“Ettus Research and its Research”

GRCon 2018

Martin Braun, Senior Git Wrangler
This is not a keynote, so I’ll skip the inspirational quotes.

If you want some good inspirational quotes, read/watch Hamming’s talk “You and your research”
  (Or go to a famous burrito chain)

Hamming asks: What are the most important problems in your domain? And are you working on them?

....well, are we?
What *are* the SDR problems?

- Do we even have any?
- Is SDR finished?
- Are SDR researchers going to be unemployed?

- ...I don’t think so.
What *are* the SDR problems?

- **Hardware:**
  - Which components will we use going forward?
  - Can we make hardware for all kinds of use cases?
  - How can we make hardware easy to use in various scenarios?

- **Frameworks & Software:**
  - Can frameworks keep up with the heterogeneity of hardware?
  - How do we keep up with increased bandwidth, lower latency requirements, ...? Can we even have one framework to rule them all?

- **Algorithms:**
  - Are our current DSP solutions still optimal in today’s SDR environment?
  - How do we optimize implementations of any given algorithm for power, latency, platform, resource utilization...

- **People:**
  - How do we train future engineers to work in the broad domain of SDR?
Example: Let’s build a phone

- It’s 2018; makers and hackers build all sorts of stuff, without having to have decades of experience
- So how would you build a pure SDR phone?
  - Which hardware do you get? You might need a lot of FPGA/RF/CPU resources. Are you OK using a rack-mountable device?
  - Would you use GNU Radio? (Spoiler: The answer is “no”)
  - What about OpenBTS, Osmocom, OAI, srsUE, etc….. they’re not generic frameworks!
  - Do we really know what the “best” implementation of a decoder/equalizer/synchronizer… is, given an arbitrary constraint space?
  - How many people do you intend to hire to do this?
What *are* the SDR problems?

**Hardware:**
- Which components will we use going forward?
- Can we make hardware for all kinds of use cases?
- How can we make hardware easy to use in various scenarios?

**Frameworks & Software:**
- Can frameworks keep up with the heterogeneity of hardware?
- How do we keep up with increased bandwidth, lower latency requirements, ...? Can we even have one framework to rule them all?

**Algorithms:**
- Are our current DSP solutions still optimal in today’s SDR environment?
- How do we optimize implementations of any given algorithm for power, latency, platform, resource utilization...

**People:**
- How do we train future engineers to work in the broad domain of SDR?
Why can’t we have a smartphone-sized device with high performance SDR capabilities?

Many vendors are providing new chips, SoCs, and platforms. Can they solve the SDR problems?

Hardware and Software are driving each other, too (TensorFlow, ML, ...)

Will we be still dealing with von-Neumann architectures in the future? Are we set up to do otherwise?

(Note: This is not an Ettus product announcement slide!)
Let’s accept the fact that we have to obey the rules of physics: More powerful devices will always be bigger.

Ettus philosophy: Cover a wide range of devices in the cost/power spectrum, provide single software API.
USRP N310

- A bit of E310, a bit of X310, and many other things!
- XC7Z100 FPGA for more RFNoC, AD9371 RFIC for higher channel density
- Embedded Linux for more control, RASM features
- SFP connectors for 10GigE access to FPGA, or slower streaming rates for embedded mode
To improve coverage of the power/SWaP space, we introduce the E320!

Bigger brother of the E310:

- XC7Z045 Zynq
- Single-board form factor for custom integration
- 10G/1G SFP for higher streaming rates off-device
- GPSDO
More channels, more USRPs, more problems?
Hopefully not: Let’s make our USRPs smarter!
Controlling multiple USRPs in a rack can be hard if there are no tools to help
Let the Embedded OS do some of the heavy lifting!
Remote updates, remote management, health monitoring, remote app deployment, ...
Hardware: White Rabbit

- How do you synchronize widely distributed SDRs?
- Don’t reinvent the wheel: CERN gave us White Rabbit
  - Sub-nanosecond synchronization across Ethernet
  - SyncE + IEEE 1588 PTP
- Open Source, so we can modify it, even upstream
- Currently available for the USRP N310/N300
Good frameworks & software APIs are the key enabler to efficient SDR development

Many open and proprietary frameworks and development environments available

We need a constructive and scientific approach at comparing and dissecting the various solutions

Many areas for research! Optimum resource allocation, scheduling strategies, ...
Frameworks: Language Support

- We don’t like to tie ourselves down:
  - We have C, C++, and Python APIs for UHD
  - Go to Brent’s talk to find out more about the Python API!
RFNoC: Native support for FPGA acceleration within GNU Radio and other frameworks/applications

- Fully meets the framework paradigm: High flexibility and high performance, some framework overhead
RFNoC remains the core architecture for all our USRPs going forward.

See RF loopback demo at the booth!

Also, see if you can still get a slot in our tutorials!

Next steps:
- Stabilize the APIs
- Flesh out software controls
Higher Bandwidths: DPDK

- In order to increase streaming bandwidth, there’s multiple angles of attack
- One is to improve the host-side load of the actual Ethernet transport handling
- DPDK will pull more network-driver tasks into userland to allow higher optimization
- Stable device support and Benchmark results to come!
Loose Ends

- Who will train the next generation of SDR engineers?
- Who will create the perfect algorithms, the optimal frameworks (or prove that we already have them)?
- Who will design the chips that drive future SDRs?

What are the big SDR solutions?

- Hardware:
  - Which components will we use going forward?
  - Can we make hardware for all kinds of use cases?
  - How can we make hardware easy to use in various scenarios?
- Frameworks & Software:
  - Can frameworks keep up with the heterogeneity of hardware?
  - How do we keep up with increased bandwidth, lower latency requirements, …? Can we even have one framework to rule them all?
- Algorithms:
  - Are our current DSP solutions still optimal in today’s SDR environment?
  - How do we optimize implementations of any given algorithm for power, latency, platform, resource utilization…?
- People:
  - How do we train future engineers to work in the broad domain of SDR?
There are many interesting problems left in the SDR domain

Ettus Research is committed to doing our part by providing the best hardware and software we can

If the GRCon community can’t solve the rest, who can?