An FPGA Spectrum Sensing Accelerator for Cognitive Radio

George Eichinger
Miriam Leeser
Kaushik Chowdhury

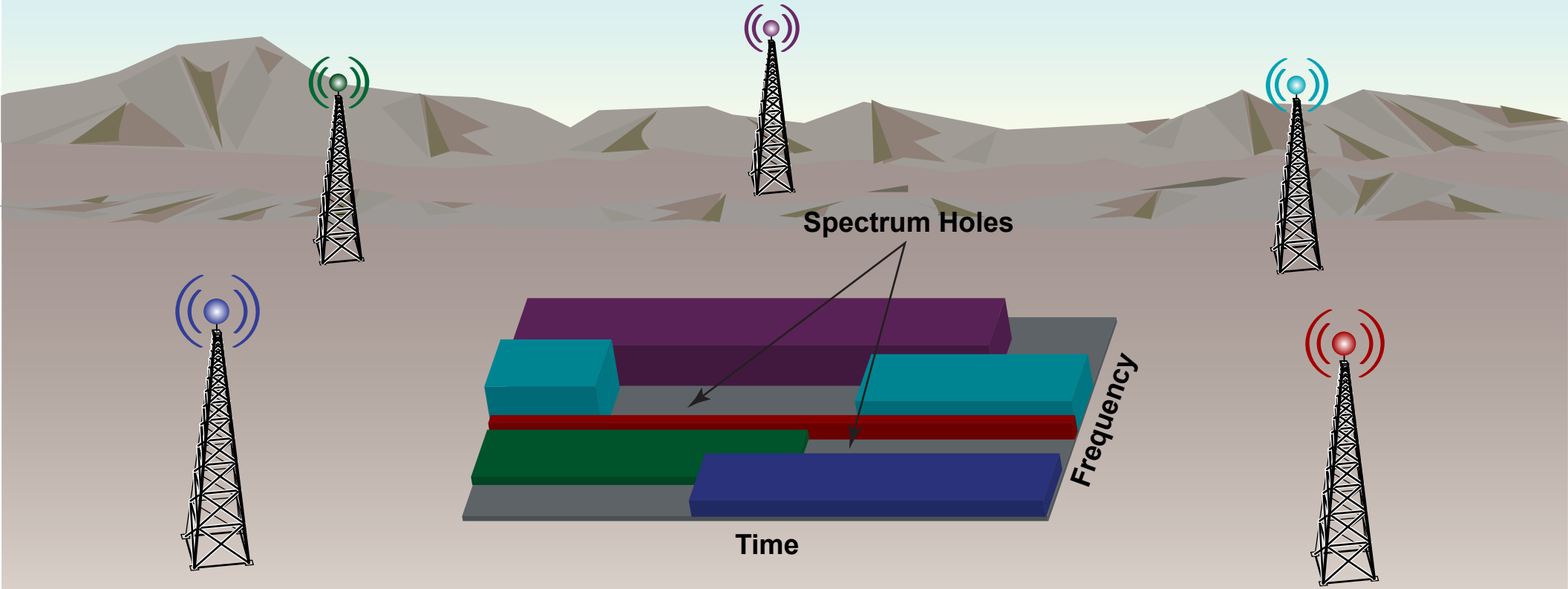
HPEC 2011

21 Sept 2011

Northeastern University

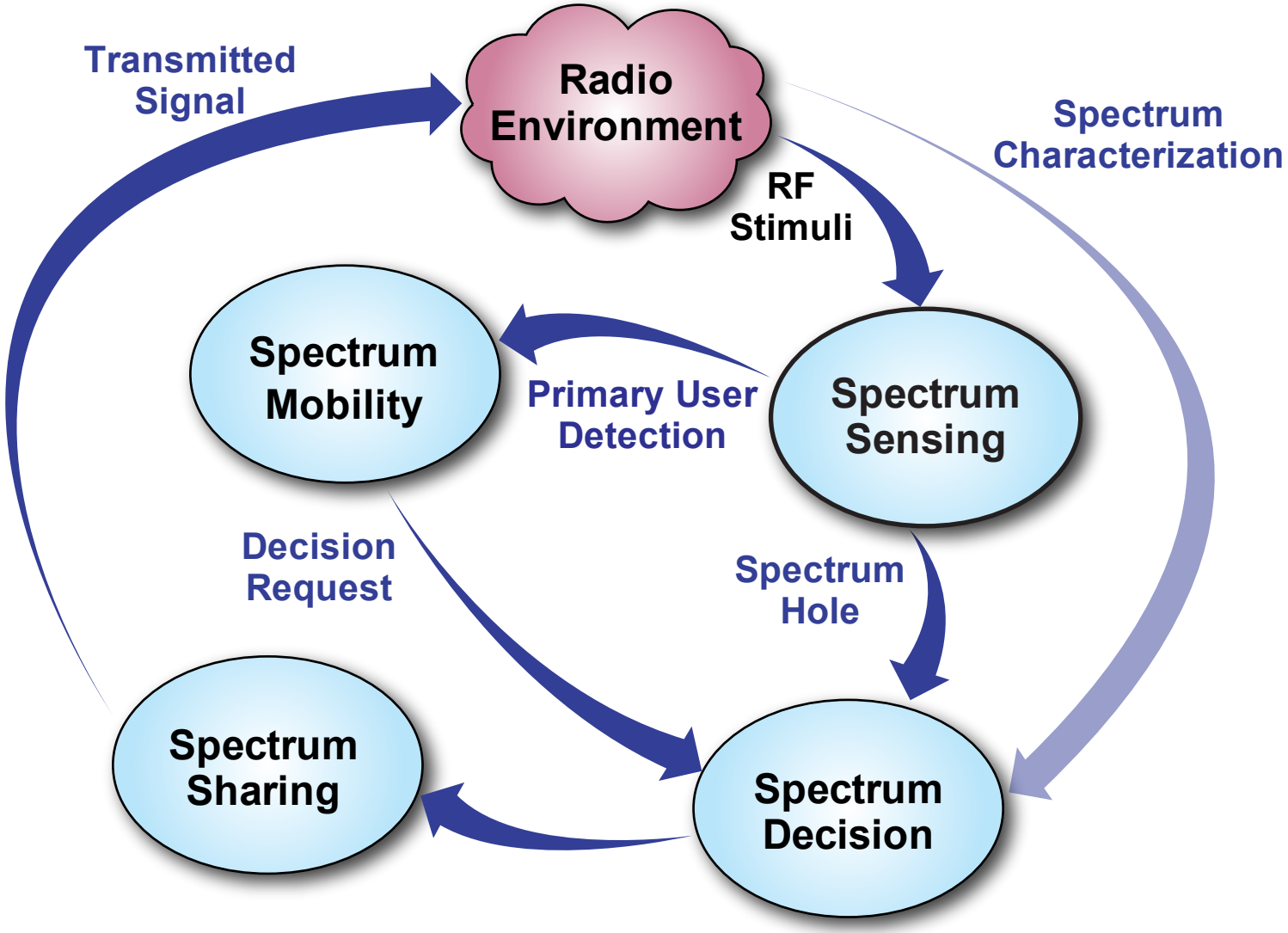
This work is sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. The opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government

