CRUSH: Cognitive Radio Universal Software Hardware

George F. Eichinger III
MIT Lincoln Laboratory
Lexington MA

Kaushik Chowdhury,
Miriam Leeser
Northeastern University
Boston, MA USA

This work is sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. The Opinions, interpretations conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

State of the Art in Software Defined Radio

+ Attach agile radio frequency front end to host computer to perform radio functions using software such as GNURadio or Matlab

+ More flexible than fixed hardware systems; all network layers can be implemented and modified in software

- Software adds latency and decreases system speed; data must be transferred between host and front end, not designed for real time operation

CRUSH is real-time capable and moves processing closer to the RF receiver/transmitter

Goals of CRUSH

+ Use FPGAs in SDR front ends, treating the hardware description language (HDL) as “software”

+ Process data close to the receiver/transmitter. Remove latency.

+ Existing SDR platforms (Ettus USRP, WARP from Rice University) have onboard, user modifiable FPGAs
  - Modifying existing HDL can be complex
  - RF components and FPGA integrated on a single board
- FPGAs and RF front ends are released on different schedules

CRUSH decouples fast evolving FPGA hardware from the RF front end

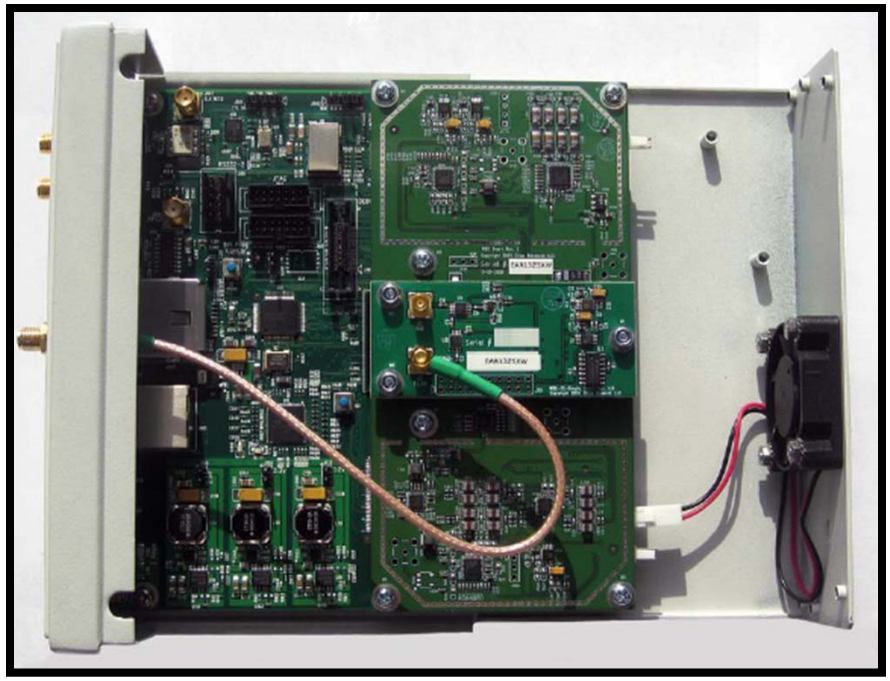
Outline

- SDR overview and motivation
- Hardware Platform
  - Cognitive Radio Universal Software Hardware (CRUSH)
- Cognitive radio overview
- Application: Spectrum sensing
- Results
- Summary

CRUSH Components

- **Hardware**
  - Xilinx ML605 FPGA Development Board
  - Ettus Research USRP N210
  - Custom Interface Board
- **Hardware Description Language (HDL) Framework**
  - USRP HDL modified for CRUSH
  - ML605 HDL created from scratch for CRUSH
- **Software**
  - Discussed later in presentation

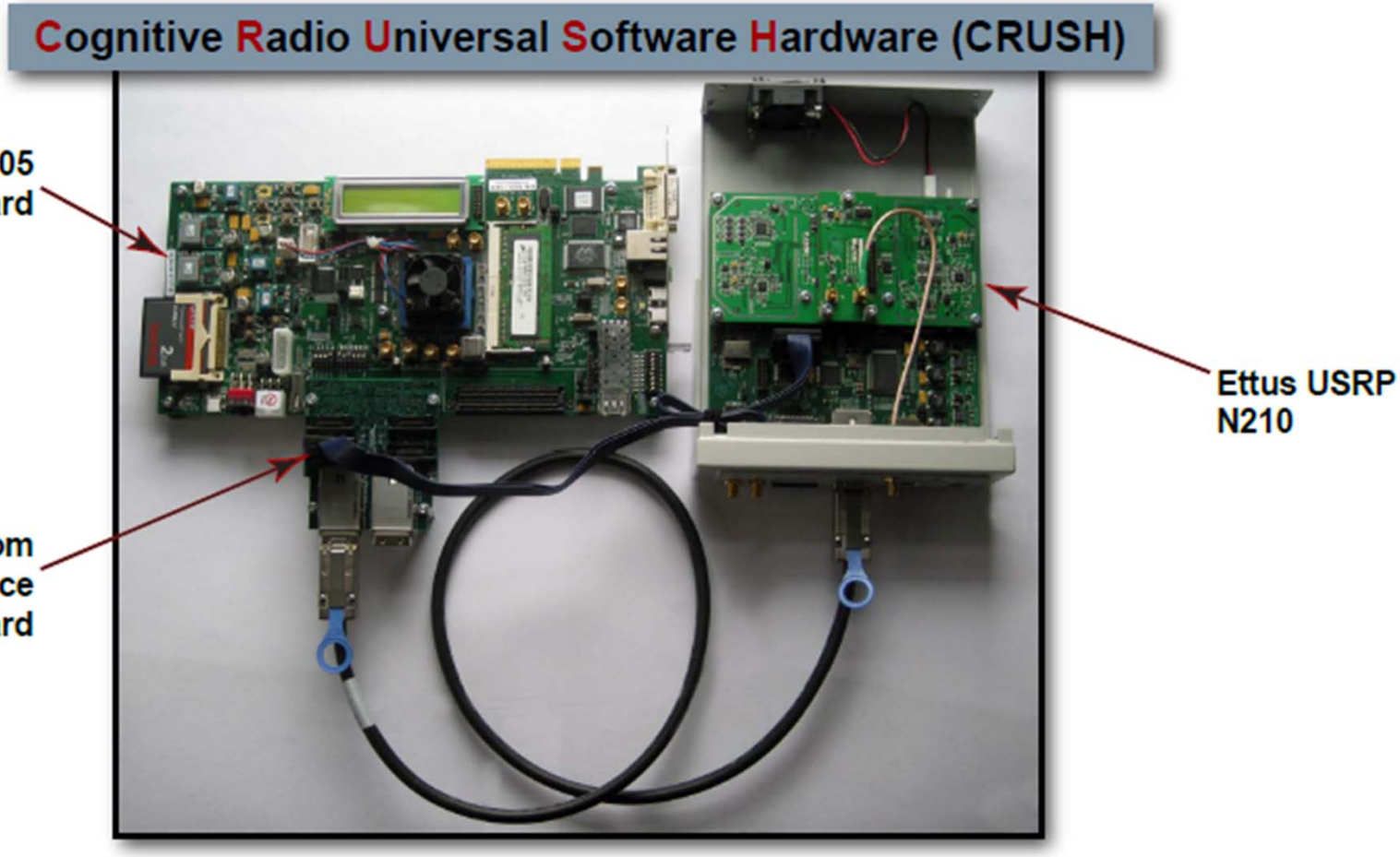
Ettus USRP N210



Universal Software Radio Peripheral

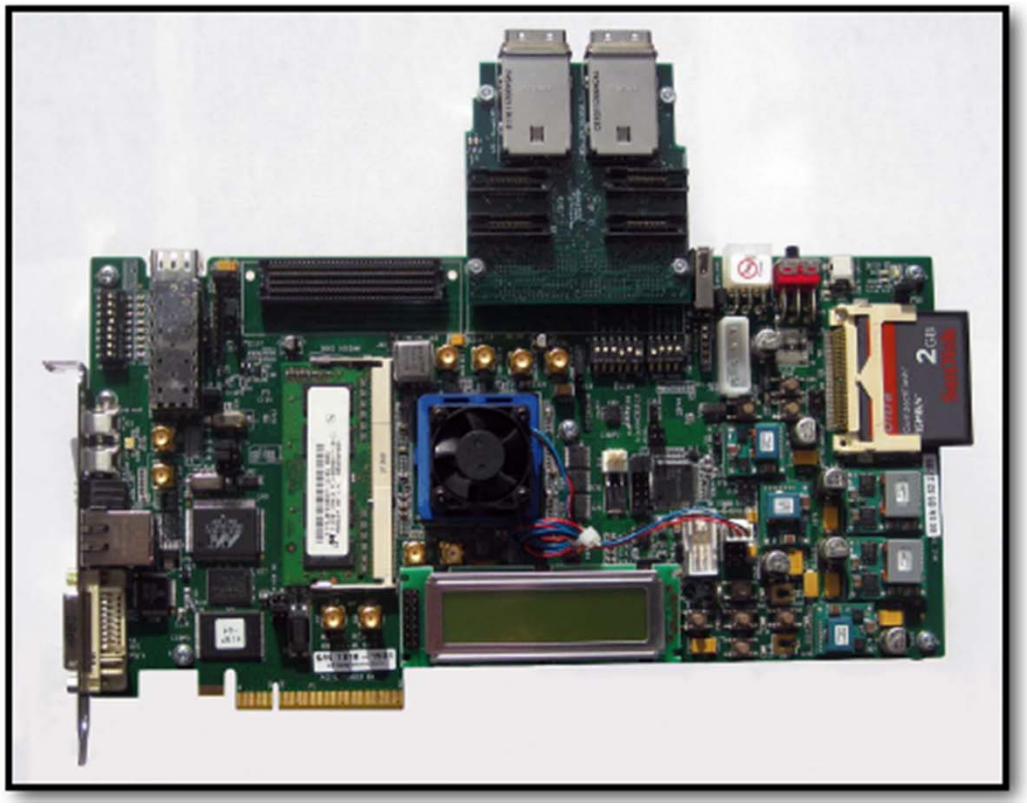
- Designed by Ettus Research
- Utilizes Xilinx Spartan 3A DSP series FPGA
- Popular academic SDR platform
- FPGA mostly filled with existing radio functionality
- USRP used as RF front end for SDR implemented in either GNUradio or Matlab
- CRUSH uses USRP N210 coupled with a high end FPGA board for SDR

Introducing CRUSH



- Standard software defined radio and FPGA development board
- Custom interface board to connect the two boards

Xilinx ML605 FPGA Development Board



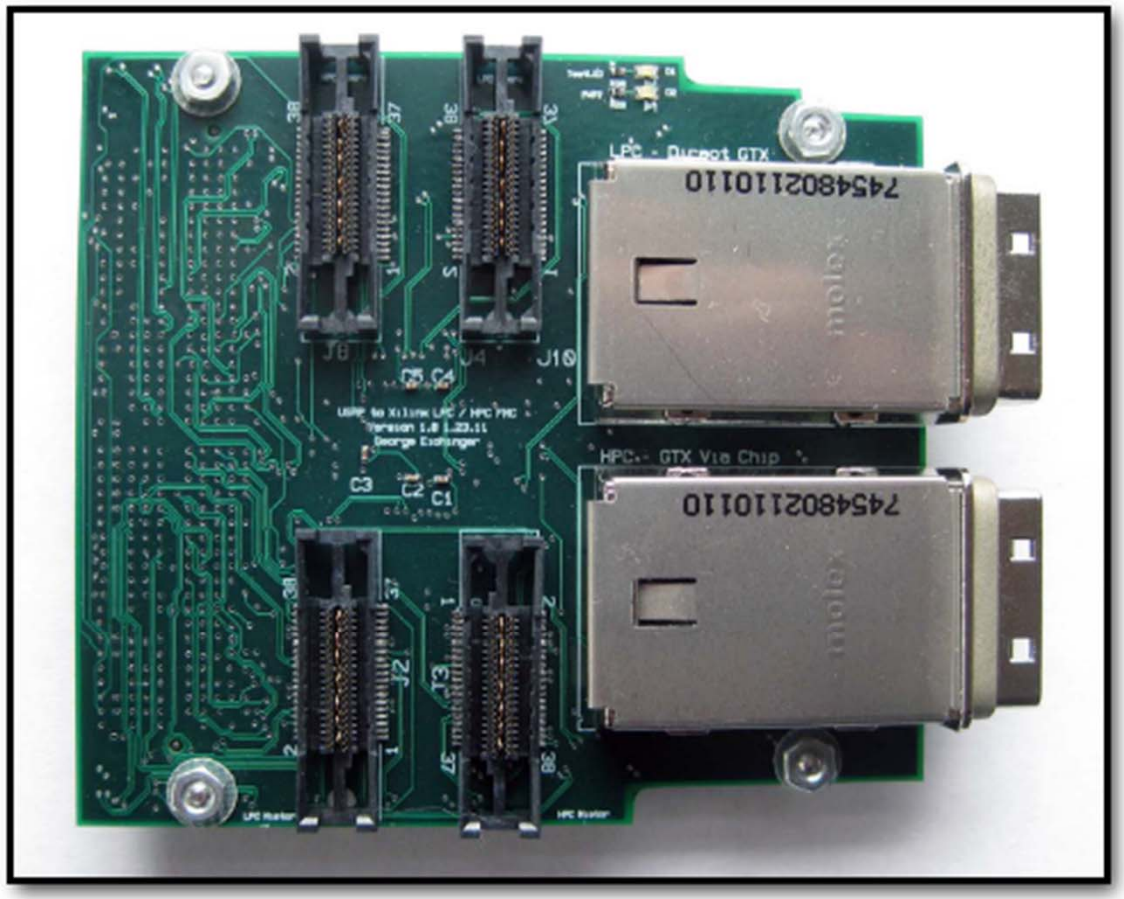
Specifications:

