# An Introduction to Software Radio

(and a bit about GNU Radio & the USRP)

**Eric Blossom** eb@comsec.com

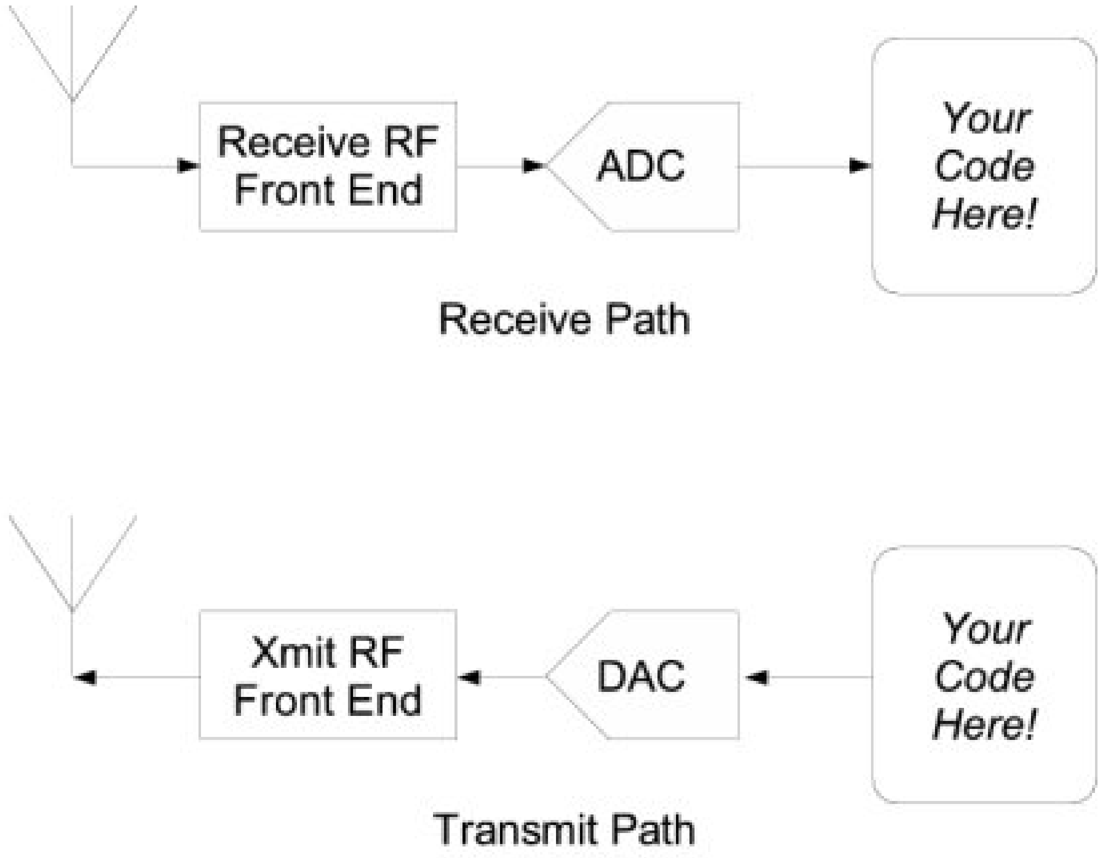
www.gnu.org/software/gnuradio
comsec.com/wiki

USENIX / Boston / June 3, 2006

## What's Software Radio?

- It's a technique for building wireless communication systems.
- Get the software as close to the antenna as you can.
- No modulation specific h/w
- Software defines the signals transmitted, sample by sample.
- Software demodulates/decodes the samples received.

# S/W Radio Block Diagram



## Pros...

- Extreme flexibility
- On the fly reconfiguration
- Can do multiple (different) things simultaneously
- Much quicker development cycle
- In-field upgrades are possible
- No soldering irons required...
  It's a simple matter of programming!

