Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, 2015, *7*, 1405-1414

Research on FPGA Prototype Simulation in MP3 Audio Decoder Design

Geng Wen-bo[1,2,*] and Zhou Zi-ang[1]

[1]*School of Physics and Electromechanical Engineering, Zhoukou Normal University, Zhoukou 466001, PR China*

[2]*Key Laboratory of Cloud Computing and Internet of Things Applications, Zhoukou Normal University, Zhoukou 466001, PR China*

Abstract: By further study of MP3 audio decoding algorithms, this paper analyzed the system structure of MP3 decoder and discusses the mapping of each module from algorithm to logic structure. Aiming at hardware implementation, it proposes the improved searching method for MP3 audio encoder, based on the features of Huffman decoding. It also adopts the intellectual property core of MP3 comprehensive filter groups and fast IMDCT algorithms. In inverse quantization module the compensated factor is used to make magnitude upgrade of SNR, without multiplications. We use coprocessor to design special instruction set and arithmetic unit by the analysis of fast IMDCT algorithm. It rewrites original C language and adopts unified DCT-based transforming core algorithm to compute the IMDCT of MP3. It can provide parallel work of IMDCT and LEON3 processor to make accelerated effects.

Keywords: MP3, decoding, IMDCT, Huffman decoder, FPGA.

1. INTRODUCTION

As the important part in the process of information storage and transmission, audio and video encoding and decoding technique is developing towards the direction of multi-standard, low power and low cost. The modules design is the key factor influencing the whole performance. Thus, it has realistic significance to study audio and video codec design and its implementation. At present, some of the solutions of audio decoder like DSP-based decoding scheme, of RISC-based processor decoding scheme and ASIC-based decoding scheme are widely used [1-3]. Although DSP-based or RISC processor decoding scheme have advantages of perfect compatibility and short development cycle, the whole power of decoder is very high. It cannot satisfy the demand for terminal on power. However, ASIC-based decoding scheme has low power advantage, its flexibility is bad and its development cycle is very long. But software and hardware co-design considers such problems.

At present, the mainstream schemes can be divided into two types, according to its algorithm circuit design. One is traditionally based on general-purpose processor or software scheme of DSP (Digital Signal Processor) [4,5]. Although such methods have been broadly applied, the general instruction set is not specially set for audio or video coding. The length of processor word and instruction set are usually not effectively adapted to practical requirement of audio disposal. Meanwhile, it is difficult to complete high execution efficiency and audio and video coding/decoding of low power under further requirement. With further improvement of diversity and users' demand for various mobile terminal production functions, it can be predicted that inherent defect is increasing prominently; The second type aims to specified audio and video codec algorithm to design special decoder or ASIC (Application Specific Integrated Circuit) [6]. The execution efficiency of this scheme is relatively high for and it has minimal work consumption. Meanwhile, it can fully take advantage of parallel execution of hardware encoding and decoding. But the defect of this method is also very obvious: fixed hardware structure is hard for practical product requirement, to be adjusted and changed flexibly. With constant development of audio-video coding and decoding standard, the scheme may possibly encounter bottleneck [7].

For above problems, this paper designs a hardware and software co-design MP3 audio decoder prototype chip. In multi-standard Huffman-decoding design, the hybrid look-up table is introduced. We determine the stage rule according to codeword laws. Then the hardware structure and code table structure corresponding to index tables are also proposed. In multi-standard inverse quantization module, the compensation factor design is introduced by system error analysis. So the SNR gets magnitude upgrade without multiplication calculation. Then the improved discrete cosine inverse transformation is accelerated in this paper. Due to the difference between IMDCT36 and DCT64, the state machine control is not suited to IMDCT36 design. Therefore, we adopt coprocessor acceleration method and design the coprocessor which is specially used for IMDCT36. It can make parallel work with LEON3 processor to improve decoding efficiency.