Cognitive Radio Overview

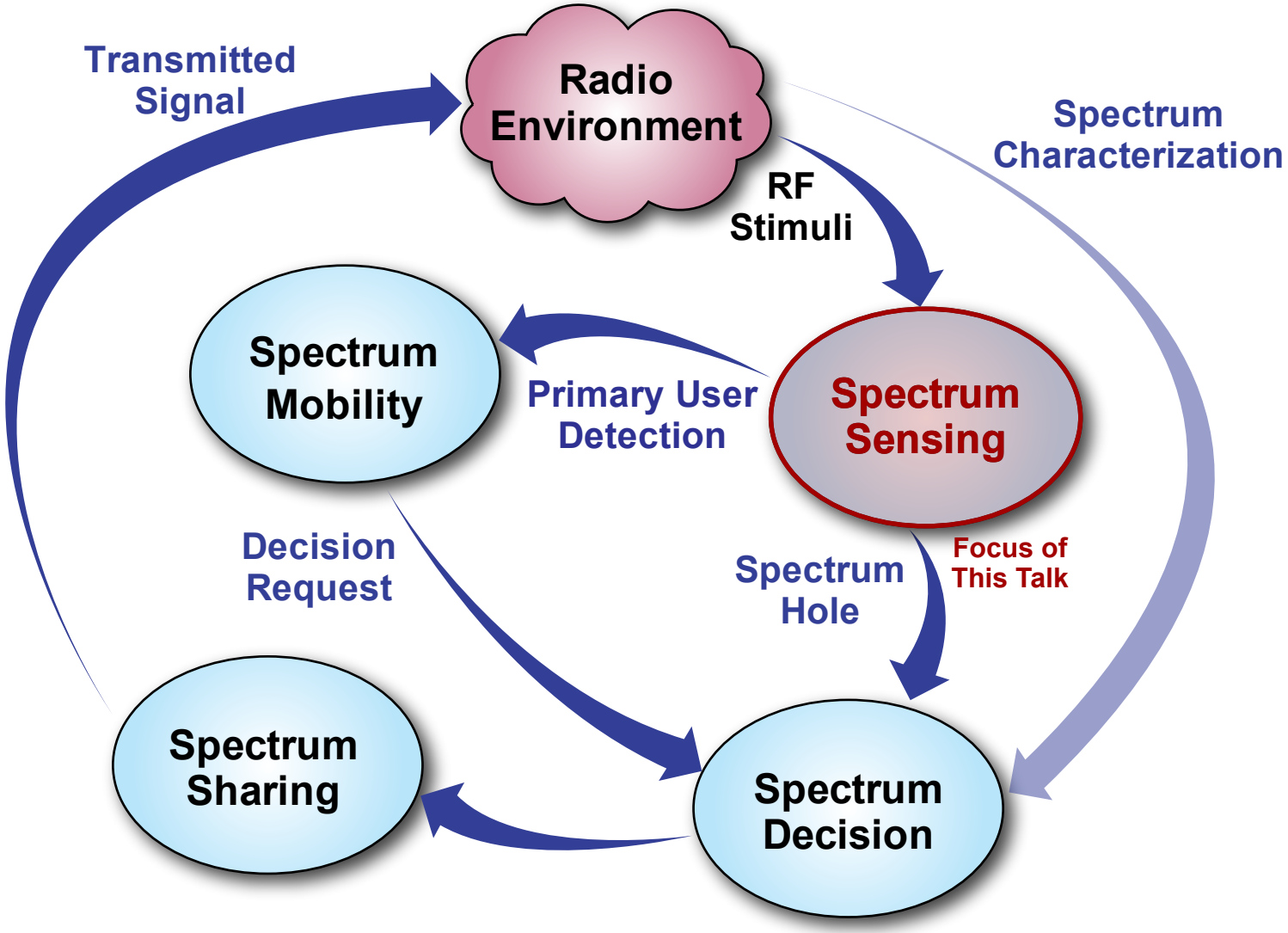


- Cognitive radios allow you to operate in unused portions of spectrum

Cognitive Cycle



Cognitive Cycle

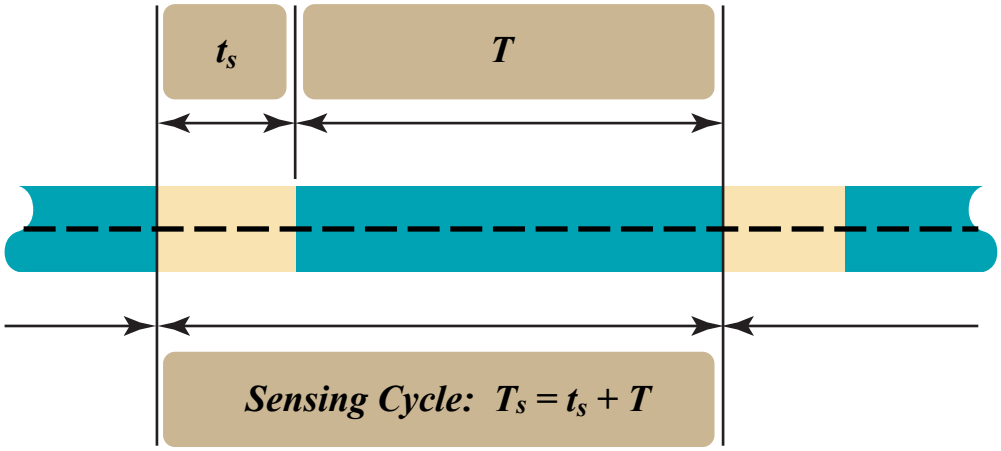


Outline

- Cognitive Radio overview
- Project motivation and goals
- Hardware platform
- Spectrum sensing algorithm
- Results of implementation
- Summary

Motivation

- 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



Project Goals

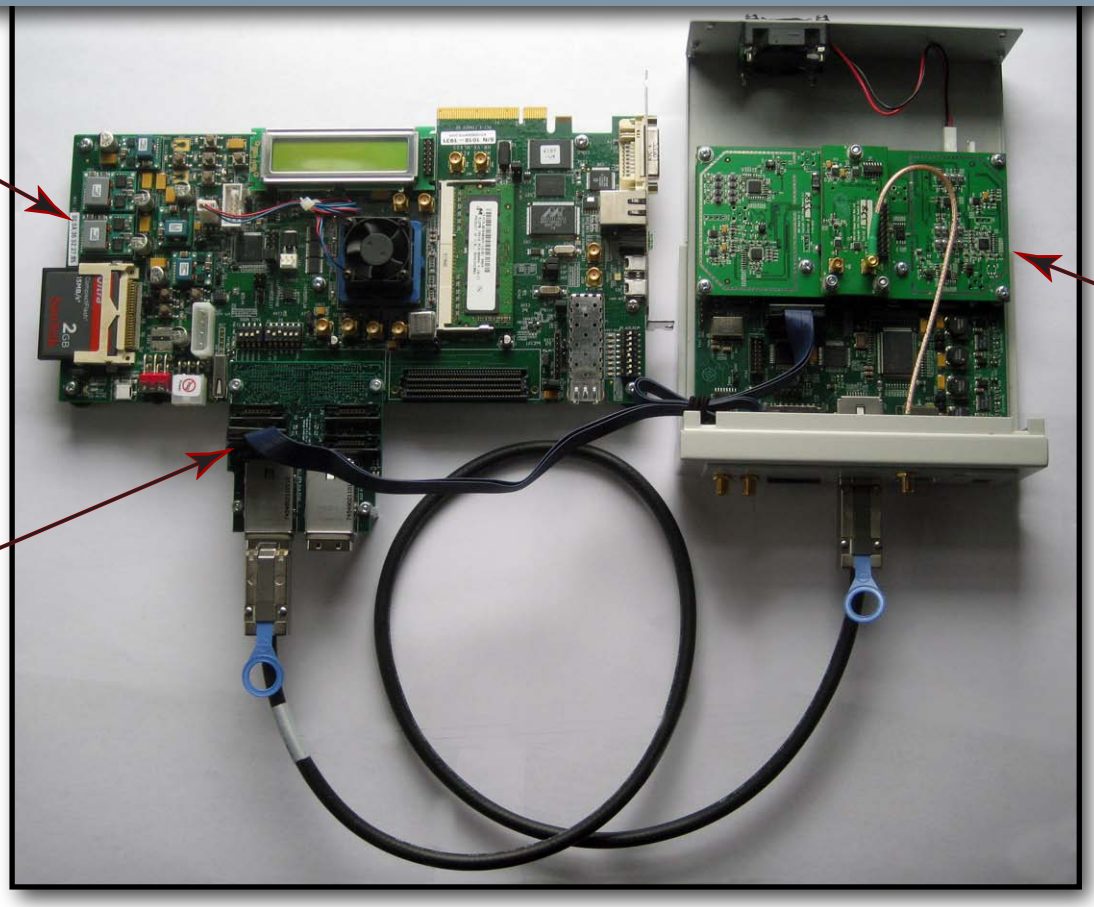
- Create platform for hardware level Cognitive Radio (CR) research
- Perform spectrum sensing as close to the receiver and as fast as possible
- Report results to host indicating whether or not a channel is free
- Design system with reconfigurable, parameterized hardware and programmable software

Outline

- Cognitive Radio overview
- Project motivation and goals
- Hardware platform
  - Cognitive Radio Universal Software Hardware (CRUSH)
- Spectrum sensing algorithm
- Results of implementation
- Summary

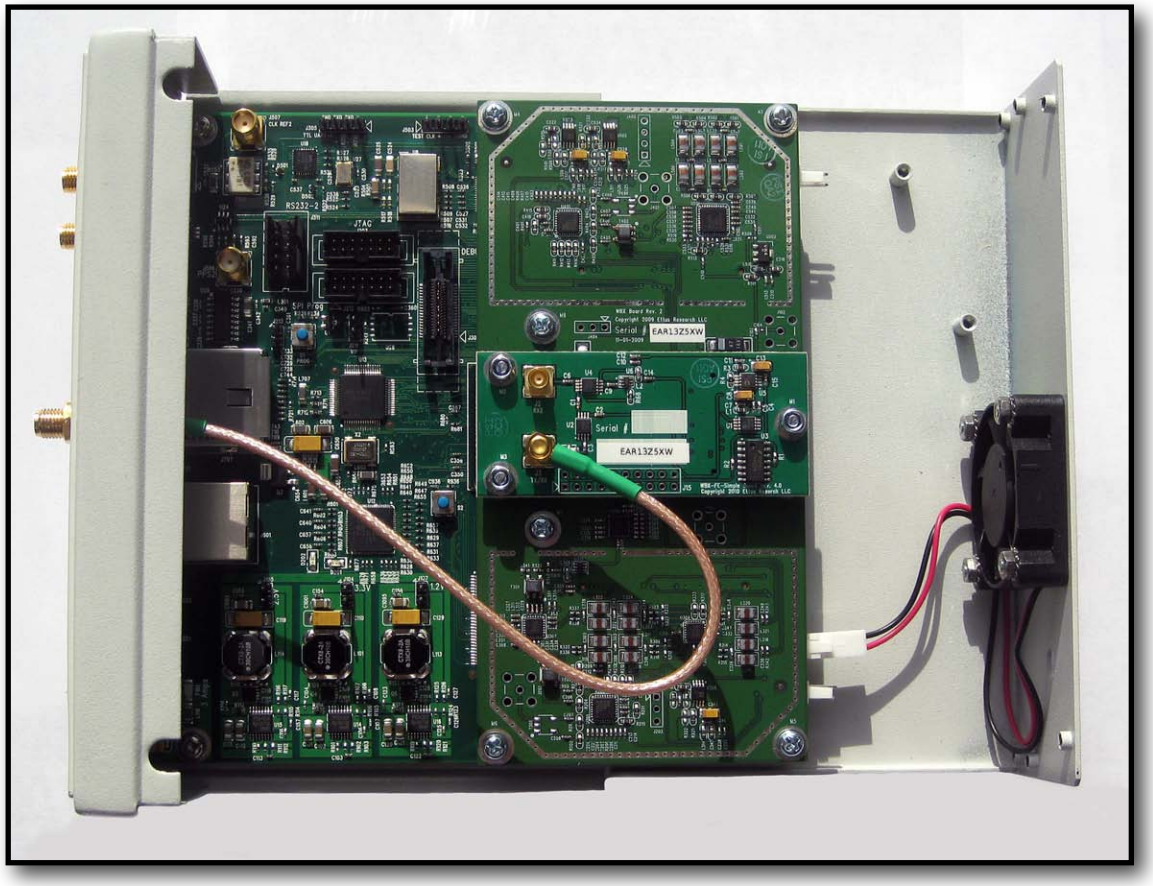
Introducing CRUSH

Cognitive Radio Universal Software Hardware (CRUSH)