## Cons...

- Relatively high power consumption relative to fixed function ASICs.
- Higher cost if flexibility not important
- High symbol rate systems require FPGA or ASIC to support data rates
- A/D performance is limiting factor

## Why now?

- Low cost of compute cycles & memory
  - General Purpose Processor (GPP)
  - Digital Signal Processor (DSP)
  - Field Programmable Gate Array (FPGA)
- A/D's and D/A's are now “good enough”

## Where is it used today?

- Military
- Research: Academic & Industry
- Cellular basestations
- SIGINT

## Expected uses

- Public Safety interoperability
- Handsets (enabled by new DSPs)
- New personal communicators
- New kinds of networks

## Wireless networking

- Life beyond WLAN and broadcast
- Software radio provides flexibility
- All parts of the stack are hackable
- Take advantage of multicast nature of the medium
- Lots of research opportunities

## Still need some h/w

- Getting from RF to samples
- Getting from samples to RF

## RF / IF / samples

- Usually two steps:
  - RF to IF (downconversion)
  - Sample at IF
- Either direct conversion or superheterodyne
- Can sample at baseband or passband
  - Nyquist:  need > 2 * bandwidth of interest

## A/D performance

- Sample rate
  - kHz to GHz
- Resolution
  - 8 to 24 bits
- Spurious free dynamic range (SFDR)
  - maxes out at about 110 dB SFDR

## Analog vs Digital Processing

- Analog:
  - Tremendous dynamic range
  - Non-ideal behavior
  - Variation from part to part
  - Variation over temp & time
- Digital:
  - Perfectly reproducible behavior
  - Complex operations are easy

## Cognitive Radio

- S/W Radio + “AI”
- Observe the environment (RF, regulatory...)
- Evolve operating configuration
  - E.g., frequency, modulation, channel coding...
- Optimize what?

## S/W Radio Tools & Frameworks

- C / C++
- MATLAB / SIMULINK
- Software Communications Architecture (SCA)
  - Used in Joint Tactical Radio System (JTRS)
  - CORBA is the answer, what was the question?
- GNU Radio (Python and C++)

## Regulatory issues

- FCC: politicians, lawyers, economists, engineers
  - s/w radio is an enabling technology
  - Helps with “spectrum scarcity”
  - How to control / regulate?
- Some argue justification for FCC is gone
  - What is “interference”?
- Property vs Commons
  - What if each cow brought its own grass?

## And on to GNU Radio...

## What's GNU Radio?

- **Free software** toolkit for:
  - **Building** and **deploying software radios**
  - **Learning** about DSP and **communication systems**
  - **Creating new** kinds of **radios**, modulations, protocols, development environments...
- Licensed under **GPL**
- A **community** effort

## GNU Radio Architecture / Impl

- **Data flow** abstraction
  - Signal processing blocks and connections between them
- Event based overlay
  - **Message Queues** and **Messages**
- Hybrid **C++ / Python** system
- Typically run on general purpose processor
- “Hello World” example

## Hello World

```
#!/usr/bin/env python

from gnuradio import gr
from gnuradio import audio

class my_graph(gr.flow_graph):

    def __init__(self):
        gr.flow_graph.__init__(self)

        sample_rate = 48000
        ampl = 0.1

        src0 = gr.sig_source_f(sample_rate, gr.GR_SIN_WAVE, 350, ampl)
        src1 = gr.sig_source_f(sample_rate, gr.GR_SIN_WAVE, 440, ampl)
        dst = audio.sink(sample_rate)
        self.connect(src0, (dst, 0))
        self.connect(src1, (dst, 1))


if __name__ == '__main__':
    try:
        my_graph().run()
    except KeyboardInterrupt:
        pass
```

## Signal Processing Blocks

- **Input streams** and **output streams**
- I/O signature
  - **Type** of each stream is **specified**
  - Blocks specifies constraints on # of streams
- Relative i/o rates
  - **Fixed** 1:1, Fixed interp 1:N, Fixed decim N:1
  - **Variable**

## Who's using GNU Radio?

- Academic researchers
- Industry / DARPA researchers
- Various government research groups
- Hackers
- Hams
- Radio Astronomers
- Scanning Probe Microscopists

## Applications

- Transceivers
- Research in wireless networking
- Ad-hoc networks
- MIMO
- STAP / Adaptive beam forming
- Cognitive Radio
- Passive Radar (PCL)
- Geolocation
- SIGINT
- Conventional Amateur stuff
- Radio Astronomy

## Cognitive Radio

- Many efforts using GNU Radio
  - DARPA ACERT (BBN)
  - Virginia Tech
  - CMU
  - Rutgers WINLAB
- Often in combination with Click Modular Router