2. FPGA-BASED MP3 DECODER

FPGA-based MP3 decoder scheme is depicted in Fig. (**1**). Due to maturity of MP3 software decoder, the decoding

1874-4443/15

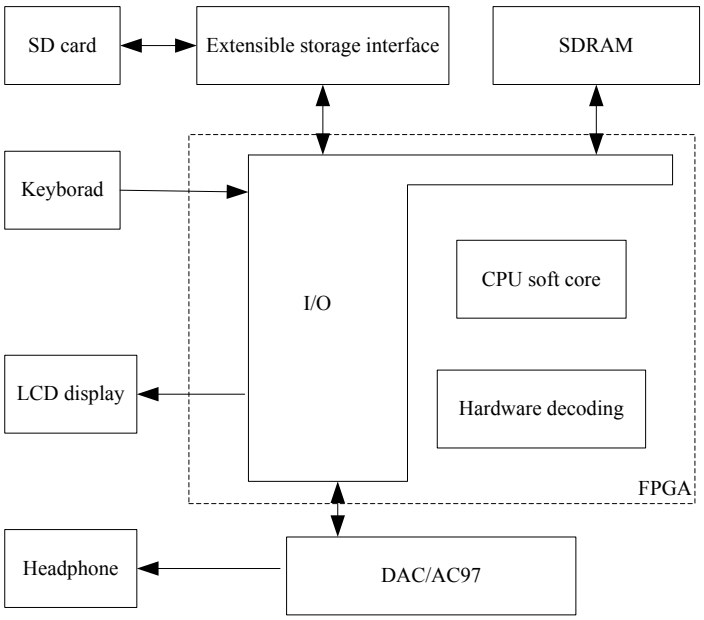

Fig. (1). FPGA-based MP3 decoder.

library of open source is very abundant. For instance, MPG123 and Libmad can be directly used in common embedded system. Therefore, an IP core (Intellectual Property Core) is mapped to the processor of FPGA and MP3 decoder becomes the simplest implementation inside IP core. The design prototype in this paper is adopting LEON3 and MP3 decoder MPG123 of open source to perform MP3 decoder function on Spartan-6 platform. Since all decoding processed are performed by software, the decoding efficiency is not high. Thus, decoding with long time consumption is conversed to be implemented by hardware such as Huffman decoder, Inverse Modified Discrete Cosine Transform (IMDCT), and sub-band synthesis filter [8]. After these modules are implemented by hardware, they become a peripheral component of main processor, to connect LEON3 via AMBA (Advanced Microcontroller Bus Architecture). Other decoding functions and management functions are still implemented by software, which can largely improve the decoding efficiency and accelerate the development process.

FPGA contains a lot of logic gates which can be connected by programs. FPGA offers flexible, variable word length and potential framework of parallel processing ability. Some FPGA also contain DSP units to perform common DSP algorithm such as DSP48A1 of Xilinx Spartan-6 series and DSP48E1 of Virtex-6/7 series. FPGA is used to perform digital signal processing, which can meet the demand for specific processing speed and performance. In the latest Xilinx device Zynq-7000EPP, it integrates industrial grade ARM binuclear Cortex-A9MPCore, so the design is divided into different parts to be easily implemented. FPGA developers can directly share existing software resources in ARM system. As long as the IP core matches the bus rules, it can be used on this platform. FPGA has better flexibility during signal processing. For instance, when 10 multiplication and accumulation (MAC) are performed on FPGA and DSP, if serial mode performs 10 MAC one by one on DSP, this operation needs 10 clock cycles; if FPGA is used for complete parallel operation, it only needs one clock cycle. FPGA and DSP are realized 10 Multiplication and Accumulation, MAC computations. If serial mode performs 10 MAC one by one on DSP, this operation needs 10 clock cycles. If FPGA is used for complete parallel operation, it only needs one clock cycle. We execute 10 MAC within 5 clock cycles and 2 MAC are implemented in parallel each time. However, if DSP processor is adopted, it is impossible to modify execution cycle according to the demands in such a flexible way. Such flexibility is determined by inverse relation between circuit area and speed. If 10 MAC is operated rapidly, FPGA can operate them in parallel within one clock cycle. However, it needs to integrate 10 arithmetic units to parallel calculate and consumes lots of hardware resources. If 10 MAC is allowed to be operated in slow speed, FPGA can be operated in serial with only 1 arithmetic unit. Therefore, the used amount of logic resources reduces to its 1/10 in FPGA chip but it needs 10 clock cycles for operation. Since parallel processing is adopted, FPGA is far superior to DSP processor in data/arithmetic throughput and flexibility. FPGA flexibility can implement algorithms on FPGA to satisfy specific application by different methods usage in design.