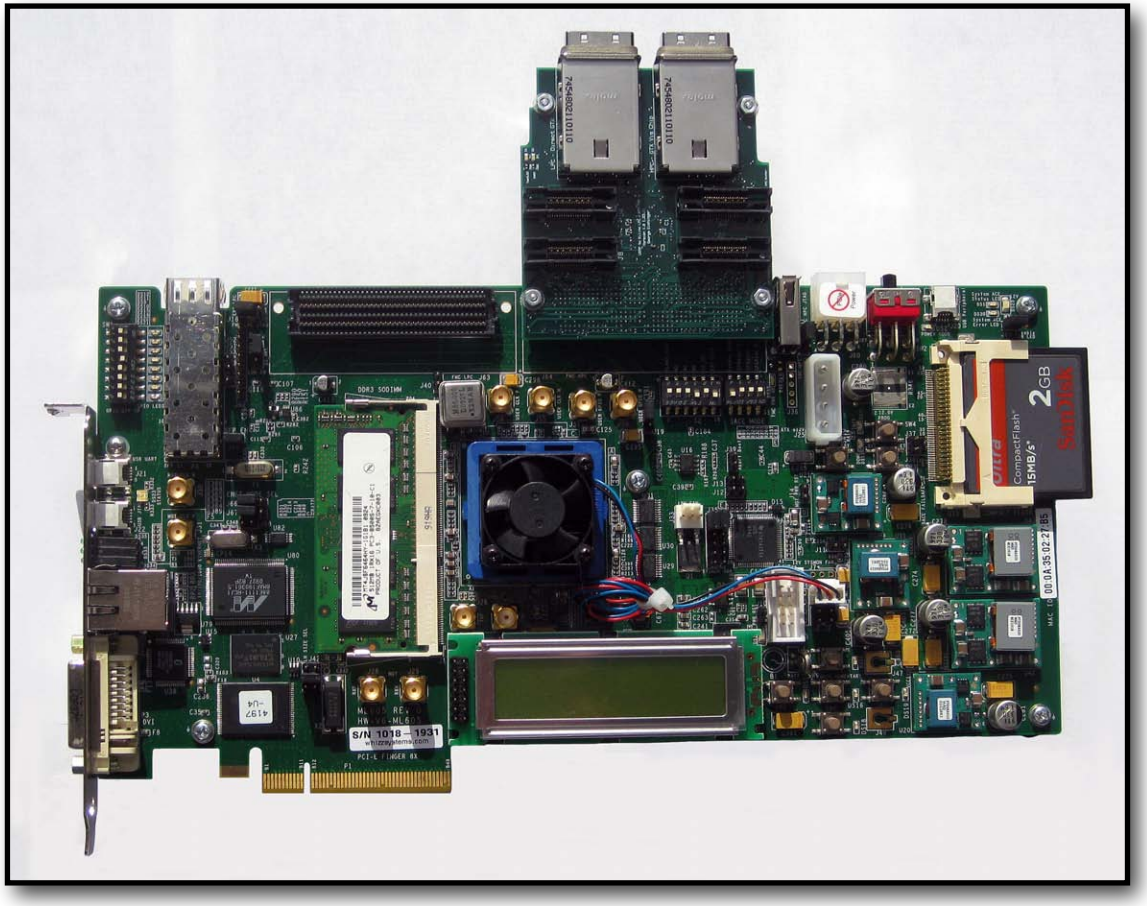
- Standard software defined radio and FPGA development board
- Custom interface board to connect the two boards

Ettus Research USRP N210



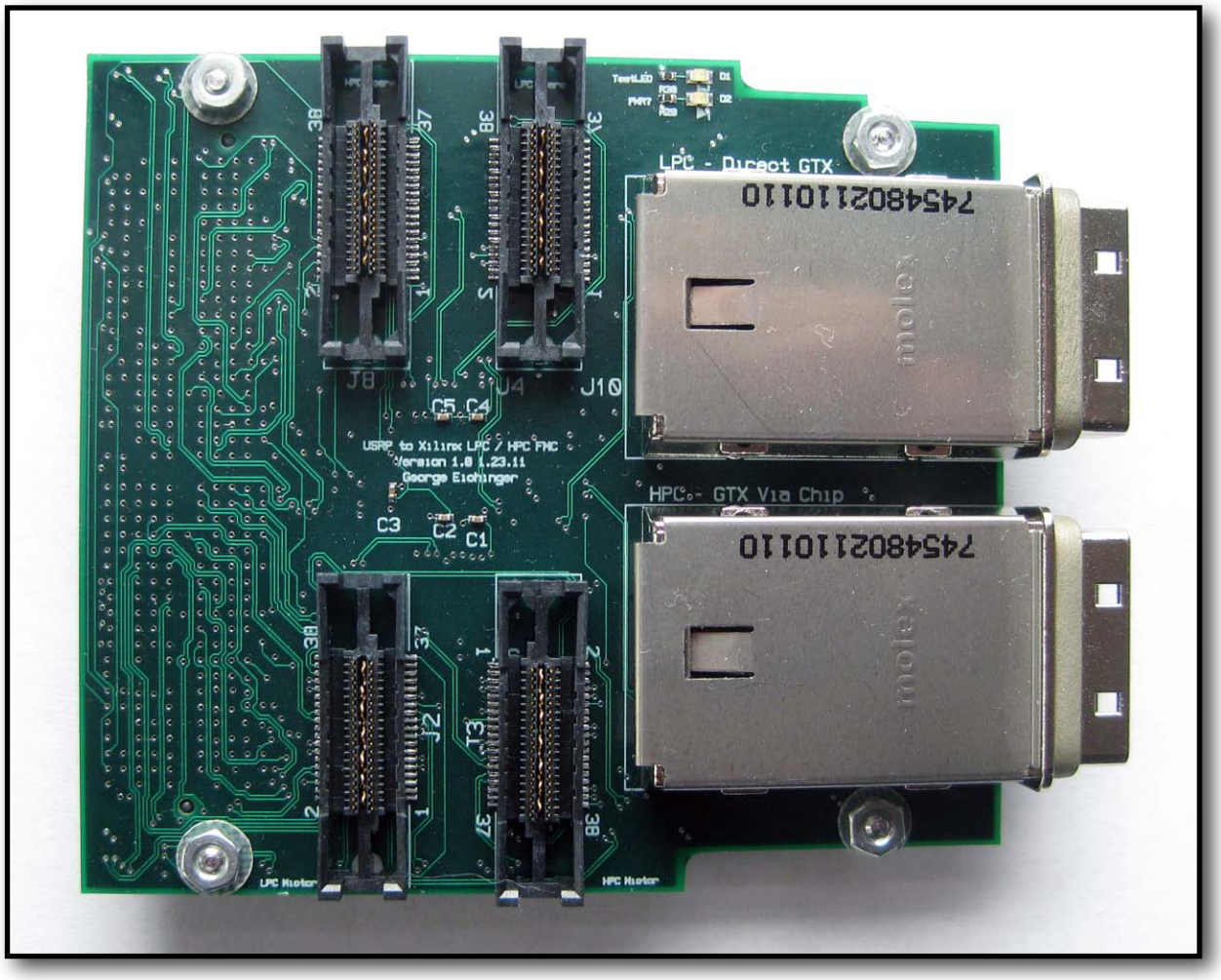
- Agile front end
- Low cost
- Xilinx Spartan 3A DSP FPGA for RX/TX
- Minor changes for CRUSH

Xilinx ML605 FPGA Development Board



- 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

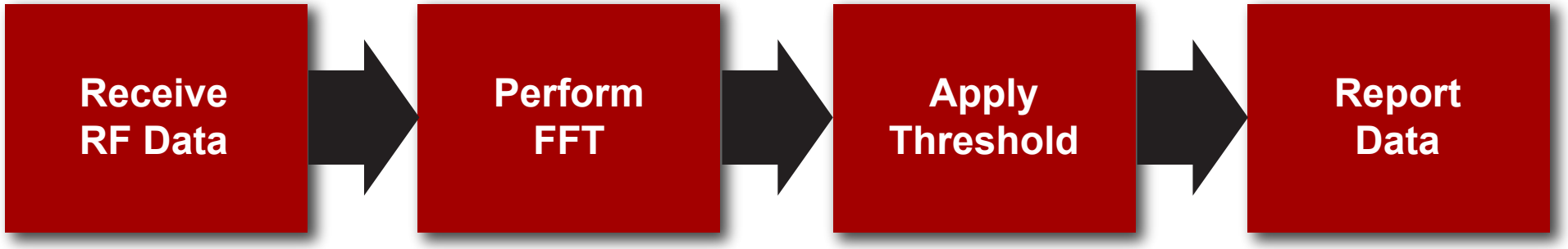


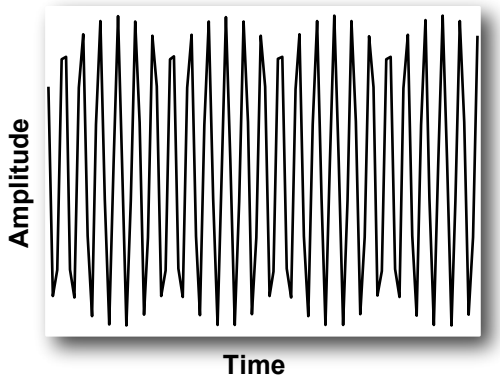
- Only custom part of CRUSH
- Allows full 100 MSPS IQ transmit and receive data
- Able to communicate with two USRPs at once
- Compatible with FMC boards such as Xilinx ML605 and SP605

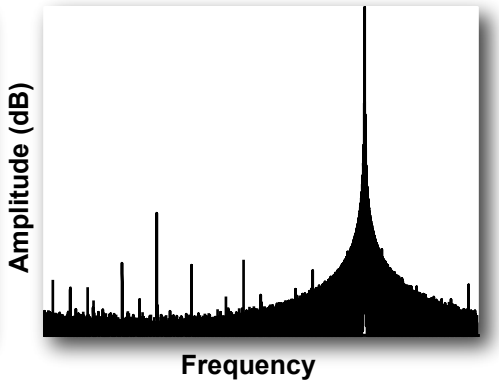
Outline

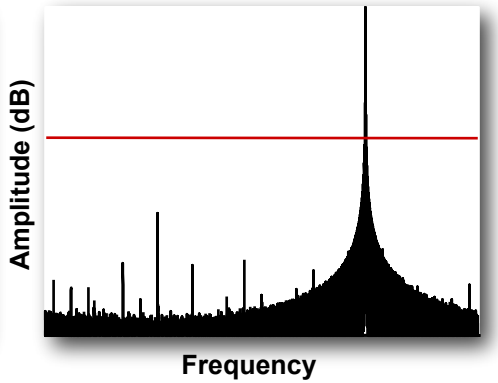
- Cognitive Radio overview
- Project motivation and goals
- Hardware platform
- Spectrum sensing algorithm
- Results of implementation
- Summary