## Waveforms

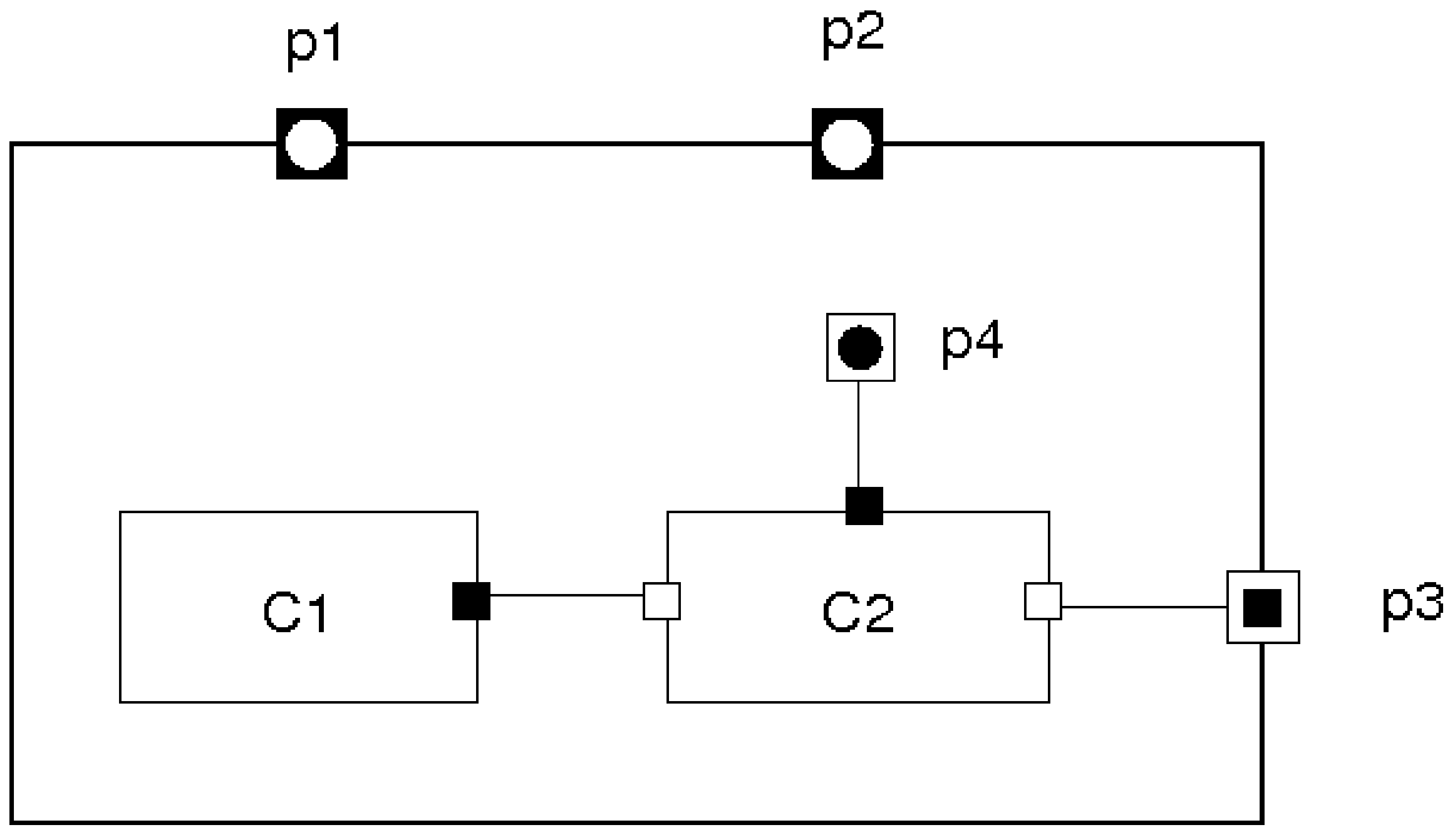
- Now:
  - AM, FM, SSB
  - ATSC VSB-8
  - FSK, GMSK, PSK
- Coming:
  - OFDM
  - Fast Freq Hopper
  - Direct Sequence

## Coming attractions...

## “Message Blocks”

- More natural support for packetized data
- Leverage existing code base
- Abstractions:
  - Blocks / Messages / Protocol classes / Ports
  - Connections between end points
- Data + metadata (packet annotation)
- Support for precise timing
- Hierarchical composition
- Nest “classic” GNU Radio within m-block