- Xilinx V6-LX240t FPGA
- 1* high pin count FMC
- 1* low pin count FMC
- 1* PCI express 8 ×
- 512 MB DDR3 RAM

Benefits:

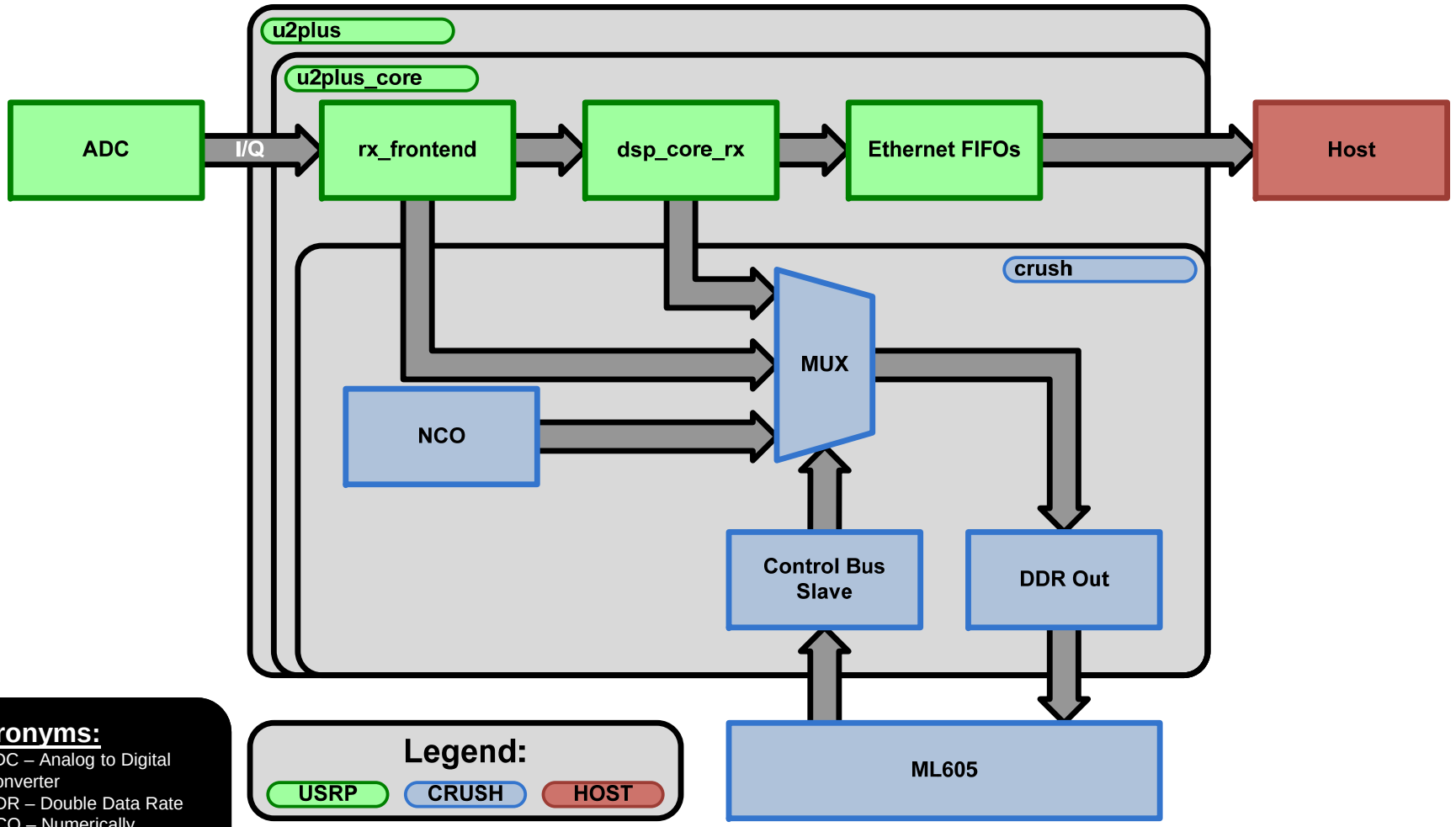
- Standard FPGA development board
- Ability to communicate at full ADC and DAC rate with the USRP
- Versatile external IO/memory
- Increases from USRP FPGA: 6.6 × more RAM, 4.5 × more LUTs, 2.5 × faster

Custom Interface Board



- 2 Mictor connectors
  - Allows for ML605 to communicate with two USRPs via the 34 pin parallel debug port
- 2 miniSAS posts
  - Transmit serial data with up to two USRPs via MIMO port
- 2 spare Mictor connectors
  - Spare FPGA IO
- FMC HPC/LPC interface
  - Fully compatible with the ML605
  - Can be used with and LPC interface like the SP605 with just one USRP
- Only custom part of CRUSH
- Allows full 100 MSPS IQ bidirectional datalink up to 800 MB/s

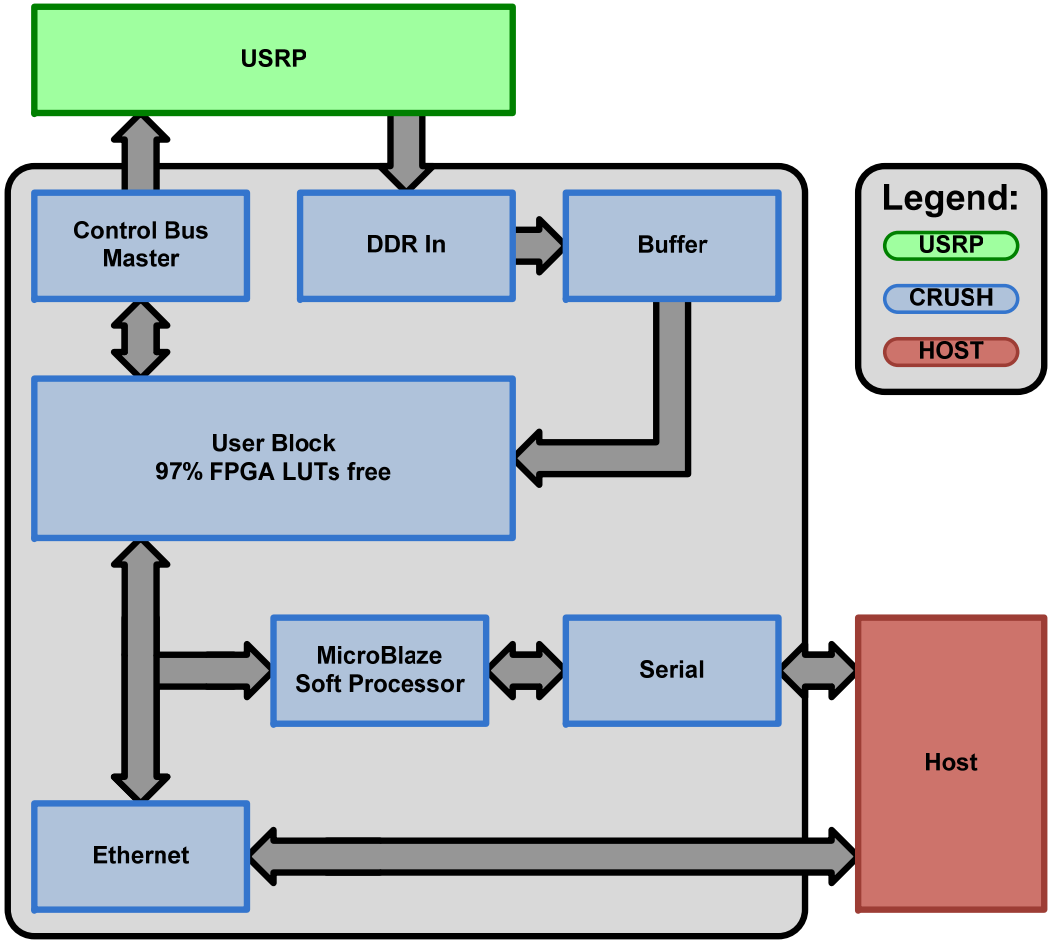
USRP HDL Framework



Acronyms:

- ADC – Analog to Digital Converter
- DDR – Double Data Rate
- NCO – Numerically Controlled Oscillator
- MUX – Multiplexer

