A Pure-ASIC Design Approach for MPEG-2 AAC Audio Decoder

Tsung-Han Tsai; Chun-Nan Liu; Yi-Wen Wang
Department of Electrical Engineering
National Central University
Taiwan

Abstract

MPEG Advanced Audio Coding (AAC) is the widely used audio coding standard and offers the high-quality multi channel surround audio. For the implementation using DSP method, the hardware is not dedicated to the algorithms and the hardware utilization efficiency and power consumption is not optimized. This paper presents a novel AAC audio decoder that is suitable for ASIC design implementation. First, each block has been analyzed to get the dedicated hardware. Then through the well scheduling and pipeline processing, a common hardware can be shared for two channel stereo signal decoding. Hence the hardware utilization and power consumption are promote effectively.

1. Introduction

The AAC audio coding is an international standard first be created in MPEG-2 AAC [1] (ISO/IEC 13818-7) and is the base of MPEG-4 general audio coding. MPEG-2 AAC audio coding has become very popular and been widely used. It is applicable for a wide range of applications from Internet audio over digital audio broadcasting to multi-channel surround sound. It achieves high compression ratio and high quality performance due to an improved time-frequency mapping cooperate with other new tools, like TNS, Predictor, etc.

There are three different profiles defined in AAC. It allows tradeoffs in audio quality and encoding/decoding complexity for different applications. Among that, the LC profile can provide nearly the highest audio quality as the Main profile, but with significant saving in memory and processing requirements. So this paper will focus on the implementation of 2-channle LC profile decoder.

Cause the implementation using DSP method can't optimize the hardware utilization efficiency and power consumption. This paper presents a novel AAC audio decoder, which is a pure-ASIC design implementation method. Each block in AAC algorithms has been analyzed and some combinations of blocks are made. Each part of them has its dedicated hardware, hence the hardware utilization and power consumption are promote effectively.

2. AAC Decoding Algorithms

The decoding flow of AAC LC profile is shown in Figure 1. There are several tools used in AAC decoder, which includes Bitstream parsing, Huffman decoding, Pulse data decoding, inverse quantizer (IQ), Rescaling, M/S and Intensity stereo, Temporal Noise Shaping (TNS), and Filterbank. Each step will be described as follows:

The first step in decoding is the Bitstream parsing, which extract the audio frame signals and the decoding information that will be used in the following decoding tools. In Huffman decoding, the input bitstream is decoded and degrouped to quantized values in the frequency domain. These quantized values are inversely quantized by the inverse quantizer tool and then scaled by the Rescale tool. The M/S and Intensity stereo tools recover and add the redundancies removed at the encoder to the frequency domain. The TNS uses prediction in the frequency domain to shape noise in the time domain. In the Filterbank block, the frequency domain signals are transformed into time domain and produce the output audio signals. The operation complexity of each tool [2] is analyzed and shown in Figure 2. From the analysis, the Filterbank and Huffman take about 80% of operation complexity. This paper proposed a dedicated hardware architecture, which can reduce the operation complexity efficiently.