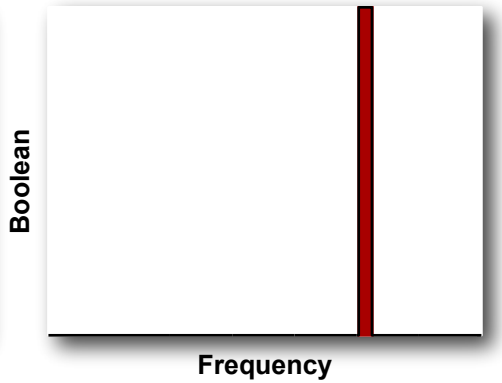
Spectrum Sensing Algorithm



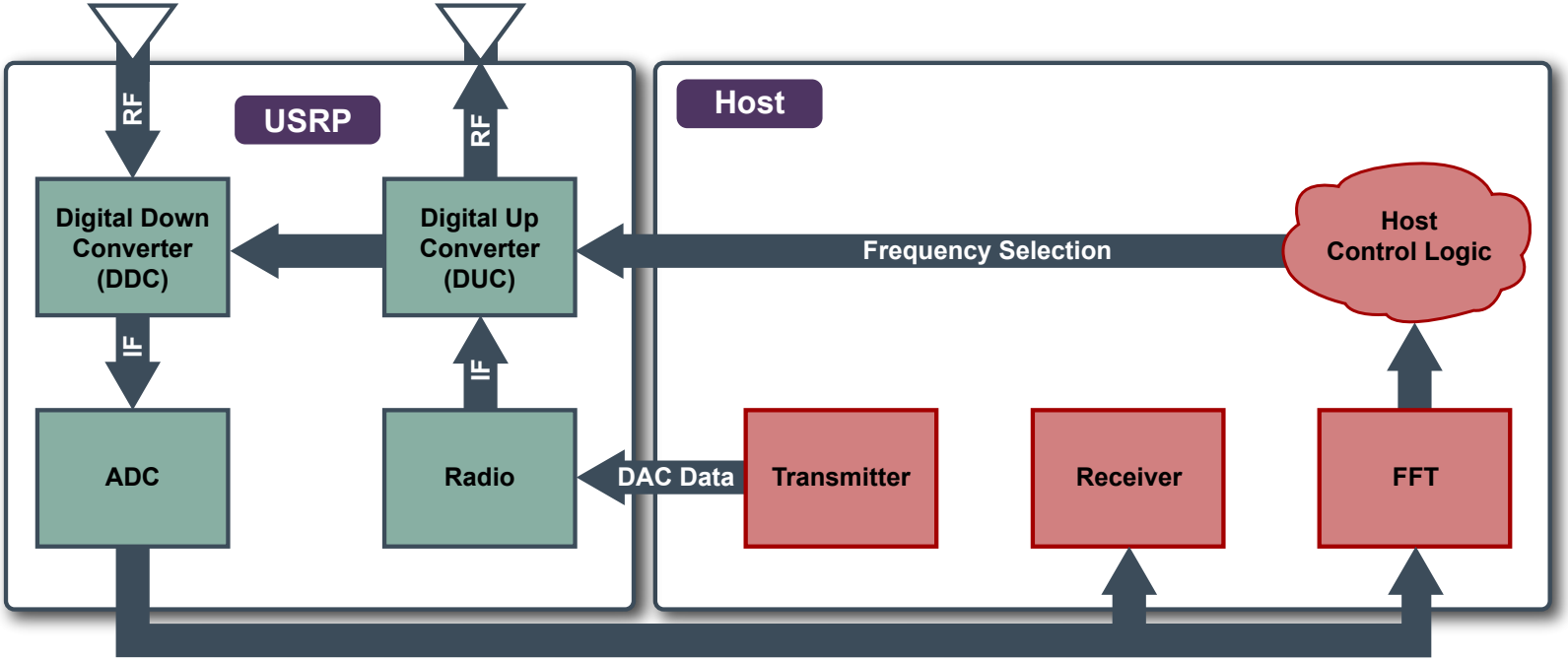






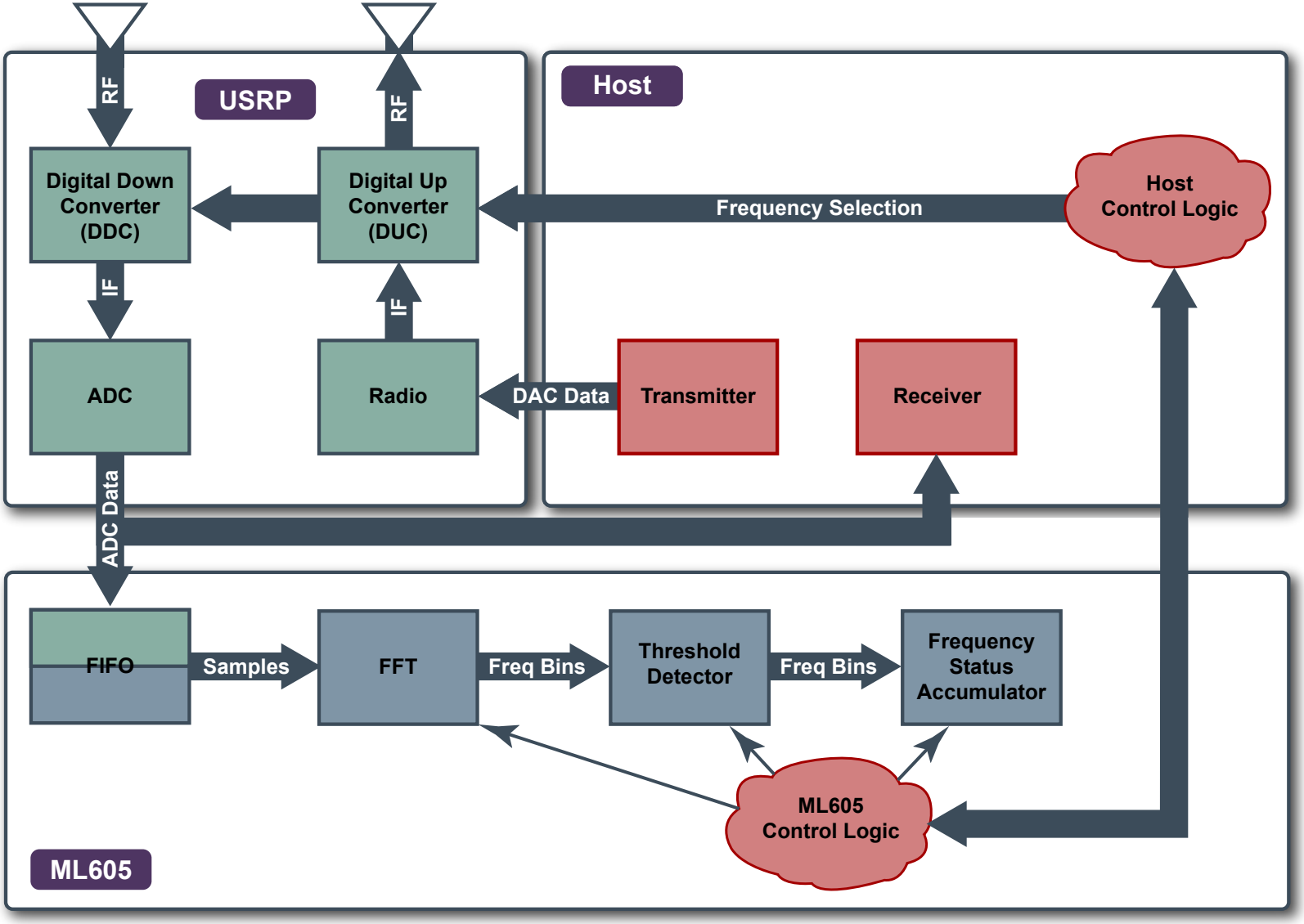


System Diagram – Spectrum Sensing without CRUSH



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

System Diagram – Spectrum Sensing with CRUSH



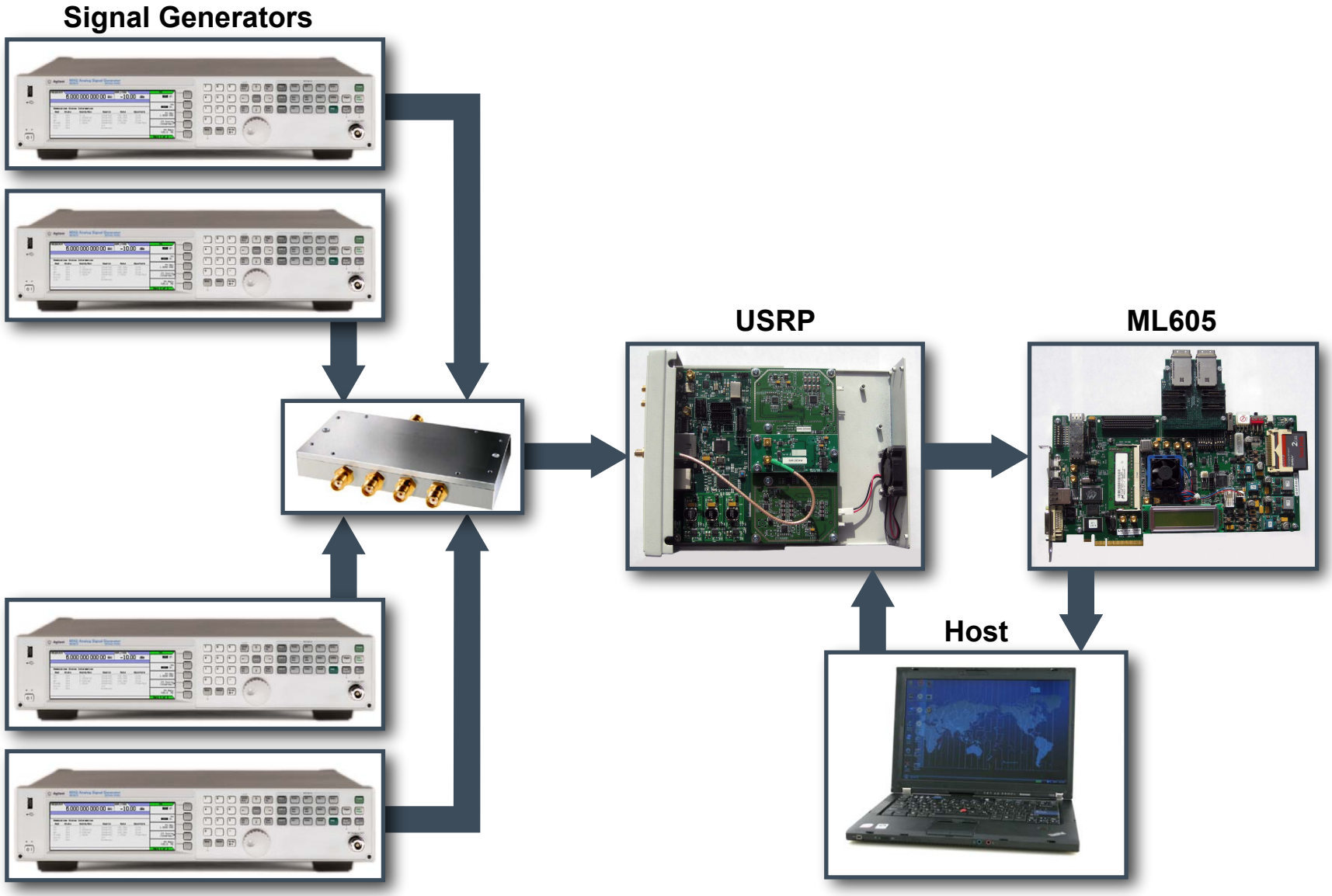
Advantages of CRUSH

- 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
- Reconfigurable hardware allows for parameters such as FFT size to be changed in real time
- New protocols with functionality partly residing on host and partly on the radio are now possible

Outline

- Cognitive Radio overview
- Project motivation and goals
- Hardware platform
- Spectrum sensing algorithm
- Results of implementation
- Summary

