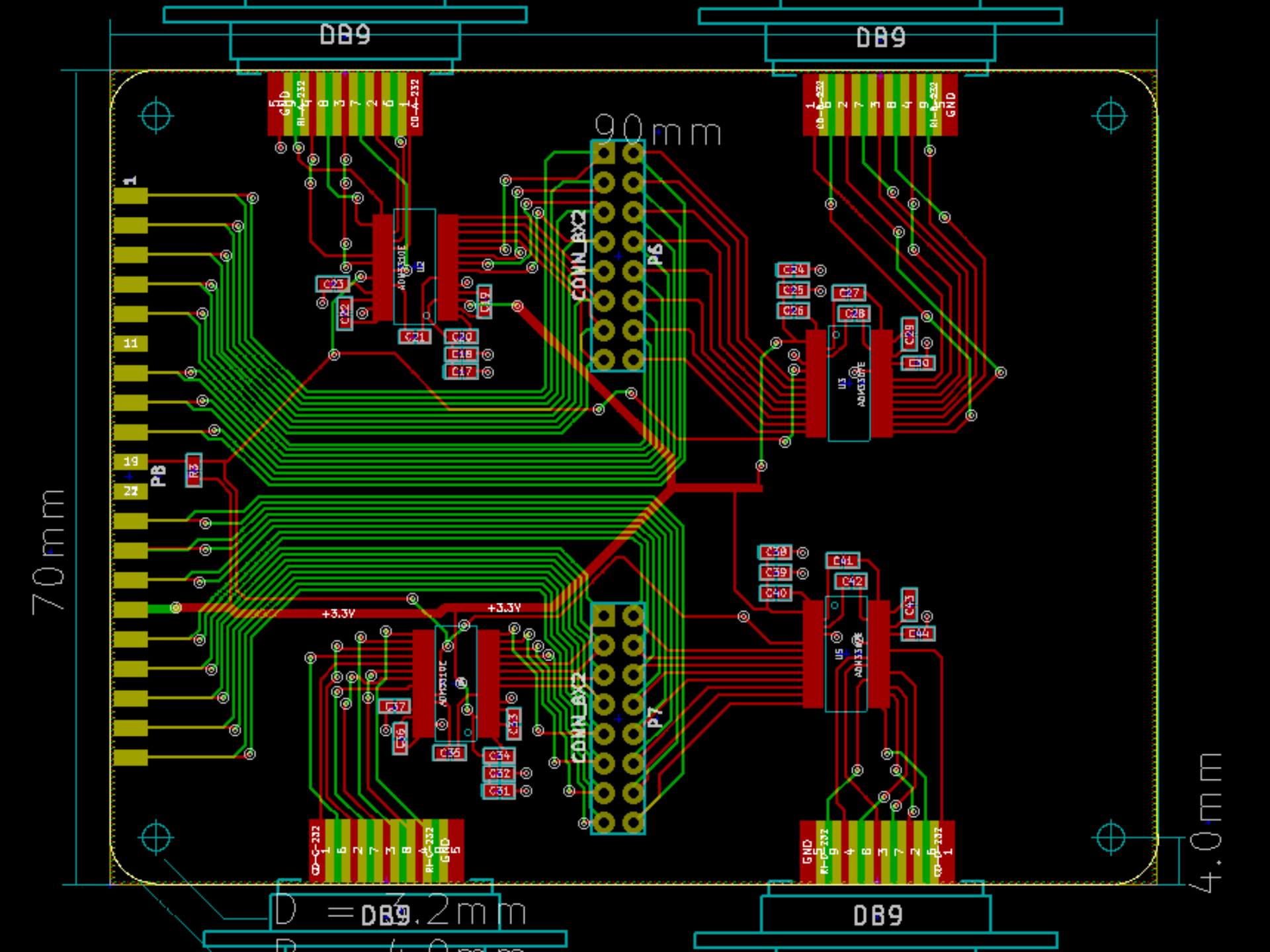
Hardware: RS-232

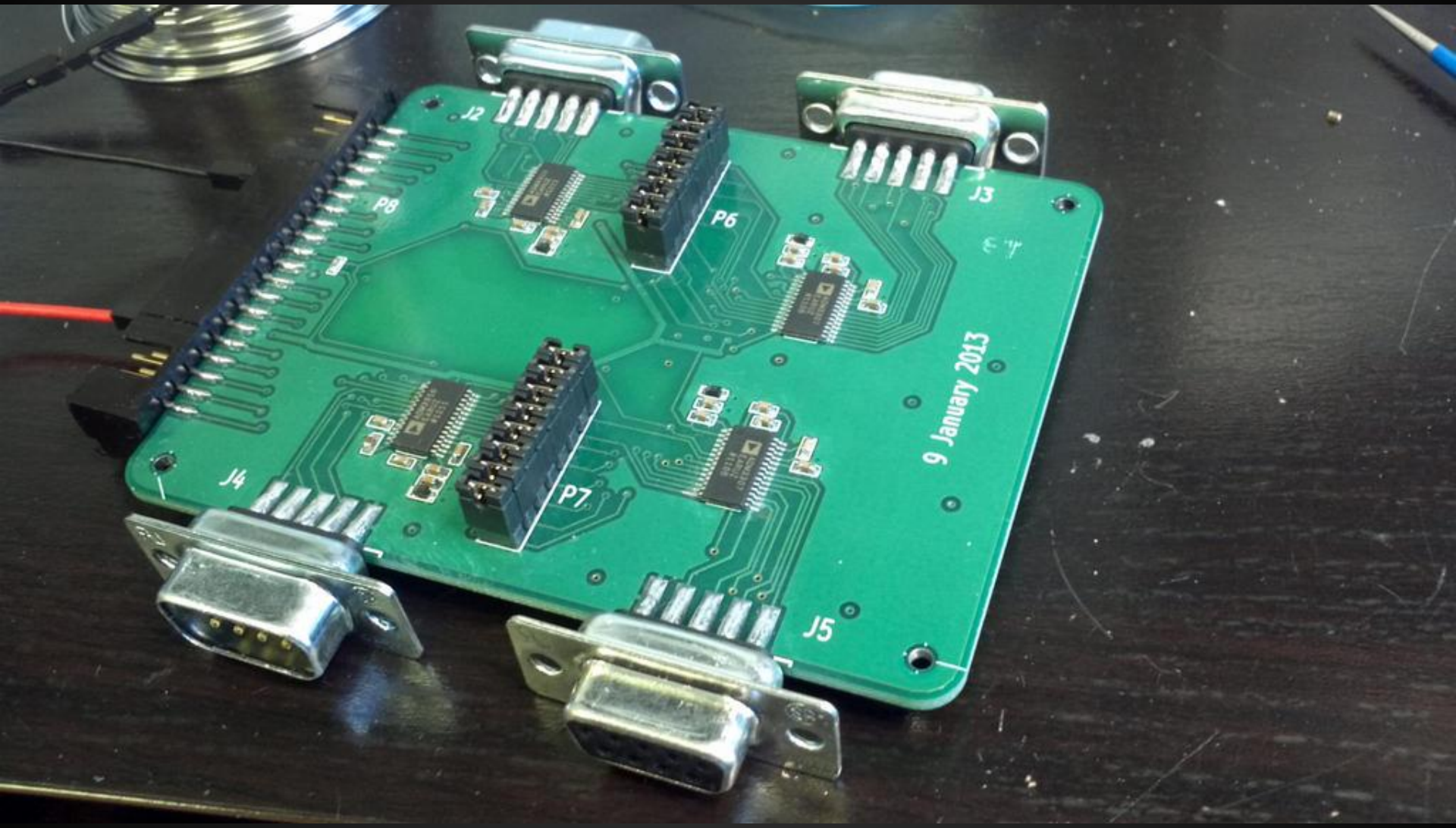
Converts 232 to TTL, routes through FPGA, then back to 232