ML605 HDL Framework



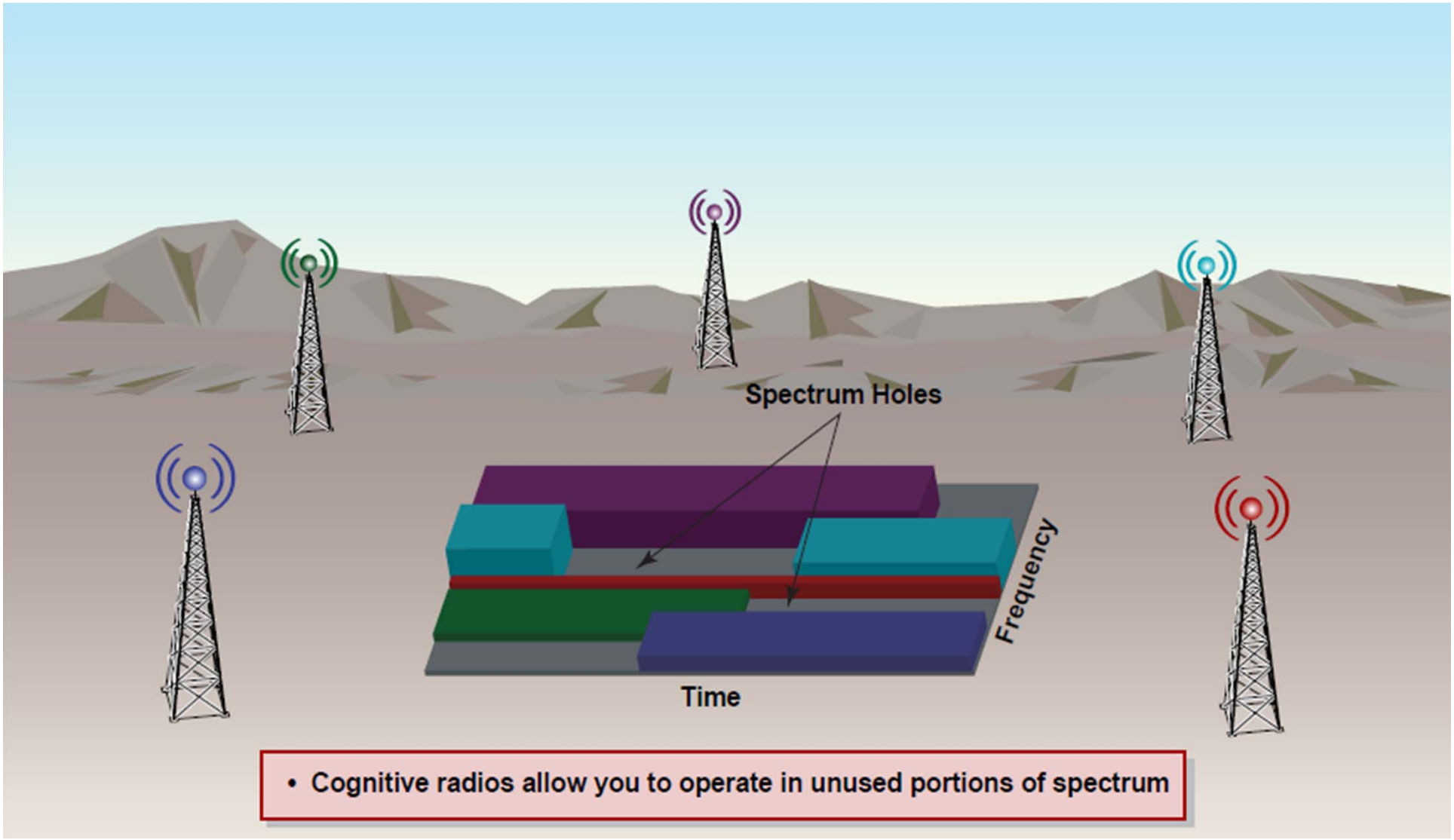
Acronyms:
- LUT – Look Up Table

Framework takes up just 3% of Logic, 97% Free for User Block

Outline

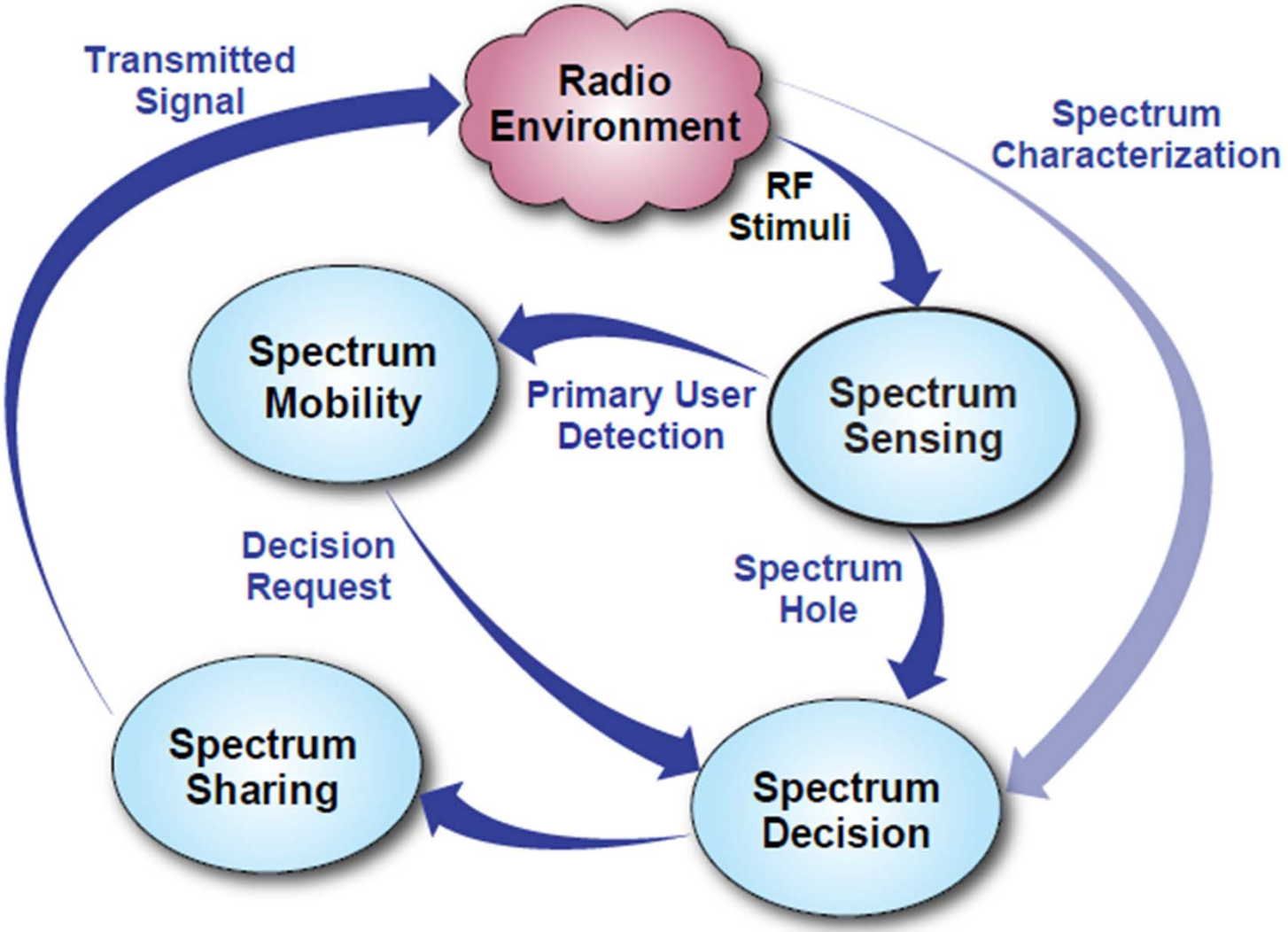
- SDR overview and motivation
- Hardware Platform
- Cognitive radio overview
- Application: Spectrum sensing
- Results
- Summary

Cognitive Radio Overview



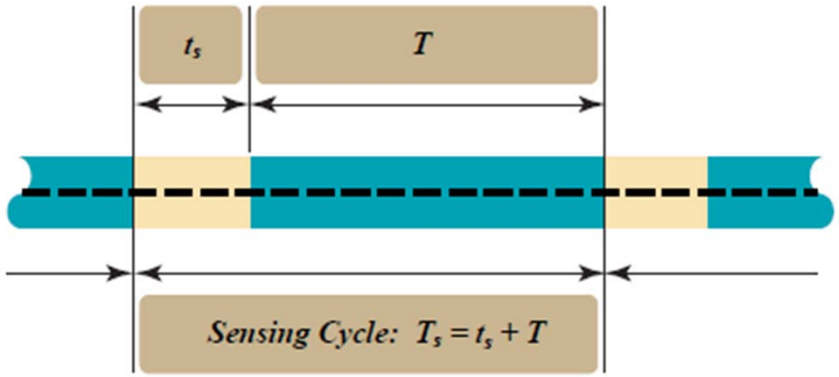
- Cognitive radios allow you to operate in unused portions of spectrum

Cognitive Cycle



CR / SDR Application: Spectrum Sensing

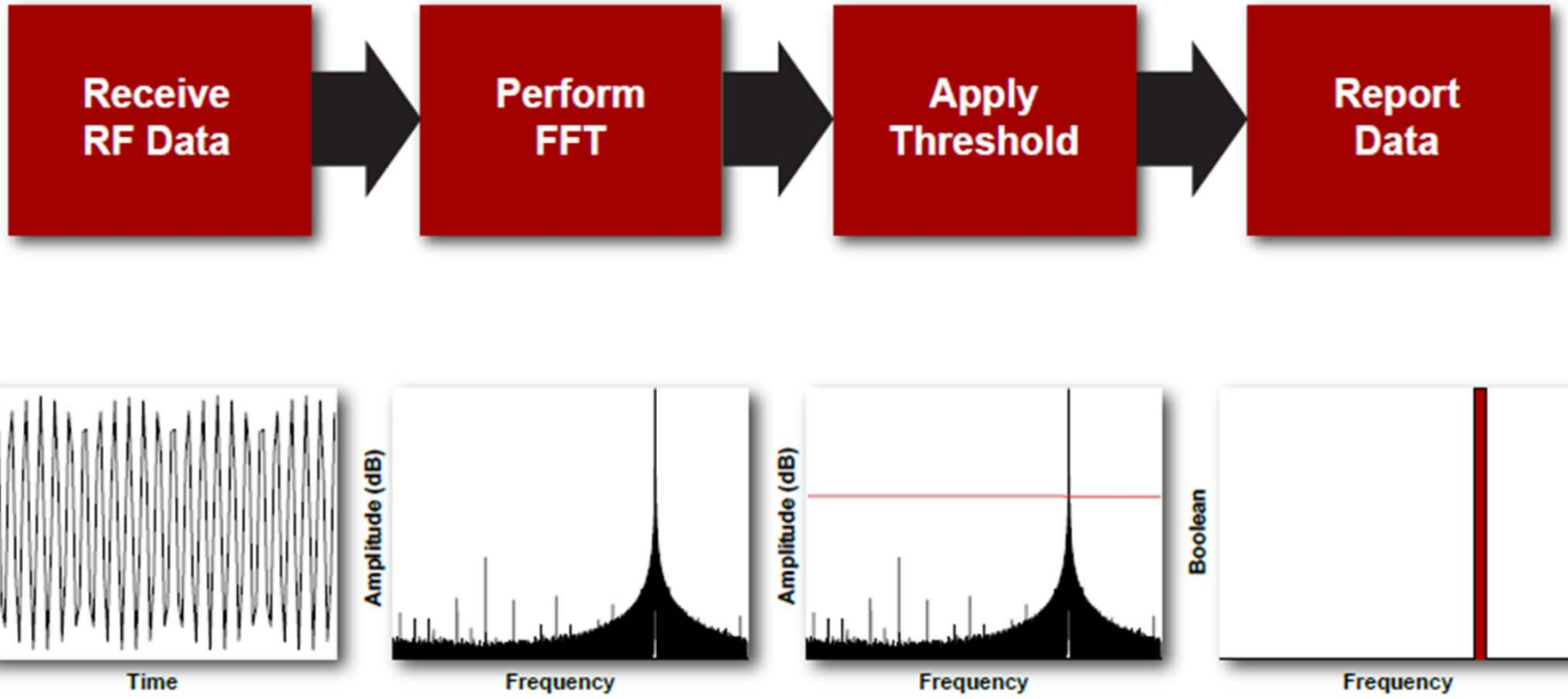
- Existing Software Defined Radio (SDR) systems take too long to perform spectrum sensing
- Software spectrum sensing involves transmitting data to and from the host computer which adds latency and processing time
- Moving spectrum sensing closer to the receiver reduces latency and makes real time spectrum sensing feasible



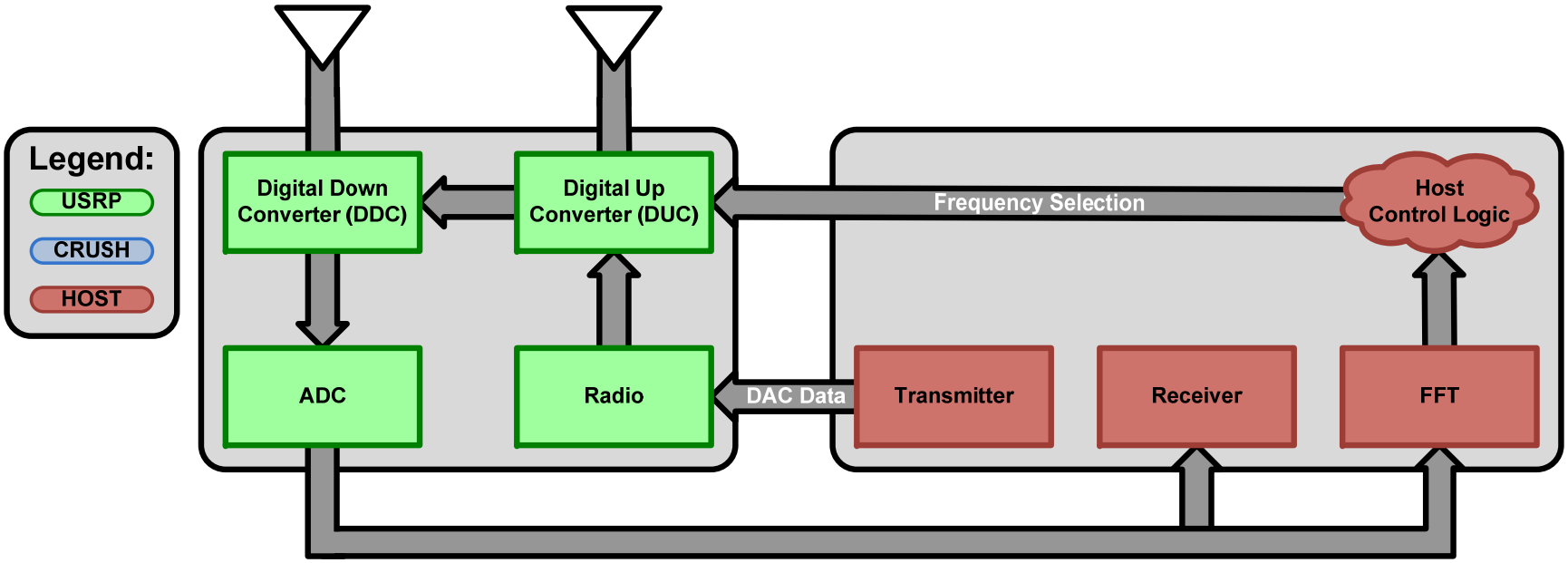
Outline

- SDR overview and motivation
- Hardware Platform
- Cognitive radio overview
- Application: Spectrum sensing
- Results
- Summary