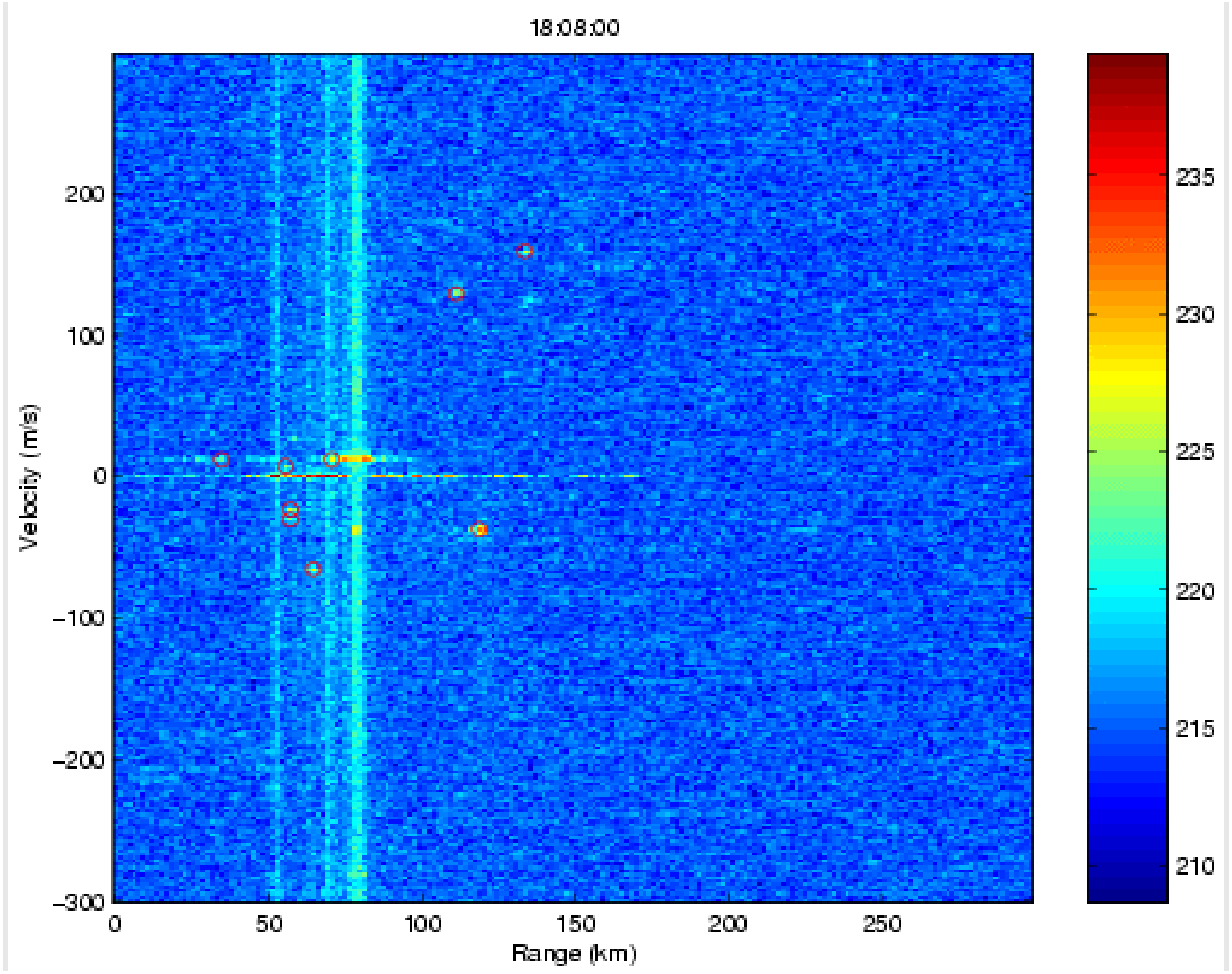
## “Message Blocks” (2)



## Passive Radar (PCL)

- Use existing transmitters (e.g., TV, Radio)
- Very high dynamic range front end
- 2 x 2 phased array
- TDOA, doppler, angle of arrival
- ESPRIT
- output: position, velocity, object class
- Superresolution techniques

## Existence proof!



## The USRP

- Why?

