DAB
Software Receiver Implementation

Andreas Müller
Supervisor: Michael Lerjen

ETH – ITET – CTL

June 13, 2008
1 Introduction
   • Task
   • Software Defined Radio
   • DAB
   • GNU Radio and USRP

2 Implementation
   • OFDM Synchronisation
   • OFDM Demodulation

3 Evaluation
   • Test Setup
   • Results

4 Conclusions

5 Questions
1 Introduction
   - Task
   - Software Defined Radio
   - DAB
   - GNU Radio and USRP

2 Implementation

3 Evaluation

4 Conclusions

5 Questions
Goal: Implementation of a real-time DAB receiver as SDR

**SDR**
Software Defined Radio $\rightarrow$ (almost) all signal processing in software

**DAB**
Digital Audio Broadcasting $\rightarrow$ digital radio technology standardized by ETSI

**Real-time**
Process data as fast as it arrives $\rightarrow$ 2 MSPS or 16 MB/s
Software Defined Radio

**Idea**
Digitize the signal and do all the signal processing in (high level, architecture independent) software.

**Strengths**
- Flexibility
- Reusable code, fast development cycle
- Cognitive radio: Adapts itself dynamically to RF environment → better spectral and power efficiency

**Weaknesses**
- Limited sample rate and dynamic range of ADCs and DACs → analog front end needed for filtering
- Resource usage, energy consumption, cost
Digital Audio Broadcasting (DAB) Specification

Modes

Four modes for different frequency ranges and RF characteristics
- Presentation: Mode I (Code: All Modes)

DAB Mode I OFDM signal
- Frames with 76 OFDM symbols (1 pilot, 75 data)
- Null symbols (energy zero) to separate frames
- 1536 subcarriers at 1 kHz & central carrier zero → 1.537 MHz
- D-QPSK modulation for each subcarrier
- Cyclic prefix: 504 samples → SFN with max. TX distance 74 km

Upper Layers
- Punctured convolutional coding
- Energy dispersal, Time interleaving
- MPEG 2 audio coding
GNU Radio

Overview
- Open source framework for real-time software radios
- Provides many common building blocks: FFT, FIR & IIR filters, mathematical operations, AGC, modulation & demodulation, ...

Flow Graph Concept
- Programmer creates a directed graph for sample flow
- Signal processing blocks are written in C++ and wired together in Python

Signal Processing Block
- `work()` function receives a number of samples from scheduler
- Block processes as many samples as possible and returns the number of consumed and produced samples
Universal Software Radio Peripheral (USRP)

**Hardware**
- Interface between computer and antenna is needed
- Most commonly used with GNU Radio: USRP

**USRP**
- Two AD9862 Mixed Signal Front-End Processors
  - 4 DACs with sampling rate 128 MSPS → 2 I/Q TX channels
  - 4 ADCs with sampling rate 64 MSPS → 2 I/Q RX channels
- Altera Cyclone FPGA for conversion to/from baseband, decimation/interpolation, multiplexing and buffering
- Cypress FX2 USB 2.0 interface
- Daughterboards according to selected frequency range
Introduction

2 Implementation
   - OFDM Synchronisation
   - OFDM Demodulation

3 Evaluation

4 Conclusions

5 Questions
### OFDM I – Synchronisation

#### Time Synchronisation
- Frame start detection must be accurate, as the other blocks depend on it.
- Can easily be done by looking at the energy of the signal (Null symbols).
- Implemented with moving sum, inverter and peak detector.

#### Frequency Synchronisation
- Small subcarrier spacing $\rightarrow$ accurate synchronisation needed.
- Fine frequency synchronisation (offsets $< \text{subcarrier spacing}$):
  - compare cyclic prefix to end of the symbol $\rightarrow$ fine frequency offset can be estimated from the phase offset.
- Coarse frequency synchronisation (offsets $> \text{subcarrier spacing}$):
  - done after fine frequency synchronisation and after FFT.
  - simply shift signal in the frequency domain $\rightarrow$ very efficient.
Besides time and frequency synchronisation, demodulation is rather straightforward:

- Sampler: Remove cyclic prefix, pack each OFDM symbol in a vector
- FFT
- Calculate phase difference (undo the D in D-QPSK)
- Magnitude equalization (only needed for soft bits, as the information is only in the phase)
- Undo frequency interleaving: Mix symbols according to sequence specified in DAB standard
- I and Q components contain independent bits → simply check if $\Re(x) > 0$ and $\Im(x) > 0$
3 Evaluation
   • Test Setup
   • Results
Test Setup

Simulation Cycle
- Generate random bytes
- Modulation
- Channel-model distorts OFDM signal
- Demodulation
- Calculate BER from original and received bytes

Channel Model
- Sampling frequency offset modeled by fractional interpolator
- Multipath propagation modeled with FIR filter
- Frequency offset (signal source + multiplication block)
- AWGN (noise source + adder block)
Results – SNR
Results – Effects of Multipath Propagation

![Graph showing BER vs Echo delay for different DAB modes and cyclic prefix lengths.]

<table>
<thead>
<tr>
<th>DAB Mode</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclic Prefix Length</td>
<td>504</td>
<td>126</td>
<td>63</td>
<td>252</td>
</tr>
</tbody>
</table>
Conclusions

- Real-time processing is possible
- FIBs successfully decoded
- No audio yet

Challenges

- Very efficient algorithms and programming needed
- Many signal processing papers are written from a primarily mathematical perspective

Advantages

- Same code for simulation and actual receiver
- Open source code of existing blocks helps understand algorithms
- Existing code can sometimes be adapted for new purposes
- GNU Radio: Large and enthusiastic community
1. Introduction
2. Implementation
3. Evaluation
4. Conclusions
5. Questions
Thank you for your attention.