Design of Low Complexity Profile MPEG 4 AAC Audio Decoder

Suyog V Pande[1], M. A. Gaikwad[2], D. R. Dandekar[3]

[1, 3]Electronics Department, Bapurao Deshmukh Engineering of College, Sewagram, Wardha, Maharashtra, India

[2]Principal, Bapurao Deshmukh Engineering of College, Sewagram, Wardha, Maharashtra, India

Abstract: ***MPEG-4 AAC is the widely used audio standard and getting more popular for commercial use. MPEG-4 AAC (Advanced Audio Coding) incorporates several innovative technologies in order to achieve high Fidelity at low bitrates.MPEG-4 AAC is the widely used audio standard and getting more popular for commercial use. This paper presents an investigation & implementation of low complexity profile high quality MPEG4 AAC Audio Decoder at a sampling frequency of 44 KHZ on a Field programmable gate array. Through this paper we discuss IMDCT filter bank, Noiseless decoder, Inverse quantiser and Scale factor application modules of MPEG-4 Advanced Audio Coding decoder more efficiently.***

Keywords: high quality, low complexity

1. Introduction

The AAC audio coding is an international standard first be created in MPEG-2 AAC [2] (ISO/IEC 13818-7) and is the base of MPEG-4 general audio coding. AAC supports up to forty-eight audio channels. Sample rates supported range from 8 kHz to 96 kHz. The LC profile achieves nearly the same audio quality as the Main profile, but with significant savings in memory and processing requirements. With this mode, it is possible to decode the bit stream into a PCM signal having one of a variety of different sample rates [3].

This paper will focus on the mono channel, sampling frequency of 44 KHz; bitrates is 264Kb/s, Low Complexity profile implementation of the coder, which represents the configuration that is best suited for consumer electronics applications [7]. In AAC audio coding, there are two kinds of audio transport formats. One is Audio Data Interchange Format (ADIF), and the other is Audio Data Transport Stream (ADTS)[2]. For the ADIF format, it puts all data controlling the decoder (like sampling frequency, mode, etc.) into a single header preceding the actual audio stream. It is useful for file exchange but does not allow for break-in or start of decoding at any point in time. For the ADTS format, it packs AAC data into frames with headers and allows decoding to begin in the middle of an audio bit stream [1]. To design MPEG 4 AAC Audio Decoder here we use ADTS Frame format.MPEG is a lossy compression, which means, some audio information is certainly lost using these compression methods. This loss can hardly be noticed because the compression method tries to control it. By using several complicated and demanding mathematical algorithms it will only lose those components of sound that are hard to be heard even in the original form [6]

1.1 Decoder design flow chart

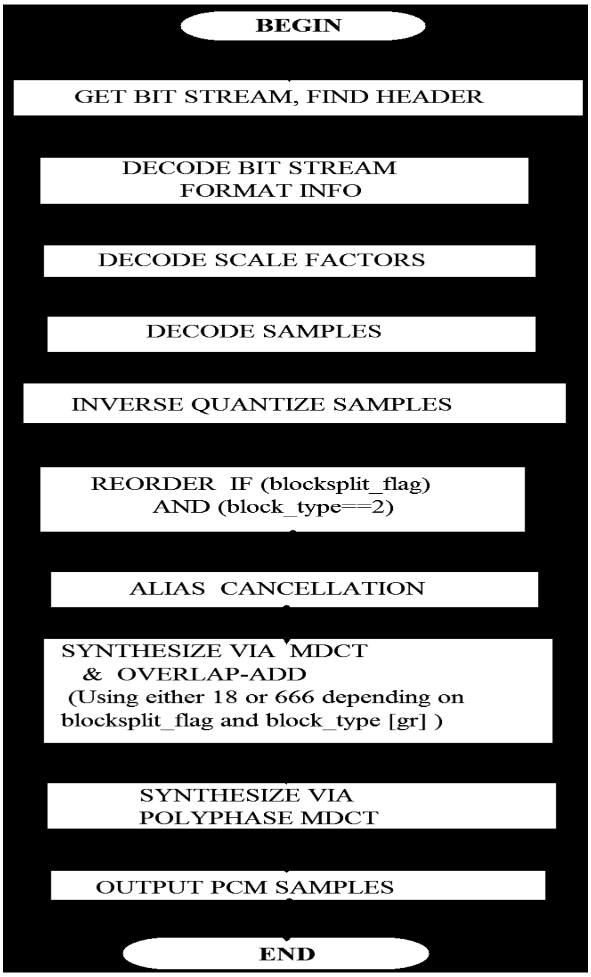

BEGIN
GET BIT STREAM, FIND HEADER
DECODE BIT STREAM FORMAT INFO
DECODE SCALE FACTORS
DECODE SAMPLES
INVERSE QUANTIZE SAMPLES
REORDER IF (blocksplit_flag) AND (block_type==2)
ALIAS CANCELLATION
SYNTHESIZE VIA MDCT & OVERLAP-ADD (Using either 18 or 666 depending on blocksplit_flag and block_type [gr])
SYNTHESIZE VIA POLYPHASE MDCT
OUTPUT PCM SAMPLES
END