FPGA has advantages of short development cycle and perfect real-time performance with good expansibility, so it can satisfy the requirement of constant changing and updating. Thus, it has higher research value. So this paper adopts the idea of software and hardware co-design to develop a platform with FPGA. On one hand, it uses existing open source software to perform hardware acceleration which shortens the design cycle; on the other hand, with appearance of lots of soft-cores, it is more convenient to perform the secondary development.

Table 1. Computation analysis of MP3 encoder modules.

Decoding Modules	Computation Percent (%)
Head and side information decoding	2.52
Data extraction	2.73
Huffman decoding	17.32
Inverse quantization	5.21
Stereo	1.34
Reordering	0.39
Antialiasing	0.51
IMDCT	22
Integrated filter	47.98

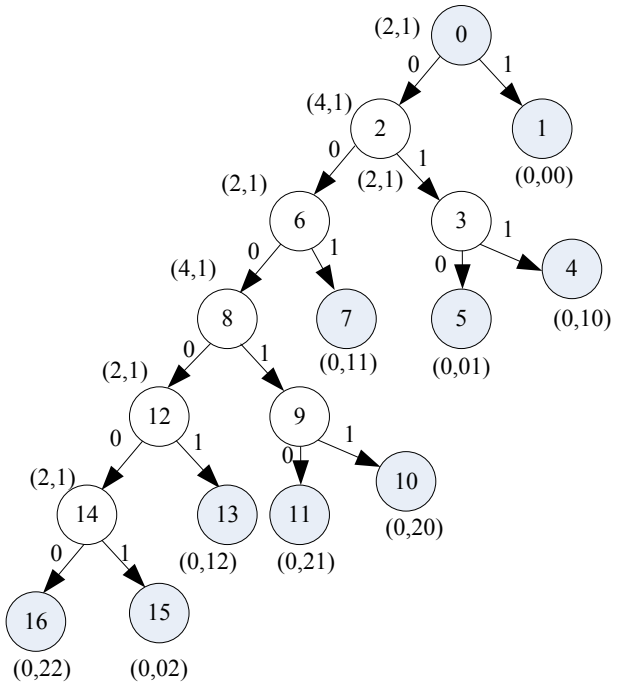

Fig. (2). Huffman tree corresponding to the Huffman code table.

3. SCHEME FOR PROTOTYPE CHİP DESİGN

Wang in literature [9] offers the percent of operation time for modules in MP3 decoder by study, as show in Table **1**. We can see that the main computation of MP3 lies in Huffman decoding, inverse quantization, IMDCT and integrated filter. Therefore, these modules are taken as the candidate modules for hardware implementation. In the next parts we will provide the improvement analysis for them.

3.1. Huffman Decoding Module

We adopt Huffman tree to save the code tables, to reduce the storage space. The code table can be loaded to a binary tree to create a Huffman tree [10]. The binary searching algorithms have better utilization of storage space and simpler control. But its processing speed is relatively slow which needs bit determination one by one. Since the object of this paper is saving the circuit resource and Huffman decoding is not the bottleneck of MP3 decoder, we choose binary tree searching algorithms for hardware implementation.

It starts from the root nodes and traverses the tree according to the input of code streams. When input is 0, it goes to the next node based on left offset; when input bit is 1, it first saves the number of right node at the bifurcation of tree, as is depicted in Fig. (**2**). This tree has 17 nodes totally from root node to leaf node, so it needs 17 storage space. That is, high