## Sound Cards, etc

- Relatively low sampling rate
  - 48 kHz or 96 kHz,  16 or 24 bits
- Good for audio input and output
- Can be used with narrow and low IF
- Examples
  - Narrow band HF (SDR 1000)
  - “Digital Radio Mundial”

## Wide Band I/O

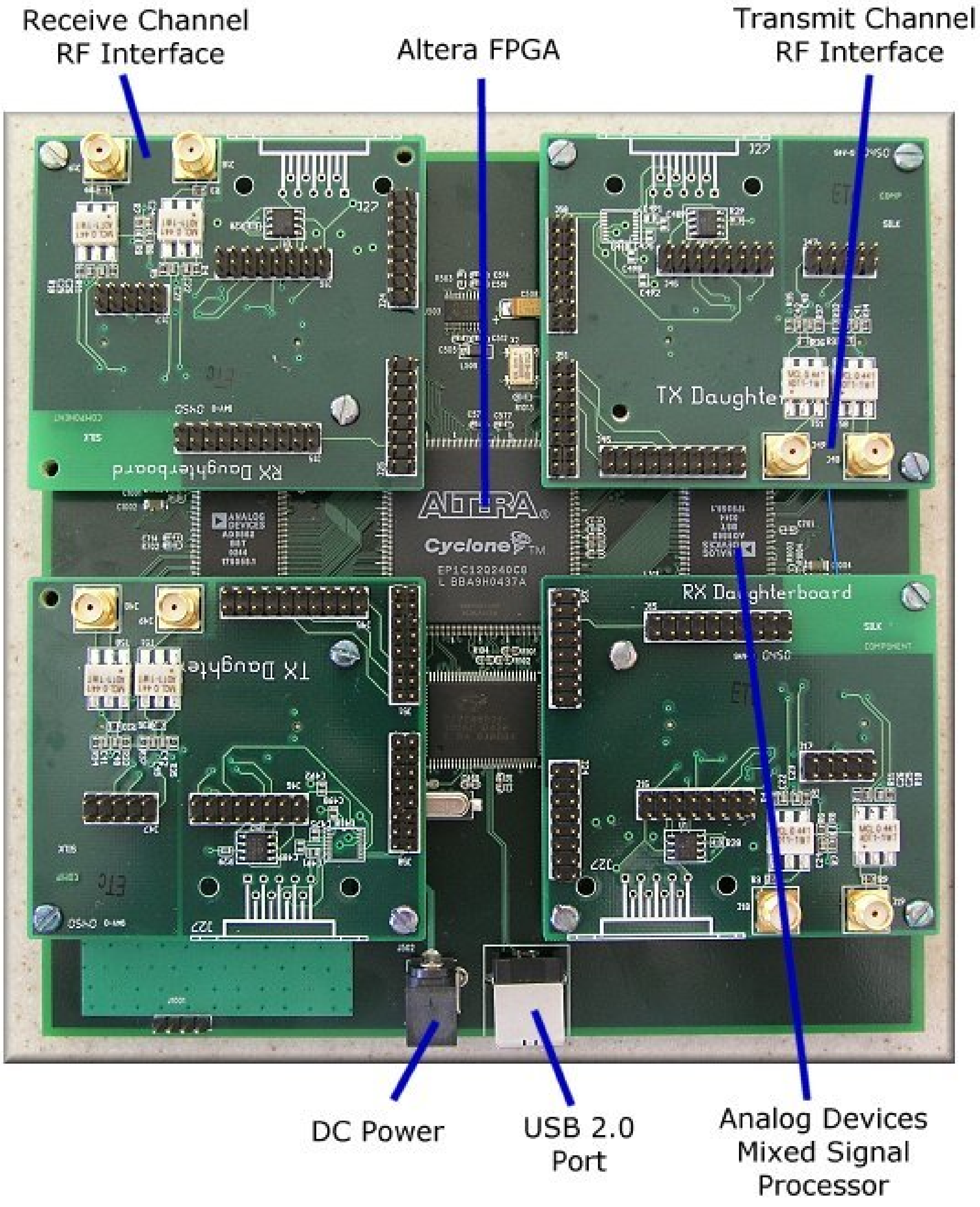
- PCI A/D and D/A Cards
  - Good Bus Bandwidth
  - Expensive to Very Expensive ($1k - $10k)
  - Still need RF Front End