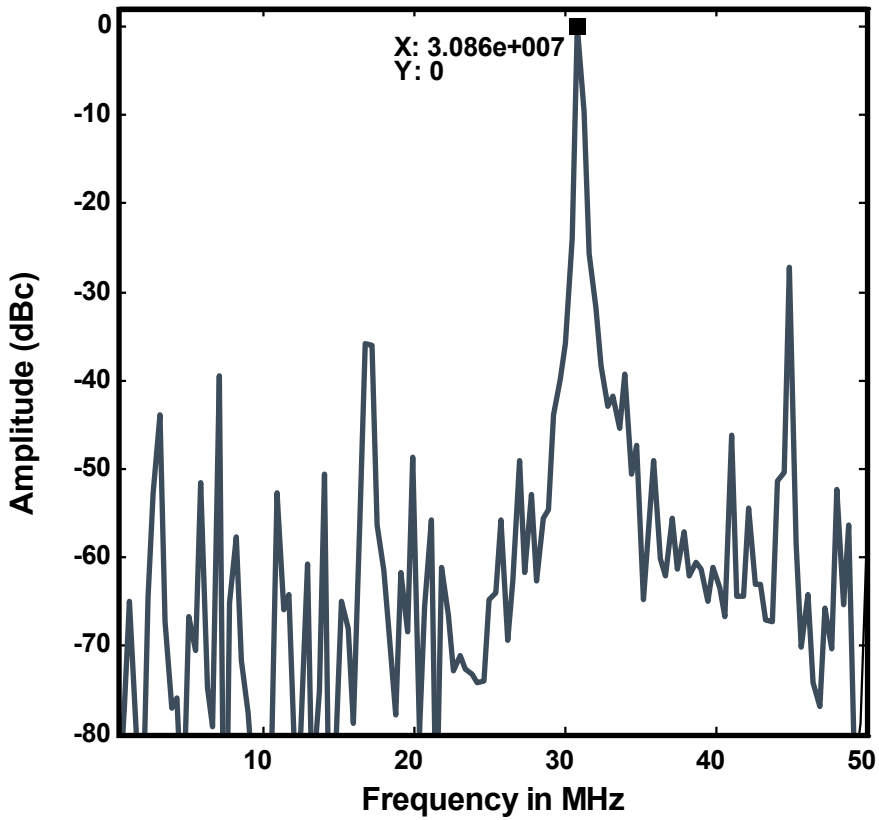
Test Setup – Functional Verification



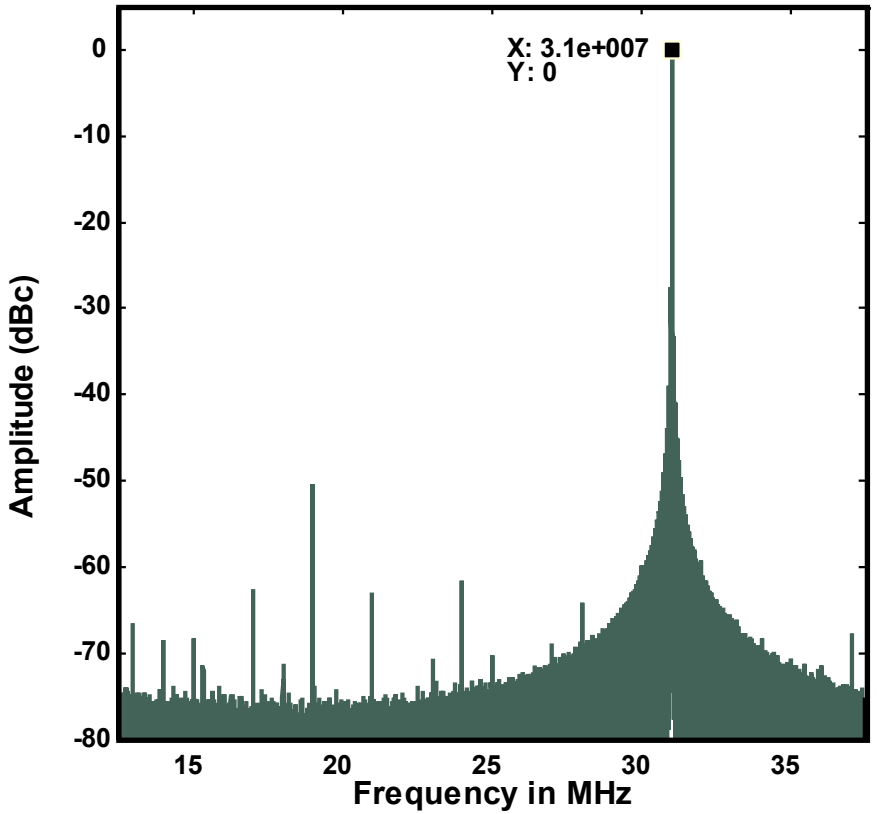
Results – Functionality Verification

- Lab measurement with one tone at 31 MHz and a center frequency of 25 MHz

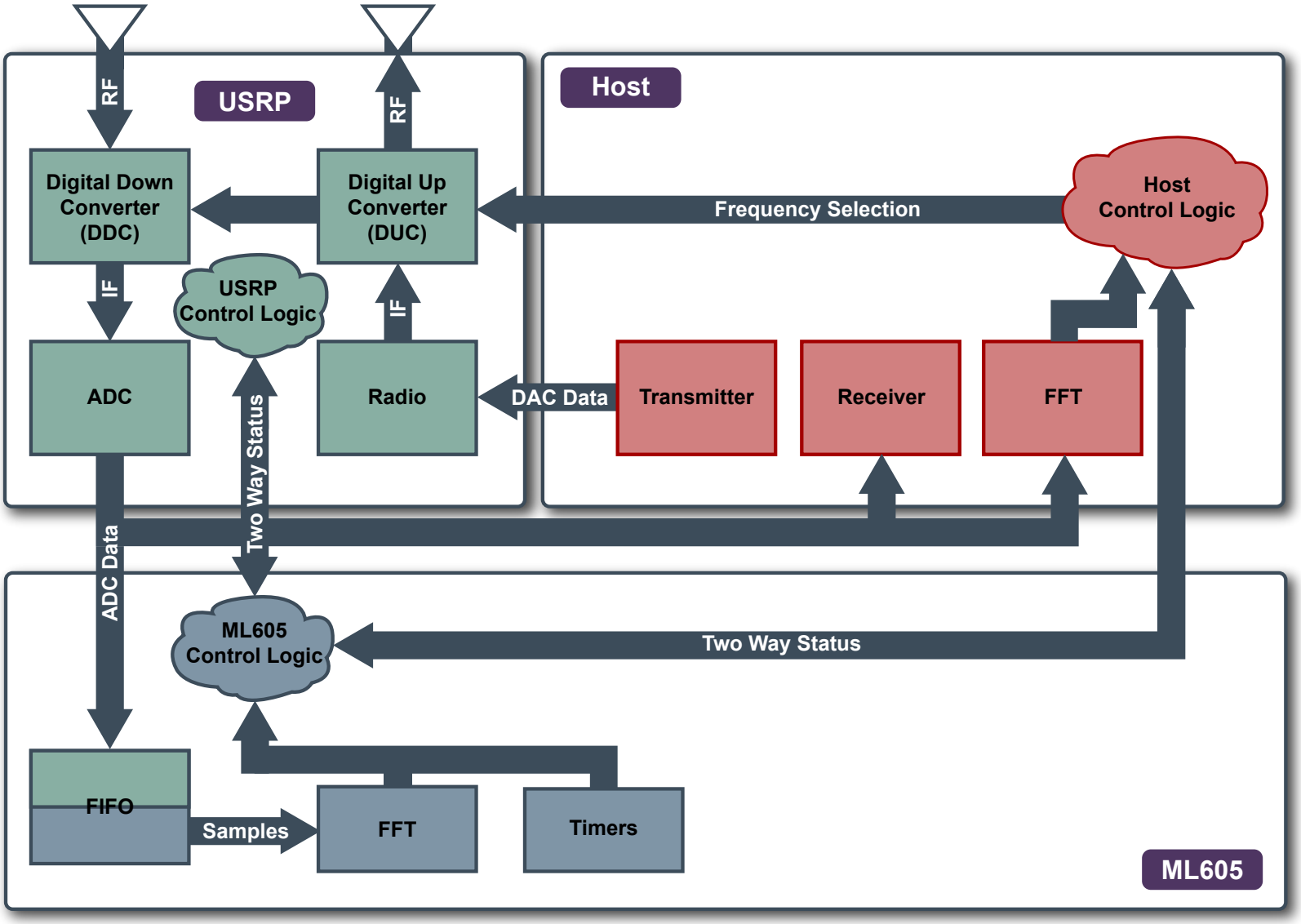
256-point FFT – Processed by FPGA



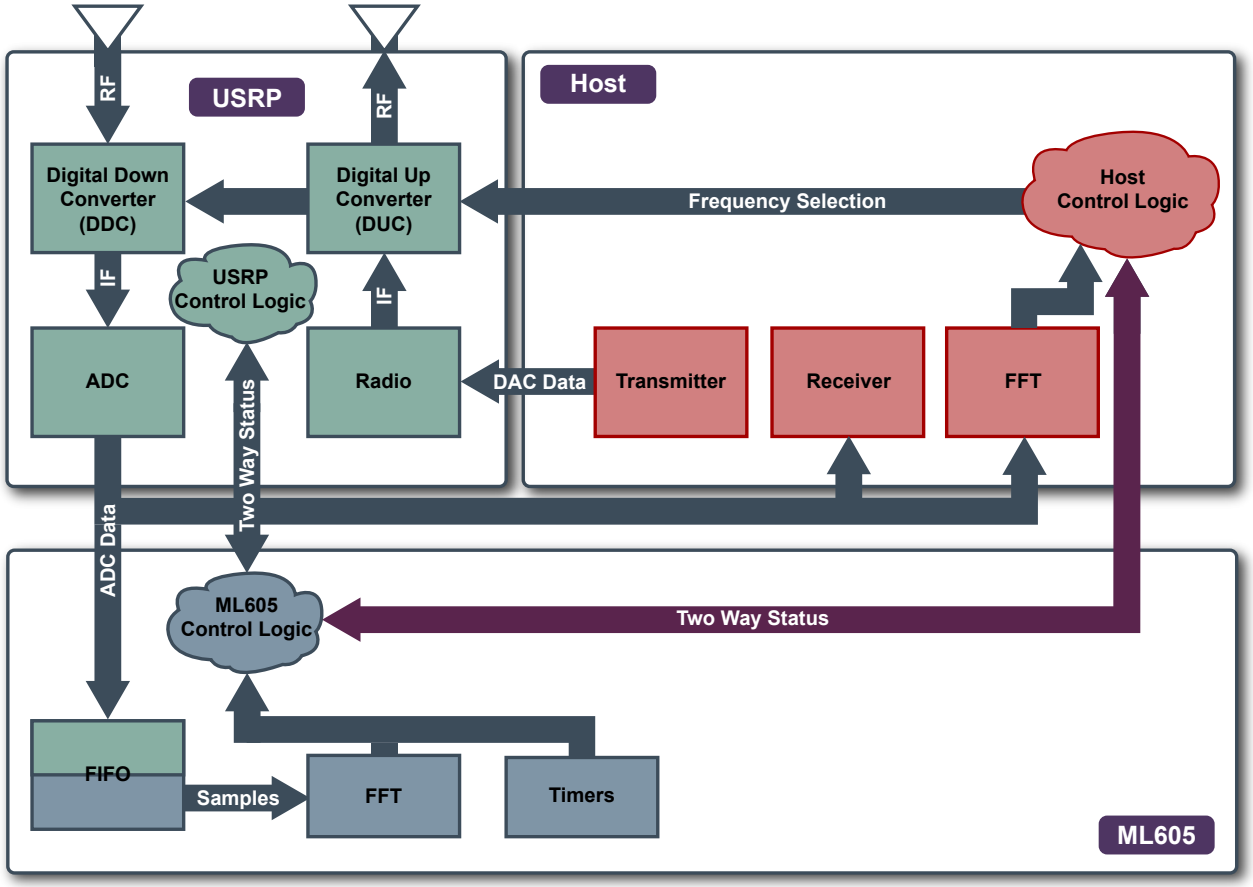
FFT of USRP data Processed in Matlab