Figure 1: MPEG 4 AAC decoder flow chart

1.2 AAC coding system

AAC uses a combination of multiple coding tools to achieve bit rate reduction. All of the coding tools described below (except for prediction) are used in both the Main and LC profiles [7]. The first block is the bit stream parser, which extracts the audio frame signals and the decoding information that are used in the following decoding tools [1] mainly noiseless decoder, requantization, scale factor ,filter bank .

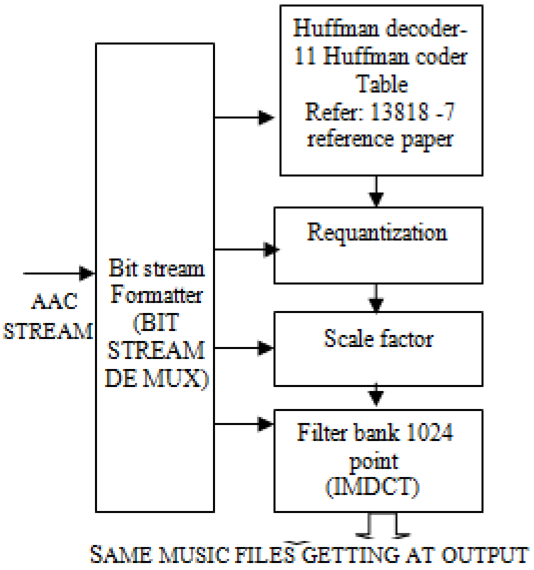
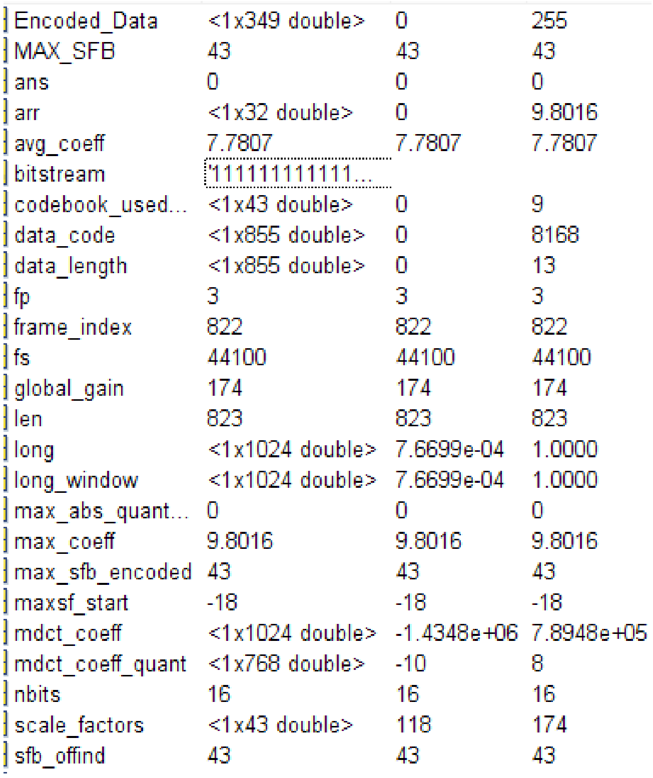


Figure 2: LC Profile of MPEG 4 AAC Audio Decoder

1.3 Decoding operation

The quantized samples are derived from the Huffman codes in the Huffman decoding block. The necessary side information needed for Huffman decoding is obtained from Huffman Info decoding block. Since the Huffman codes are variable length codes, the Huffman encoding of the quantized samples results in a variable frame size [4.]The next step after Huffman decoding, is the re-quantization. The re-quantizer re-quantizes the Huffman decoder output using the scale factors. The alias reduction block is used to reduce the unavoidable aliasing effects of the encoding poly phase filter bank. The IMDCT block converts the frequency domain samples to frequency sub band samples [2][6].