## VXI / cPCI / ...

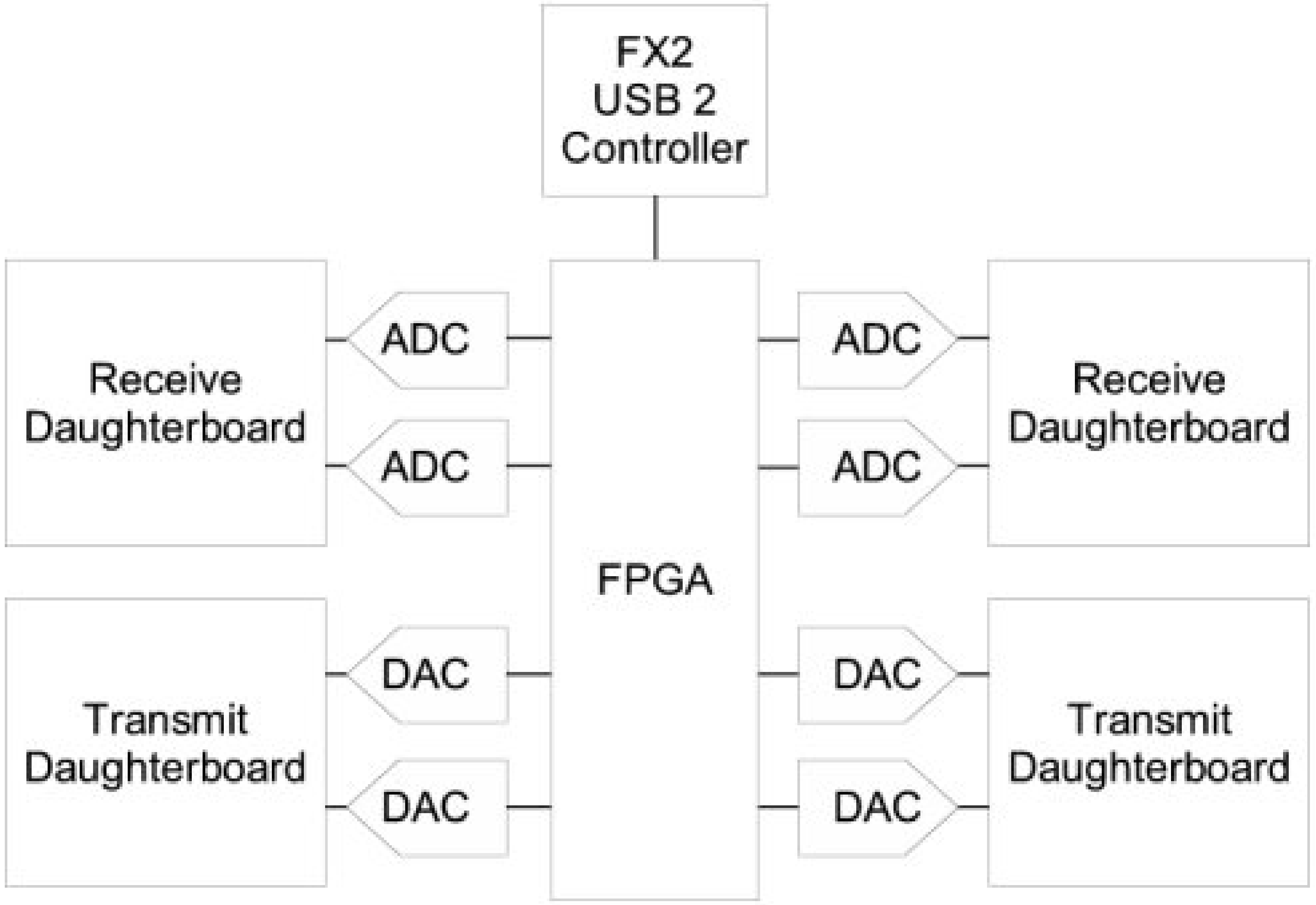
- Card cages full of cards
  - RF Front Ends
  - Digital Receiver / Transmitter
    - Typically A/D, D/A + FPGA or ASIC
  - FPGA / DSP / GPP
- High speed interconnect
- Lots of choices
- Typically very expensive.

