Berkeley
UNIVERSITY OF CALIFORNIA

The Collaboration for Astronomy Signal Processing and Electronics Research in 2017

Jack Hickish

Radio Astronomy Lab, UC Berkeley
jackh@berkeley.edu

September 7, 2016

Outline

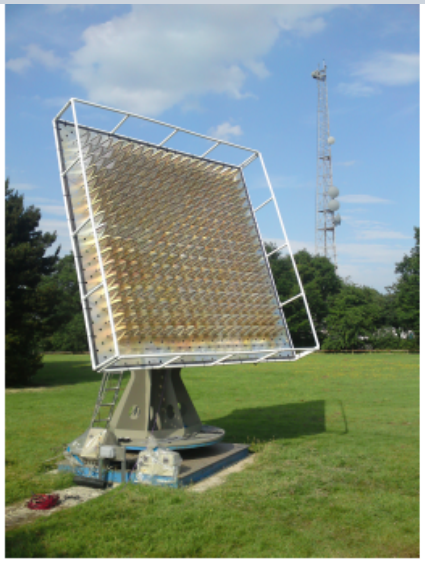
Acknowledgements

- Local Organizing Committee
  - Michael Burke
  - Dave Hawkins
  - Ryan Monroe
  - Jonathon Kocz
  - Terry Filiba Schrager
  - Melissa Soriano
  - Cody Vaudrin
- Busyweek-ers
  - Adam Isaacson
  - Jonathon Kocz
  - Wes New
  - Amish Patel

The Age of Digital Radio Astronomy

Large Dishes

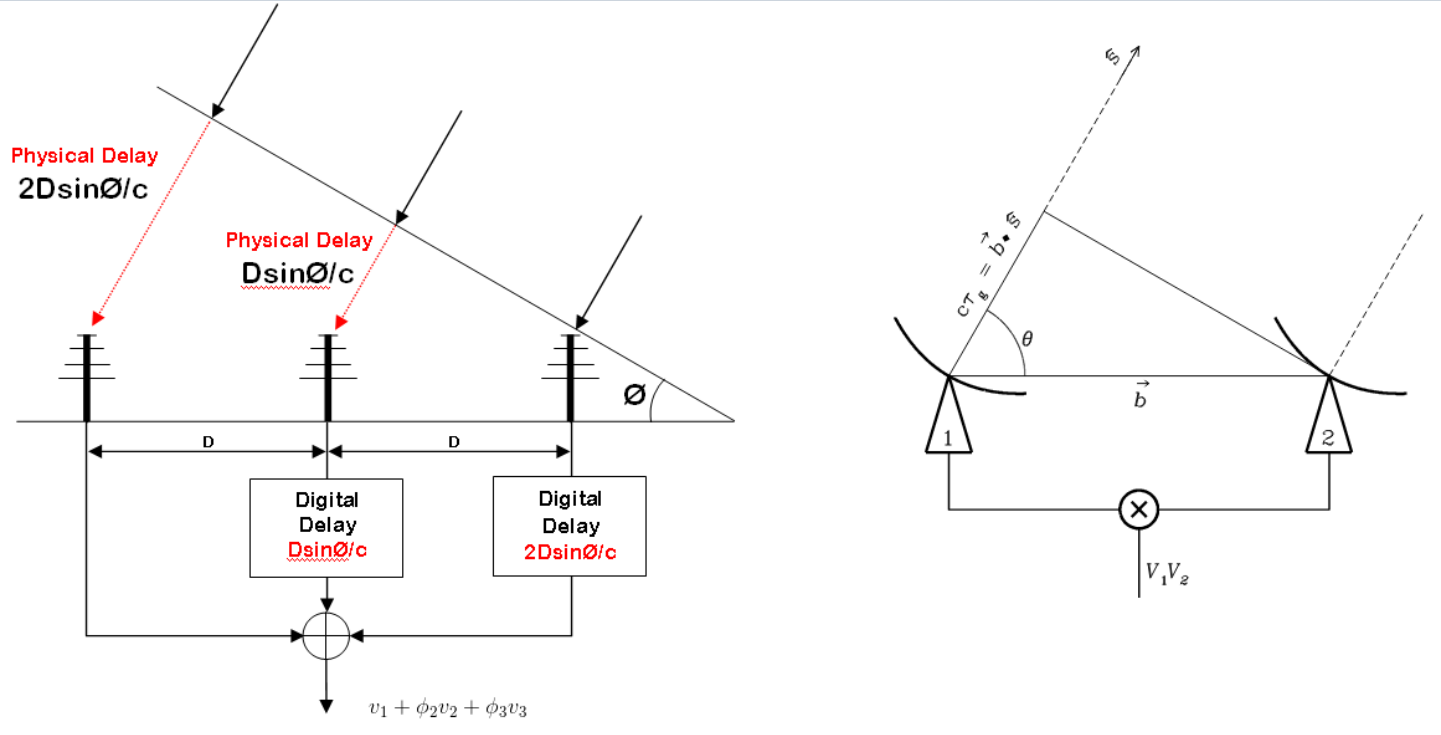
Arrays of small telescopes



Phased Arrays

EXPENSIVE ELEMENTS → **EXPENSIVE COMPUTATION**

The Age of Digital Radio Astronomy



More is more. Best is best.

- More bandwidth
- More field-of-view
- More sensitivity
- More resolution (time, frequency, spatial)

Moore's law in correlators

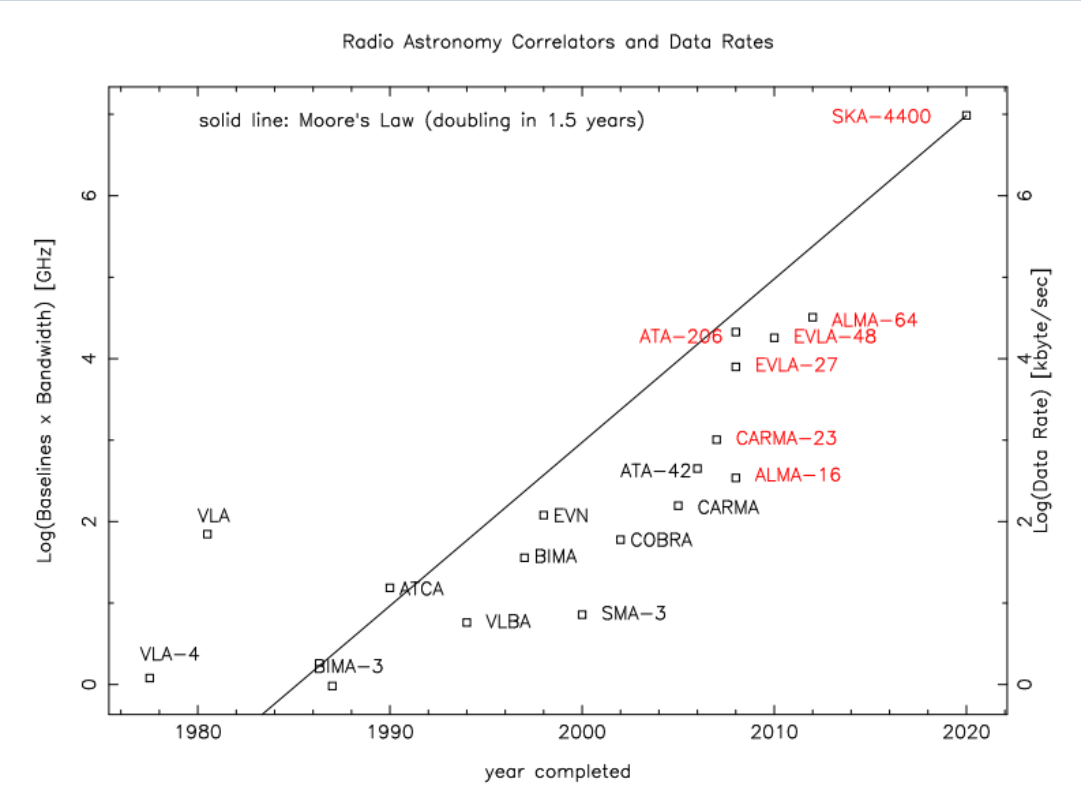
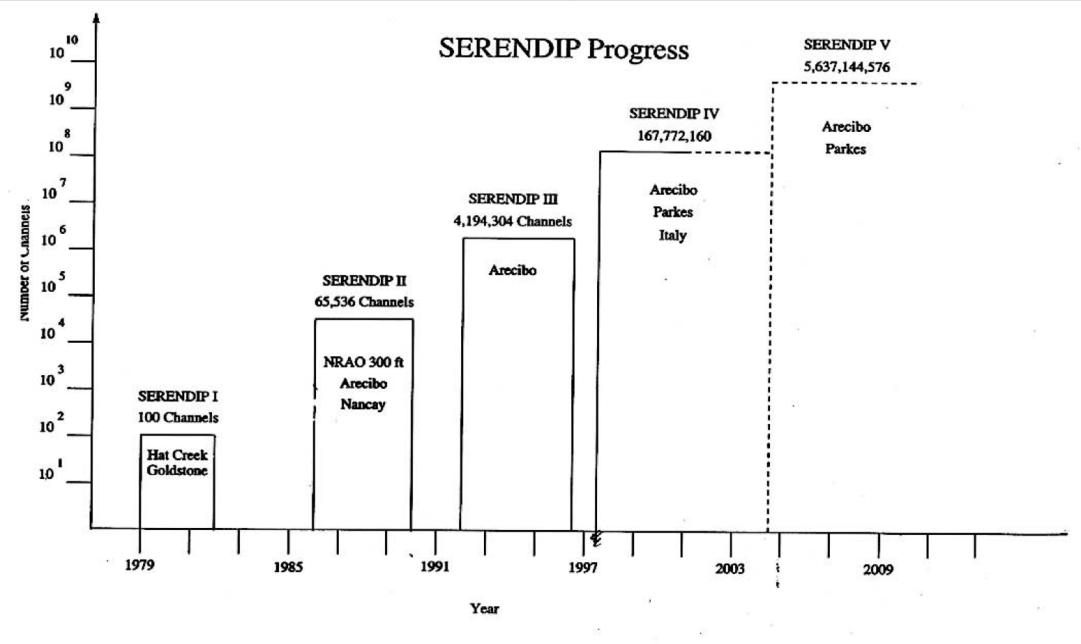