Monitors all signals; TXD, RXD, DTE, DTR, RTS, etc

Able to jumper single signals w/out decode

Current design uses DE2 2x40 parallel connector, final design will use mezzanine


DB9
DB9
90mm
70mm
4.0mm
D =39.2mm
DB9
DB9
CONN_8X2
P6
CONN_8X2
P7
+3.3V
+3.3V
PB
GND
GND


J2
P8
P6
J3
J4
P7
J5
9 January 2013

RS-232 Goals

Complete logging of all signals, including carrier sense, etc

Proof of concept for FPGA based MITM of signals

Logging serial console data alongside Ethernet

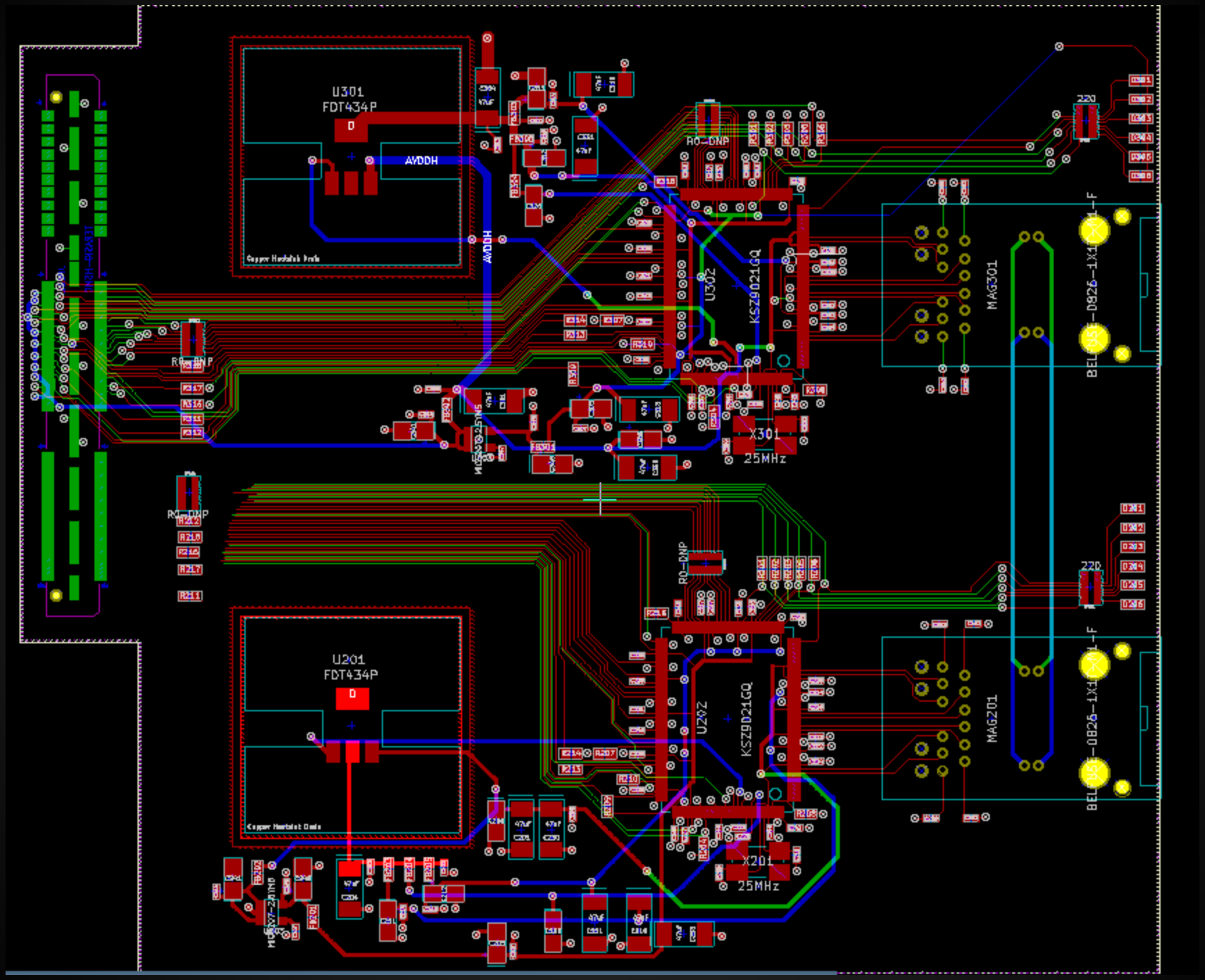
Hardware - Gig-E Tap

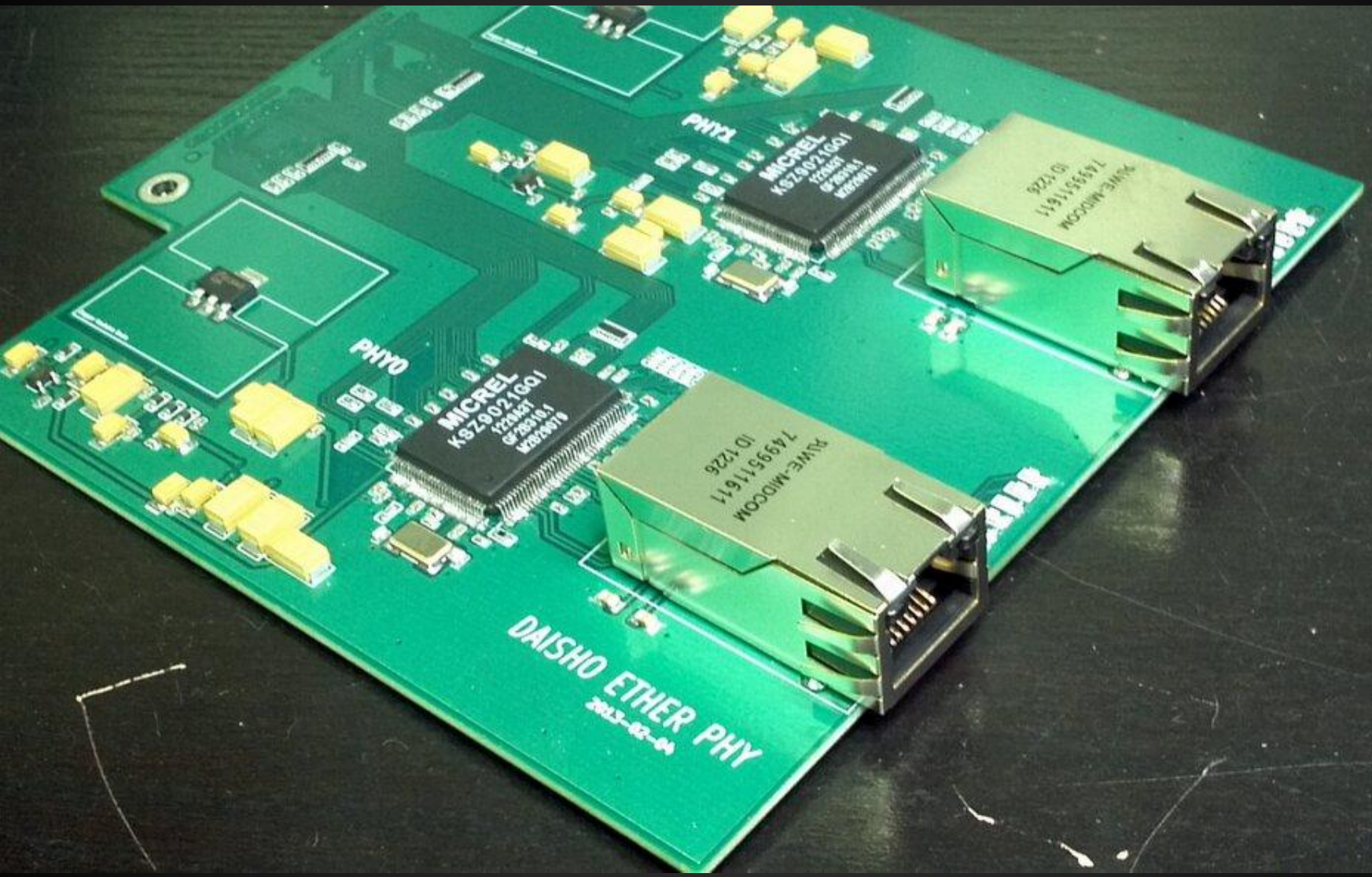
Two independent Gig-E PHYs, 10/100/1000

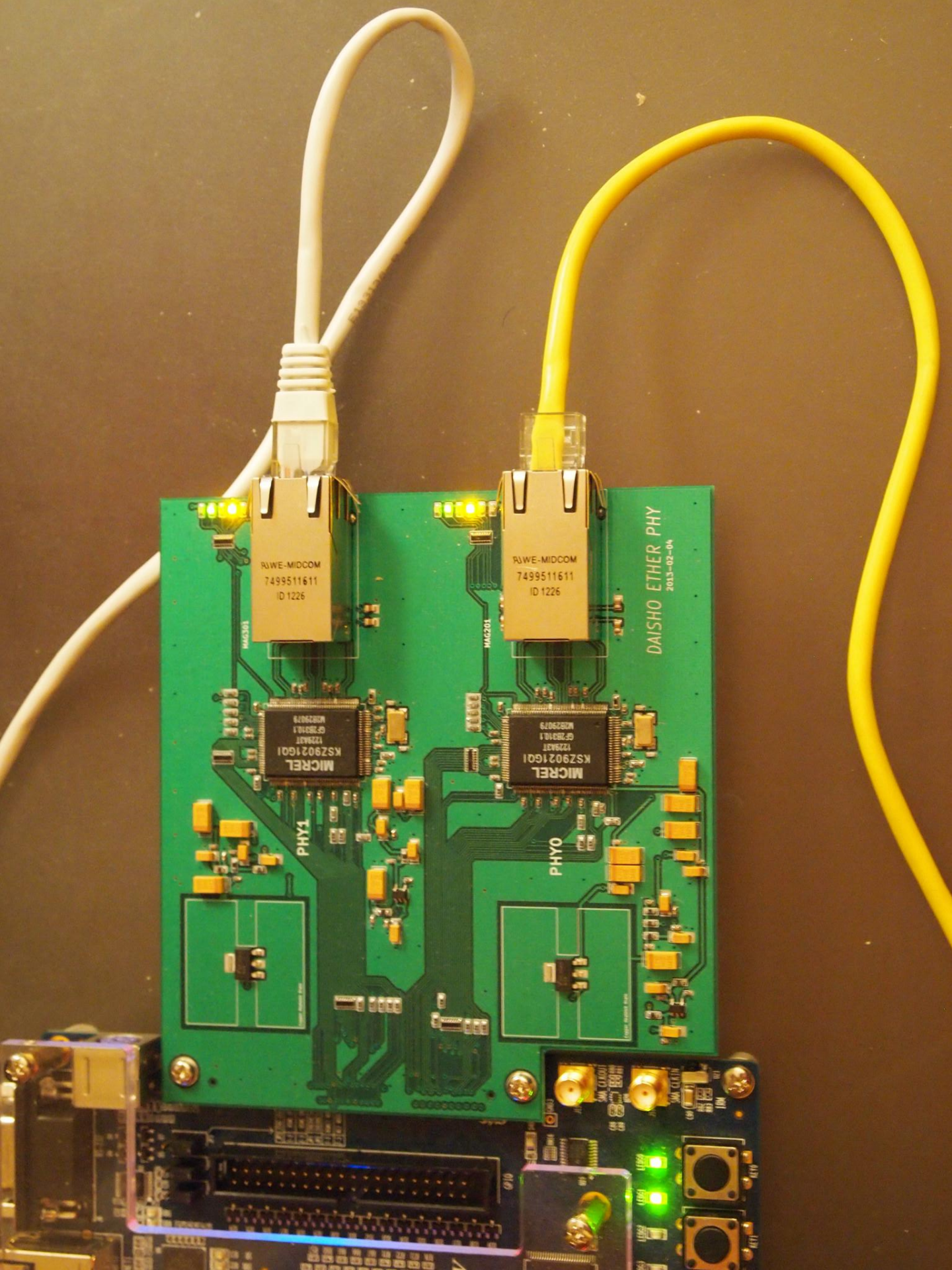
Dumps packet to FPGA, FPGA writes packet back to other PHY