Spectrum Sensing – Energy Detection

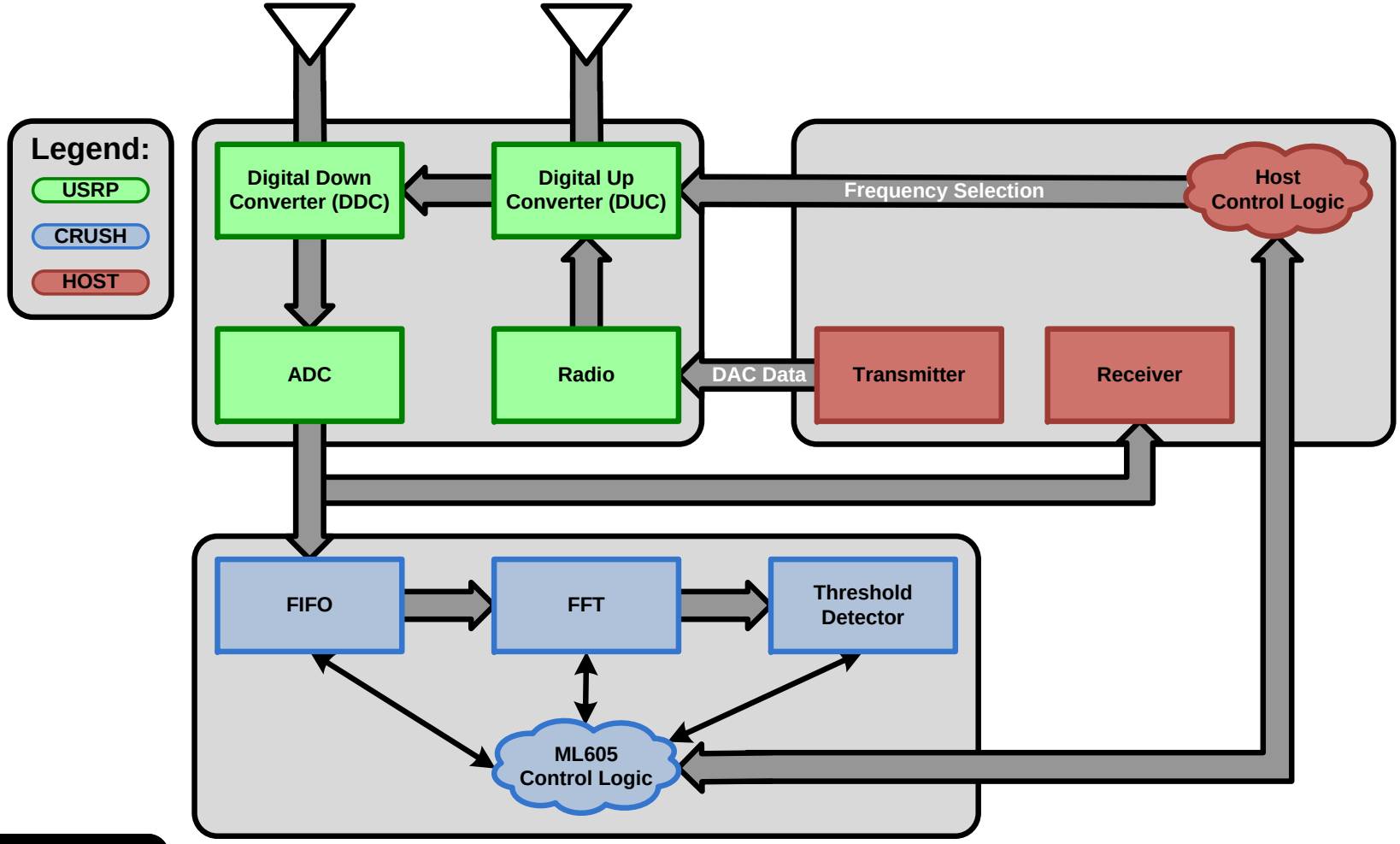


System Diagram – Without CRUSH



- All processing occurs on the host
- No real-time guarantee

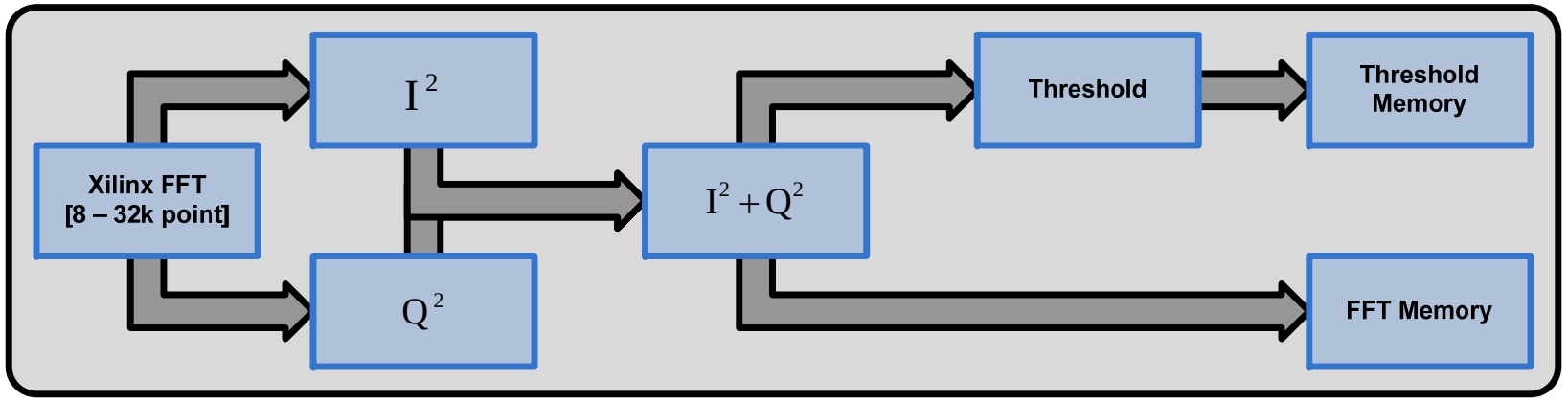
System Diagram – With CRUSH



Acronyms:

- FIFO – First In First Out
- FFT – Fast Fourier Transform

User Block



- Xilinx FFT
  - 8-4k point size
  - Streaming
  - Scalable
- I and Q Magnitude
- Thresholding
  - User specified
- Result Storage
  - FFT
  - Threshold

Advantages

- Ability to process received data in real time
  - FPGA: Parallel clock driven data bus
  - Host: Serial packetized data
- Higher throughput datalink
  - FPGA: 100 MHz 32 bit DDR interface (800 MB/s)
  - Host: Gigabit Ethernet (125 MB/s)
- Less processing load on the host
  - More time for high level policy / protocol execution
- Reconfigure hardware allows for parameters such as FFT size to be changed in real time
- New protocols with functionality partly residing on the host and partly on the radio are now possible

Outline

- SDR overview and motivation
- Hardware Platform
- Cognitive radio overview
- Application: Spectrum sensing
- Results
- Summary

Results Breakdown

- Test Setup
  - CRUSH Software
  - CRUSH Equipment Configuration
- Functional Verification
  - Prove the functionality of the FFT used for Spectrum Sensing
  - Visually show a comparison of USRP and CRUSH
- FFT Timing
  - Show modifications needed for precise timing
  - Look at a 256-point example in detail
  - Compare the time for completion of an FFT on both CRUSH and the USRP
- End to End CRUSH Timing
  - Show modifications needed for timing on the host
  - Look at the end to end timing of CRUSH measured by the host
- Real World Examples
  - CRUSH Matlab Demo
  - Free Space Spectrum Sensing

Test Setup - Software

USRP Hardware Driver (UHD) C++ Code:

+ Fast
+ Integrated in GNURadio
+ Ability to also control USRP
+ Fully configurable
- No graphical support
- Not user friendly

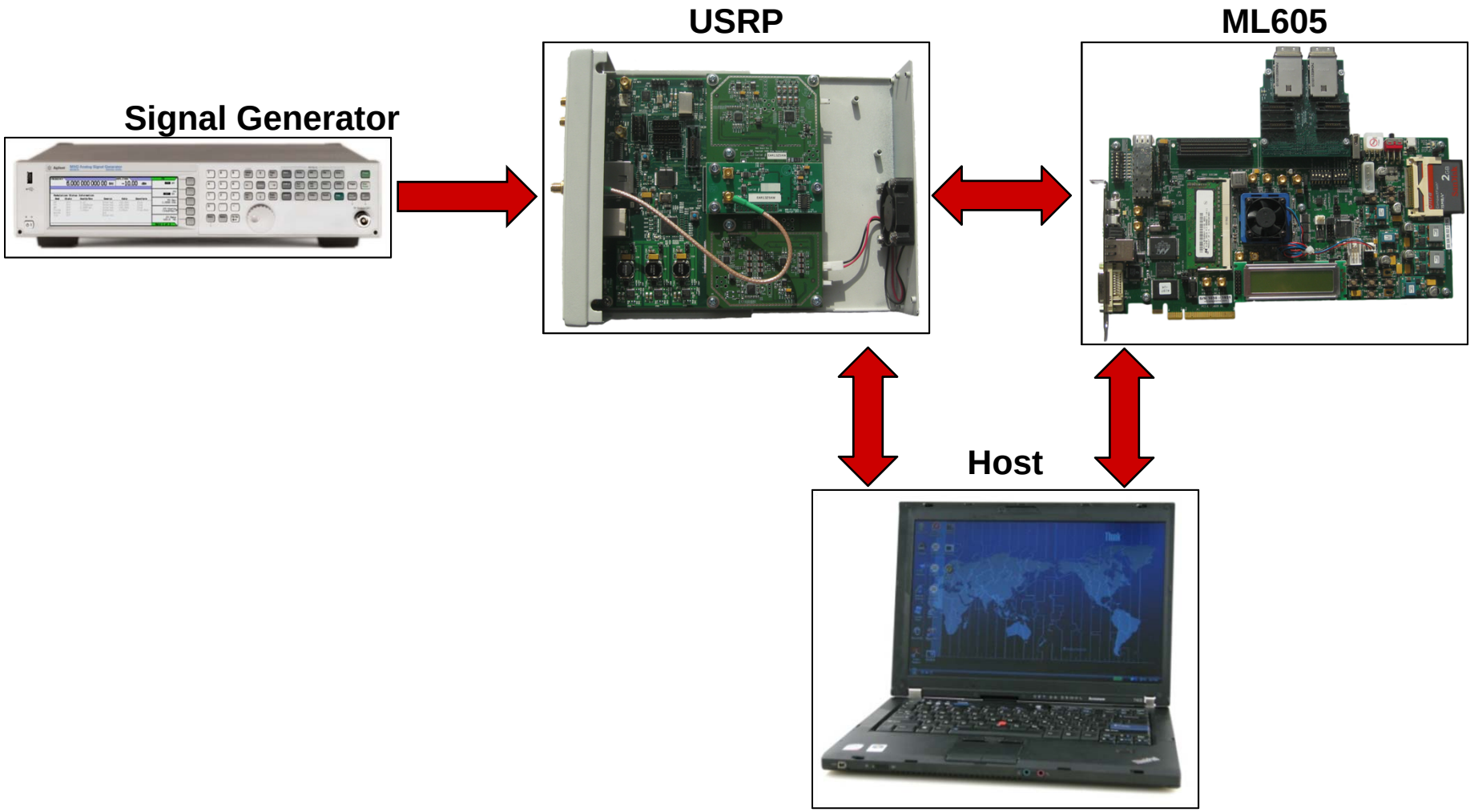
Matlab Spectrum Sensing Demo:

+ Quickly demonstrate CRUSH
+ Dynamically vary parameters
+ Manual or automatic updates
+ Graphs data for quick analysis
- Not real time (ms vs us)
- This demo not integrated into SDR

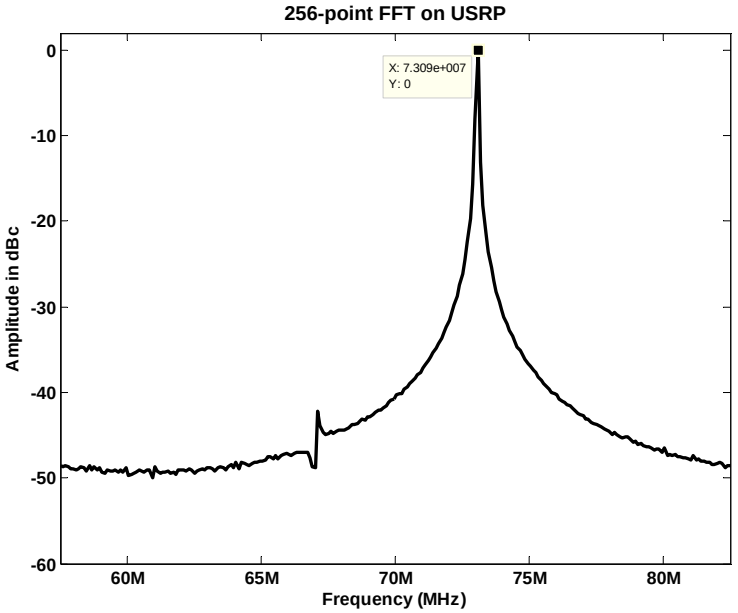
Debug Serial Port:

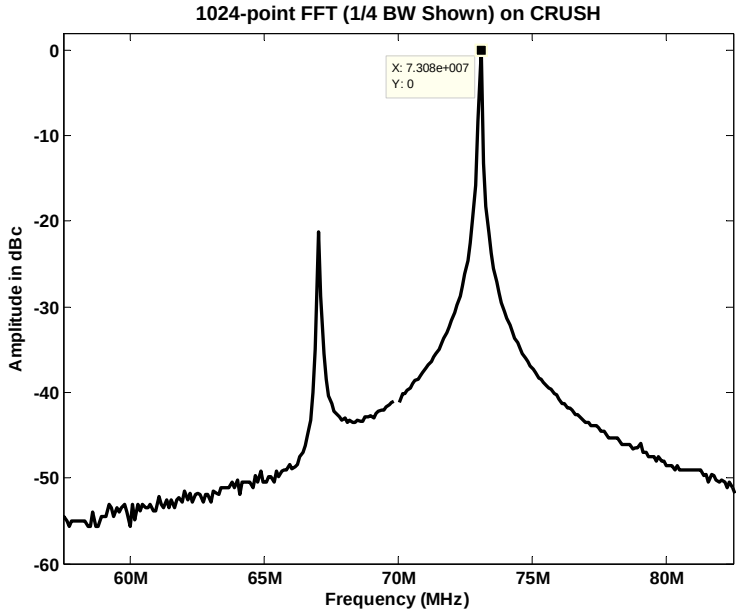
+ Debug Interface
+ Access more detailed settings
+ Contains all functions of UHD
+ Reprogrammable in software
- Slow
- Not user friendly