Figure: Credit: Mel Wright (`https://casper.berkeley.edu/wiki/Videos`)

Moore's law in spectrometers



Radio-Astronomy DSP Parameters

- Number of antennas ($1 \rightarrow \sim 1000$)
- Bandwidth (100 MHz $\rightarrow$ 100 GHz)
- Frequency channels ($\sim 2^{10} \rightarrow \sim 2^{30+}$)
- Averaging period (None $\rightarrow$ 10 s)

- Many Tb/s interconnect, many complex Tops/s

> *"The data collected by the SKA in a single day would take nearly two million years to playback on an ipod."*
> *https://www.skatelescope.org/amazingfacts/*

Building DSP systems for Radio Telescopes

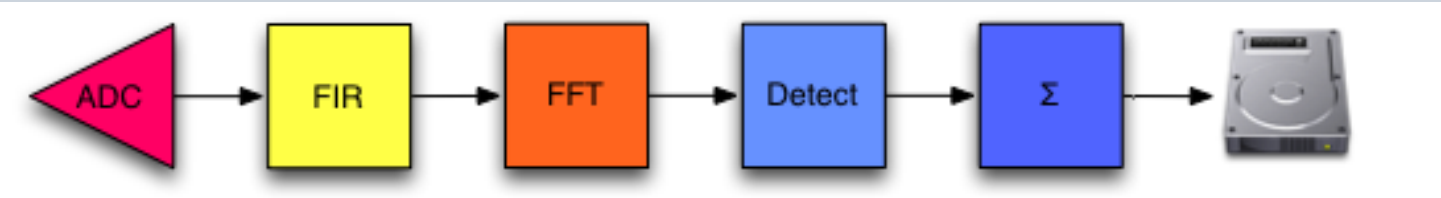
- Large data input rates
- Large computation rates
- Simple operations
- Many common components between different telescopes
- Highly parallel

Radio Astronomy DSP Parameters

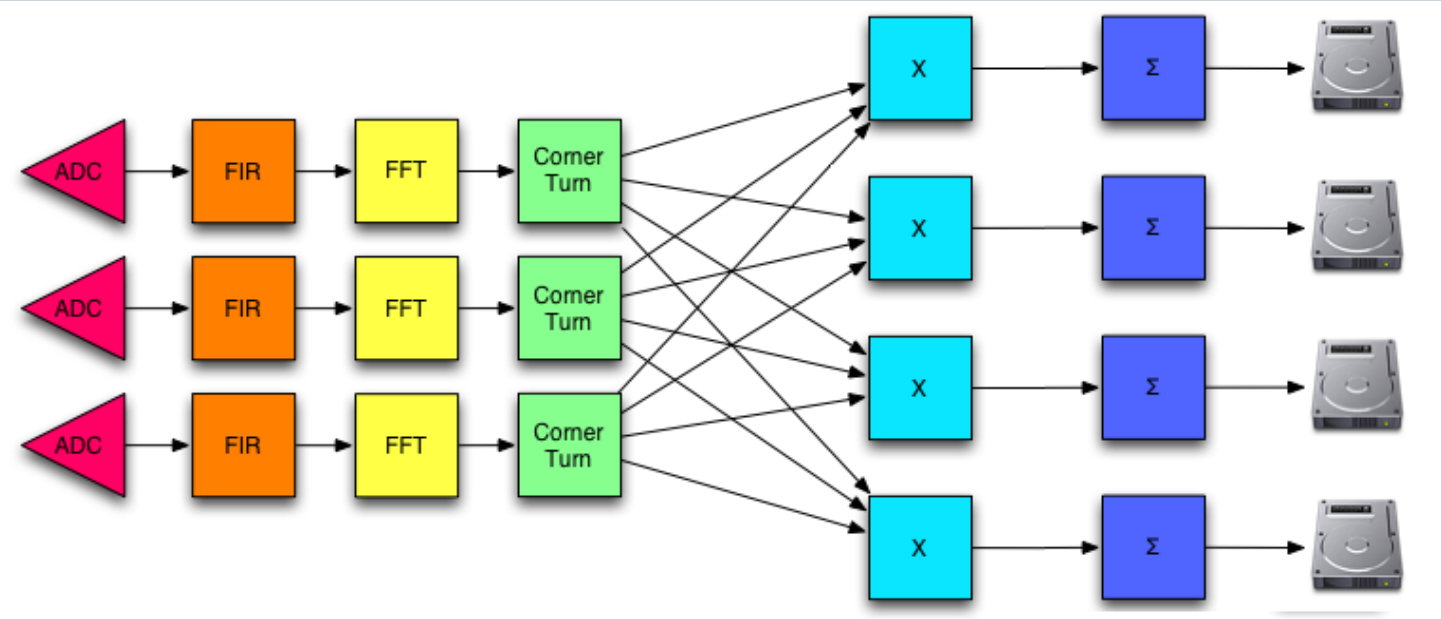
A large variety of Radio Astronomy instruments can be build from a small number of parameterised

- Filters (F-Engines)
- Correlation Engines (X-Engines)
- Beamforming Engines (B-Engines)
- Interconnect