Integrated jack magnetics, plug-and-go

4-layer PCB; more complicated but still reasonable


U301
FDT434P
D
AVDDH
Copper Heatsink Pads
AVDDH
R0_DNP
U302
KSZ9021GQ
MAG301
BEL_SS-0826-1X1-F
X301
25MHz
R0_DNP
R0_DNP
U201
FDT434P
D
Copper Heatsink Pads
U202
KSZ9021GQ
MAG201
BEL_SS-0826-1X1-F
X201
25MHz


PHY1
MICREL
KSZ9021GQI
PHY0
MICREL
KSZ9021GQI
DAISHO ETHER PHY


RJWE-MIDCOM
7499511611
ID 1226
RJWE-MIDCOM
7499511611
ID 1226
DAISHO ETHER PHY
2013-02-04
MAG01
MAG02
MICREL
KSZ9021GAI
MICREL
KSZ9021GAI
PHY1
PHY0

Gig-E Goals

Support for 1 Gbps data rate which can't be passively tapped

High precision timestamping of packets

Precision relative timestamping of each side of link

"Invisible" monitoring

PHY vs MAC

PHY transceiver encodes bitstream and transmits electrical signals

MAC layer implements the Ethernet standard and is an Ethernet device on the network

Switches and network interfaces are MAC layer devices

Daisho has no MAC layer; it's PHY-only

Why we care about Gig-E

There are lots of existing Gig-E taps

They all implement a PHY+MAC - port mirroring switches are Ethernet devices on the network