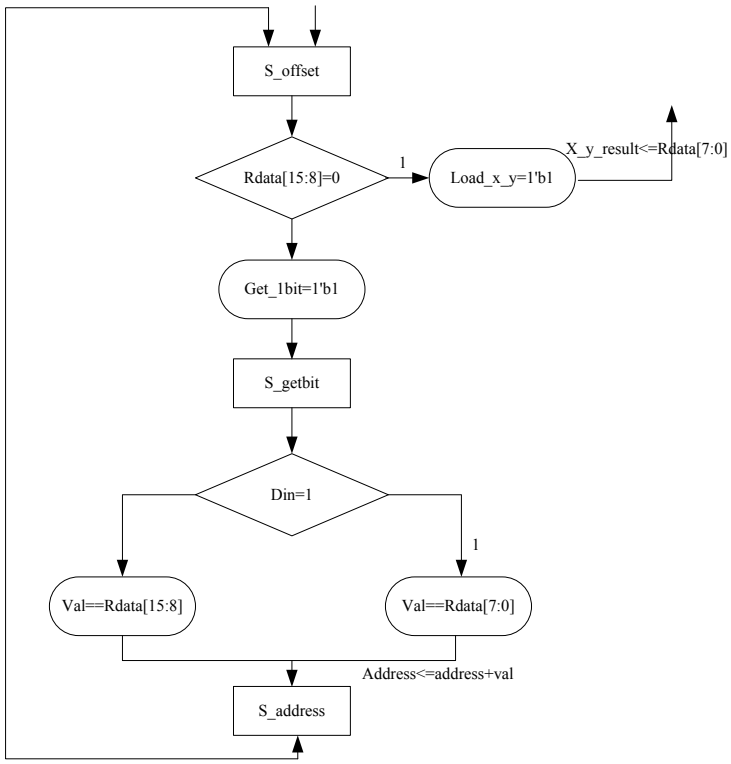

Fig. (3). State control graph of Huffman module.

8 bits denote left offset and low 8 bits denote right offset. The large-value area needs to save 15 code tables and small-value area needs to save 2 Huffman tables. The Huffman tree corresponding to these code tables are saved in ROM unit for query in decoding.

The state control for hardware Implementation is depicted as Fig. (**3**). At the state of *S_offset*, it makes decision from Huffman ROM which provides the node number: if the left offset of its high 8 bits *Rdata* [15:8] is 0, this node is leaf node and the decoding is over. *Rdata* [7:0] of low 8 bits will be taken as the final decoding result *x_y_result*. Otherwise, it generates control signal *get_1bit* to read a codeword from RAM. The state *S_getbit* acquires input data *Din*. If its value is 1, it makes offset backwards according to the right offset *Rdata* [7:0], and the address will be added to *val* to point to the new storage space. *S_address* reads the number of new node from Huffman ROM according to the address and returns the state of *S_offset* to determine whether this node is leaf node. Otherwise, it continuing read the codeword to determine the next address of node.

Then the Huffman decoding runs by default procedures. After each decoding it will generate two or four frequency line value that is saved in public RAM units. When the decoding of great-value and small-value area is over and arrives at the zero-value border, we can directly write 0 to public RAM unit until 576 frequency value can be acquired. This module needs to record the border value of zero-area, which is used for sequent computing module as output signals, to reduce the data computation.

3.2. Inverse Quantization Module

The top encapsulated port of inverse quantization with compensated factor is designed as the following Fig. (**4**):

Compared to Huffman module, the state machine of inverse quantization module with compensated factors is relatively simple. The whole module is designed under the total framework of control logic and data path. The state transition of control logic is depicted as Fig. (**5**):

The meanings of each state in above figure are:

SAMPLE is state of lookup table. It has two processes of lookup table. The first is the lookup table for value *is*. We can acquire the value of 4/3 power corresponding to the lookup table directly, for the value not more than 1026. For the other *is* value we can get corresponding 4/3 power by look-up table, after it is divided by 8, and finish the operation of multiplication by offset. Second is the look-up table of compensated factors. For value larger than 1026, we can determine the compensated factors multiplex group by its size and the remainder divided by 8. The two states of look-up run parallel, to exert the advantage of parallel processing.

SCALE index computation and processing state: this state is charge of the computation of index A during the inverse process of MP3, and the index process for *is* acquired by lookup table. We adopt 32-bits wide in the preparation of code table. It is multiplied by corresponding 2^x according to the size of $is^{4/3}$. The fractional part is transformed to integral part for storage. The shift anti-operation is performed after the index computation.