Test Setup – Runtime Analysis

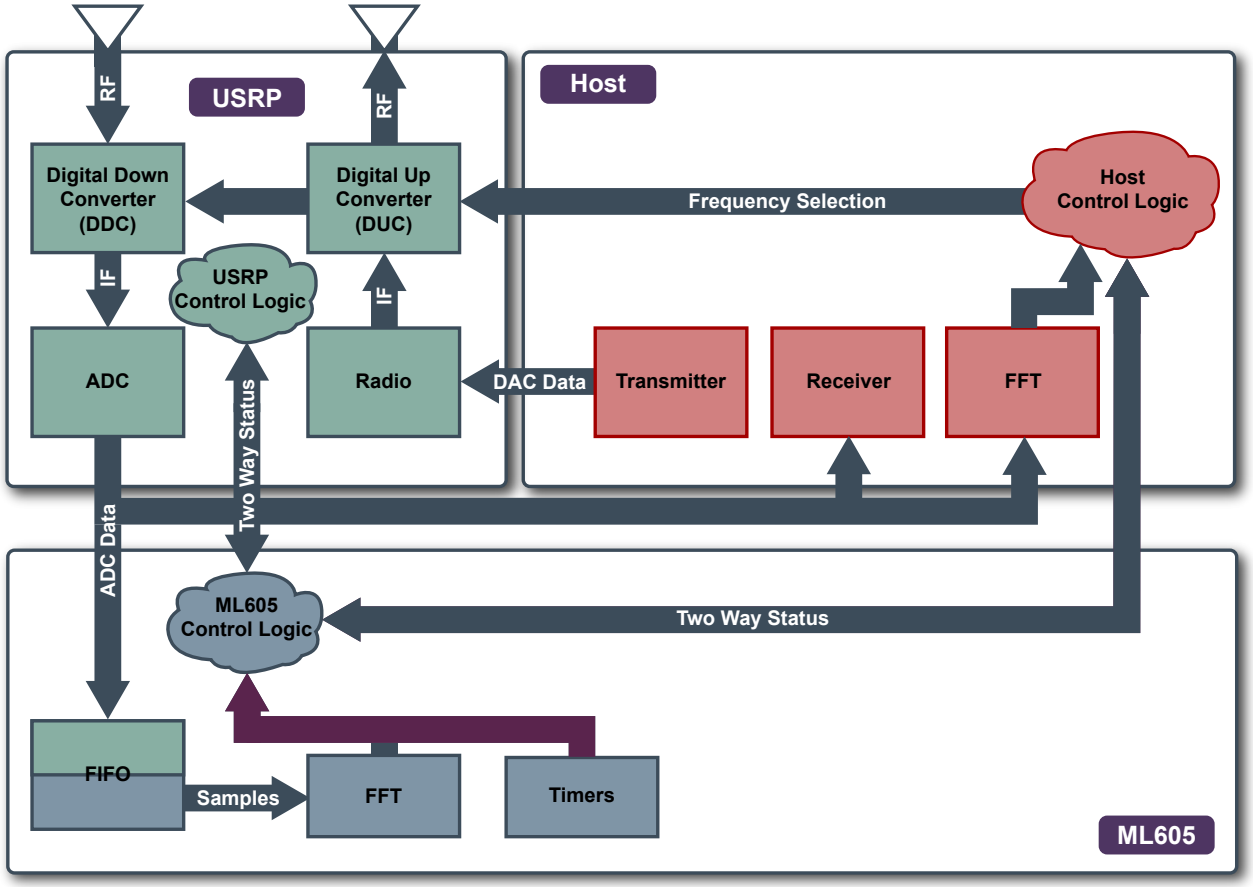


Test Setup – Runtime Analysis



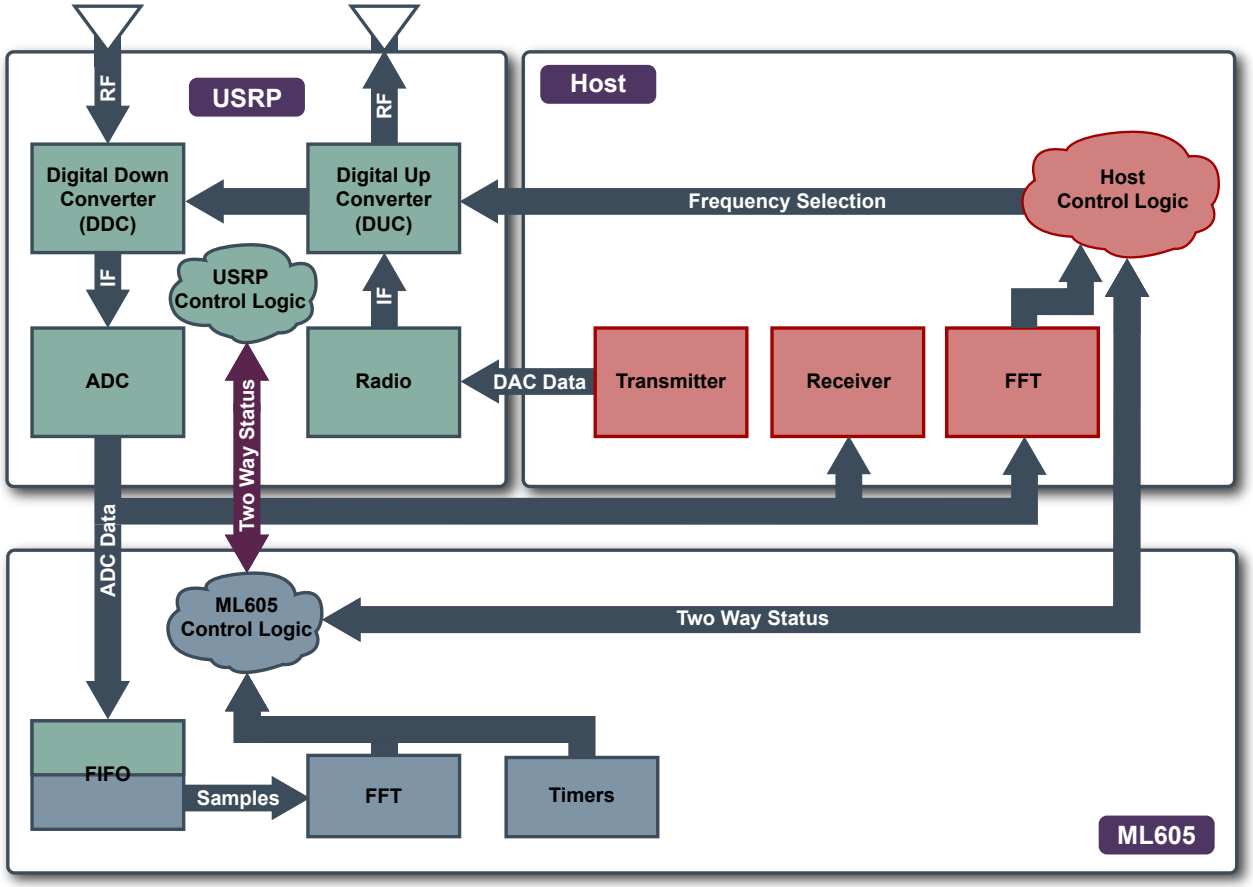
1. **Host -> FPGA (start test)**
2. FPGA (reset timers)
3. FPGA -> USRP (start test)
4. USRP -> Host/FPGA (test pattern)
5. FPGA FFT Done (stops timer)
6. Host -> FPGA (FFT Done)
7. Host FFT Done (stops timer)