Test Setup – Equipment Configuration



Results – Functional Verification

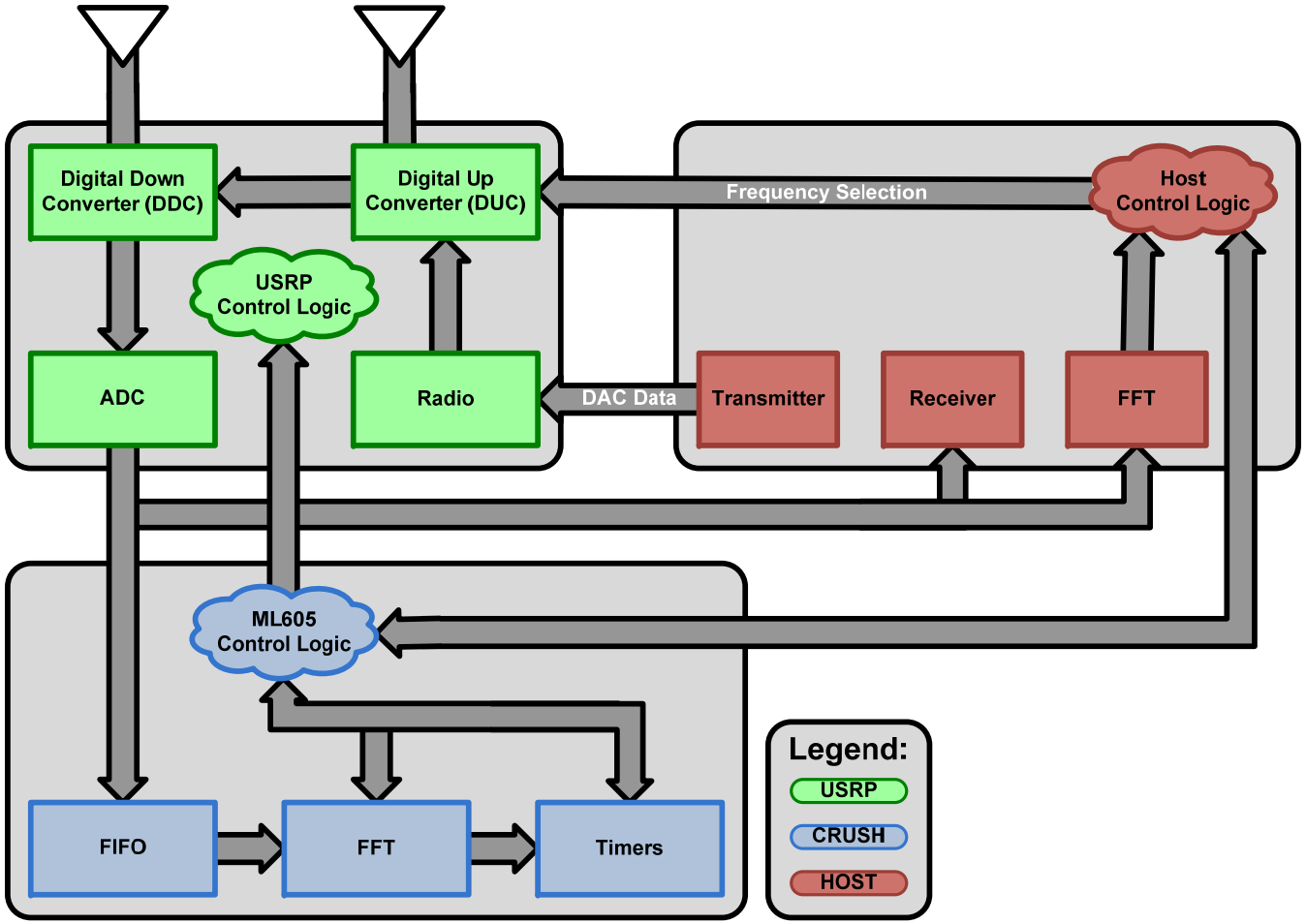




- Data recorded via USRP software
- 73 MHz CW tone
- 70 MHz Center Frequency
- 256-point FFT, 25 MHz Bandwidth
- USRP Data is filtered and Decimated by the dsp_core block

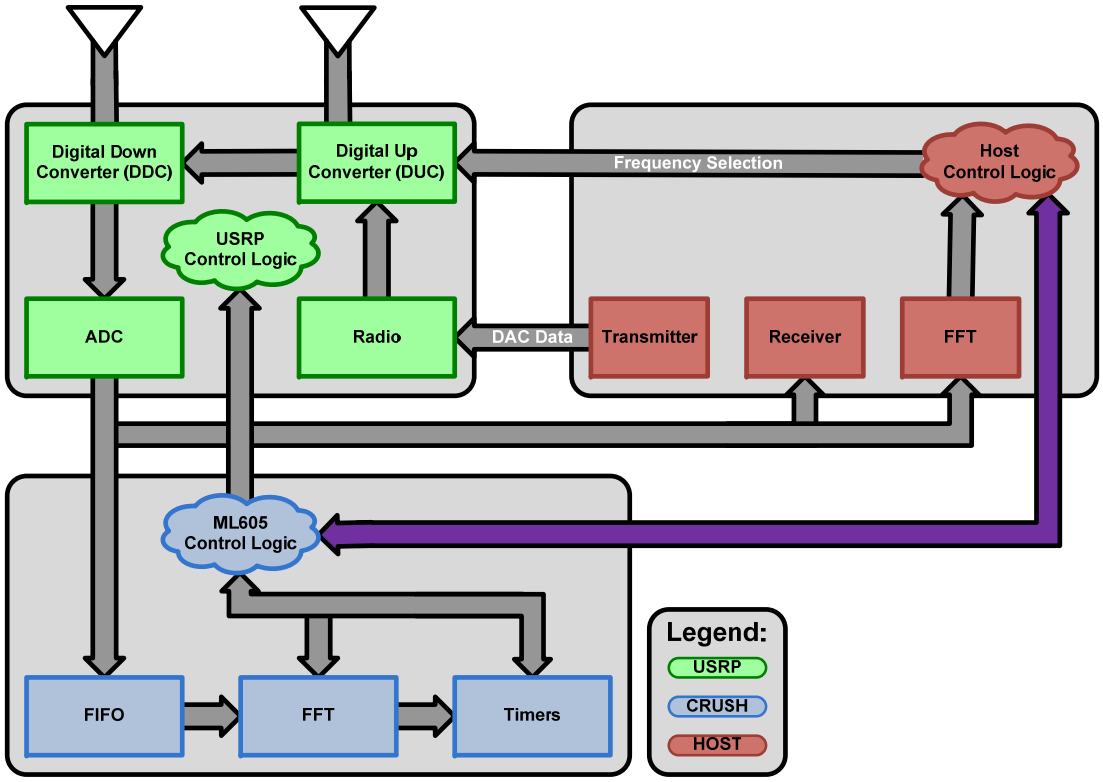
- Data sent over Ethernet using CRUSH and recorded using Matlab
- 1024-point FFT, 100 MHz Bandwidth, 25 MHz shown to match USRP
- Verifies FFT in CRUSH

FFT Timing – Required Modifications



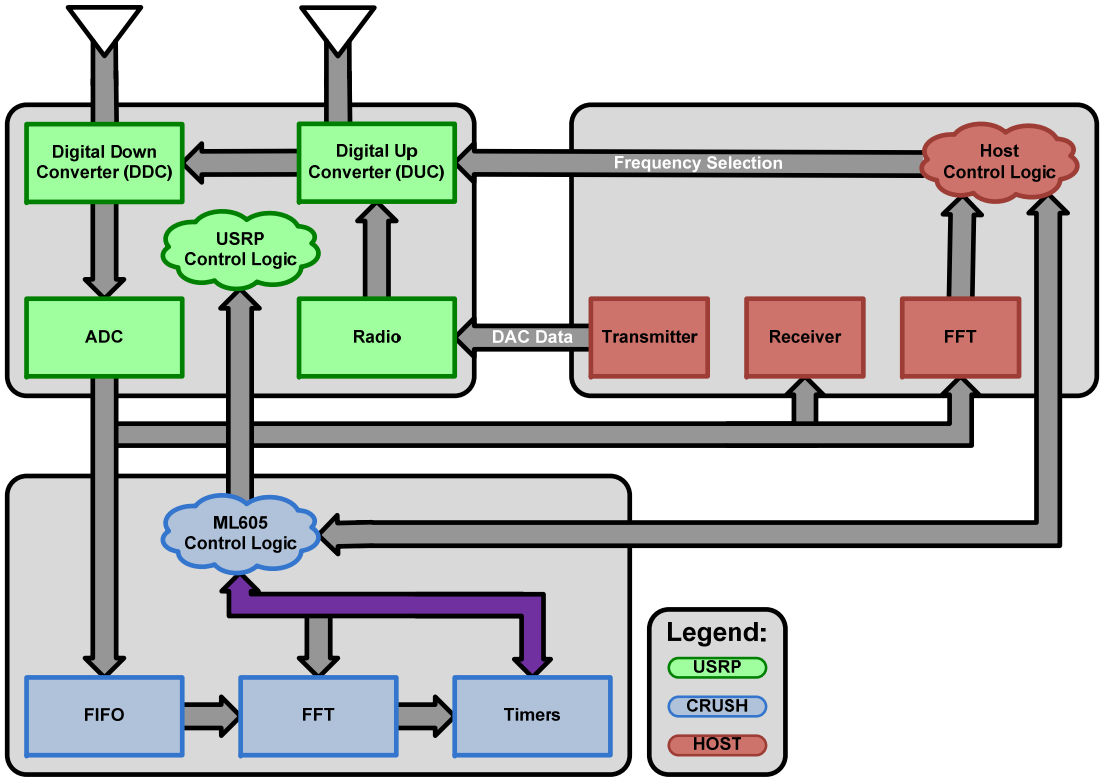
- Added Timers to ML605 and special control logic to the USRP

FFT Timing – Detailed Steps



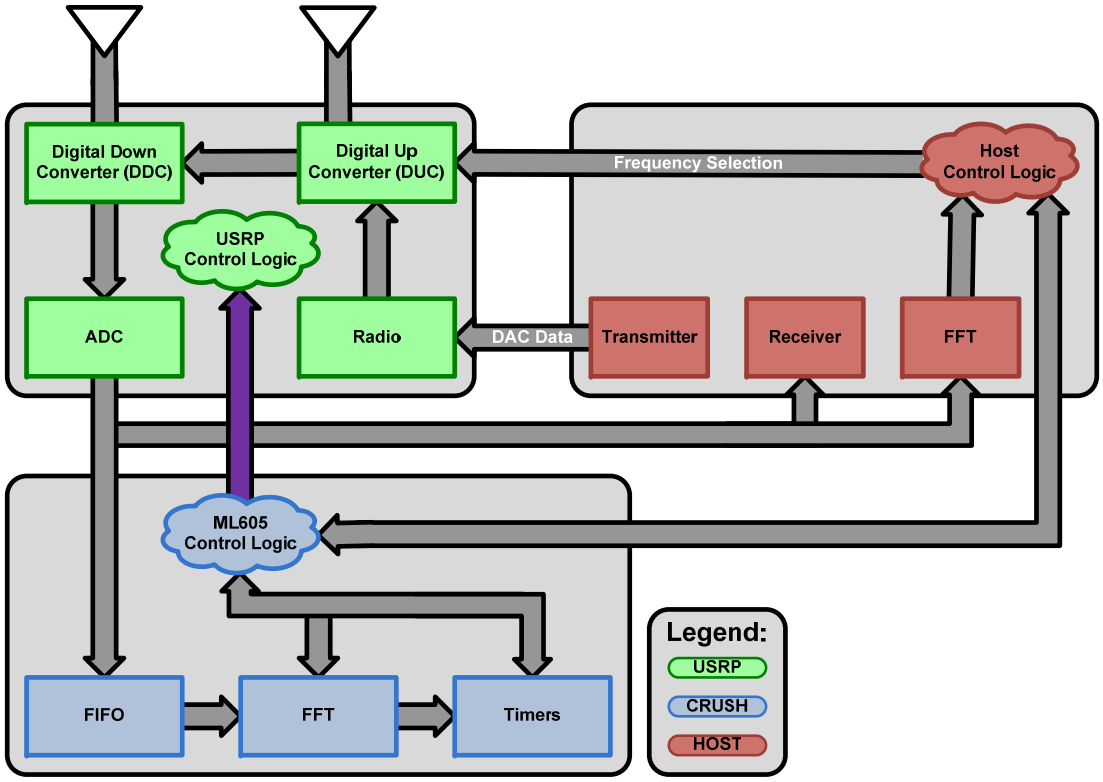
1. **Host -> ML605 (start test)**
2. ML605 (reset timers)
3. ML605 -> USRP (start test)
4. USRP -> Host / ML605 (test pattern)
5. USRP / ML605 Processing
6. ML605 FFT Done (stops timer)
7. Host -> ML605 (FFT Done)
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