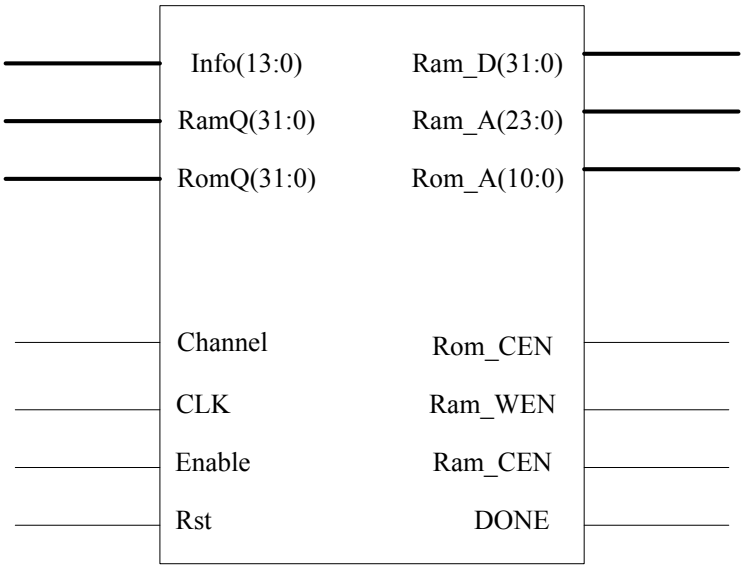

Fig. (4). Top encapsulation with compensated factors of inverse quantization module.

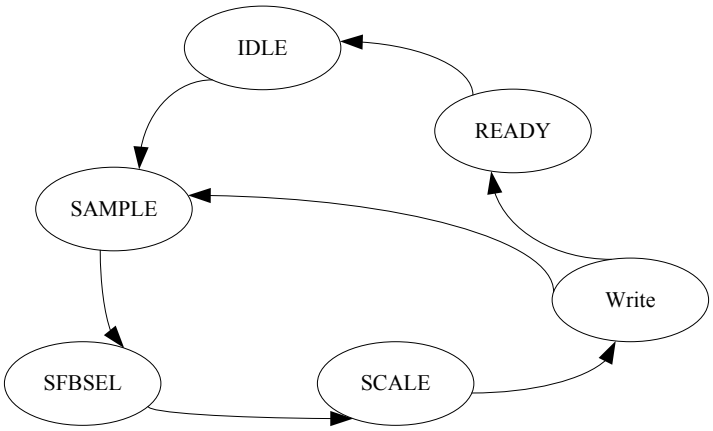

Fig. (5). State transition graph of inverse quantization module.

WRITE state: this state will write *is* back to the address of public Ram. It determines whether all the processes of *is* is finished. If so, it goes to the completion state; otherwise, it returns to lookup table state and runs the inverse quantization transition of next *is*.

Completion state READY: this state finishes the remaining procedures. When all the inverse quantization is finished, it goes to the idle state.

Code table preparation and design of compensated factors: as mentioned above, in the program for decoding, we adopt float number to denote and compute the data with fractional part. Since the float number constitutes the order number and mantissa, the representation and computing process are hard to be implemented in hardware. So the decimal is multiplied by N_{th} power of 2 and represented by integer data. When the bit wide is above 30 we can still obtain better SNR.

3.3. Fast Algorithm of IMDCT

IMDCT transforms a sub-band signal in particle from frequency domain to time domain, and the equation is:

$$x^i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos[\frac{\pi}{2n}(2i+1+\frac{\pi}{2})(2k+1)], 0 \le i \le n-1 \qquad (1)$$

X_k is input frequency signal; x^i is output time domain signal; n is the number of band lines. For short blocks, $n=12$; for long blocks $n=36$.

Marovich in literature [11] demonstrated that the method of Konstantinides [12] can be used to speed up IMDCT with 12 points and 36 points. It only needs to compute the result of IMDCT with 6 points and 18 points, as $DCT_{32\to64}$.