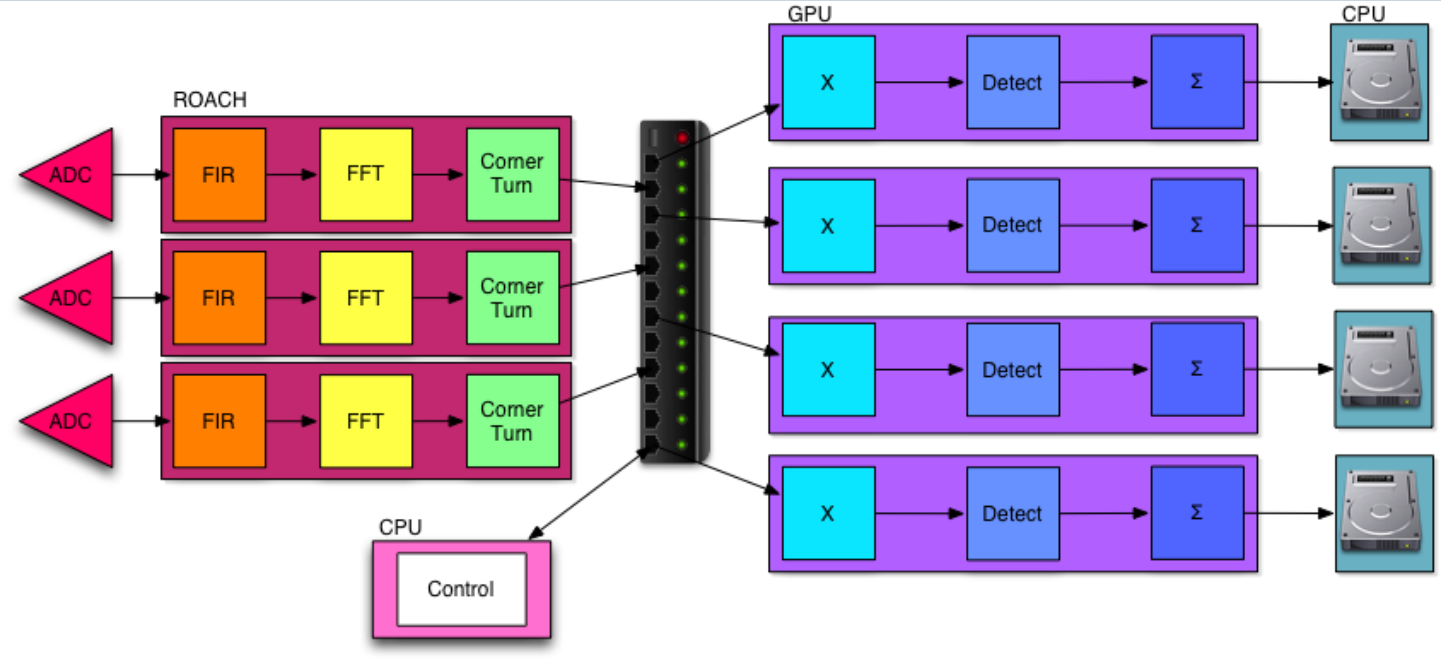
A Simple Spectrometer



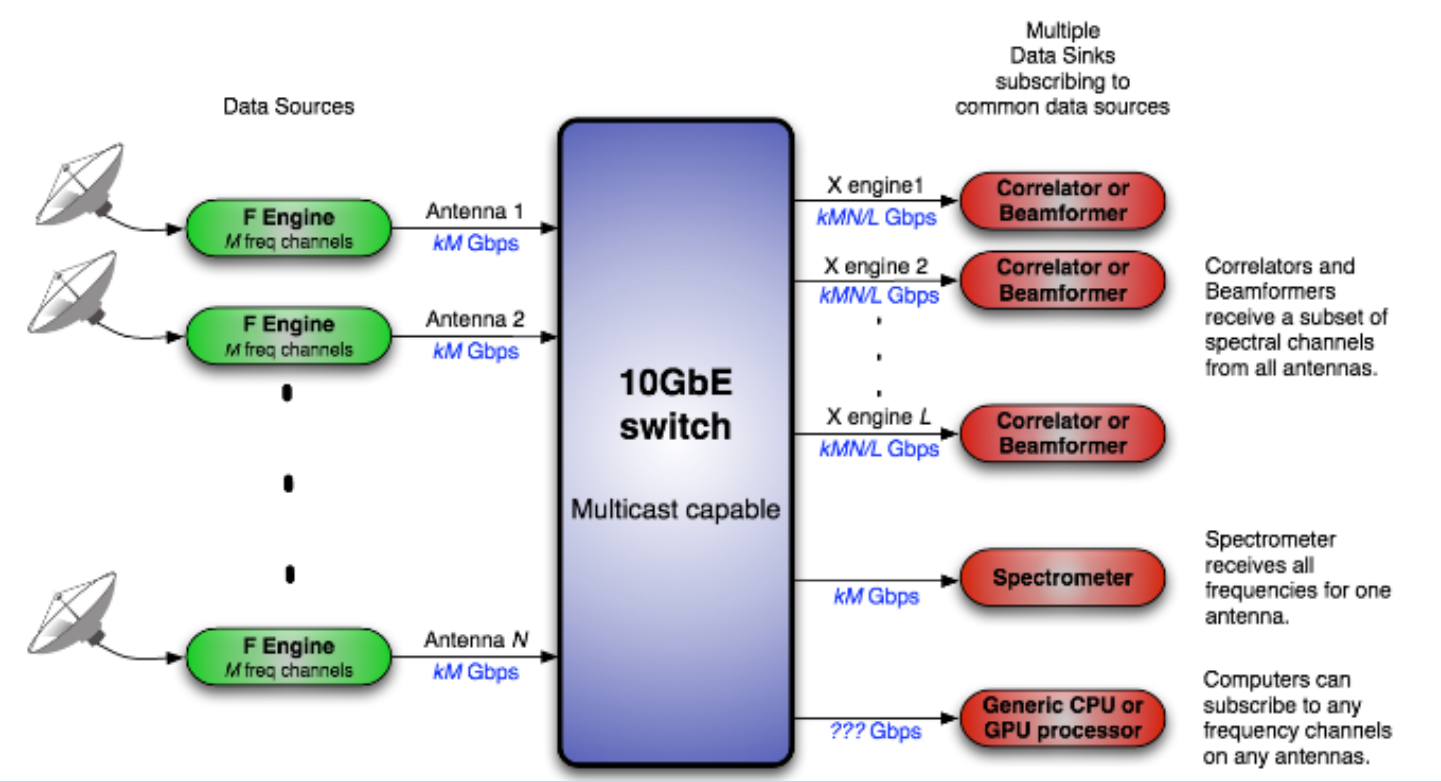
A Multi-Antenna System



A Multi-Antenna System



A Multi-User System



CASPER

Center for Astronomy Signal Processing and Electronics Research

CASPER

~~Center~~ for Astronomy Signal Processing and Electronics Research Collaboration

CASPER

~~Center~~ for Astronomy Signal Processing and Electronics Research

~~Collaboration~~

Community?

CASPER

"The primary goal of CASPER is to streamline and simplify the design flow of radio astronomy instrumentation by promoting design reuse through the development of platform-independent, open-source hardware and software."