Test Setup – Runtime Analysis



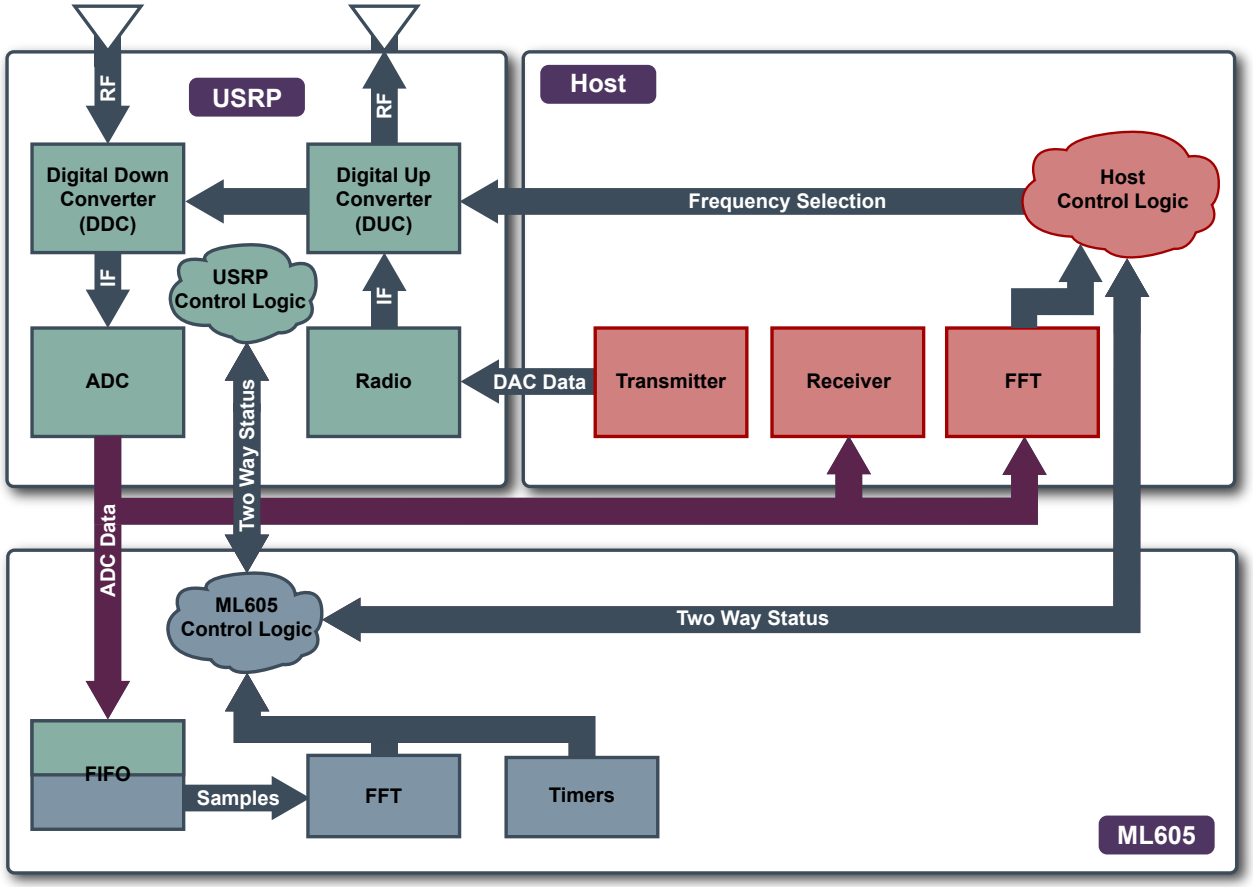
1. Host -> FPGA (start test)
2. **FPGA (reset timers)**
3. FPGA -> USRP (start test)
4. USRP -> Host/FPGA (test pattern)
5. FPGA FFT Done (stops timer)
6. Host -> FPGA (FFT Done)
7. Host FFT Done (stops timer)

Test Setup – Runtime Analysis



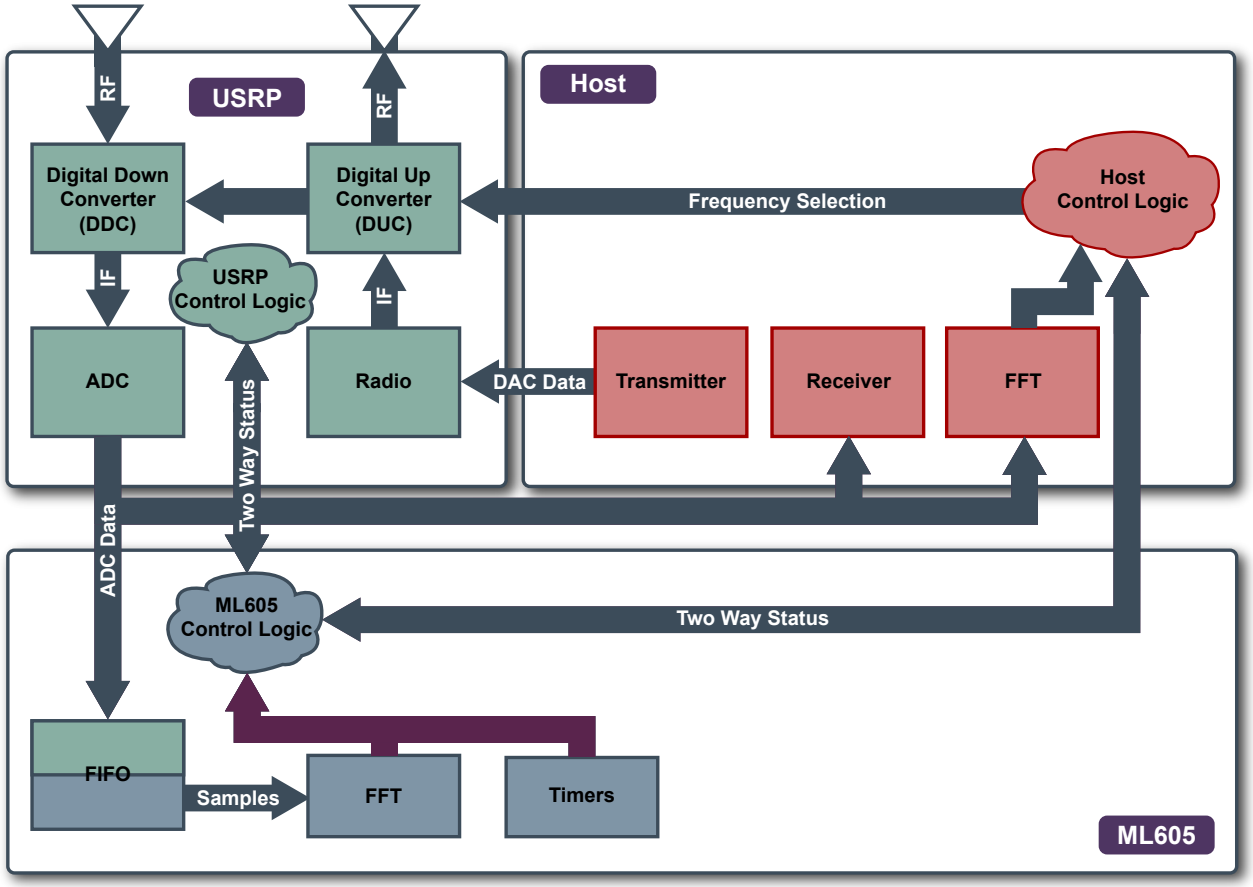
1. Host -> FPGA (start test)
2. FPGA (reset timers)
3. **FPGA -> USRP (start test)**
4. USRP -> Host/FPGA (test pattern)
5. FPGA FFT Done (stops timer)
6. Host -> FPGA (FFT Done)
7. Host FFT Done (stops timer)