First we take any wav sample file (song) given to design low complexity profile encoder which gives AAC bit stream (coded form). The performance and result of each different block implemented in matlab & simulate it. Result getting in matlab as shown below. The AAC bit stream comes from encoder is given to proposed LC profile decoder which gives same wav samples applied at encoder side. The performance and result of each different block at decoder is implemented in VHDL using Model Sim as the simulation tool is presented. Testing and simulation were made to ensure full functionality of the design. Song of 128bitrate, 44 KHz and single channel is selected for the simulation

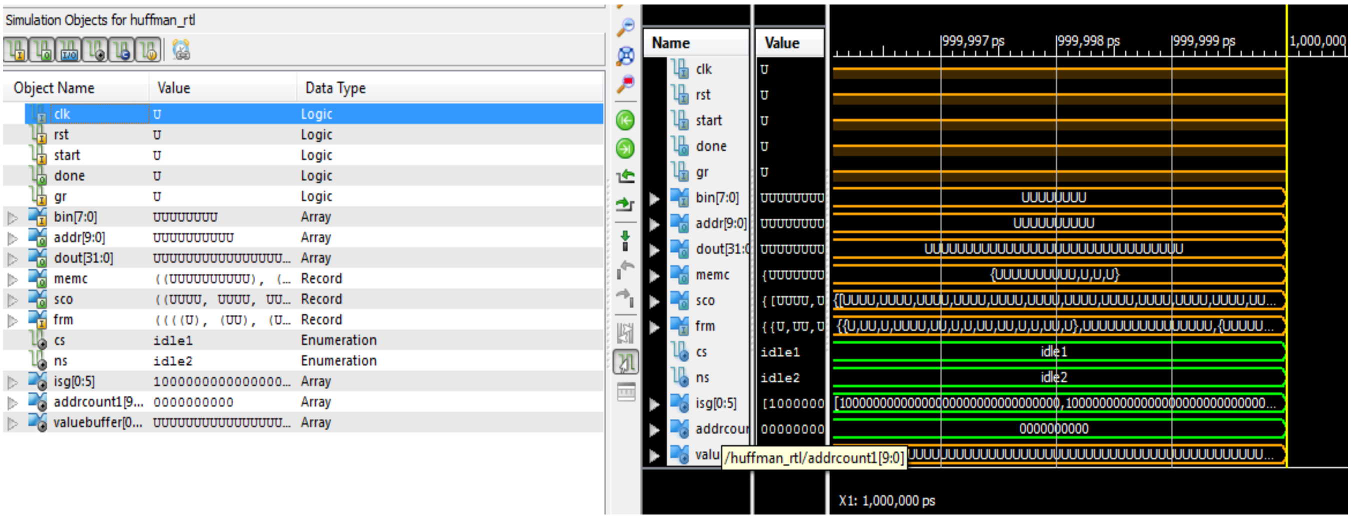


Encoded_Data	<1x349 double>	0	255
MAX_SFB	43	43	43
ans	0	0	0
arr	<1x32 double>	0	9.8016
avg_coeff	7.7807	7.7807	7.7807
bitstream	'111111111111...		
codebook_used...	<1x43 double>	0	9
data_code	<1x855 double>	0	8168
data_length	<1x855 double>	0	13
fp	3	3	3
frame_index	822	822	822
fs	44100	44100	44100
global_gain	174	174	174
len	823	823	823
long	<1x1024 double>	7.6699e-04	1.0000
long_window	<1x1024 double>	7.6699e-04	1.0000
max_abs_quant...	0	0	0
max_coeff	9.8016	9.8016	9.8016
max_sfb_encoded	43	43	43
maxsf_start	-18	-18	-18
mdct_coeff	<1x1024 double>	-1.4348e+06	7.8948e+05
mdct_coeff_quant	<1x768 double>	-10	8
nbits	16	16	16
scale_factors	<1x43 double>	118	174
sfb_offind	43	43	43

Figure 3: Encoder o/p in Matlab

2. Huffman Decoder

The Huffman decoder compares the input Huffman code bits with the 32 Huffman code tables to find a match to represent the Huffman code bits[4][6]. The Huffman code tables are predefined base on statistics of ISO standard 11172-3[5]. The Huffman coding is a loss-less source coding scheme [4] .The techniques used is represents most common letters using fewer bits where as less common letters using more bits.



Simulation Objects for huffman_rtl

Object Name	Value	Data Type
clk	U	Logic
rst	U	Logic
start	U	Logic
done	U	Logic
gr	U	Logic
bin[7:0]	UUUUUUUU	Array
addr[9:0]	UUUUUUUUUU	Array
dout[31:0]	UUUUUUUUUUUUUUUU...	Array
memc	((UUUUUUUUUU), (...	Record
sco	((UUUU, UUUU, UU...	Record
frm	((((U), (UU), (U...	Record
cs	idle1	Enumeration
ns	idle2	Enumeration
isg[0:5]	1000000000000000...	Array
addrcount1[9...	0000000000	Array
valuebuffer[0...	UUUUUUUUUUUUUUUU...	Array

Name	Value
clk	U
rst	U
start	U
done	U
gr	U
bin[7:0]	UUUUUUUU
addr[9:0]	UUUUUUUU
dout[31:0	UUUUUUUU
memc	{UUUUUUU
sco	{[UUUU,U
frm	{{U,UU,U
cs	idle1
ns	idle2
isg[0:5]	[1000000
addrcou	00000000
valu	