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

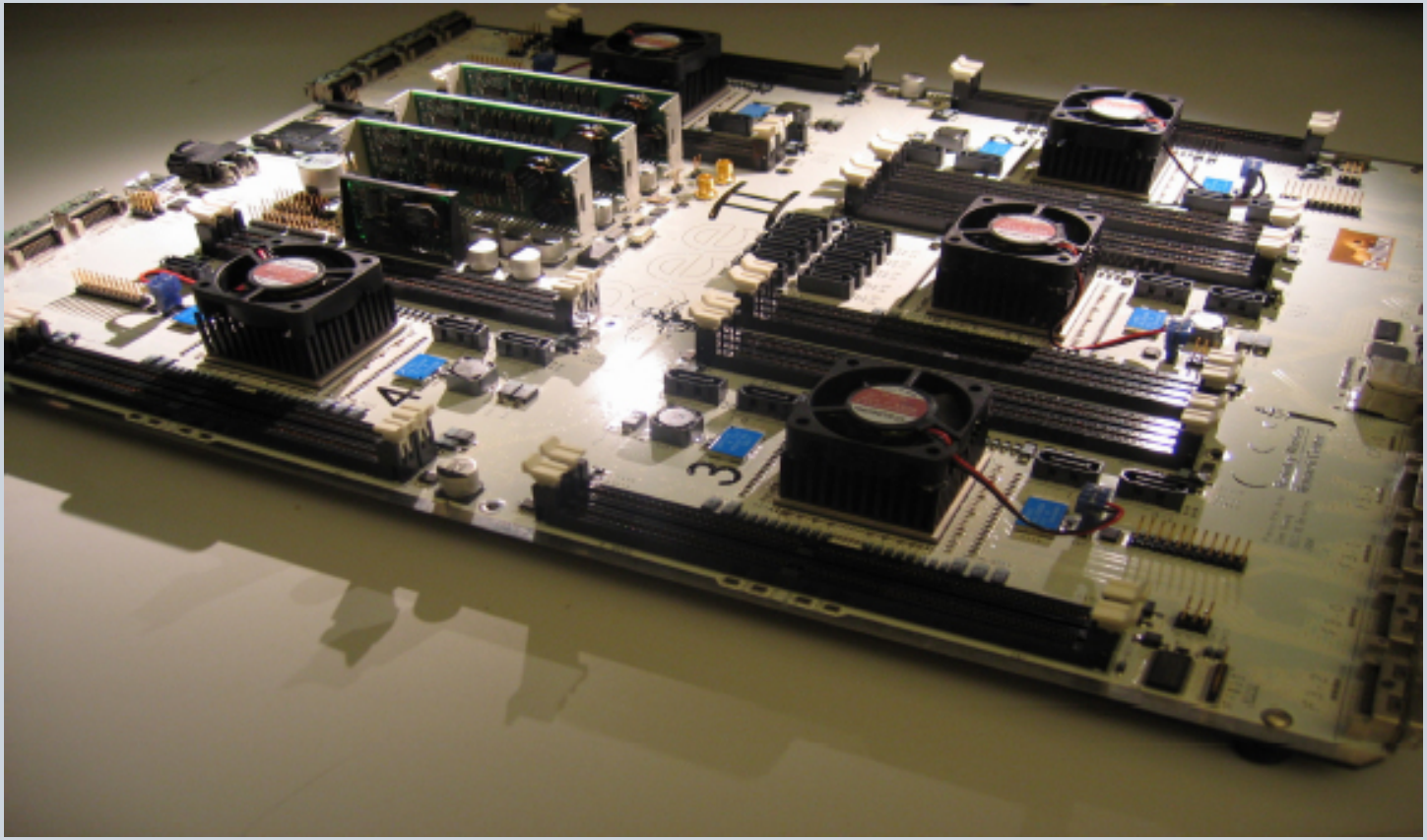
- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

CASPER

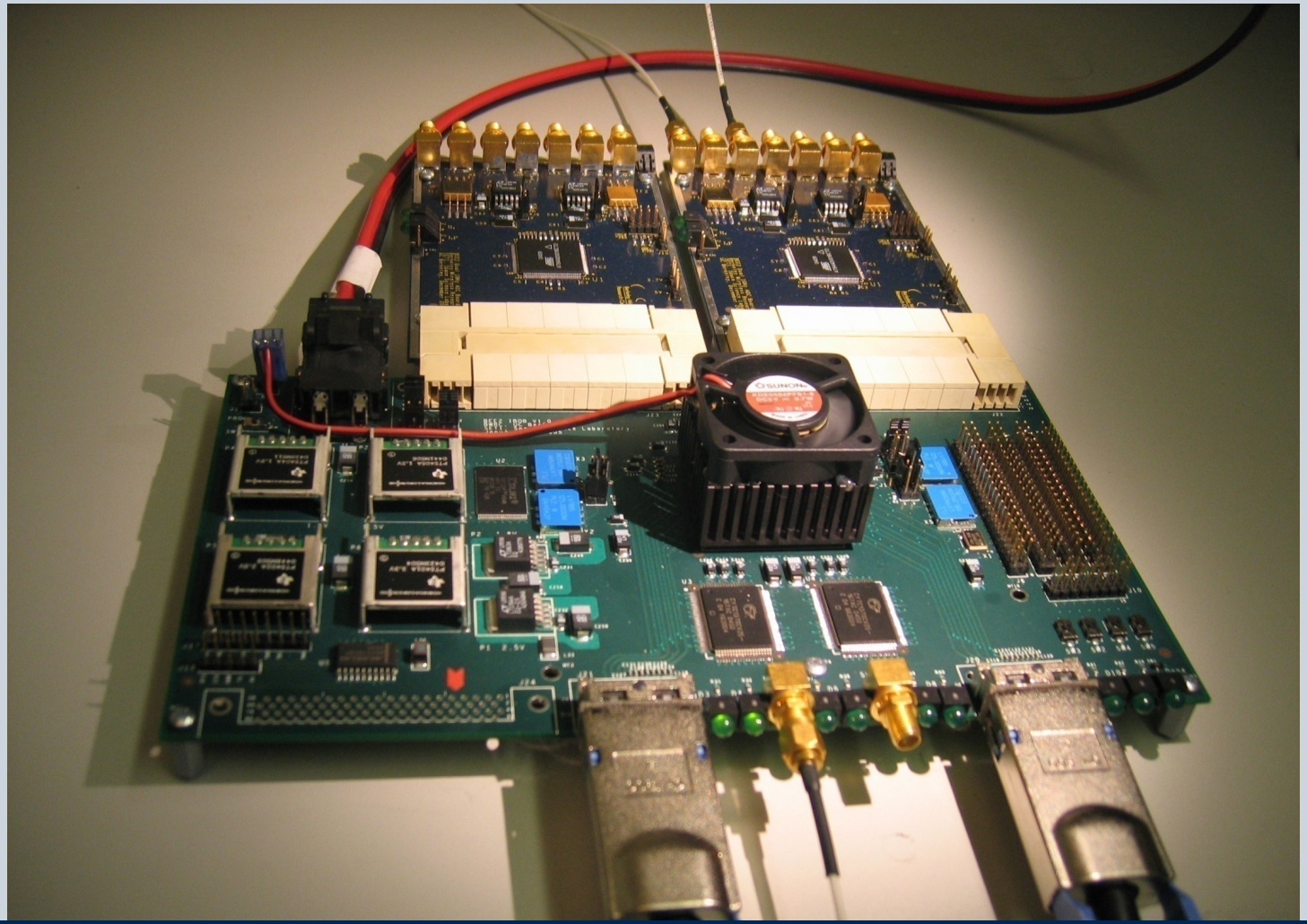
- Simplify
  - Leverage industry standards (eg, Ethernet for interconnect)
  - Small number of custom [FPGA] platforms
  - Optimize for ease of use (not ops/watt, ops/rack unit)
  - Low knowledge-barrier for users
- Re-use
  - General purpose hardware
  - General Purpose libraries
  - Modular, upgradable piecemeal
  - Flexible, scalable architectures