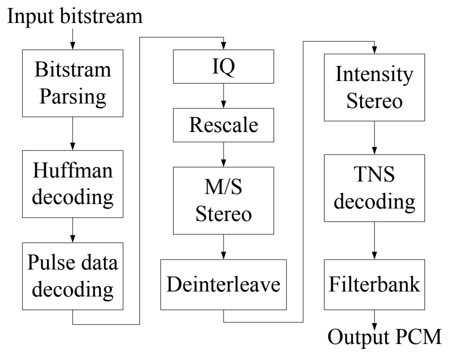

Figure 1. Decoding flow of AAC LC profile

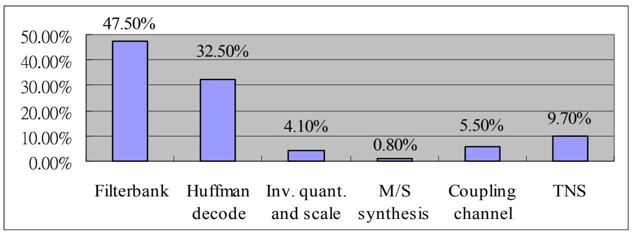

Figure 2. Ratios of operation complexity of each block

3. Architectural Design

Before the architecture design, the characteristic of each block is analyzed. From Table 1, the bitstream parsing and Huffman decoding block have large decision-making operations. The IQ block needs table lookup and little arithmetic operations. The others need largely arithmetic operations. Besides, the operation cycles and number of memory access for each tool of the proposed design is also shown in Table 1. From the analysis, the blocks in the decoding flow can be combined as shown in Figure 3. Each block will be described as follows.

Table 1. Characteristic and operation analysis of decoding flow

Block	Operation Types	Operation Cycles	Memory Access
Bitstream parsing	Decision	255x2	0
Huffman	Decision	1436x2	0
IQ	Table lookup, Addition	1024x2	Read: 1024x2
Rescale	Multiplication	1024x2	Write: 1024
M/S	Addition	1024	Read: 1024 Write: 1024x2
Intensity	Multiplication	1024	Read: 1024 Write: 1024
TNS	Multiplication, Addition	8832x2	Read: 1024x2 Write: 1024x2
Filterbank	Multiplication, Addition	16384x2	Read: 21504x2 Write: 12288x2

*One left channel frame and one right channel frame

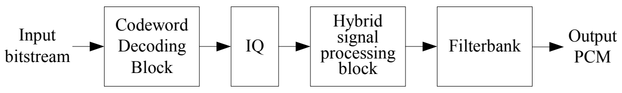

Figure 3. The decoding block diagram proposed

3.1 Codeword Decoding Block

There are three blocks in codeword decoding block including bitstream parsing, Huffman decoding and pulse data decoding. The overall architecture of this block is shown in Figure 4. The bitstream parser contains a barrel shifter and two register. A finite state machine is used to control it according to the decoded bitstream information.
The Huffman decoding part contains 12 tables, which is implemented using the PLA based method. By this method, only logic operations are needed and no other memory access is performed. It can reduce the power consumption largely and the throughput is higher than the conventional binary tree search method. The worst case (including the ESC sequence and signed bits) to decode a codeword is 2.5 cycles. Besides, the parser can be reused to get the input of Huffman decoding. For the pulse data decoding, it is just a add operation which is combined in the Huffman block.

3.2 Inverse Quantization

The inverse quantization in AAC is described as the following formula:

$$y = \text{sign}(x)\,|\,x\,|^{4/3}$$

where x is the quantized subband sample ($|x| < 8192$), and y is the inversely quantized subband sample.
For the implementation using table lookup with 8192 entries is rather impractical due to memory constraint. Therefore a combination of table lookup and interpolation method [3] we proposed before is adopted. In this method only 256 size lookup table is used and the accuracy without significant loss.

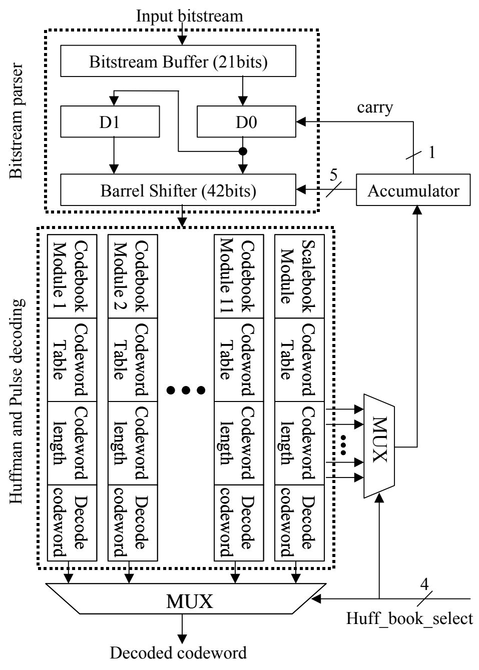

Figure 4. Block diagram of codeword decoding block

3.3 Hybrid Signal Processing Block

The hybrid signal processing block contains four tools, which are Rescale, TNS, M/S and Intensity stereo decoding. The operation of each tool is summarized as follows:

Rescale : $L_{RS} = L_{IQ} * \text{gain}$
$R_{RS} = R_{IQ} * \text{gain}$
M/S: $L_{MS} = L_{RS}$ or $L_{RS} + R_{RS}$
$R_{MS} = R_{RS}$ or $L_{RS} - R_{RS}$
Intensity : $L_{IS} = L_{MS}$
$R_{IS} = L_{MS} * \text{Scale}$
TNS : $L_T(n) = L_{IS}(n)-A_1*L_T(n-1)-\cdots-A_{order}* L_T(n\text{-}order)$
$R_T(n) = R_{IS}(n)-B_1*R_T(n-1)-\cdots-B_{order}* R_T(n\text{-}order)$
Where $\text{gain} = 2^{0.25*(Sf-100)}$
$\text{Scale} = 0.5^{0.25*\text{is\_position}}$

From the description above, these tools need only multiplication and add/sub operations and the processing of them are sequential. Therefore a common hardware that can be reused is preferred. The block diagram of this hybrid signal processing block is shown in Figure 5. In this design, three temporary storage memories are necessary. They are used for M/S, Intensity and TNS decoding since some processing of them are on the channel pairs, the immediate values must be stored for feature use. Besides, for the pipeline operation of left and right channel, RAM2 and RAM3 are also served as pipeline buffers.

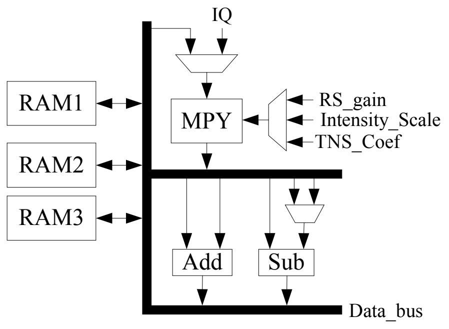

Figure 5. The block diagram of common hardware block

3.4 Filterbank

As shown in Figure 6, the AAC decoder filterbank consists of an inverse modified discrete cosine transform (IMDCT), and a windowing and overlap-add function. Since the window function has a significant effect on the filterbank frequency response, the filterbank has been designed to allow a switch in window shape (sine window and Kaiser-Bessel Derived (KBD) window) to best adapt to input signal conditions. Besides, there are four types of window sequence can be used for block switching including long, long start, long stop and eight short. Figure 7 shows the shape of individual sequence and the switching conditions of them.

In AAC, the filterbank is the most important part as shown in Figure 2. It occupies the high computation load in decoding flow and can't be solved efficiently by a simple architecture. In this paper, we use the N/4 point IFFT algorithm [4] as the fast IMDCT algorithm. This algorithm can reduce the computational complexity greatly but the control is more complicated. Figure 8 shows our architecture for this algorithm of filterbank. The kernel is a radix-2 butterfly unit with one MPY and two Add/Sub units.

Data input of IMDCT buffer is switched between RAM2 and RAM3 in the common hardware block for pipeline processing between left and right channel. The temporal data result from the operation of each IFFT stage is store in the memory of IFFT buffer real and IFFT buffer imag. As shown in Figure 7, the results of IMDCT need to be windowed by one type of window sequence. For the long and long stop type frames, the first half windowed result (white) is direct overlap and add with the previous frame and store at the output buffer. The second half part (gray) will be stored in the overlap buffer for the overlap-add operation of next frame. For the long start type frame, the first half windowed result is also direct overlap-add with previous frame. The second half part only overlap 128 points with the first eight short type frame, so only this part is stored in the overlap buffer. The result before this part is direct store at the output buffer and the result after this part is ignored since they are all zeros. For the eight short type frame, each time 256-point windowed results are produced. The first half is direct overlap-add and stored to the output buffer. The second part (gray) is store at the overlap buffer for the overlap-add operation with the next 256-point windowed results.