999,997 ps
999,998 ps
999,999 ps
1,000,000
UUUUUUUU
UUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
{UUUUUUUUUU,U,U,U}
{[UUUU,UUUU,UUUU,UUUU,UUUU,UUUU,UUUU,UUUU,UUUU,UUUU,UUUU,UU...
{{U,UU,U,UUUU,UU,U,U,UU,UU,U,U,UU,U},UUUUUUUUUUUUUUUU,{UUUUU...
idle1
idle2
[10000000000000000000000000000000,10000000000000000000000000000...
0000000000
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU...
/huffman_rtl/addrcount1[9:0]
X1: 1,000,000 ps

Figure 4: Waveform output of main data read and transfers to bit reservoir

3. Inverse Modified Discrete Cosine Transform

The filter bank tool, being the important part in the AAC decoder, converts the frequency domain signal into time-domain signal. It consists of N-point IMDCT, windowing and overlap-add operation. In that, IMDCT is the important step The Inverse Modified Discrete Cosine Transform (IMDCT) transforms each sub band from frequency domain to time domain [9]. It in cooperated with the synthesis polybasic filter bank to produce the time samples *xi* from the input frequency line X (k) [4].

$$xi = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos\left[\frac{\pi}{2n}\left[2i+1+\frac{n}{2}\right](2k+1)\right]$$

Where n is the number of the windowed samples, n is equal to 12 for a short block and n is equal to 36 for a long block. For a case with n = 36, the IMDCT is an 18 point DCT that generate 36 poly phase filter sub-band samples from 18 input frequency lines[8][4] These samples are multiplied with a 36 point window before it can be used by the next step in the decoding process [4]

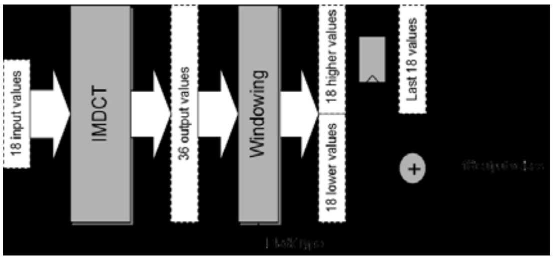


Figure 5: IMDCT design flow

4. Frequency Inversion

The overlap output consists of 18 time samples for each 32 poly phase sub bands. All the odd subsamples in the odd sub bands are negated by multiplying by -1 before processing the time samples into synthesis poly phase filter bank[2][6]

a) Controller

The controller block is designed to control the state. Figure shows the input & output of the controller block

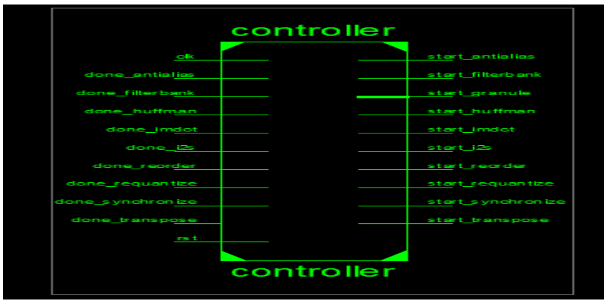


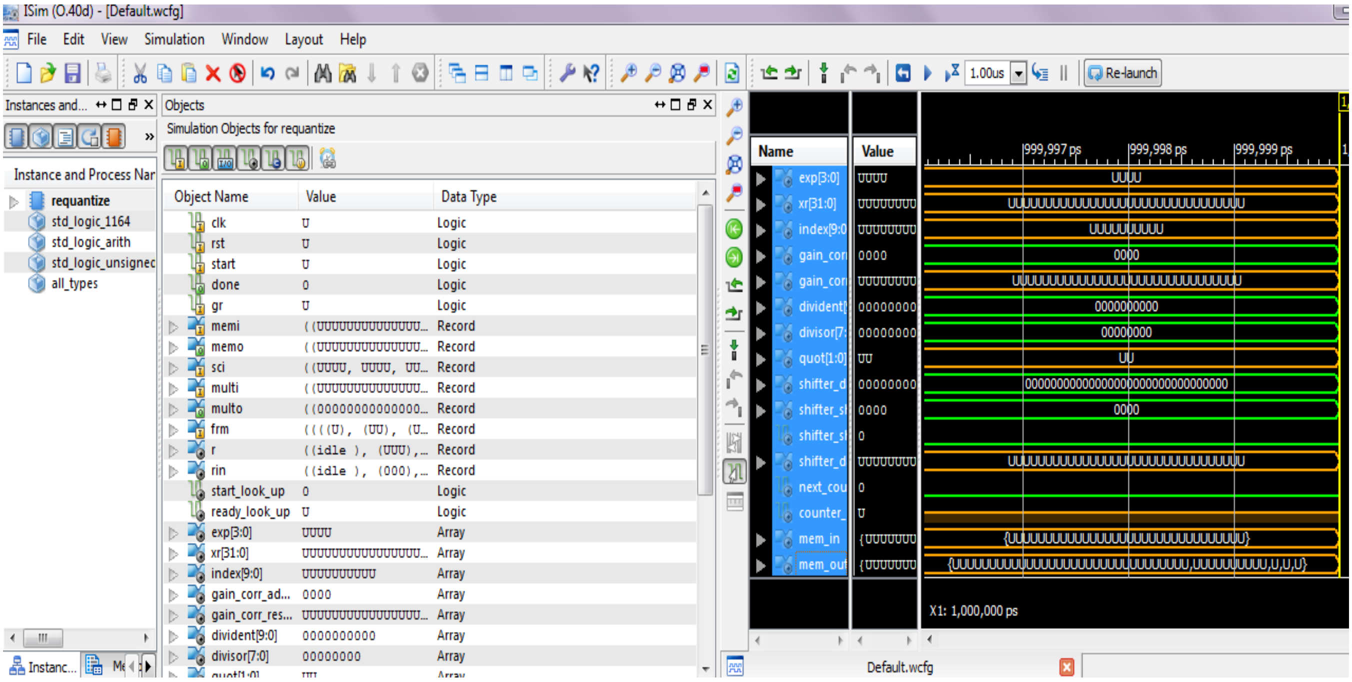
Figure 6: Input & Output of controller block