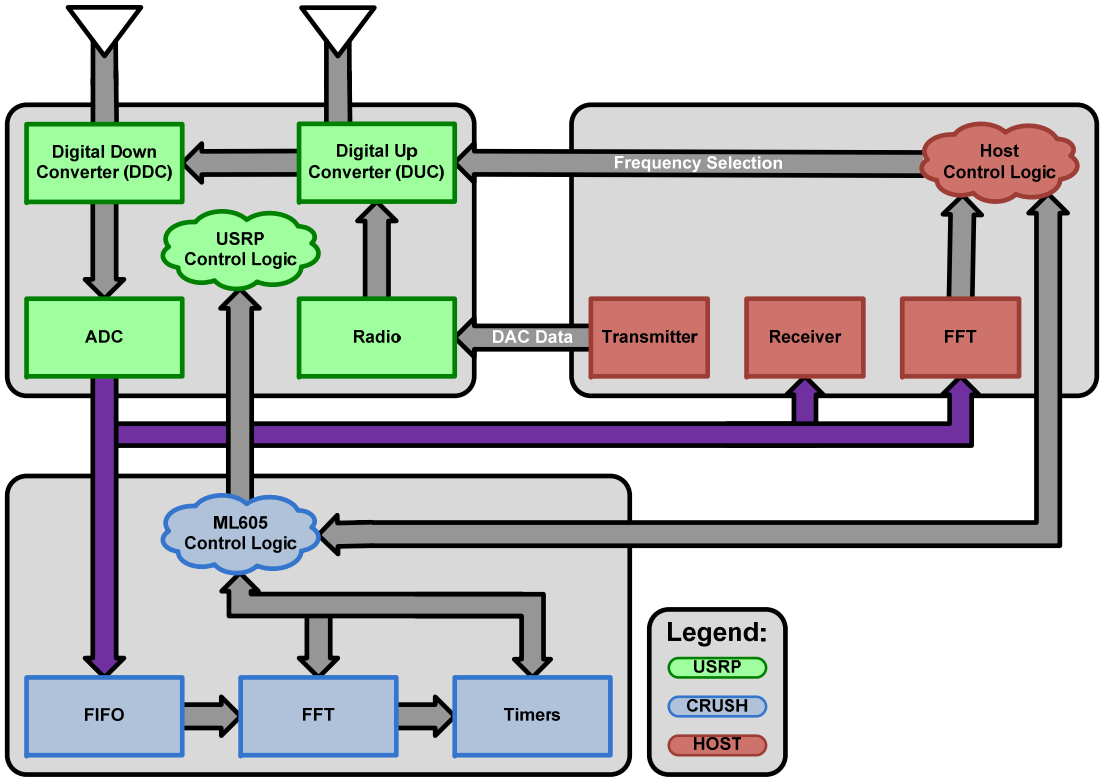
1. Host -> ML605 (start test)
2. **ML605 (reset timers)**
3. ML605 -> USRP (start test)
4. USRP -> Host / ML605 (test pattern)
5. USRP / ML605 Processing
6. ML605 FFT Done (stops timer)
7. Host -> ML605 (FFT Done)
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



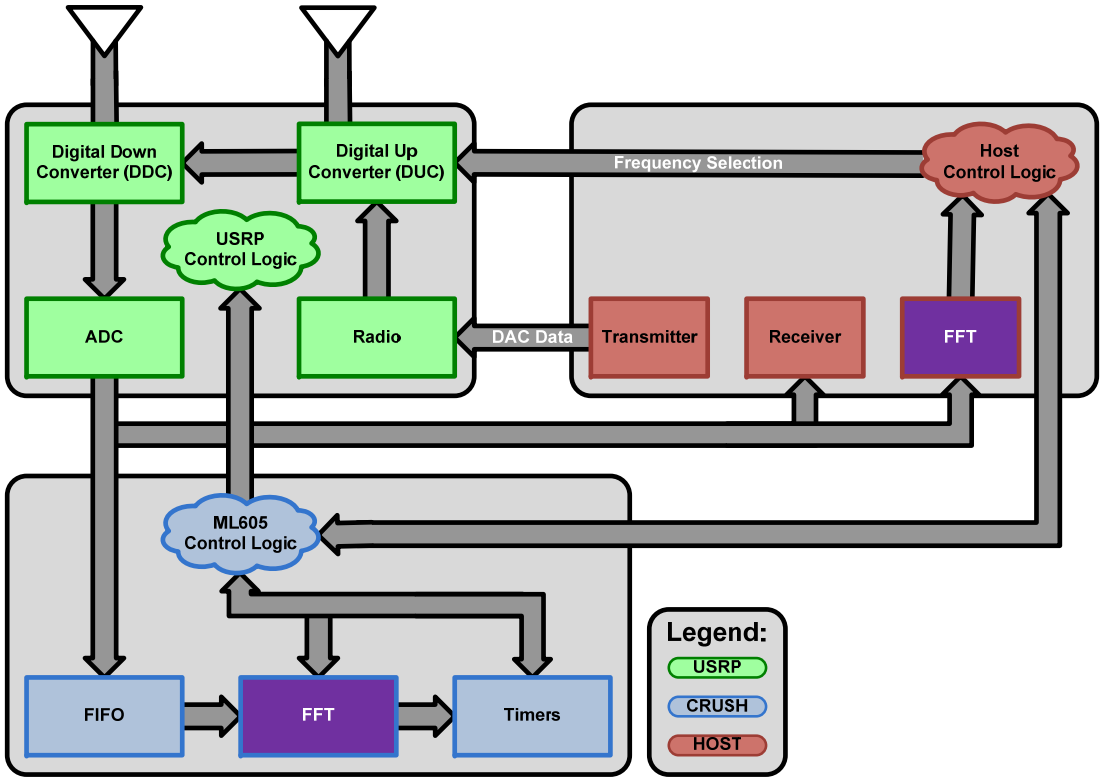
1. Host -> ML605 (start test)
2. ML605 (reset timers)
3. **ML605 -> USRP (start test)**
4. USRP -> Host / ML605 (test pattern)
5. USRP / ML605 Processing
6. ML605 FFT Done (stops timer)
7. Host -> ML605 (FFT Done)
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



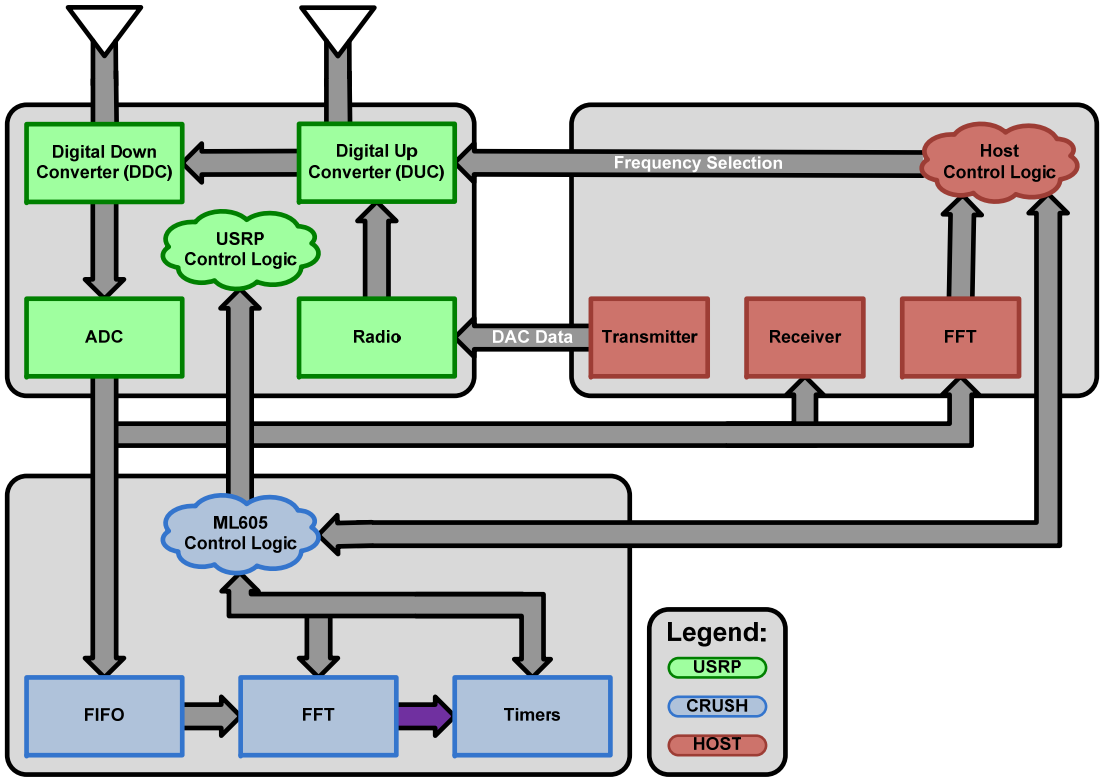
1. Host -> ML605 (start test)
2. ML605 (reset timers)
3. ML605 -> USRP (start test)
4. **USRP -> Host / ML605 (test pattern)**
5. USRP / ML605 Processing
6. ML605 FFT Done (stops timer)
7. Host -> ML605 (FFT Done)
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



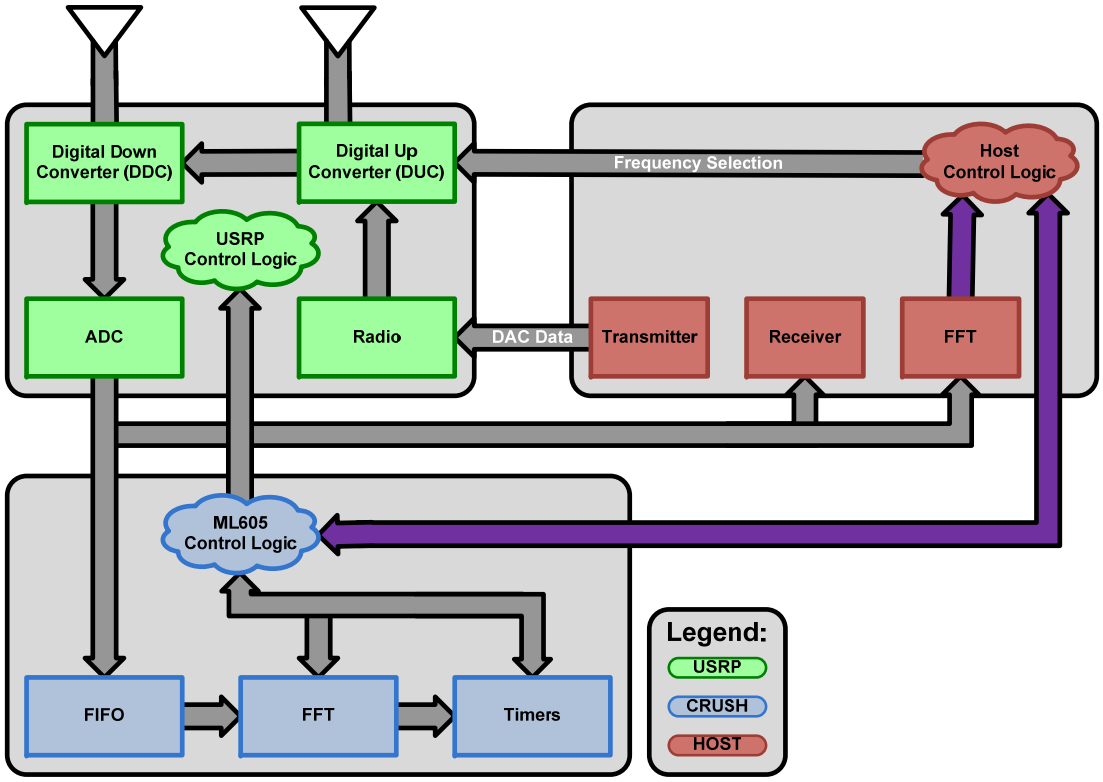
1. Host -> ML605 (start test)
2. ML605 (reset timers)
3. ML605 -> USRP (start test)
4. USRP -> Host / ML605 (test pattern)
5. **USRP / ML605 Processing**
6. ML605 FFT Done (stops timer)
7. Host -> ML605 (FFT Done)
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



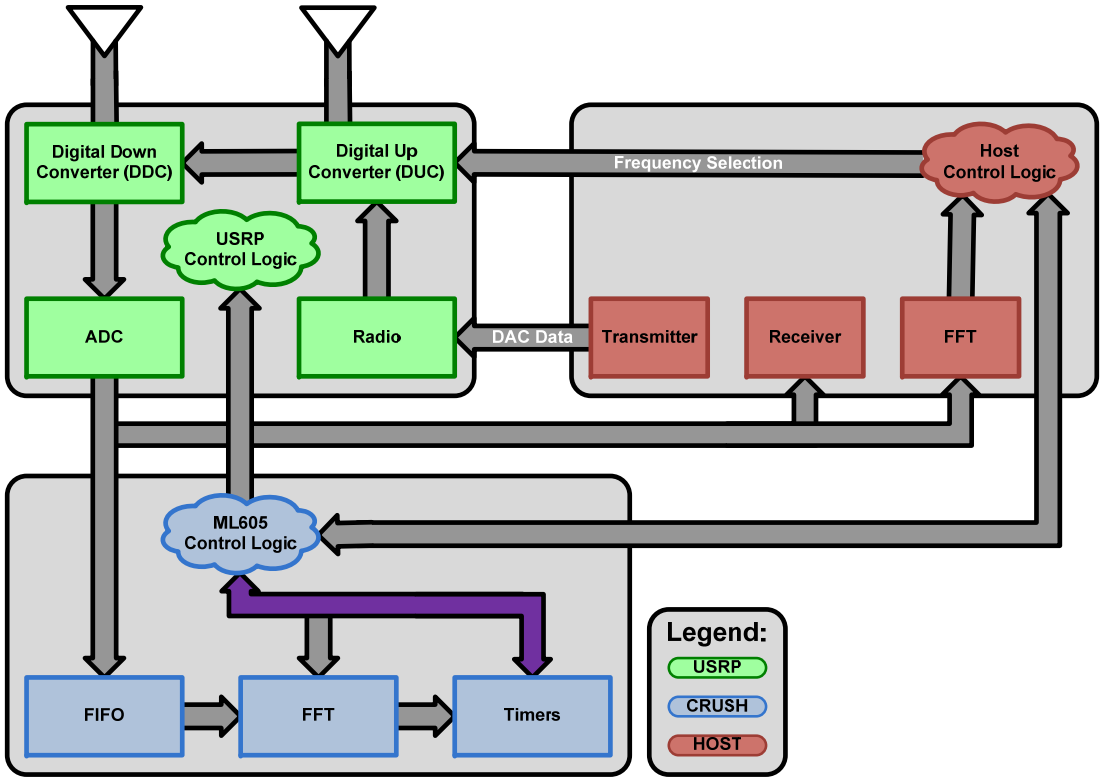
1. Host -> ML605 (start test)
2. ML605 (reset timers)
3. ML605 -> USRP (start test)
4. USRP -> Host / ML605 (test pattern)
5. USRP / ML605 Processing
6. **ML605 FFT Done (stops timer)**
7. Host -> ML605 (FFT Done)
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



