Master’s Thesis

Implementation of MPEG-4 AAC Decoder for Digital Radio Navigator on FPGA

Mohammed Shaaban
Lund University – Sweden
2012-11-16
AGENDA

Introduction

SurfOnHertz Project

MPEG-4 Decoder Overview

Proposed Decoder Architecture

Results

Conclusion
• Digital radios receivers.

• New functionalities besides receiving radio stations.

• Traditional to smart radios receivers world.

• Limited bandwidth channel.

• Audio compression.
SURFONHERTZ PROJECT

Source: DIGITAL RADIO MONDIALE (DRM) A BROADCASTER’S GUIDE
- **Inputs**
  - Parallel $N$ AAC Coded Streams

- **Outputs**
  - Parallel $N$ PCM Audio Streams
MPEG-4 Decoder Overview

AAC file \rightarrow \text{MPEG-4 AAC Decoder (decompression)} \rightarrow \text{Wav file}
\text{PCM audio samples}

**Advantages:**
- more efficient than mp3
- High performance
- High quality features.

**Applications**
- Internet audio
- Digital audio podcasting
- Mobile phones media players that supports AAC files

Source: DIGITAL RADIO MONDIALE (DRM) A BROADCASTER’S GUIDE

Mohammed Shaaban | Lund University | 2012-11-12 | 6
**MPEG-4 Decoder Overview**

- **Data**
- **Control**
- **Required tools**
- **Optional tools**

**Required tools**
- MPEG-2 Profiles
  - Main Profile
  - Low-Complexity (LC)
  - Scalable Sampling Rate (SSR)

**Optional tools**

**MPEG-4 Decoder Flowchart**

1. **AAC Bitstream**
2. **Bit Stream Demux**
3. **Huffman Decoder**
4. **Inverse Quantization**
5. **Rescaling**
6. **Mid/Side**
7. **Intensity/Coupling**
8. **TNS**
9. **Filterbank**
10. **Gain Control**
11. **PCM Audio Samples**
AAC Bitstream Types

- One header for each frame.
- Data broadcasting applications.

- One header in the whole file.
- Files based applications.
AAC ADTS Frame Structure

- **Header**
- **RDB[0]**
- **RDB[1]**
- **RDB[2]**
- **RDB[3]**
- **RDB[4]**
- **RDB[5]**
- **RDB[6]**
- **RDB[7]**

**Example of RDB contents**
- **Syn_ele_id**
- **FILL_ELE**
- **Syn_ele_id**
- **SCE**
- **Syn_ele_id**
- **TRIM**

- **ID = 6**
- **ID = 0**
- **ID = 7**

**SCE: Single Channel Element**
- **Global gain**
- **Section data**
- **Scale factor data**
- **Pulse data**
- **TNS data**
- **Spectral data**

**Side Information**

**1024 samples**
MPEG-4 Decoding Algorithm

Start

Input Bitstream

Is Sync. Word detected

Yes

Parse Frame Header

Read syn_ele_id

id = FIL

Yes

Read Fill Data

id = SCE

No

No

Read Side_info

Huffman Decoder

Spectral data (x_qunt)

Inverse Quantization

(x_inv_qunt)

Rescaling

(x_rescale)

Yes

IDMCT

Y

2048 samples

Windowing

Z (2048)

Z2_Prev

Z_1 (1024)

Z_2 (1024)

Overlapping

Time_out

Store for next frame
PROPOSED DECODER SPECIFICATION

1. ADTS Bitstream
2. Single Channel
3. LC Profile
4. Fixed-Point
**PROPOSED DECODER SPECIFICATION**

**Input:**
AAC ADTS Bitstream

32-bit

**Output:**
PCM Samples

16-bit

MPEG-4 AAC Audio Decoder
PROPOSED DECODER ARCHITECTURE

Input:
AAC ADTS Bitstream

Demux & Huffman → IQ & Rescaling → Filter Bank → PCM 16-bit

Output:
PCM Samples

Mohammed Shaaban
Lund University
2012-11-16
PROPOSED DECODER ARCHITECTURE

Mohammed Shaaban
Lund University
2012-11-16
Proposed Decoder Architecture

Input: AAC ADTS Bitstream

Demux & Huffman

IQ & Rescaling

Filter Bank

PCM 16-bit

Output: PCM Samples

Mohammed Shaaban
Lund University
2012-11-16
PROPOSED DECODER ARCHITECTURE

(Bitstream data)

Input_Buffer

Reg_1

Reg_0

Barrel Shifter

32-bit

21-bit

16-bit

32-bit

Huffman Decoder

Demux & Huff

IQ & Resca

Filter Bank

PCM 16-bit

DEMUX-HUFF

Demux_Controller

TO Spectral Data RAM

TO Scale Factor Data RAM

Mohammed Shaaban

Lund University

2012-11-16
PROPOSED DECODER ARCHITECTURE

REG_1 | REG_0
---|---
011111100000011101
011111100000011101
011111100000011101
011111100000011101

Barrel shifter output: B G A

<table>
<thead>
<tr>
<th>DECODED CODE</th>
<th>CODE WORD</th>
<th>CODE LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>01</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>101</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>110</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>1110</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>11110</td>
<td>5</td>
</tr>
<tr>
<td>H</td>
<td>111110</td>
<td>5</td>
</tr>
</tbody>
</table>