b) Synchronizer

The synchronizer block is the first stage in the pipeline. It interfaces with the input MP3 bit-stream. It will verify for the correct MP3 file and find the beginning of a new frame. When a correct MP3 file is read it will read the header and side information [4]. Finally the main data is read and transferred

c) Requantizer

The task of the requantizer is to rescale the Huffman decoded scaled and quantized frequency lines by using the scale factors decoded from Huffman block [4]



Simulation Objects for requantize

Object Name	Value	Data Type
clk	U	Logic
rst	U	Logic
start	U	Logic
done	0	Logic
gr	U	Logic
memi	((UUUUUUUUUUUUUUU...	Record
memo	((UUUUUUUUUUUUUUU...	Record
sci	((UUUU, UUUU, UU...	Record
multi	((UUUUUUUUUUUUUUU...	Record
multo	((000000000000000...	Record
frm	((((U), (UU), (U...	Record
r	((idle), (UUU),...	Record
rin	((idle), (000),...	Record
start_look_up	0	Logic
ready_look_up	U	Logic
exp[3:0]	UUUU	Array
xr[31:0]	UUUUUUUUUUUUUUUUU...	Array
index[9:0]	UUUUUUUUUU	Array
gain_corr_ad...	0000	Array
gain_corr_res...	UUUUUUUUUUUUUUUUU...	Array
divident[9:0]	0000000000	Array
divisor[7:0]	00000000	Array
quot[1:0]	UU	Array

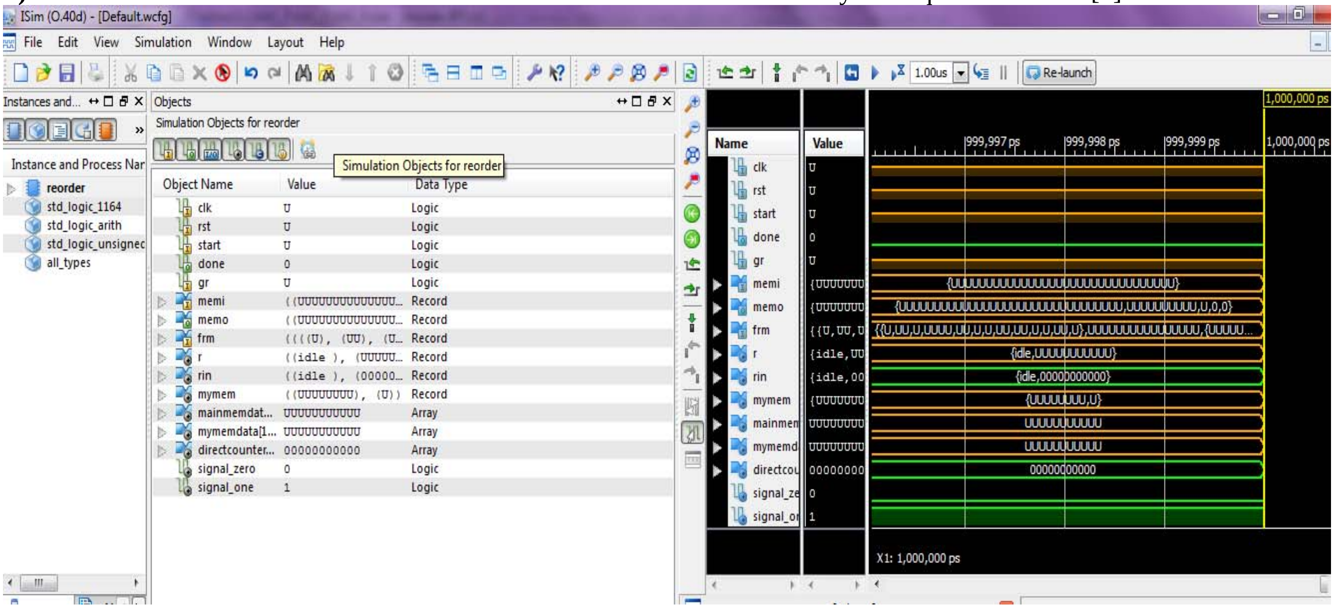
Name	Value	999,997 ps – 999,999 ps
exp[3:0]	UUUU	UUUU
xr[31:0]	UUUUUUUU	UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
index[9:0	UUUUUUUU	UUUUUUUUUU
gain_corı	0000	0000
gain_corı	UUUUUUUU	UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
divident[	00000000	0000000000
divisor[7:	00000000	00000000
quot[1:0]	UU	UU
shifter_d	00000000	00000000000000000000000000000000
shifter_sl	0000	0000
shifter_sl	0	
shifter_d	UUUUUUUU	UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
next_cou	0	
counter_	U	
mem_in	{UUUUUUU	{UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU}
mem_out	{UUUUUUU	{UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU,UUUUUUUUUU,U,U,U}