Outline

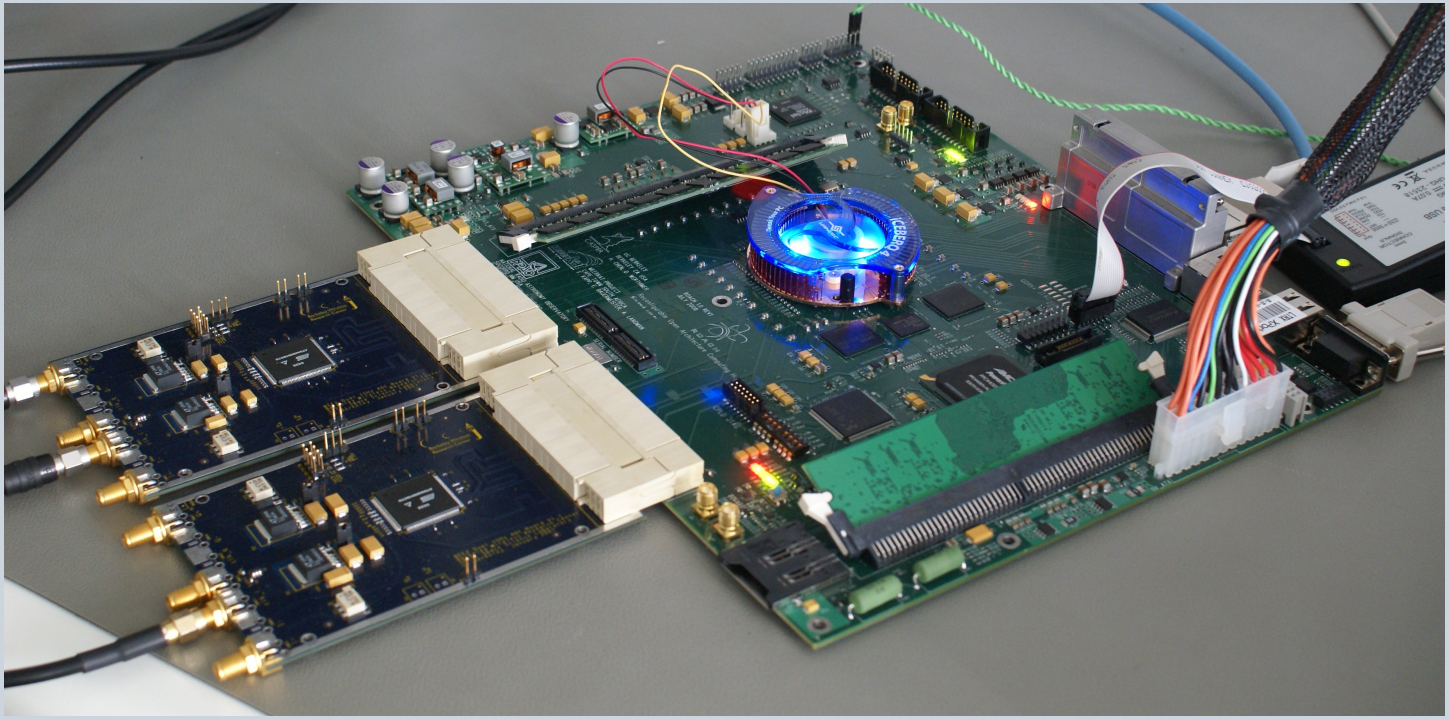
BEE2 (Virtex 2 Pro) 2005—



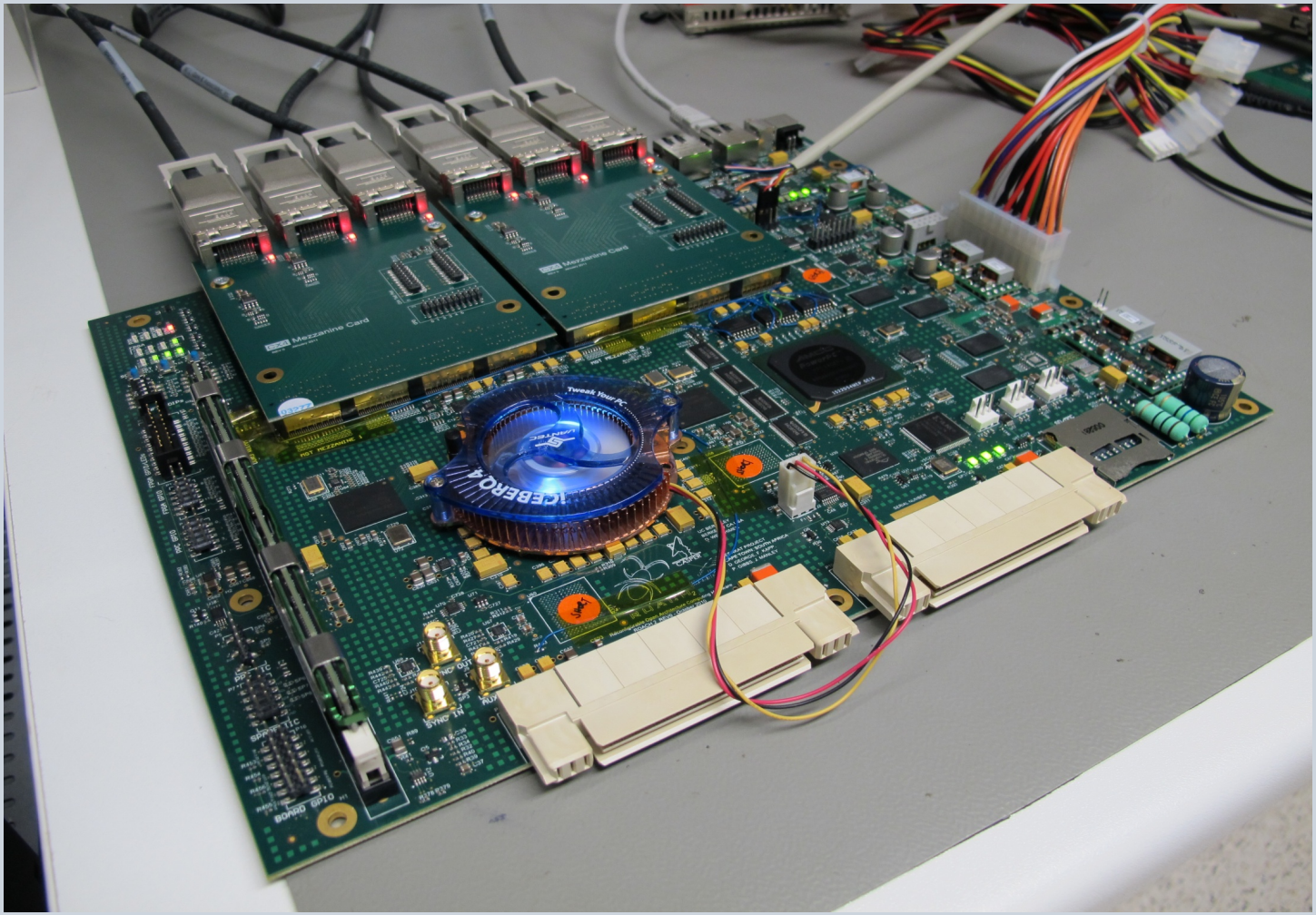
iBOB (Virtex 2 Pro) 2005—



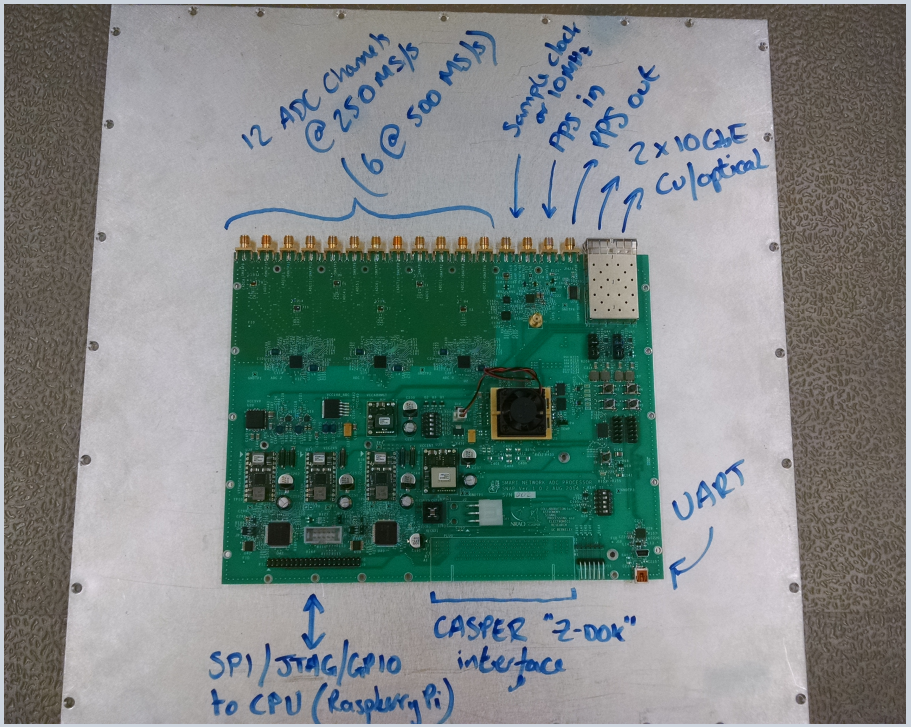
ROACH (Virtex 5 SX95T) 2009—



ROACH2 (Virtex 6 SX475T) 2010—

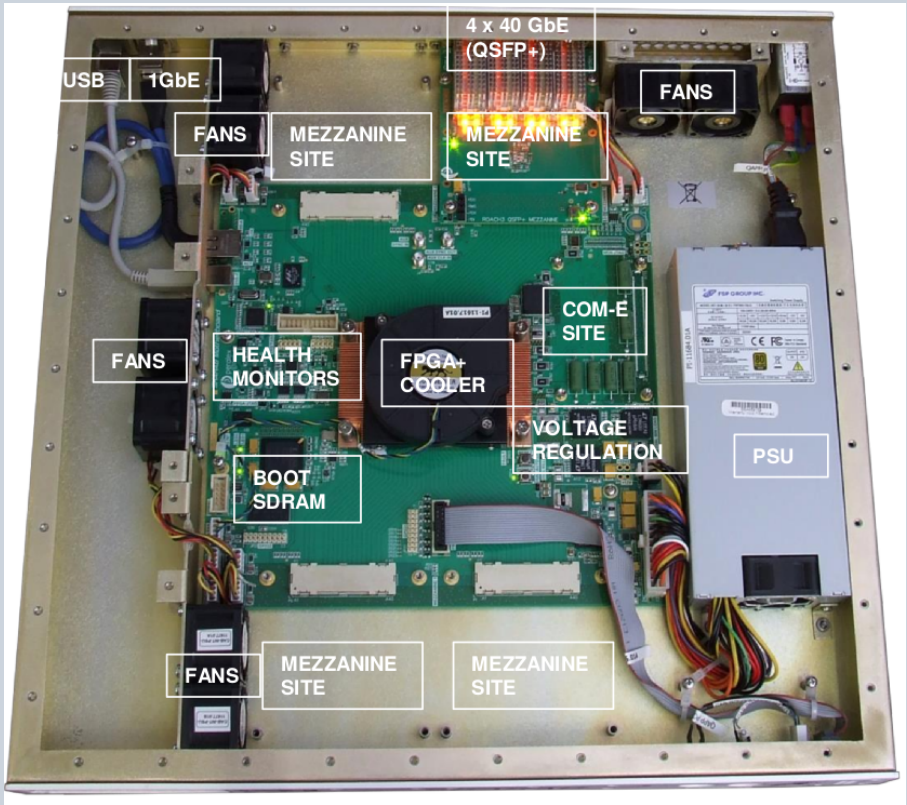


SNAP (Kintex 7 160T/325T/410T) 2016—