Mohammed Shaaban
Lund University
2012-11-16
17
**PROPOSED DECODER ARCHITECTURE**

**Data_IN** (from the barrel shifter output)

- **Codebook No.**
  - **HCB_Mux**
  - **HCB_Len_Mux**

- **Reg_W**, **Reg_X**, **Reg_Y**, **Reg_Z**

- **Huff_Code_len**, **HSF_len**

- **Coeff_Mux**

- **TO Spectral Data RAM**

- **Global_Gian**
  - **To Scale factors RAM**

---

**Huffman Decoder**

Mohammed Shaaban

Lund University

2012-11-16
**PROPOSED DECODER ARCHITECTURE**

**TABLE II**

**EXAMPLE OF HUFFMAN CODEBOOK AND ITS CORRESPONDING OUTPUTS**

<table>
<thead>
<tr>
<th>Index</th>
<th>Huffman Codeword</th>
<th>Codeword length</th>
<th>Decode codeword y</th>
<th>Decode codeword z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Huffman Table Implementation**

- Parallel search
- Less Area
- One Clock Cycle decoding

---

**Image:**

Tsung-Han Tsai

"Low-Power System Design for MPEG-2/4AAC Audio Decoder Using Pure ASIC Approach"


---

**Fig. 7.** Parallel PLA implementation of Table II.

Mohammed Shaaban

Lund University

2012-11-16

19
The interpolation calculation is taken from FAAD2 2005
**PROPOSED DECODER ARCHITECTURE**

**IQ_RESCE MODULE**

Demux & Huff → IQ & Resca → Filter Bank → PCM 16-bit

\[
X_{\text{rescale}} = X_{\text{inv_qunt}} \times \text{gain}
\]

\[
\text{gain} = 2^{0.25*(sf-100)}
\]

**RESCALING IMPLEMENTED ALGORITHM**

% Exponential part calculation
\[
EXP = sf >> 2 \quad \text{(divide by 4)}
\]

\[
EXP = EXP - 25
\]

% multiplication using shift
\[
X_{\text{rescale}} = IQ << EXP
\]

*The rescaling calculation is taken from FAAD2 2005*
PROPOSED DECODER ARCHITECTURE

IQ_RESC MODULE

Demux & Huff → IQ & Resca → Filter Bank → PCM 16-bit

Spectral data

Address Calc → Comparator

Mux

\( X^{4/3} \) Table

Interpolation Circuit

Mux

Rescaling

To Filter Bank

Mohammed Shaaban
Lund University
2012-11-16
PROPOSED DECODER ARCHITECTURE

IQ_RESC MODULE

Demux & Huff → IQ & Resca → Filter Bank → PCM 16-bit

Diagram details...
PROPOSED DECODER ARCHITECTURE

FILTER BANK MODULE

Demux & Huff    IQ & Resca    Filter Bank    PCM 16-bit

Input
Resc_spect (Freq.)

IMDCT → Windowing → Overlapping

Output
PCM (Time)
**PROPOSED DECODER ARCHITECTURE**

**IMDCT ALGORITHM**

\[ Y(k) \]

Pre-iFFT

\[ Y_1(k) \]

iFFT

\[ Y_2(k) \]

Post-iFFT

\[ Y_3(k) \]

Reordering

\[ x(k) \]

\[ X(k) \text{ is the rescaled 1024 samples} \]

\[ Y(k) = -X(2k) + jX \left( \frac{N}{2} - 1 - 2k \right) \]

Complex multiplication:

\[ Y_1(k) = Y(k) \times Z(k) \]

N/4-point iFFT:

\[ Y_2(k) = \frac{2}{N} \sum_{k=0}^{\frac{N}{4}-1} Y_1(k) e^{\frac{2\pi}{N/4} i k}, \quad 0 \leq i < \frac{N}{4} \]