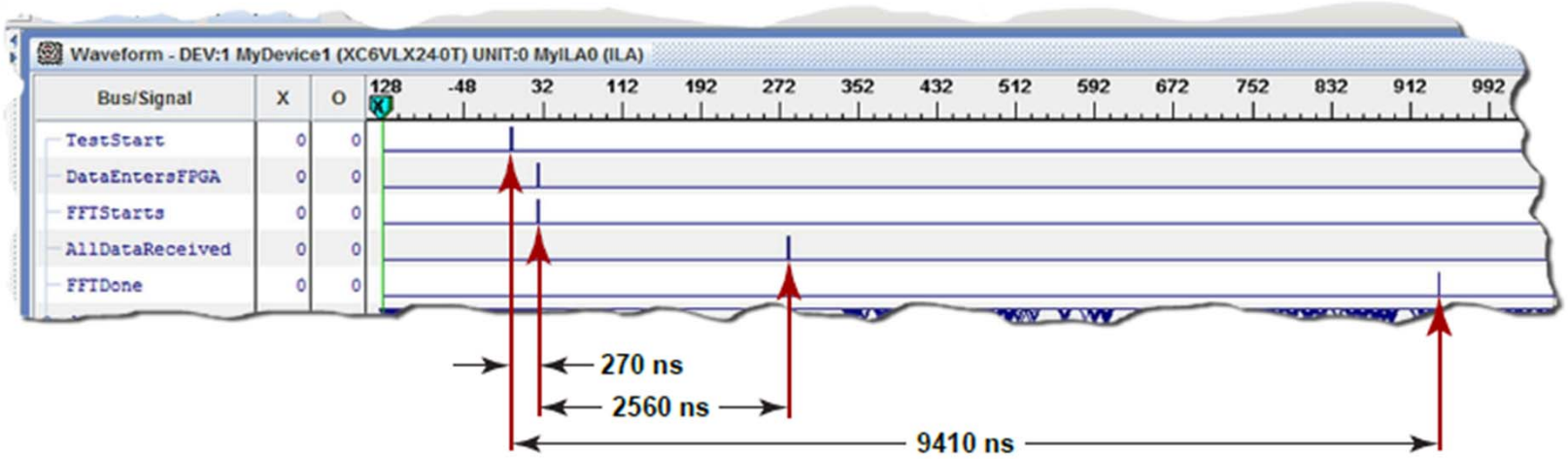
1. Host -> ML605 (start test)
2. ML605 (reset timers)
3. ML605 -> USRP (start test)
4. USRP -> Host / ML605 (test pattern)
5. USRP / ML605 Processing
6. ML605 FFT Done (stops timer)
7. **Host -> ML605 (FFT Done)**
8. Host FFT Done (stops timer)

FFT Timing – Detailed Steps



1. Host -> ML605 (start test)
2. ML605 (reset timers)
3. ML605 -> USRP (start test)
4. USRP -> Host / ML605 (test pattern)
5. USRP / ML605 Processing
6. ML605 FFT Done (stops timer)
7. Host -> ML605 (FFT Done)
8. **Host FFT Done (stops timer)**

256-point FFT Example



Action	Clocks	Time (µs)	Incremental (µs)
Start Test	0	0	0
Data enters FPGA Clock Domain	27	0.27	0.27
FFT Starts	27	0.27	0
All Data inside FPGA	256	2.56	2.83
FFT Complete	941	9.41	6.58
Total	**941**	**9.41**	**9.41**

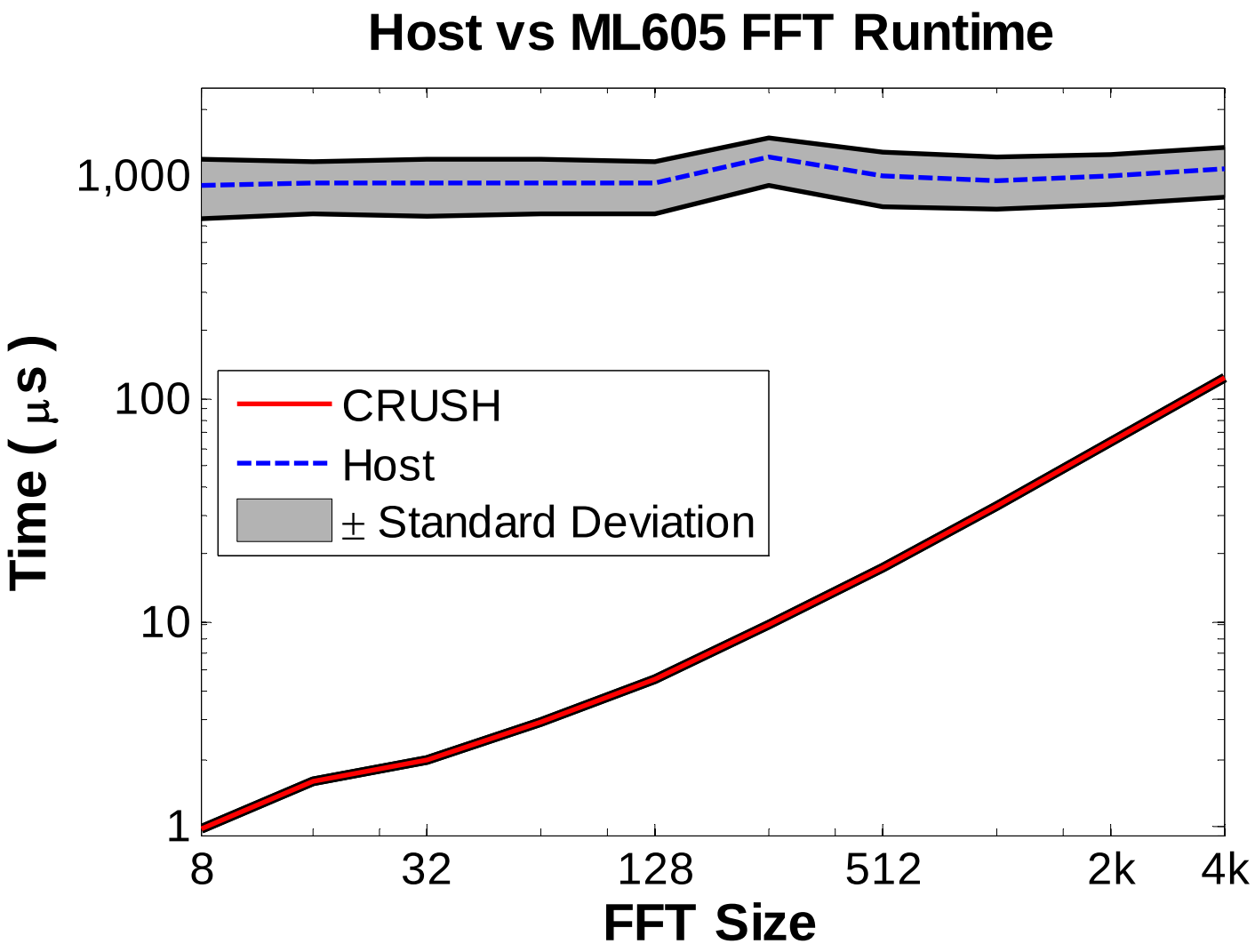
- **FFT Starts on first word, streams thereafter**

Results – FFT Timing Analysis

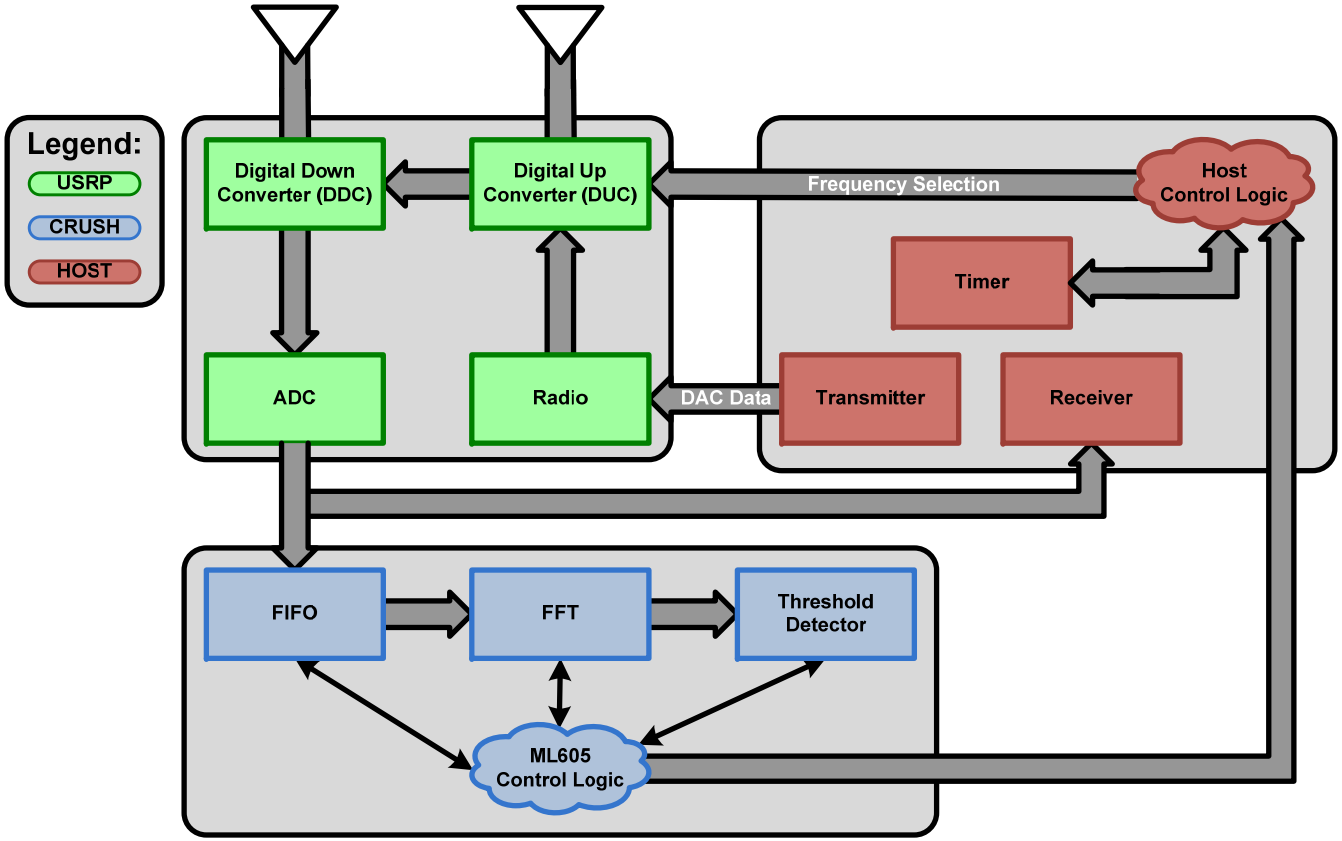
FFT Size	FPGA Average (µs)	Host Average (µs)	Speed-up (×)
8	1.17	907.72	774
16	1.91	915.89	479
32	2.38	920.07	386
64	3.56	925.47	259
128	5.47	916.54	167
256	9.56	1198.19	125
512	17.23	1003.83	58
1024	32.84	955.30	29
2048	63.55	995.26	15
4096	125.24	1071.79	8

- Speed-up between 8 and 774 ×
- FPGA timing scales log linear with FFT size
- Host timing driven by packet transmit time and internal buffering