Bridging can be detected & creates traffic

Dual PHY with byte-duping is as close to "passive" as we can get with Gig-E

Hardware - HDMI Tap

2 HDMI ports (In / Out)

Single high-speed SerDes (serializer/deserializer) to parallelize data to FPGA

6 layer PCB design, very high-speed parallel data

Hardware - HDMI Tap

Using SerDes lets us get at least 1080p

We'll hopefully even support 4K

Huge number of IO lines, strains capability of development hardware

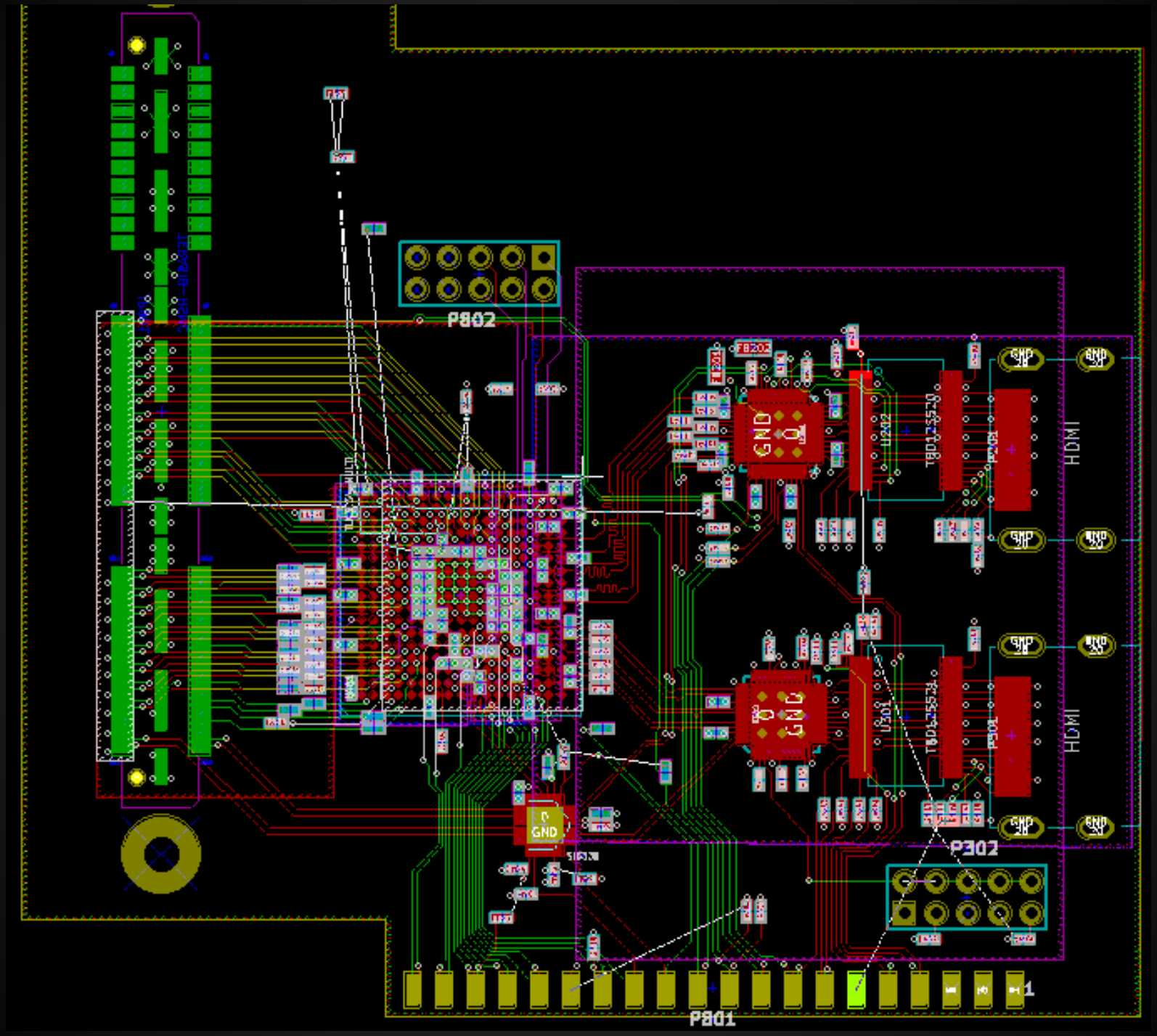
Alternate HDMI methods

Could plumb HDMI directly to FPGA and decode differential signals

NeTV does this; limited to 720p/1080i

Requires FPGA be absurdly fast to handle multi-GHz signals for 1080p and up

SerDes converts serial to parallel and allows for slower individual data lines


P802
HDMI
HDMI
P802
P801

Why HDMI is Interesting

Can be a complicated protocol

What else is going on besides encryption?