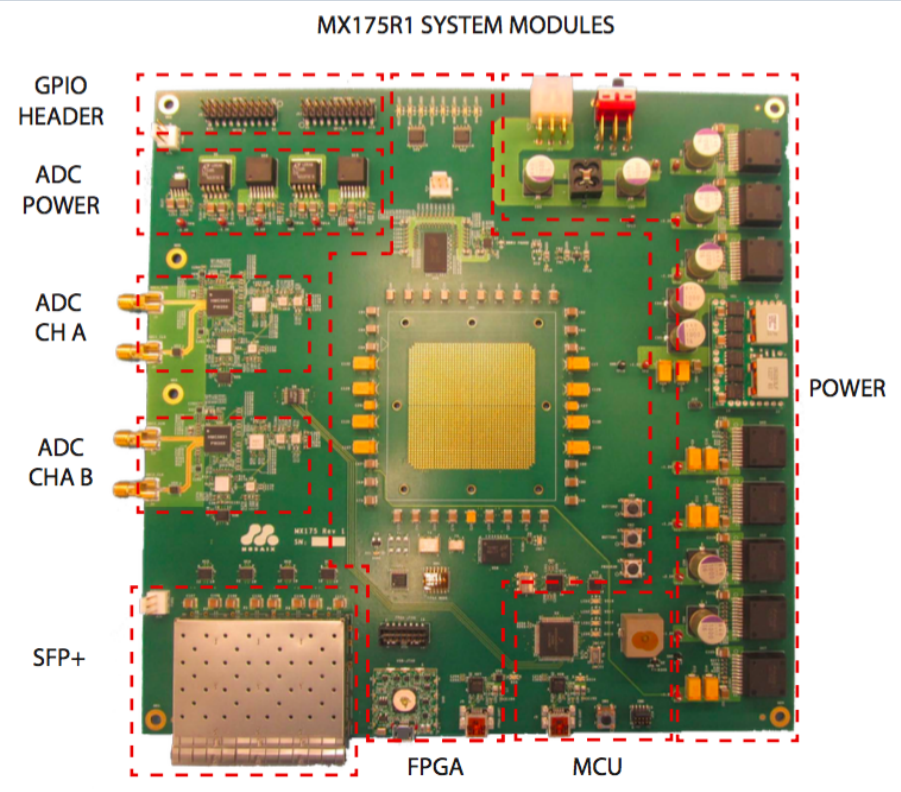
- 600-1540 DSPs
- 3 onboard HMCAD1511 digitizers
- 3x1 Gsps / 6x500 Msps / 12x250 Msps
- 1 x ZDOK
- 2 x 10 GbE IO
- approx. $3k

SKARAB (Virtex 7 690T) 2016—



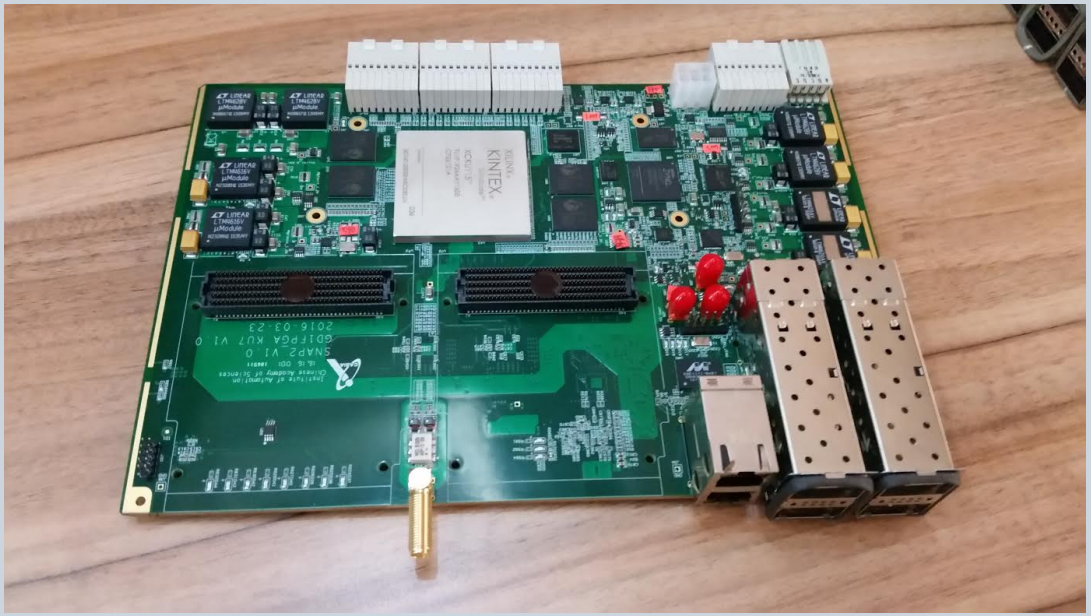
- 3600 DSP slices
- 4 mezzanine card sites
- HMC high-bandwidth memory
- up to 16 x 40 GbE interfaces