Test Setup – Runtime Analysis



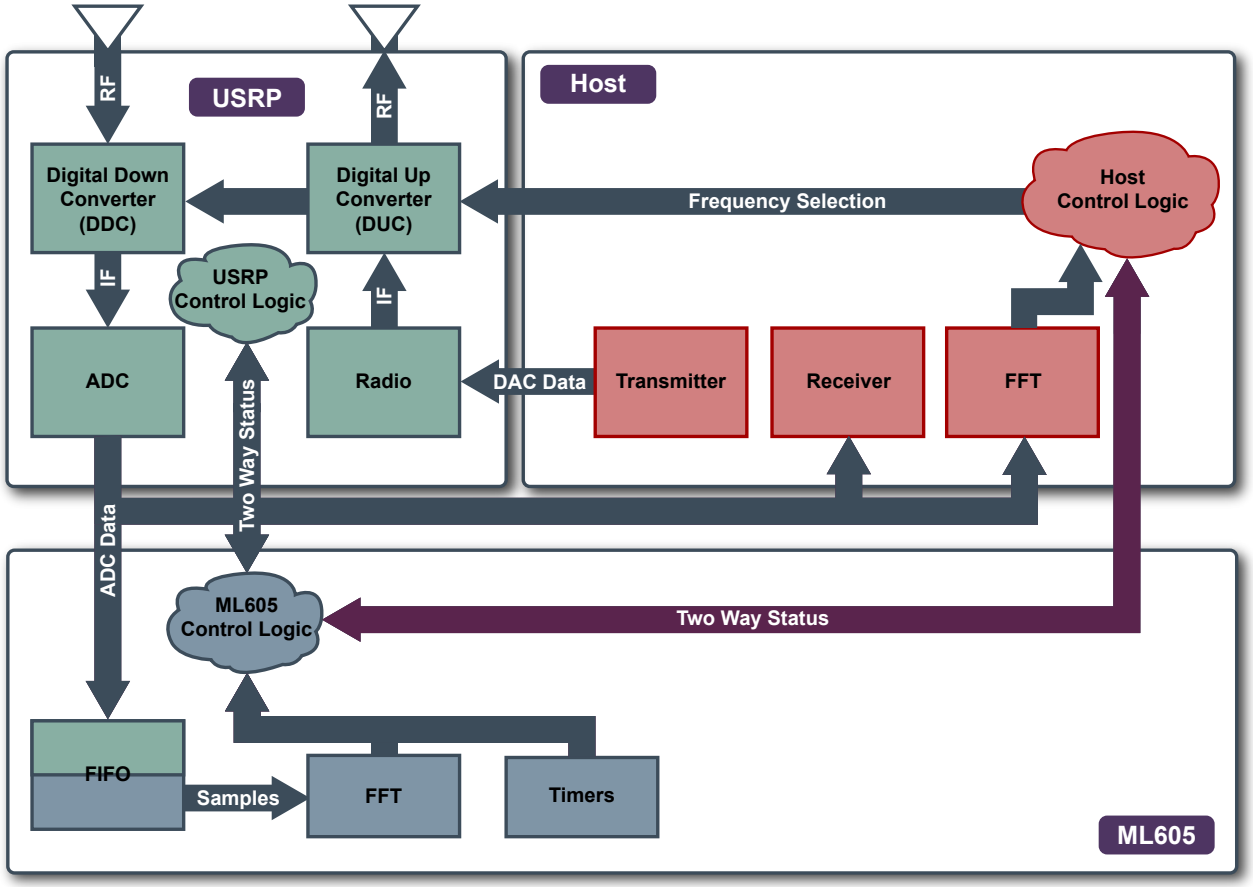
1. Host -> FPGA (start test)
2. FPGA (reset timers)
3. FPGA -> USRP (start test)
4. **USRP -> Host/FPGA (test pattern)**
5. FPGA FFT Done (stops timer)
6. Host -> FPGA (FFT Done)
7. Host FFT Done (stops timer)

Test Setup – Runtime Analysis



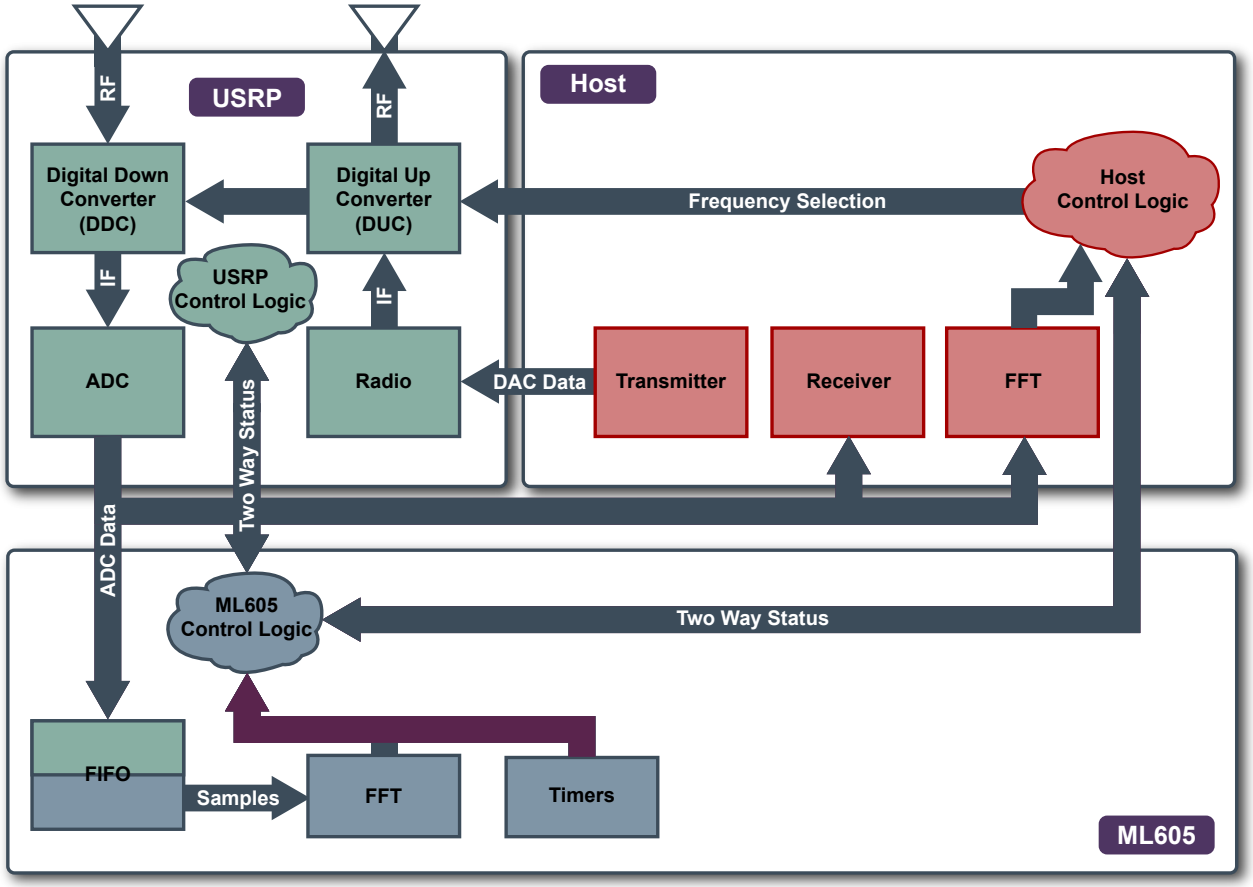
1. Host -> FPGA (start test)
2. FPGA (reset timers)
3. FPGA -> USRP (start test)
4. USRP -> Host/FPGA (test pattern)
5. **FPGA FFT Done (stops timer)**
6. Host -> FPGA (FFT Done)
7. Host FFT Done (stops timer)

Test Setup – Runtime Analysis



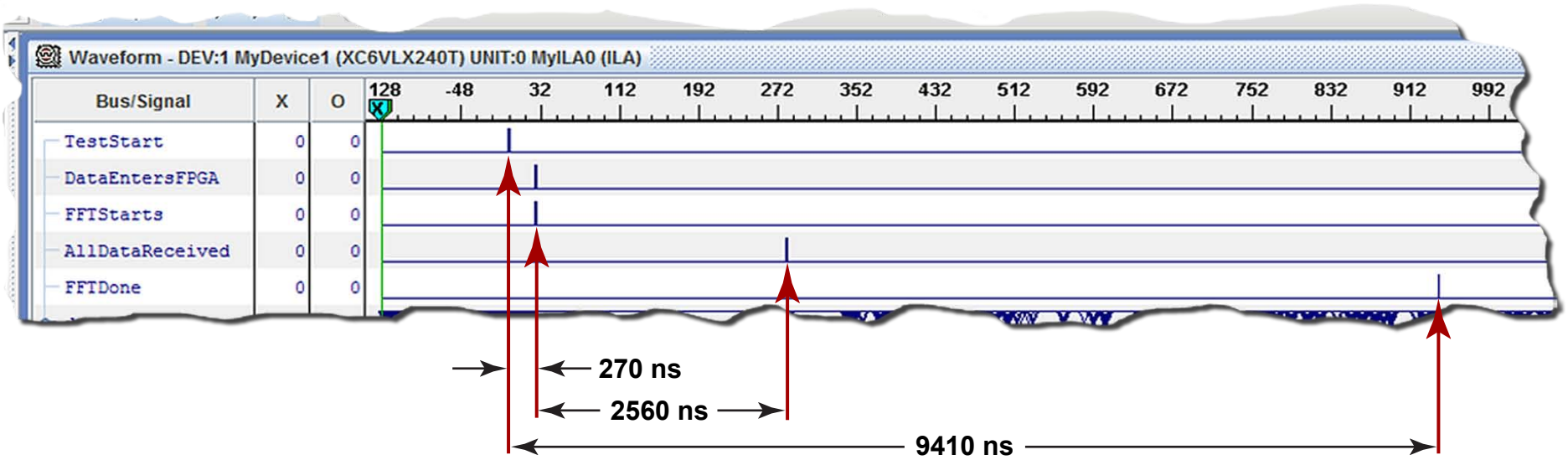
1. Host -> FPGA (start test)
2. FPGA (reset timers)
3. FPGA -> USRP (start test)
4. USRP -> Host/FPGA (test pattern)
5. FPGA FFT Done (stops timer)
6. **Host -> FPGA (FFT Done)**
7. Host FFT Done (stops timer)

Test Setup – Runtime Analysis



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

256-point FPGA FFT Timing Analysis



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**

Results – Runtime 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 – Runtime 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

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 receiver
- Implemented spectrum sensing on CRUSH
  - Achieved more than 100 × performance for FFT sizes of interest
  - Reduced load on host computer
  - Fully configurable

Future Work

- Integrate the spectrum sensing module into research on Cognitve 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

Questions?

Miriam Leeser
mel@coe.neu.edu
http://www.coe.neu.edu/Research/rcl/index.php

Kaushik Chowdhury
krc@ece.neu.edu
http://indigo.ece.neu.edu/~krc/#research

George Eichinger
eichinger.g@husky.neu.edu