Has a 100 Mbps Ethernet channel

I2C communications bus

Hardware - Mainboard

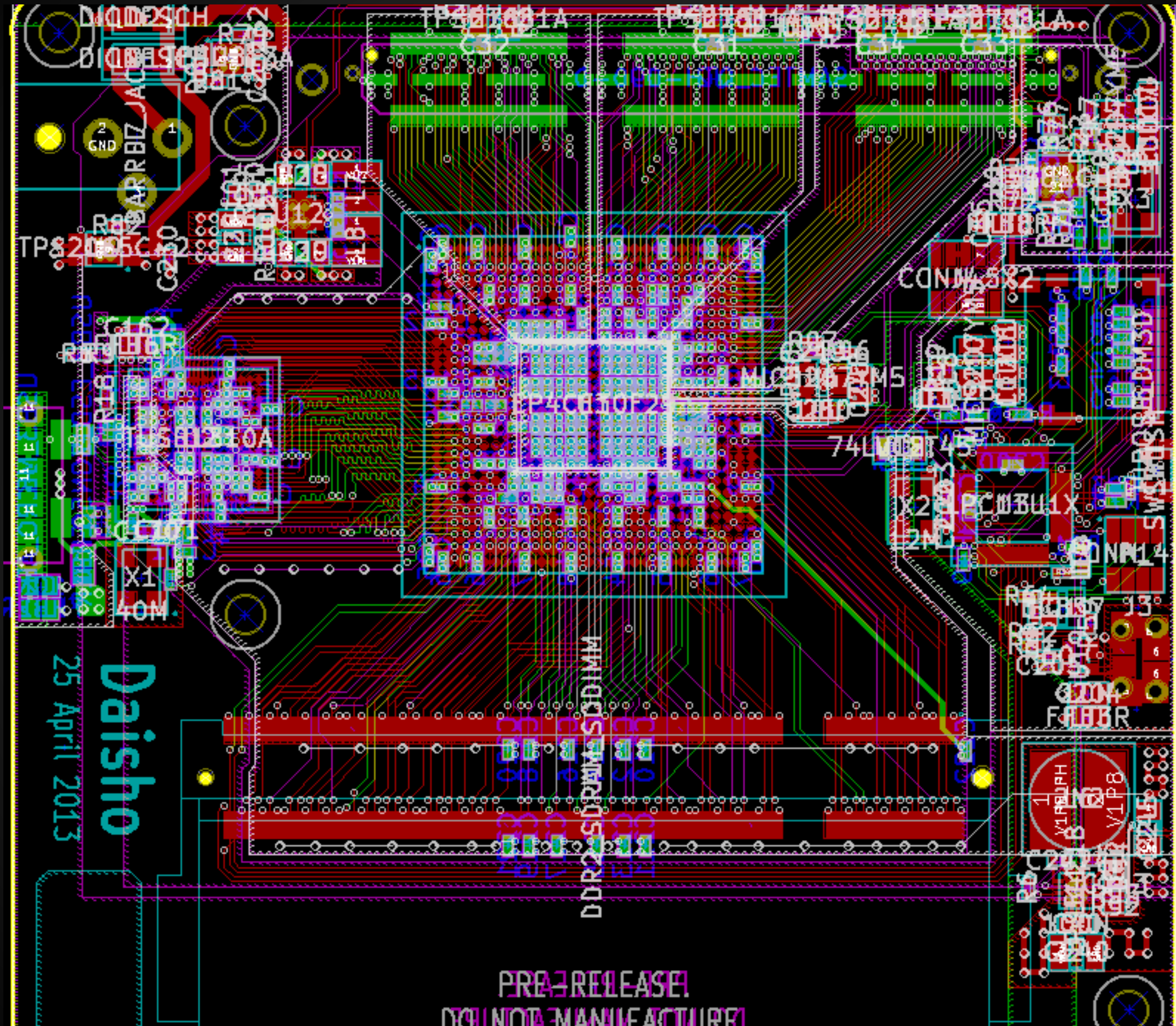
Altera Cyclone IV FPGA

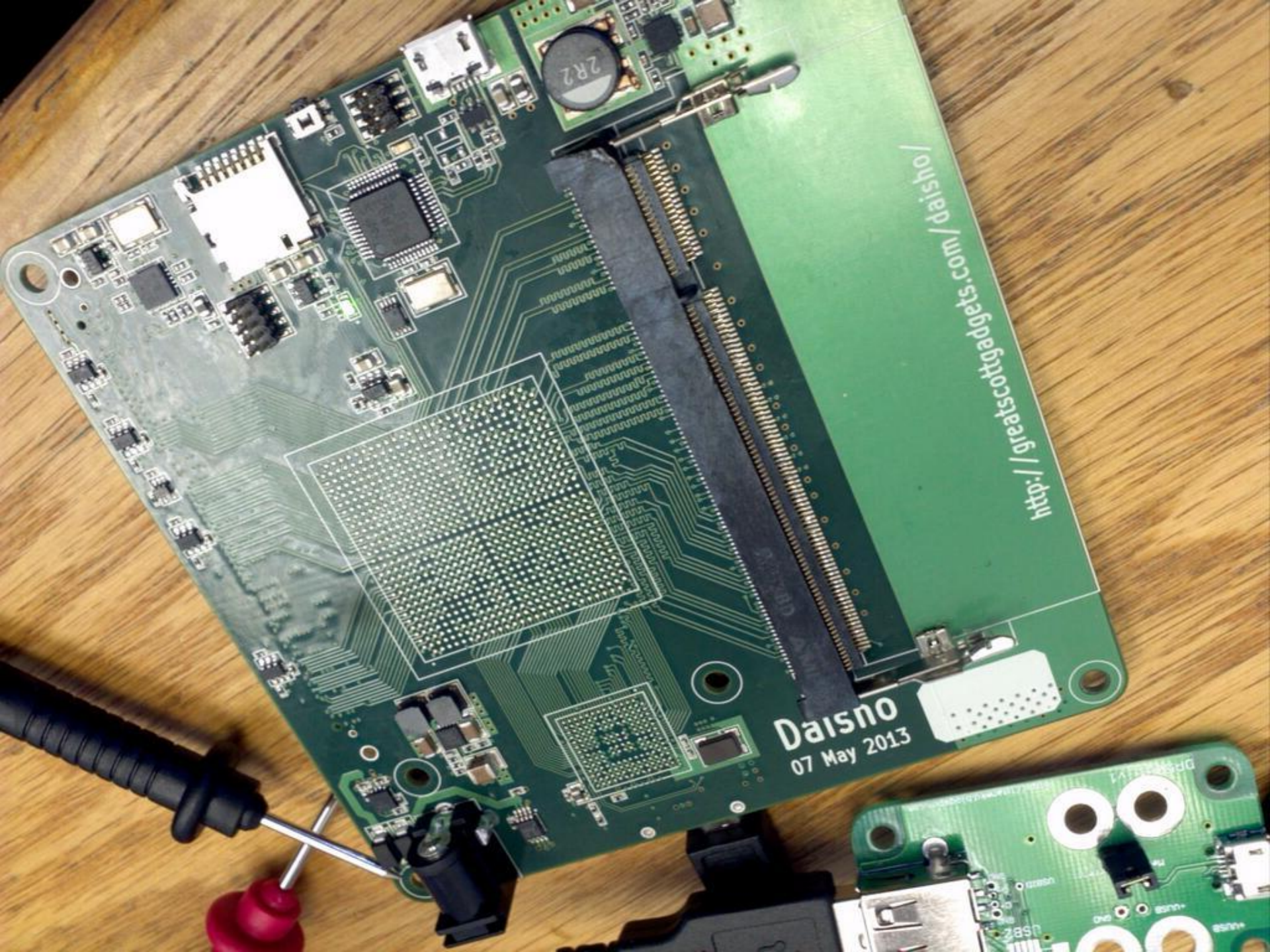
NXP ARM MCU for bootstrapping FPGA

DDR2 RAM

USB 3.0 to host

Designed by Jared Boone / ShareBrained


Daisho
25 April 2013
PRE-RELEASE
DO NOT MANUFACTURE
DDR2 SODIMM
CONN_5X2
X1
40M
GND


2R2
http://greatscottgadgets.com/daisho/
Daisho
07 May 2013

ALTERA
Cyclone IV
EP4CE30F29C8N
G CCAAA1313A
KOREA
VENTURA TECHNOLOGY GROUP INC
D2-53GG130MV-666
25463
Daisho
07 May 2013

PCB Design

Lots of EE-CAD tools to pick from, both OSS and commercial

We try to only use open toolchains

Eagle is "free" but limited, has funky licensing requirements for complex designs

KiCad! (Fully OSS / Unencumbered license)

KICAD

The finest PCB tool that Open Source has to offer.

KiCad

Capable of N-layer boards (no license limit)