X1: 1,000,000 ps

Figure 7: Waveform of output for requantizer block

d) Reorder

The task of the reorder is to sort the 576 frequency lines that are created by the requantizer block [8].



Simulation Objects for reorder

Object Name	Value	Data Type
clk	U	Logic
rst	U	Logic
start	U	Logic
done	0	Logic
gr	U	Logic
memi	((UUUUUUUUUUUUUUU...	Record
memo	((UUUUUUUUUUUUUUU...	Record
frm	((((U), (UU), (U...	Record
r	((idle), (UUUUU...	Record
rin	((idle), (00000...	Record
mymem	((UUUUUUUU), (U))	Record
mainmemdat...	UUUUUUUUUUU	Array
mymemdata[1...	UUUUUUUUUUU	Array
directcounter...	00000000000	Array
signal_zero	0	Logic
signal_one	1	Logic

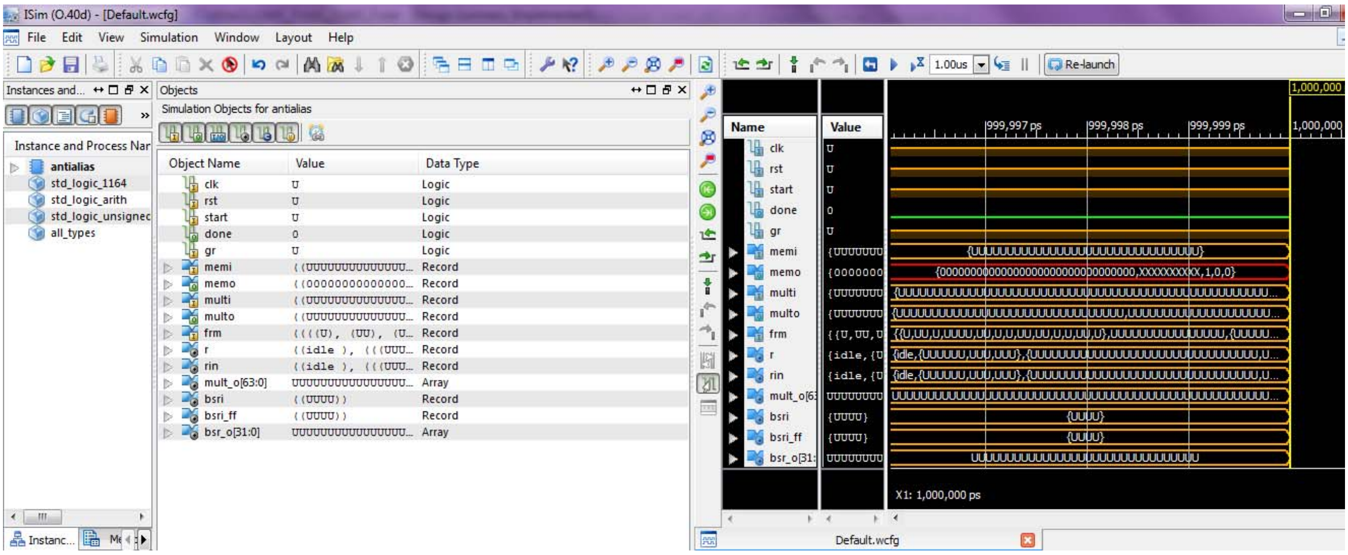
Name	Value	999,997 ps – 1,000,000 ps
clk	U	
rst	U	
start	U	
done	0	
gr	U	
memi	{UUUUUUU	{UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU}
memo	{UUUUUUU	{UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU,UUUUUUUUUU,U,0,0}
frm	{{U,UU,U	{{U,UU,U,UUUU,UU,U,U,UU,UU,U,U,UU,U},UUUUUUUUUUUUUUUUU,{UUUUU...
r	{idle,UU	{idle,UUUUUUUUUUU}
rin	{idle,00	{idle,00000000000}
mymem	{UUUUUUU	{UUUUUUUU,U}
mainmem	UUUUUUUU	UUUUUUUUUUU
mymemd	UUUUUUUU	UUUUUUUUUUU
directcou	00000000	00000000000
signal_ze	0	
signal_or	1	