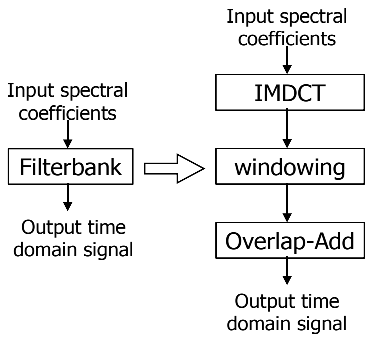
Input spectral coefficients
Filterbank
Output time domain signal
Input spectral coefficients
IMDCT
windowing
Overlap-Add
Output time domain signal

Figure 6. Decoding flow of AAC Filterbank

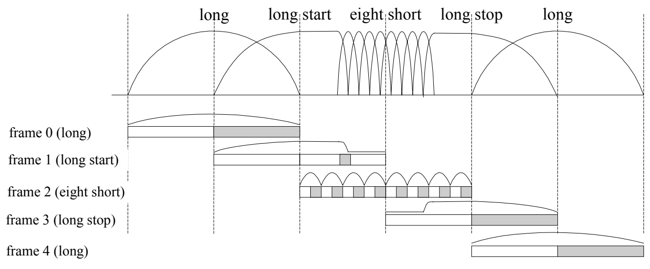

Figure 7. Window switch and Overlap-Add buffer accessing

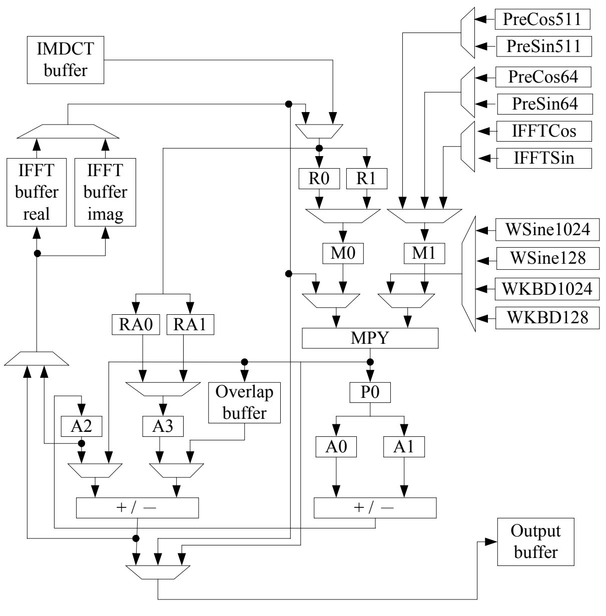

Figure 8. Overall architecture of filterbank

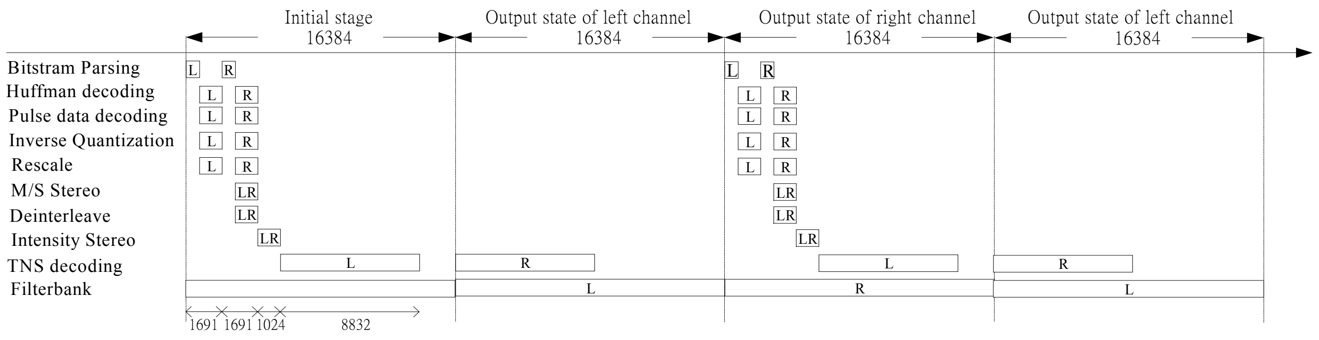

Figure 9. Pipeline scheduling of decoding processing

3.5 Pipeline Scheduling

In Table 1 we have analyzed the operation cycles need in each block for one frame. However, since the decoding contains two channels, some blocks need to wait the data of the other channel. Besides, the hardware sharing between two channels is the design consideration. Therefore the hardware scheduling of each block is analyzed and the result is shown in Figure 9. In this scheduling diagram, some blocks can be processed parallel in one pipeline stage. And through the pipeline processing, we can get one frame of left or right channel at each stage.

4. Implementation Results

The proposed architecture is designed with Verilog, compiled and simulated with SYNOPSYS tool and implemented using TSMC 0.25 um technology. Figure 10 shows the floorplan of the AAC decoder chip and the chip specification is summarized in Table 2.

To verify the implemented system, the result of the floating-point simulation in C language reference source code and fixed-point simulation in Verilog design are compared with each other for verification of the algorithms. The maximum error of C and Verilog is only 2 bit of the LSB for 16 PCM output.