MX175 (Virtex 7 690T + 2xHMCAD5831 ADC) 2016—



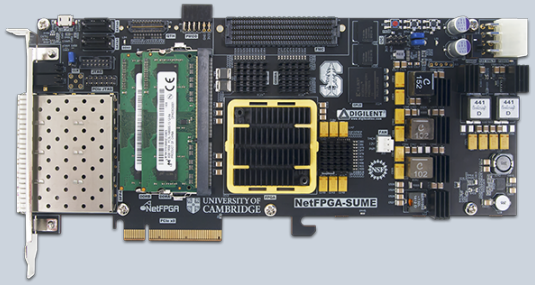
- 2 onboard 26 Gsps digitizers
- Same FPGA as SKARAB
- ? 4 x 40 GbE IO

SNAP2 (Kintex Ultrascale KU115) 2017—



- 160 Gb/s IO
- Expansion card up to $\approx \infty$ Gb/s
- FMC interfaces – some ADCs in development
- FMC - ZDOK adapter
- 5520 DSP Slices
- Est. $15k

COTS options—

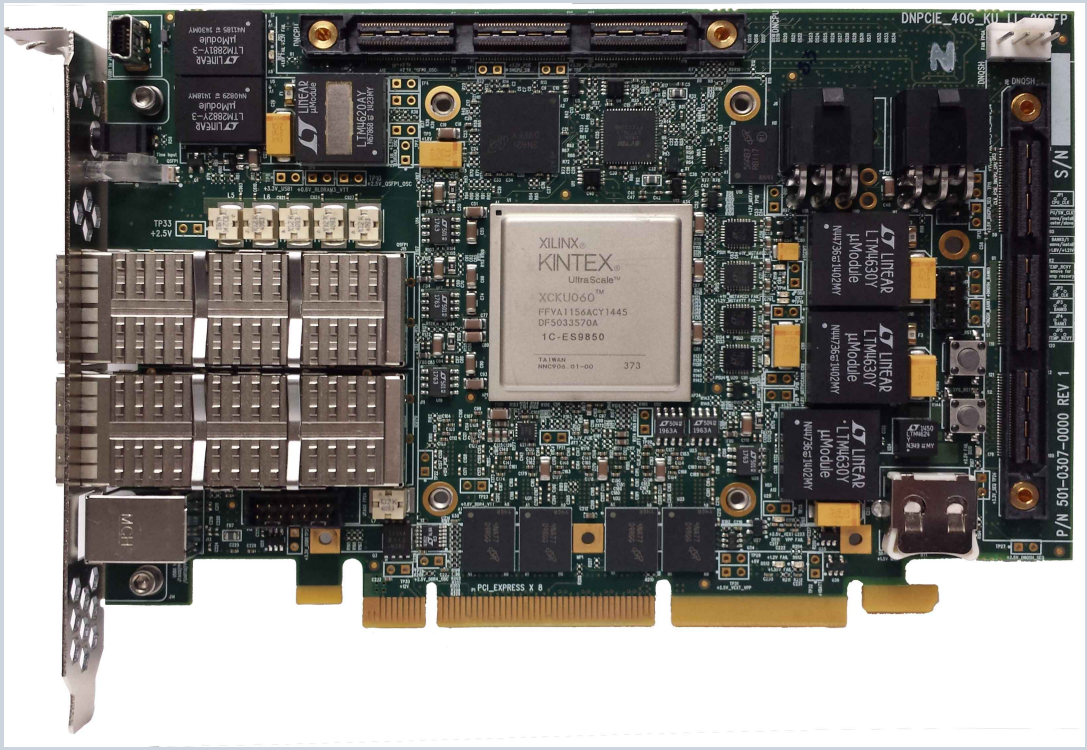


- NetFPGA-SUME
- Virtex 7 690T
- 3600 DSP Slices
- 4 x 10 GbE
- DDR3 & QDR
- $6995

- Alpha Data ADM-PCIE-7V3
- Virtex 7 690T
- 3600 DSP Slices
- 2 x 10 GbE
- 2 x 8 GB DDR3
- $3200

- Alpha Data ADM-PCIE-KU3
- Kintex U.S. KU60
- 2760 DSP slices
- 2 x 40 GbE
- 2 x 8 GB DDR3
- $2795

COTS options—

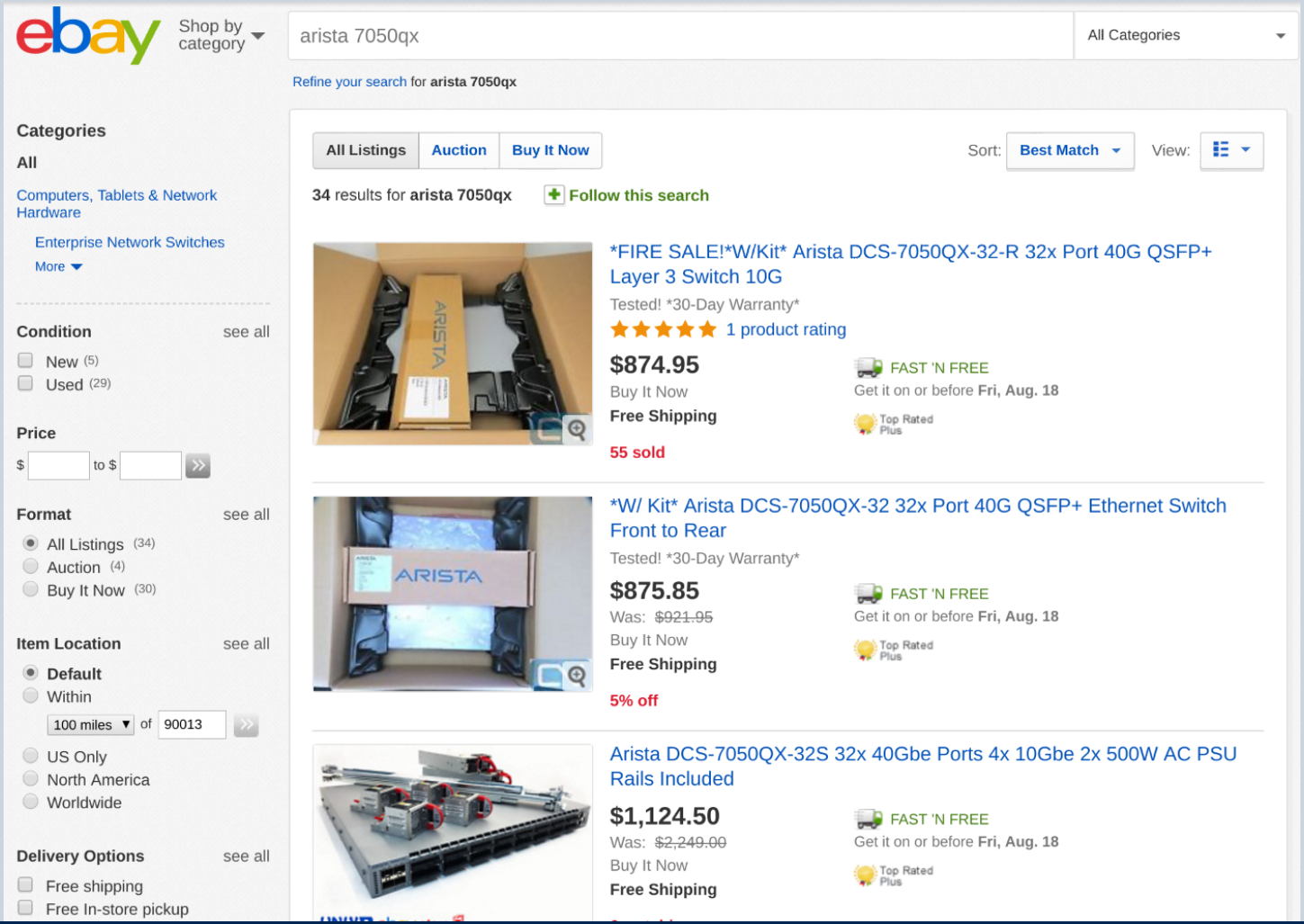


DINIGROUP: “Uncle of Godzilla’s Bad Hair Day”