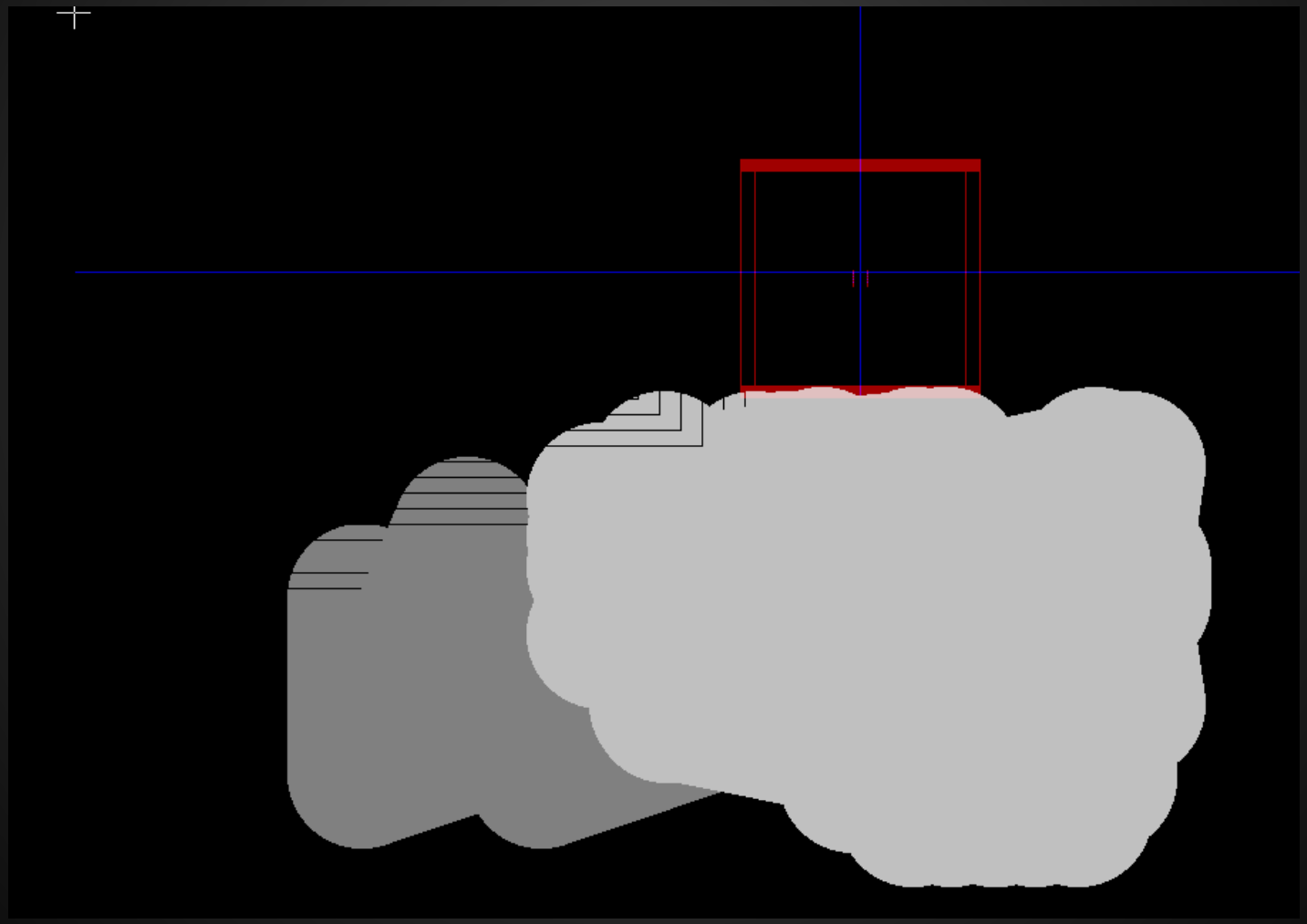
No size restrictions

Friendly (well, friendlier) file formats, all text

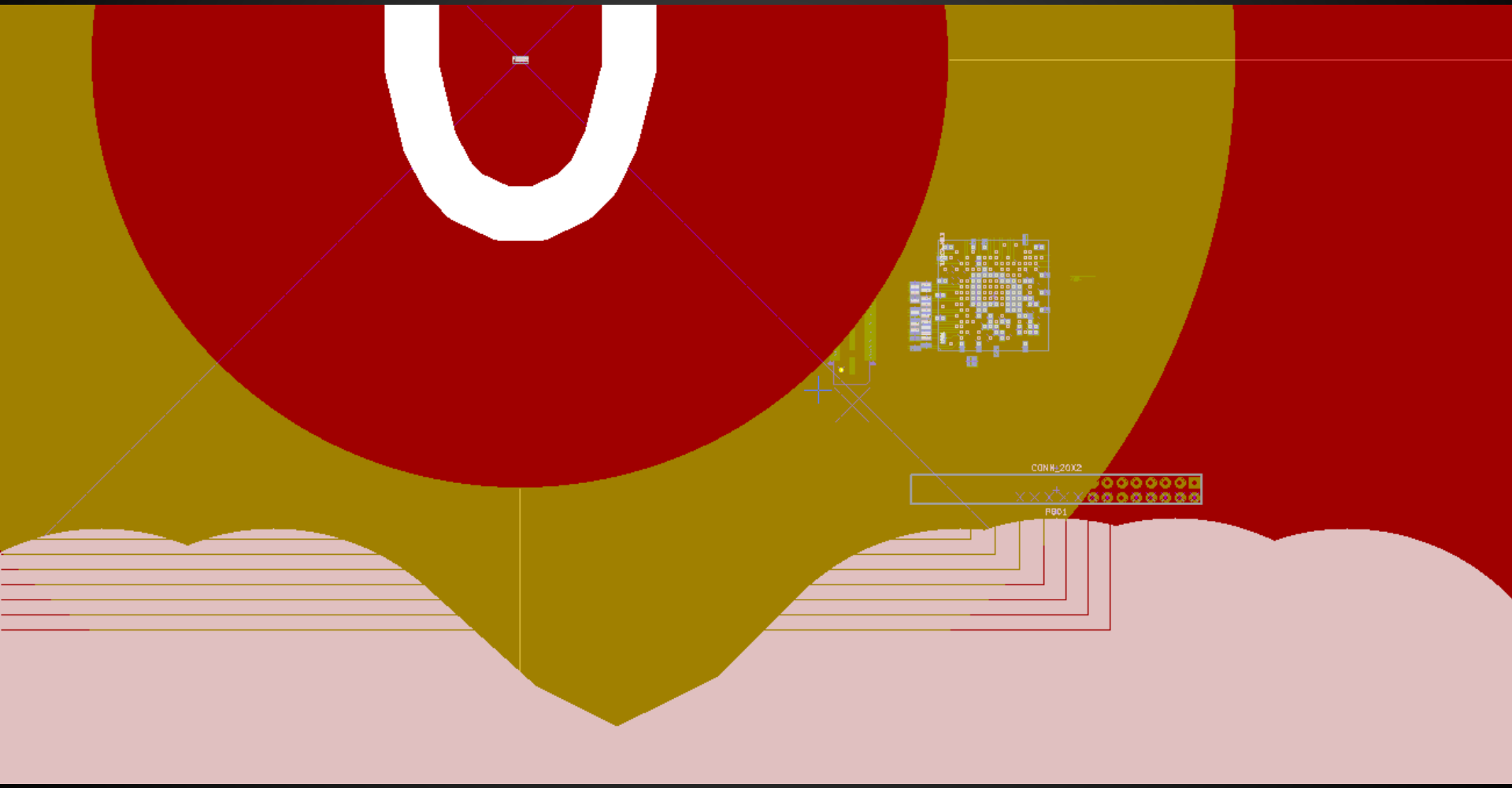
Truly OSS - which is good ***and*** bad at times

Sometimes does... ... odd things

Seriously, KiCAD? WTF.



I don't even...