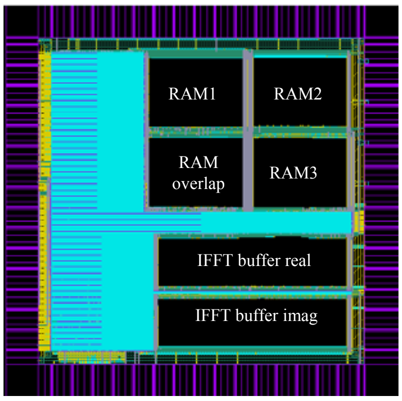

Figure 10. Floorplan of AAC decoder chip

Table 2. Chip summary of AAC decoder

Technology	TSMC .25 μm
Number of pins	64
Number of gate counts	82216.99
Power consumption	158.11 mW
Chip area	3002.1x3002.1 μ^2m

5. Conclusion

In this paper we presented a novel AAC audio decoder with pure-ASIC design approach. All blocks in the decoding flow has its dedicated hardware for the purpose of reducing operation complexity and increasing the performance. This decoder can decode 2-channel LC profile bitstreams. By the hardware scheduling method the hardware can be shared between two channels. Besides, the processing of left and right channel has been pipelined. Therefore, this decoder is efficient and the hardware utilization and power consumption are promote effectively.

References

[1] ISO/IEC JTC1/SC29/WG11 No.1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding, AAC)", April 1997.

[2] S. R. Quackenbush and Y. Toguri, "Revised Report on Complexity of MPEG-2 AAC Tools", ISO/IEC JTC1/SC29/WG11 N2005, February. 1998.

[3] T. H. Tsai and C. C. Yen, "A high quality re-quantization/quantization method for MP3 and MPEG-4 AAC audio coding", IEEE International Symposium on Circuits and Systems, Vol.3, pp.851 -854, 2002

[4] Duhamel, P.; Mahieux, Y. and Petit, J.P.,"A fast algorithm for the implementation of filter banks based on `time domain aliasing cancellation'", International Conference on Acoustics, Speech, and Signal Processing, Vol.3, pp.2209 -2212, Apr. 1991.