X1: 1,000,000 ps

Figure 8: Waveform of Block_type of side information read from requantizer

e) Antialias

The task of anti aliasing is to implement the butterfly calculation. The butterfly reads the input data from memory design with some counters and multiplier.



Object Name	Value	Data Type
clk	U	Logic
rst	U	Logic
start	U	Logic
done	0	Logic
gr	U	Logic
memi	((UUUUUUUUUUUUUU...	Record
memo	((00000000000000...	Record
multi	((UUUUUUUUUUUUUU...	Record
multo	((UUUUUUUUUUUUUU...	Record
frm	((((U), (UU), (U...	Record
r	((idle), (((UUU...	Record
rin	((idle), (((UUU...	Record
mult_o[63:0]	UUUUUUUUUUUUUUUU...	Array
bsri	((UUUU))	Record
bsri_ff	((UUUU))	Record
bsr_o[31:0]	UUUUUUUUUUUUUUUU...	Array

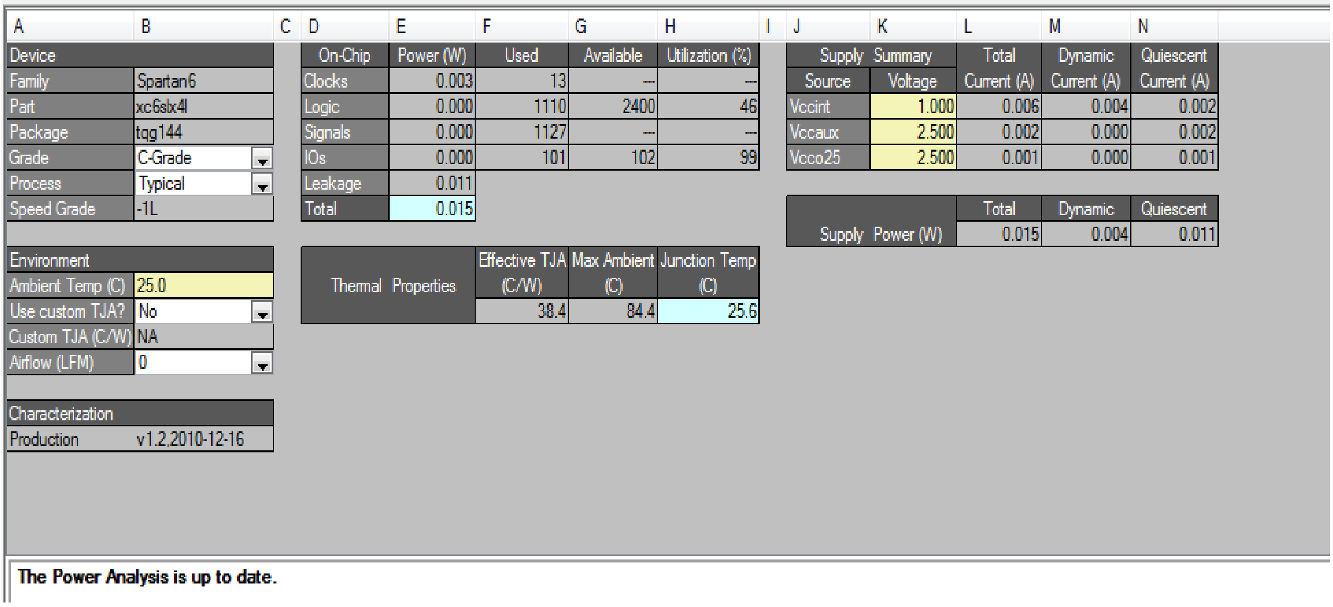
Figure 9: Waveform of antialias butterfly calculations

5. Comparison Results

References	Existing system [1]	Proposed We design
Power(mw)	64	15
Technology	FPGA (SPARTAN 6)	FPGA (SPARTAN 6)

Figure 10: Comparison of Existing system with proposed design

Fig 11 shows the power analysis report produce by Xilinx ISIM after simulation process is completed. Power consumption of designed architecture is 15mW in FPGA Spartan 6.