KiCad Challenges

Development version doesn't play nice with stable version (New PCB format, different units of measurement)

Not very good at moving components once they're placed

No helpful auto-tools for length matching, BGA routing, etc - for very complex designs, can feel "write-only"

PCB Design Challenges

High speed digital signals can behave very oddly

Fab requirements become integral to the design

Home assembly is still *possible*, but you probably wouldn't *want* to

Prototype size runs are VERY expensive

PCB Requirements

Many of the designs (those using BGA) are more than 4 layer, which puts us outside many prototype fab capabilities

Via-in-Pad (holes inside pads) is expensive, but unavoidable at this complexity

Have to work with fab on layer stack-up, impedance control, etc - high speed signals very sensitive to it

Mainboard PCB Specifications

8 copper layers

5 mil trace width

5 mil trace isolation

8 mil vias, many via-in-pad

4.85 mil annular ring

Mainboard Firmware

World's first open source USB 3.0 core

Minimal implementation of protocols to ship data to host

Front-ends don't currently require independent FPGA bitstream, but will contain identifiers to allow dynamic bitstream loading if necessary

Software

libdaisho

- Userspace driver based on libusb

Wireshark integration

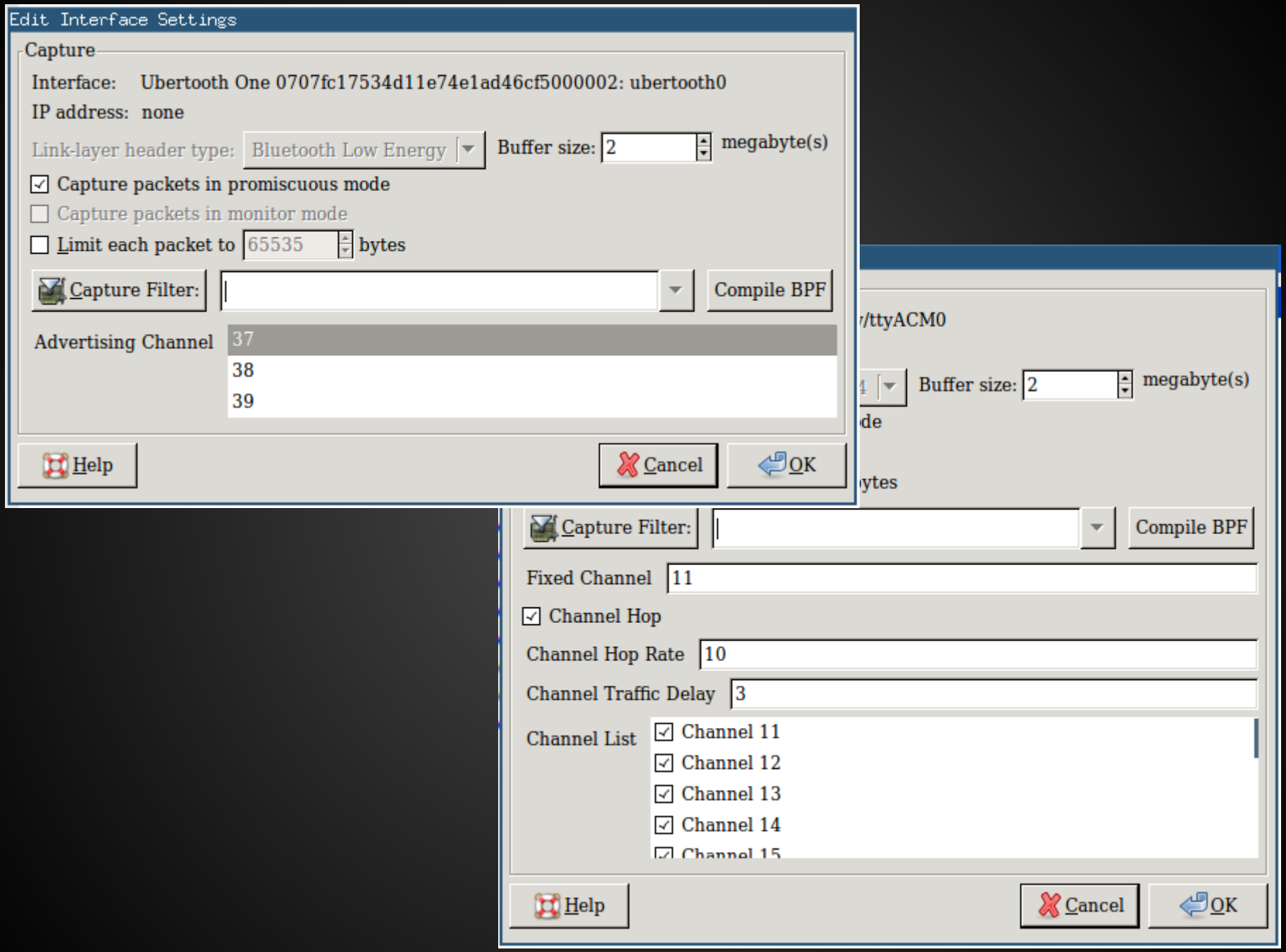
- Using extcap

Extcap

Under development / initial version submitted to Wireshark in time for Black Hat

Allows a simple config grammar to define GTK UI for Wireshark

Allows non-netdev capture with minimal developer effort


Edit Interface Settings
Capture
Interface: Ubertooth One 0707fc17534d11e74e1ad46cf5000002: ubertooth0
IP address: none
Link-layer header type: Bluetooth Low Energy
Buffer size: 2
megabyte(s)
Capture packets in promiscuous mode
Capture packets in monitor mode
Limit each packet to 65535 bytes
Capture Filter:
Compile BPF
Advertising Channel
37
38
39
Help
Cancel
OK
/ttyACM0
Buffer size: 2
megabyte(s)
Capture Filter:
Compile BPF
Fixed Channel 11
Channel Hop
Channel Hop Rate 10
Channel Traffic Delay 3
Channel List
Channel 11
Channel 12
Channel 13
Channel 14
Help
Cancel
OK

Software

USB3 enumeration is completely under our control

Possible for front-end boards to present as PHY-specific interfaces to host (Gig-E presenting as 2 CDC-ACM Gig-E interfaces, for instance)

Also possible to simply do bulk IO via LibUSB

Demonstration

What's Next?

Full mainboard bring-up

Re-target front-ends for our mainboard instead of DE2 development environment

Auto-identifying connected frontends via ID chips

Possible Future Targets

DisplayPort
DVI
SATA
SAS
Telephone/DSL
Wideband SDR
Fiber Gig-E

more...?

Thanks

DARPA CFT Program

BIT Systems

Questions?

http://greatscottgadgets.com/daisho

https://github.com/mossmann/daisho

#daisho on irc.freenode.net

Black Hat feedback is appreciated