An FPGA Spectrum Sensing Accelerator for Cognitive Radio

George Eichinger
Miriam Leeser
Kaushik Chowdhury

NEWSDR’11

01 October 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 Radios Overview

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

Radio Environment

Spectrum Mobility

Spectrum Sensing

Spectrum Sharing

Spectrum Decision

Transmitted Signal

RF Stimuli

Primary User Detection

Spectrum Hole

Decision Request

Spectrum Characterization
Cognitive Cycle

- Transmitted Signal
- Spectrum Mobility
- Decision Request
- Spectrum Sharing
- Spectrum Sensing
- Primary User Detection
- Spectrum Hole
- RF Stimuli
- Radio Environment
- Spectrum Decision
- Focus of This Talk
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**

\[ T_s = t_s + T \]

*Sensing Cycle: \( T_s = t_s + T \)*
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

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
Outline

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

1. Receive RF Data
2. Perform FFT
3. Apply Threshold
4. Report Data

Amplitude vs. Time
Amplitude (dB) vs. Frequency
Amplitude (dB) vs. Frequency
Boolean vs. Frequency
System Diagram – Spectrum Sensing without CRUSH

- All processing occurs on the host
- No real-time guarantee
System Diagram – Spectrum Sensing with CRUSH

- Digital Down Converter (DDC)
- Digital Up Converter (DUC)
- ADC
- Radio
- FIF0
- FFT
- Threshold Detector
- Frequency Status Accumulator
- ML605
- Host
- USRP
- Frequency Selection
- DAC Data
- ADC Data
- ML605 Control Logic
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

Signal Generators

USRP

ML605

Host
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)
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

<table>
<thead>
<tr>
<th>Action</th>
<th>Clocks</th>
<th>Time (µs)</th>
<th>Incremental (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Test</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Data enters FPGA Clock Domain</td>
<td>27</td>
<td>0.27</td>
<td>0.27</td>
</tr>
<tr>
<td>FFT Starts</td>
<td>27</td>
<td>0.27</td>
<td>0</td>
</tr>
<tr>
<td>All Data inside FPGA</td>
<td>256</td>
<td>2.56</td>
<td>2.83</td>
</tr>
<tr>
<td>FFT Complete</td>
<td>941</td>
<td>9.41</td>
<td>6.58</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>941</strong></td>
<td><strong>9.41</strong></td>
<td><strong>9.41</strong></td>
</tr>
</tbody>
</table>
### Results – Runtime Analysis

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

- Speed-up between 8 and 774 x
- FPGA timing scales log linear with FFT size
- Host timing driven by packet transmit time and internal buffering
## Results – Runtime Analysis

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

- 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 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
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
Issues Using the USRP N210 FPGA

- USRP N210 has a Xilinx Spartan 3A-DSP3400A
- Timing closure is an issue
  - 100 MHz clock speed makes timing closure more difficult
- Long build time and complex existing firmware
- Spartan 3A DSP line is two generations behind the latest Xilinx products
- USRP N210 FPGA is an improvement over the FPGA used in the USRP 2 but still not enough
  - 64 k Xilinx FFT takes 367 RAM36 blocks, more than the entire USRP N210 FPGA
- No ability for partial reconfiguration applications
- Limited external IO and memory