Device	
Family	Spartan6
Part	xc6slx4l
Package	tqg144
Grade	C-Grade
Process	Typical
Speed Grade	-1L

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.003	13	---	---
Logic	0.000	1110	2400	46
Signals	0.000	1127	---	---
IOs	0.000	101	102	99
Leakage	0.011			
Total	0.015			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.006	0.004	0.002
Vccaux	2.500	0.002	0.000	0.002
Vcco25	2.500	0.001	0.000	0.001

	Total	Dynamic	Quiescent
Supply Power (W)	0.015	0.004	0.011

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	38.4	84.4	25.6

Characterization	
Production	v1.2,2010-12-16

The Power Analysis is up to date.

Figure 11: Power analysis report of MP4 AAC Audio Decoder

6. Performance Parameter

Sr no	wav	Parameters	
		Peak signal to noise ratio	Mean square error
1	Sample1.wav	85.8775	1.6801e-004
2	Sample2.wav	57.0195	0.1292

Sr no	wav	Parameters	
		Mean Frequency_diff	papr_diff
1	Sample1.wav	1.6754e-005	1.0000
2	Sample2.wav	3.9171e-004	1.0000

In above table Sample 1 & sample 2 are 2 sample audio wav file

7. Parameters

I. Peak signal to noise ratio:

The ratio between the maximum possible power of signal and the power of corrupting noise. It is used to estimate the quality of reconstructed audio signal with respect to original audio signal.

It is calculated as PSNR =Peak Signal to Noise Ratio(y, y1).

II.Peak average power ratio difference:

It is defined as, papr_diff = ((peak1-peak2)/mean_frequency_diff)

III.Mean square error:

This finds the mean of the squared errors:

MSE = mean (errors. ^2)

Each element is squared separately, and then the mean of the resulting vector is found.

8. Conclusion

Thus the real time MP4 and AAC decoding system supporting LC profile for audio decoder have designed. The output of this project processed by many blocks before hear the audio services. To verify the designed system by using simulation. The result of simulation showed in both Xilinx ISE 13.1 ISIM and MATLAB R2010a software provide the same decoded output

References

[1] Low power fpga solution for dab audio decoder by R.Amutha, PR.Bhuvaneswari IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308

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

[3] Design and implementation of aac decoders” by Matthew A. Watson and Peter Buettner Dolby Laboratories, Inc. San Francisco, CA 94103

[4] Mp3 Decoder Design & Implementation Using VHDL by Ravindar Sharma, Ghanshyam Kumar Singh, Ram Mohan Mehra

[5] *Information Technology - Generic coding of moving picture and associated audio, Part 3: Advanced Audio Coding*, ISO/IEC 11172-3

[6] The Theory Behind Mp3 by Rassol Raissi December 2002

[7] Design and implementation of aac decoders” by Matthew A. Watson and Peter Buettner Dolby Laboratories, Inc. San Francisco, CA 94103

[8] MP3 DECODER in Theory and Practice by Praveen Sripada

[9] An Implementation of Filterbank for MPEG-2 AAC on FPGA by Fuhui Du, Gaoming Du, Yukun Song, Duoli Zhang, Minglun Gao

Author Profile

Mr. Suyog. V. Pande received the B.E. Degree in Electronics & Tele-Communication Engineering in the year 2008 and pursuing M.Tech Degree in Electronics in Bapurao Deshmukh Engineering of college, Sewagram. His areas of interest include communication, VLSI Design, Signal Processing.

Prof. Deepak. R. Dandekar Currently he is an Associate Professor in P.G Department of Electronics Engineering, Bapurao Deshmukh college of Engineering, Sewagram. His interest includes VLSI Circuits, Wireless Sensor Network.

Dr. Mahendra. A. Gaikwad did his BE in Electronics Engineering in 1991 from Nagpur University. He did his MBA in Marketing Management from Nagpur University & he has completed his MCM from Nagpur University. He did his Master’s Degree in Personal Administration from Nagpur University. He did his M.Tech in Communication Engineering from Indian Institute of Technology; Bombay in 1998.He did his PhD on “Network on chip Architecture using Perfect Difference Network Topology” at VNIT, Nagpur. Currently he is working as Principal at Bapurao Deshmukh College of Engineering; Sewagram (Wardha).He is author of 66 technical research papers published in various International & National journals. He is the life member of Professional bodies like Indian society for Technical Education, Institution of Engineers (India), Indian society for Telecommunication Engineers. He is also invitee member of Institution of Engineers (India), Nagpur local chapter, Nagpur. He is member of Computer Society of India.