Implementation of a DRM+ transmitter in the GNU Radio software radio framework

Felix Wunsch
Mentor: Jens Elsner
Outline

- Introduction
- DRM Standard
- Implementation in GNU Radio
- Performance
- Demonstration
- Conclusion
Introduction

- Why digital radio?
  - News and data streams possible
  - Multiple services on one channel
  - Better performance in low SNR environments

- DRM: Digital Radio Mondiale
  - Standard for transmission below 30 MHz (DRM30)
  - Inaugural transmission took place in 2003
  - DRM+ was added to support 30 MHz - 174 MHz band in 2009

- First open source implementation of DRM+
Introduction – Development Process

- Understand the standard
  - Design transmitter and its inverse in MATLAB

- Test against Dream
  - Implements DRM30 receiver
  - Developed at the TU Darmstadt

- Test against commercial DRM receiver

- Port to GNU Radio

- Create unit tests and optimize performance
DRM Standard

- OFDM system

- Variable data rate, bandwidth and error robustness

- Subset of MPEG-4 standard used for audio coding
  - AAC, CELP, HVXC

- Three logical channels
  - Main Service Channel (MSC)
  - Service Description Channel (SDC)
  - Fast Access Channel (FAC)

- All stages‘ parameters tightly intertwined to optimize performance
DRM Standard – Physical Layer

Two main parameters:
- RM: Robustness Mode (A - E, error robustness)
- SO: Spectrum Occupancy (0 - 5, occupied bandwidth)

<table>
<thead>
<tr>
<th></th>
<th>DRM30</th>
<th>DRM+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>4.5 kHz ... 20 kHz</td>
<td>100 kHz</td>
</tr>
<tr>
<td>Symbol Duration</td>
<td>9.33 ms ... 24 ms</td>
<td>2.25 ms</td>
</tr>
<tr>
<td>Guard Interval</td>
<td>2.66 ms ... 7.33 ms</td>
<td>0.25 ms</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>4.8 kbps ... 72 kbps</td>
<td>37.3 kbps ... 186.4 kbps</td>
</tr>
<tr>
<td>Frame Duration</td>
<td>400 ms</td>
<td>100 ms</td>
</tr>
</tbody>
</table>
DRM Standard – Flow graph

Conceptual DRM transmission block diagram

- Audio Source
- AAC Encoder
- Mux
- Scrambler
- MLC
- Cell Interleaver
- Pilot Generator
- OFDM Cell Mapping
- OFDM
- complex baseband

Multi Level Coding (MLC)
- Channel Encoding
- Bit Interleaver
- Bit to Symbol
Conceptual DRM transmission block diagram

- Audio Source
  - AAC Encoder
  - Mux
  - Scrambler
  - MLC
  - Cell Interleaver
  - Pilot Generator
  - OFDM Cell Mapping
  - OFDM
  - complex baseband

- Data Source
- SDC
- FAC

Multi Level Coding (MLC)
- Channel Encoding
- Bit Interleaver
- Bit to Symbol
Conceptual DRM transmission block diagram

Audio Source

Data Source

AAC Encoder

Mux

Scrambler

MLC

Cell Interleaver

Pilot Generator

OFDM Cell Mapping

OFDM

complex baseband

SDC

FAC

Multi Level Coding (MLC)

Channel Encoding

Bit Interleaver

Bit to Symbol
Conceptual DRM transmission block diagram

Audio Source -> Audio Encoder
Data Source -> Mux

Felix Wunsch
Implementation of a DRM+ transmitter in the GNU Radio software radio framework
Conceptual DRM transmission block diagram

1. Audio Source
2. AAC Encoder
3. Mux
4. Scrambler
5. MLC
6. Cell Interleaver
7. Pilot Generator
8. OFDM Cell Mapping
9. OFDM
10. Bit Domain
11. Symbol Domain
12. SDC
13. FAC
14. Multi Level Coding (MLC)
15. Channel Encoding
16. Bit Interleaver
17. Bit to Symbol
Conceptual DRM transmission block diagram

Audio Source → Mux → Scrambler → MLC → OFDM Cell Mapping → OFDM → complex baseband

Data Source → Mux → Scrambler → MLC → OFDM Cell Mapping → OFDM → complex baseband

SDC → Mux → Scrambler → MLC → OFDM Cell Mapping → OFDM → complex baseband

FAC → Mux → Scrambler → MLC → OFDM Cell Mapping → OFDM → complex baseband

Multi Level Coding (MLC)
- Channel Encoding
- Bit Interleaver
- Bit to Symbol
DRM Standard – Flow graph (cont.)

- Different pilot cells
  - Frequency reference cells
  - Time reference cells
  - Gain reference cells
  - AFS reference cells

- Variable no. of subcarriers
- Variable carrier spacing
Implementation of a DRM+ transmitter in the GNU Radio software radio framework

Communications Engineering Lab
Prof. Dr. rer. nat. Friedrich K. Jondral
Implementation in GNU Radio

- New module „gr-drm“
- Fully integrated into GRC
  - No deep knowledge of DRM required
  - Flow graphs can be used as-is
- As modular as possible
  - Generic blocks
    - Puncturing
    - Interleaver
    - …
- http://github.com/kit-cel/gr-drm
Implementation in GNU Radio – Wish List

- Reload XML definitions
  - Saves time

- Hier block transparency
  - More depth
## Performance

### Test environment
- Intel i5-2520M
- GNU Radio 3.6.1
- Ubuntu 12.04 (64 bit)
- USRP N210 with BasicTX

### Transmitter configuration
- RM A, SO 3 (10 kHz), 64 QAM
- 26.56 kbps (AAC mono)

- Real-time capable
- Low CPU consumption
- Suboptimal buffer allocation
- AAC encoder library dominates CPU consumption

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2323</td>
<td>felixwun</td>
<td>-30</td>
<td>0</td>
<td>3172m</td>
<td>298m</td>
<td>255m</td>
<td>S</td>
<td>19</td>
<td>3.8</td>
<td>3:06.79</td>
<td>python</td>
</tr>
</tbody>
</table>
Demonstration

- Signal generation with gr-drm GRC flow graph
- Decoding with NEWSTAR DR-111 (live)
- Parameters can be changed directly in GRC
- Configuration: 10 kHz bandwidth, RM A, 24 kHz wav-file, 64-QAM
Conclusion

- Great GSoC experience
  - Over 5000 lines of code
  - Daily interaction with mentor
  - Very helpful community

- DRM30 transmitter has been implemented in GNU Radio
  - DRM+ is implemented but untested

- Next steps:
  - Integration of Fraunhofer AAC encoder
  - Implementation of DRM Receiver in GNU Radio

- DRM+ might become the new standard for digital broadcasting in Brazil
Implementation of a DRM+ transmitter in the GNU Radio software radio framework

Q & A