Results – FFT Runtime

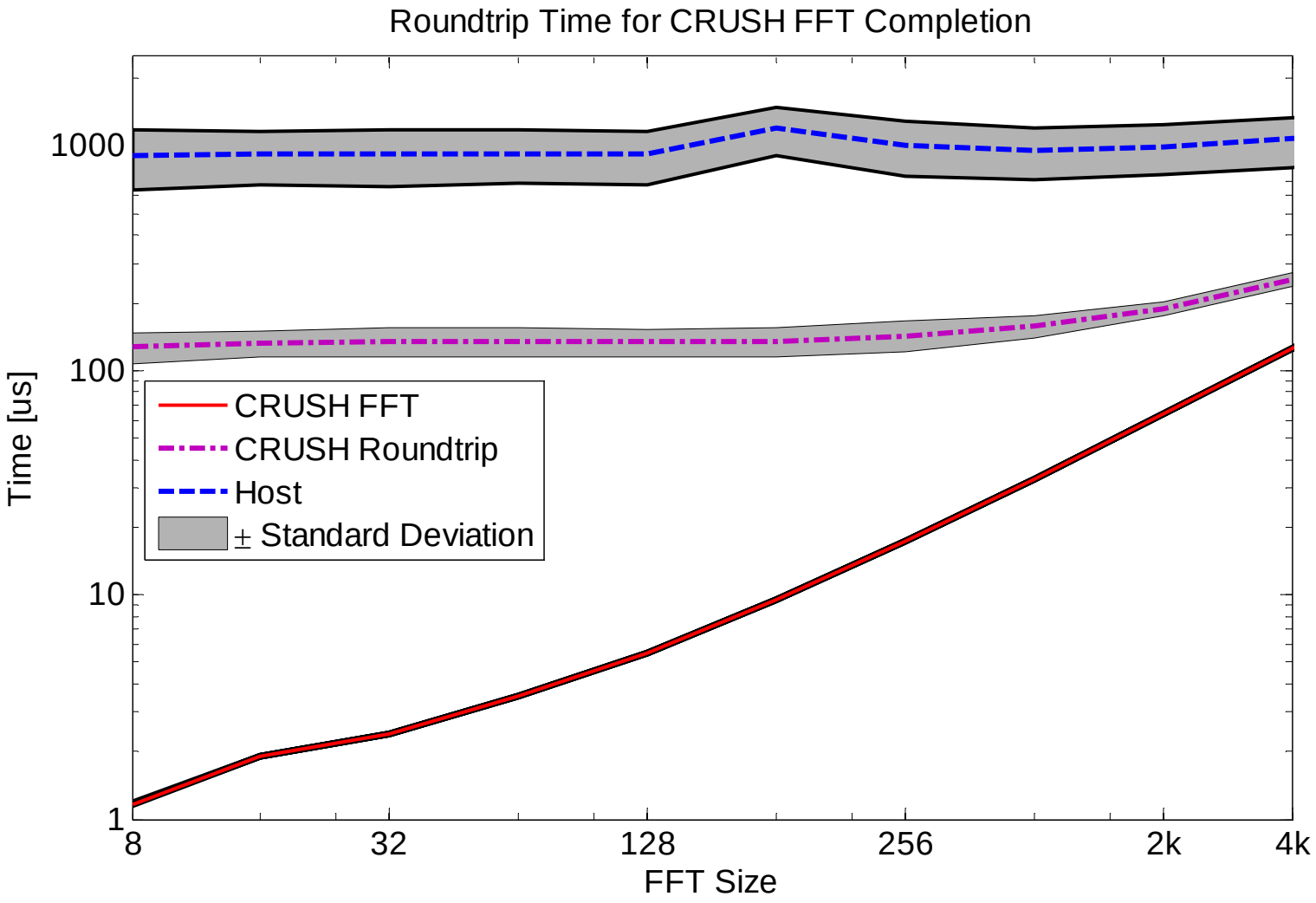


Results – End to End CRUSH Timing



- Moved timer into the host, allows for end to end timing analysis
- Measured and recorded through the UHD C++ Code

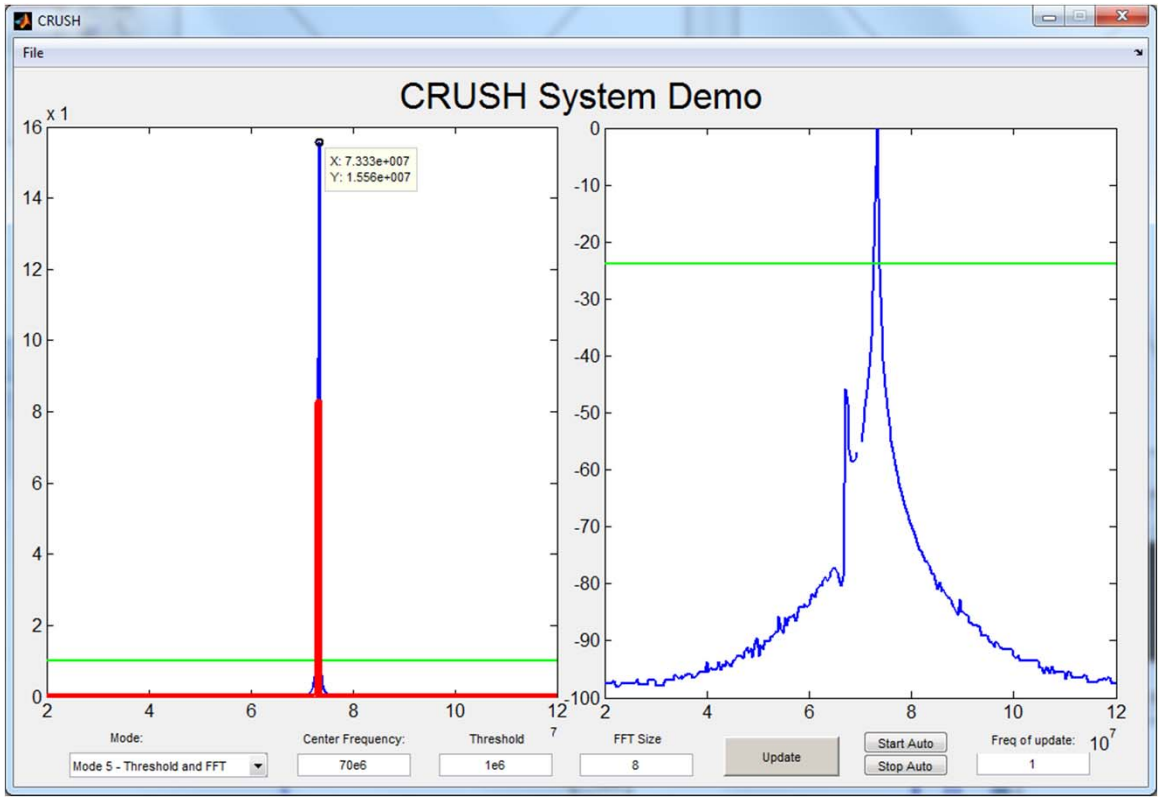
Results - End-to-end Timing



Real World Example: Free Space Test

- Free space test @ 915 MHz – 1W transmitter in ISM band

Real World Example: Matlab Demo



- **Blue** represents the FFT values from CRUSH
- **Red** represents the threshold values
- **Green** is the threshold
- Parameters adjustable on bottom of demo
- Left side is a Magnitude plot
- Right side is a dB plot

Conclusion

- Created CRUSH platform
  - Combined powerful FPGA with versatile RF front end
  - Produced custom interface board to allow high speed data transfer
  - Moved processing closer to the receiver
  - Decoupled the fast evolving FPGA platform from the custom RF front end
- Implemented spectrum sensing on CRUSH
  - Achieved more than 100x performance for FFT on point sizes of interest
  - Roundtrip timing beats USRP by 10x
  - Reduced load on host computer
  - Fully configurable

Future Work

- Integrate the spectrum sensing module into research on Cognitive Radio at Northeastern University
- Explore other methods of performing hardware accelerated spectrum sensing such as wavelet analysis
- Utilize the CRUSH platform to migrate additional software radio functions into reconfigurable hardware
- Perform non-radio research with the CRUSH platform utilizing its RF front end and FPGA back end

CRUSH software and users manual will be available from September at

http://www.coe.neu.edu/Research/rcl//projects/CRUSH.php

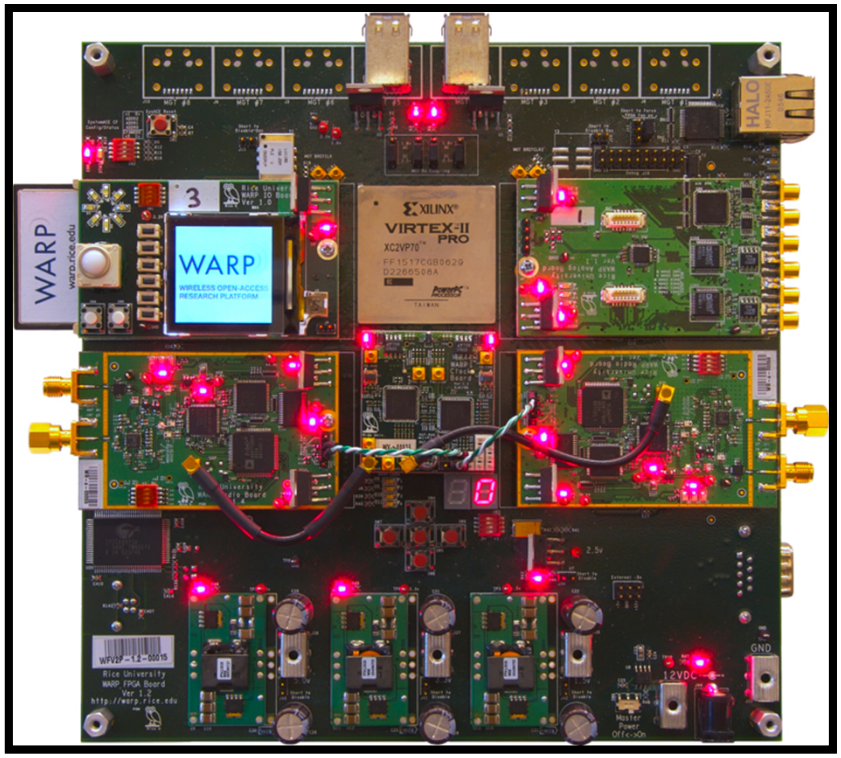
Miriam Leeser
mel@coe.neu.edu
Kaushik Chowdhury
George Eichinger

This work is sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. The Opinions, interpretations conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

The MathWorks™

Backup Slides

Related Work - WARP



*http://warp.rice.edu/trac/

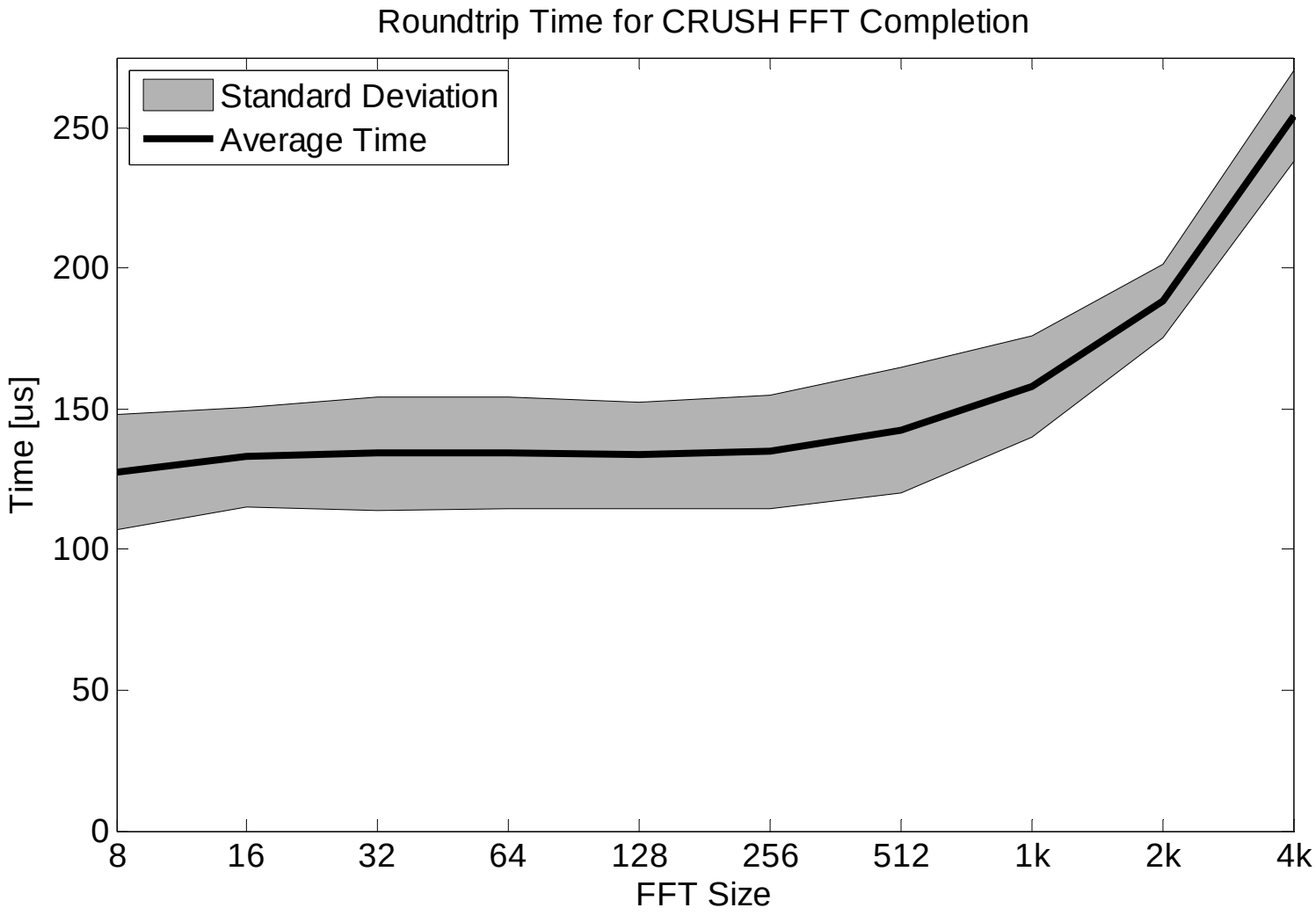
Wireless Open-Access Research Platform

- Designed by Rice University
- Latest uses Xilinx Virtex 4 FX FPGA
- 4 daughterboard slots for ADC, DAC or IO
- More processing power than USRP but the hardware is becoming dated
- Design is based on FX series FPGA which has been discontinued
- CRUSH decouples the FPGA from the RF portion and reduces the effect of old hardware

Related Work - GNURadio

- Both a Visual (GNURadio Companion) and programming (python) environment for SDR implementations
- Fully supports the USRP and other common SDR RF front ends
- Backend of GNURadio is highly optimized C libraries and front end is python scripts to connect the C libraries together
- Users can quickly get the system up and running and observe spectrum via FFTs, send simple data and implement radio protocols
- http://gnuradio.org/redmine/projects/gnuradio/wiki

Results – End to End CRUSH Timing



Serial Debug Port