## USRP

- 80% solution at 10% of the cost
- Low cost
- Small / portable
- Design is completely open
- Multiple coherent channels


Receive Channel RF Interface
Altera FPGA
Transmit Channel RF Interface
RX Daughterboard
TX Daughterboard
TX Daughterboard
RX Daughterboard
ALTERA
Cyclone
DC Power
USB 2.0 Port
Analog Devices Mixed Signal Processor

# USRP Block Diagram



## Available RF Daughterboards

- 400 MHz – 500 MHz transceiver
- 800 MHz – 1 GHz transceiver
- 2.4 – 2.5 GHz transceiver
- 50 MHz – 800 MHz receive only
- 800 MHz – 2.4 GHz receive only
- Basic Tx and Rx (baseband i/o)

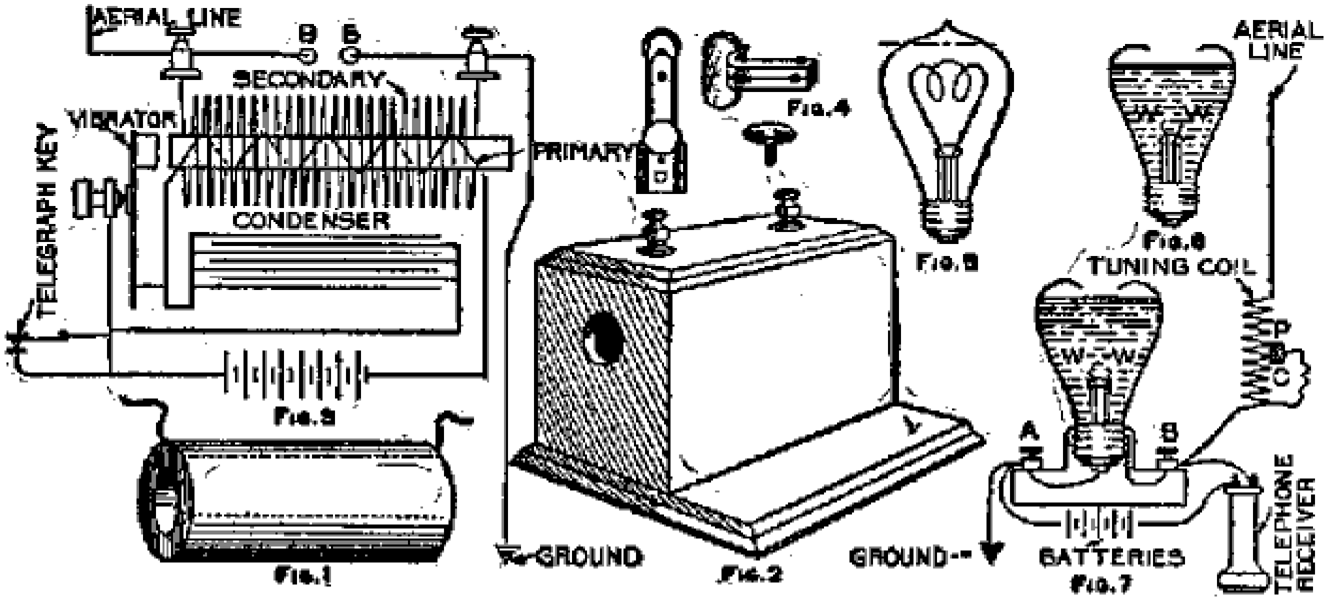
## emulab.net

- University of Utah networking testbed
- Expect 20 nodes around campus by end of year.  Uses USRP hardware with:
  - 2.4 GHz transceivers (?)
  - 400 MHz – 500 MHz transceivers (?)
  - 50 MHz – 800 MHz receive only

## Resources

- GNU Radio:
  - http://www.gnu.org/software/gnuradio
  - discuss-gnuradio mailing list
  - http://comsec.com/wiki
- USRP:
  - http://www.ettus.com

# Questions?


AERIAL LINE
B
S
SECONDARY
VIBRATOR
PRIMARY
TELEGRAPH KEY
CONDENSER
Fig. 3
Fig. 1
GROUND
Fig. 2
Fig. 4
Fig. 5
Fig. 6
AERIAL LINE
TUNING COIL
W W
A
B
P
GROUND
BATTERIES
Fig. 7
TELEPHONE RECEIVER