Complex multiplication:

\[ Y_3(k) = Y_3(k) \times Z(k) \]

\[ Z(k) = e^{j \left( \frac{2\pi}{N} k + \frac{\pi}{4N} \right)} \]

\[ 0 \leq k < N/4 \]
PROPOSED DECODER ARCHITECTURE

IMDCT MODULE

Data_IN

Coef_ROM 512_R

Reg_IM

Reg_RE

D

Coef_ROM 64_R

Coef_ROM 512_I

Coef_ROM 64_I

D

D

Complex_Mul

Xr

Xi

Cr

Ci

Zr

Zi

In_r

Out_i

In_i

Out_r

IFFT

Re_RAM

Im_RAM

Data_Out

Mohammed Shaaban

Lund University

2012-11-16

26
PROPOSED DECODER ARCHITECTURE

WIN-OV MODULE

Demux & Huff → IQ & Resca → Filter Bank → PGM 16-bit → Demux & Huff

IMDCT → Windowing → Overlapping

WINDOWING: $Z = M \times C$

OVERLAPPING: $Y = Z + X$

<table>
<thead>
<tr>
<th>M</th>
<th>IMDCT result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>Win. Result</td>
</tr>
<tr>
<td>C</td>
<td>Sine or KBD function</td>
</tr>
<tr>
<td>X</td>
<td>Z, Y or Z_prev</td>
</tr>
</tbody>
</table>

PIPEDLINE ARCHITECTURE
Proposed Decoder Architecture

PCM_16 Module

- 32-bit to 16-bit conversion
- Pure combinational circuit
Simulation Results

Implementation Results
SIMULATION RESULTS

- SW: Calculate the error.
- HW: Calculate execution time.

\[ \text{rms} = \sqrt{\frac{(H_{out} - S_{out})^2}{N}} \]
SIMULATION RESULTS
RESULTS

SIMULATION RESULTS

\[ \text{RMS} = 5.61 \times 10^{-4} \]
Simulation Results

Number of Operation Cycles in Each Block

<table>
<thead>
<tr>
<th>Block</th>
<th>Clock Cycles</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demux</td>
<td>1.8K</td>
<td>14.92%</td>
</tr>
<tr>
<td>IQ_RESC</td>
<td>4.3K</td>
<td>33.73%</td>
</tr>
<tr>
<td>IMDCT</td>
<td>4.4K</td>
<td>34.99%</td>
</tr>
<tr>
<td>Win_ov</td>
<td>2.1K</td>
<td>16.35%</td>
</tr>
<tr>
<td>Filter bank (IMDCT+OV)</td>
<td>6.5K</td>
<td>51.35%</td>
</tr>
<tr>
<td>Total</td>
<td>12.6K</td>
<td>100%</td>
</tr>
</tbody>
</table>

Exe. Time

- Demux: 15%
- IQ_RESC: 34%
- Filter Bank: 51%
## IMPLEMENTATION RESULTS

**Device Family**

<table>
<thead>
<tr>
<th>Logic Elements</th>
<th>30.653 LEs</th>
<th>27% of available LEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP Elements</td>
<td>276</td>
<td>52% of available elements</td>
</tr>
<tr>
<td>Memory Bits</td>
<td>30.72 KB</td>
<td>6% of available memory bits</td>
</tr>
<tr>
<td>Maximum Frequency</td>
<td>46.66 MHz</td>
<td></td>
</tr>
</tbody>
</table>

**ALTERA CYCLON IV E FPGA Utilization**

Mohammed Shaaban

Lund University

2012-11-16
## Results

<table>
<thead>
<tr>
<th>Module</th>
<th>Logic Elements</th>
<th>Memory Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMUX_HUFF</td>
<td>3.3K (10.80%)</td>
<td>2.38 KB (7.73%)</td>
</tr>
<tr>
<td>IQ_RESC</td>
<td>3.0K (9.67%)</td>
<td>4.00 KB (13.02%)</td>
</tr>
<tr>
<td>IMDCT</td>
<td>16.9K (55.05%)</td>
<td>20.34 KB (66.23%)</td>
</tr>
<tr>
<td>WIN_OV</td>
<td>6.9K (22.40%)</td>
<td>4.00 KB (13.02%)</td>
</tr>
<tr>
<td>Filterbank (IMDCT+Win_OV)</td>
<td>23.8K (77.45%)</td>
<td>24.34 KB (79.25%)</td>
</tr>
<tr>
<td>PCM Converter</td>
<td>0.006K (0.19%)</td>
<td>0</td>
</tr>
<tr>
<td>Main Controller</td>
<td>0.6K (2.08%)</td>
<td>0</td>
</tr>
</tbody>
</table>

## Implementation Results

### Logic Elements

- DEMUX: 11%
- IQ_RESC: 10%
- Filter Bank: 8%
- MISC: 77%

### Memory Bits

- DEMUX: 13%
- IQ_RESC: 79%
- Filter Bank: 8%
CONCLUSION

• The A full MPGE-4 decoder has been successfully simulated, synthesized and implemented on CYCLONE-IV E [EP4CE115F29C7] FPGA using an ALTERA DE-115 prototyping board.

• The minimum required tools

• Decode an ADTC bitstream, Mono channel, LC profile.

• Operating clock frequency up to 46 MHz

• Pipeline and hardware sharing techniques has been adopted in the design

• An IFFT IP core is used to give more accuracy for the IMDCT calculation.

• The results were monitored using TEKTRONIX TLA5240 to verify the decoder functionality.
TANK YOU