GPUs

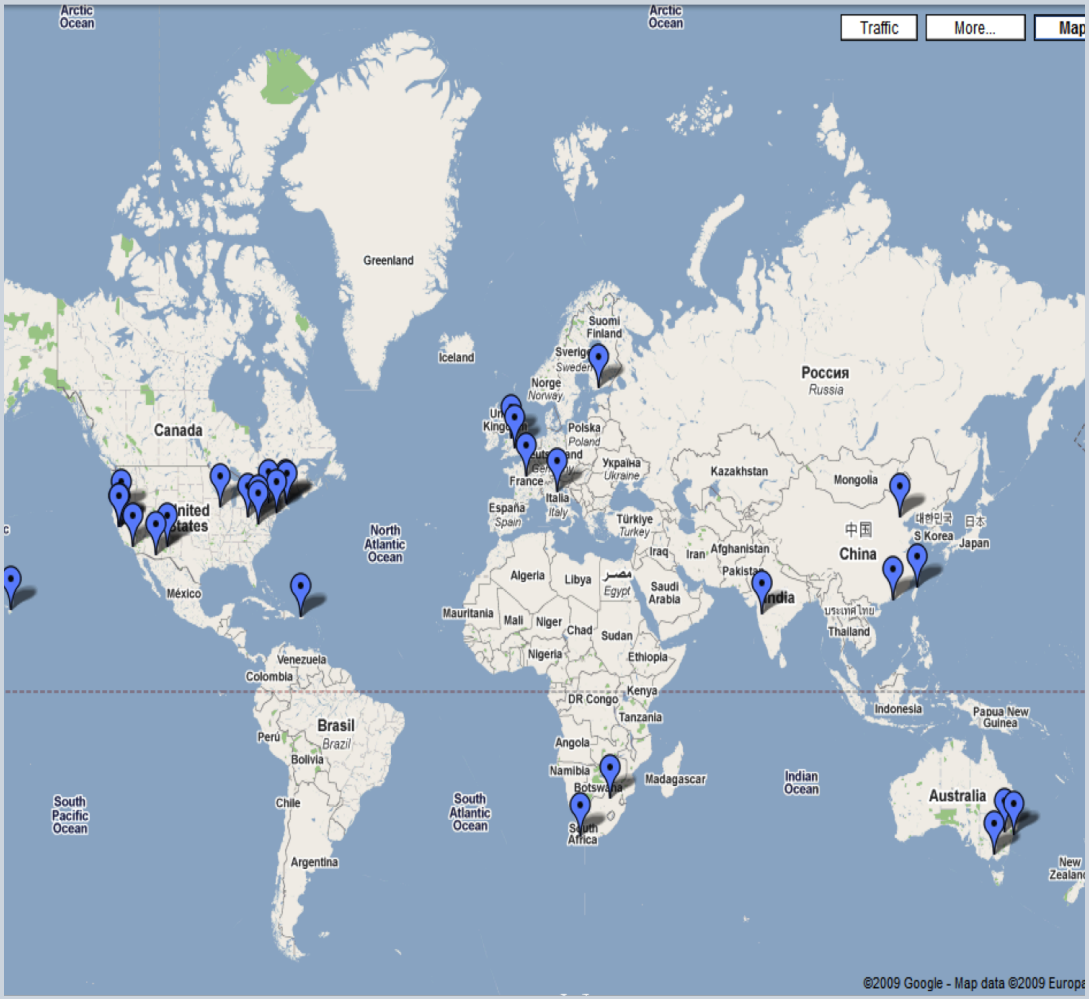
Fig. credit: Trusted Reviews

"Switches are free"



Outline

World Domination



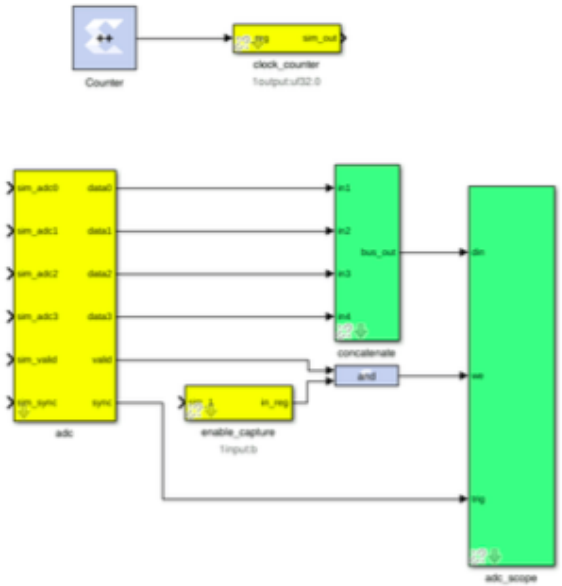
World Domination

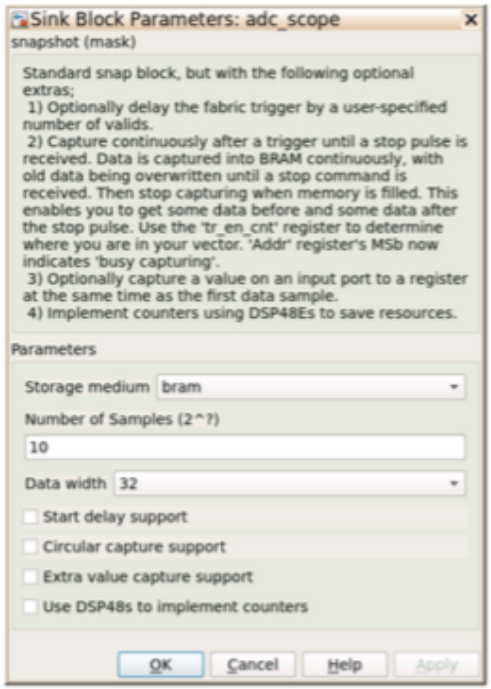
Spectrometers: Fly's Eye, GUPPI, CASPSR, BPSR, GAVRT, SERENDIP V.v, HiTREKS, Skynet, RATTY, cycSpec, C-BASS, HIPSR, KuPol, VEGAS, ALMA Phasing Project, Leuschner, R2DBE, DSN Transient Observatory, VGOS, AVN-Ghana, COMAP

World Domination

Correlators & Beamformers: KAT7, PAPER, ATA, LEDA, ARI, MAD, Medicina FFTT, GMRT, MITEoR, AMI, MeerKAT, FLAG, BIRALES, Starburst, AMiBA, EOVSA, SWARM, HERA

How? In a nutshell






```
from corr import katcp_wrapper

# connect to a CASPER board with
# hostname "roach_host"
fpga = katcp_wrapper.FpgaClient("roach_host")

# read a 32-bit register with a
# simulink name "counter"
val = fpga.read_int("counter")

# write a 32-bit register with
# simulink name "enable_capture"
fpga.write_int("enable_capture", 1)

# read 4096 bytes of FPGA memory
mem = fpga.read("adc_scope", 4096)
```

Outline

Not all roses

- Documentation is too sparse
- Simulink is intuitive, but it is slooooooow (and enraging).
- Severe version compatiblity headaches.
- Poor version control support.
- Diverging development at different institutions
- Unit tests
- Where are my new boards?!?!

The Future (at Berkeley)

- More co-ordination of developers (retreats in France?)
- More documentation. I actually promise.
- Faster paths to supporting new hardware
- Standardizing beyond the toolflow (pipelines, control software, etc.)
- In need of people to help.
- Please help.

Conclusions

- Getting everyone using [approximately] the same hardware / software has been a huge victory.
- There's loads of cheap hardware around (not just FPGAs)!
- COTs hardware is our focus in Berkeley.
- The CASPER toolflow is often liked in principle, but less so in practice. Hopefully we've made some steps to improve the lives of users and developers.
- Eager for fresh blood

Thanks

Thanks (and enjoy the workshop*)

*once developers have met you, it's much harder for them to ignore your emails.