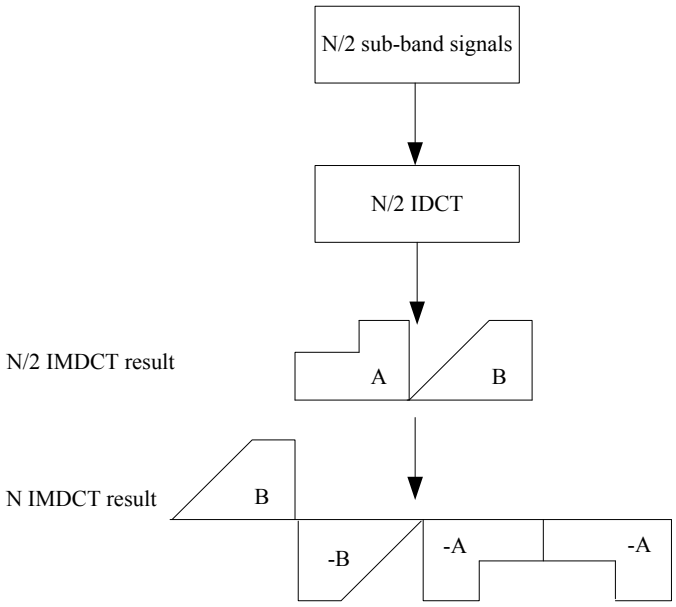

Fig. (6). Operation process of IMDCT36.

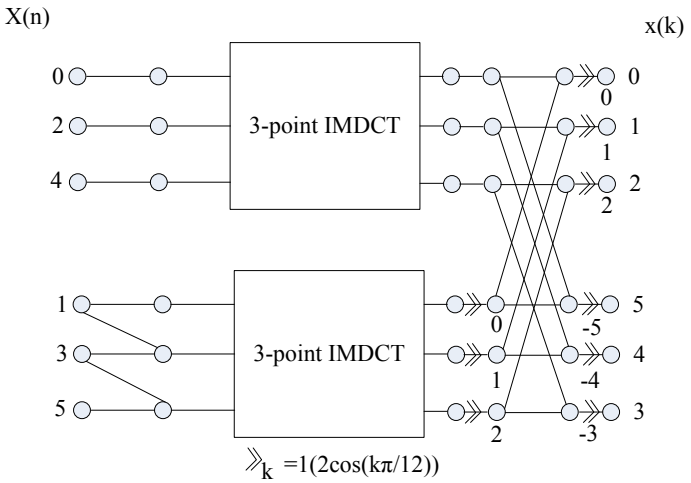

Fig. (7). IMDCT decomposition with 4 and 6 points.

Then we get the result of IMDCT with 12-point and 36 points based on the relation described in Fig. (**6**). After decomposition, we can get the results of IMDCT with 12 points and 36 points by computation of 3, 4, and 5 points.

For 6-point short blocks, they are divided into odd part and even part by serial number. Then IMDCT transformation with 3 points is performed first. Then the result is performed corresponding multiply-add operations to get the result of 6-point IMDCT. The process is shown as Fig. (**7**).

The equation of 3–point IMDCT is:

$$\begin{cases} x_0 = X_0 + (\sqrt{3}/2)X_1 + (1/2)X_2 \\ x_1 = X_0 - X_2 \\ x_2 = X_0 - (\sqrt{3}/2)X_1 + (1/2)X_2 \end{cases} \quad (2)$$

We add intermediate variables T_0 and T_1 to reduce the computation. The final computation contains two multiplications and four additions:

$$\begin{cases} T_0 = X_0 + (1/2)X_2 \\ T_1 = (\sqrt{3}/2)X_1 \\ x_0 = T_0 + T_1 \\ x_1 = X_0 - X_2 \\ x_2 = T_0 - T_1 \end{cases} \quad (3)$$

After this simplification, one 6-point IMDCT needs 13 multiplications and 27 additions. Similarly, 36-point IMDCT transformation can be acquired by 4-point and 5-point IMDCT transformation.

Table 2. Comparison of rapid algorithm and original algorithm.

N	Original Algorithm		Fast Algorithm	
	Multiplications	Additions	Multiplications	Additions
12	72	60	13	27
36	648	612	81	149

Table 3. Audio clips for audio module verification.

Code	Sampling Rate (kHz)	Code Rate (kbps)	Number of Channels
001.mp3	44.1	128	2
002.mp3	48	320	2
003.mp3	44.1	96	2
004.aac	44.1	128	2
005.aac	48	192	2

The original IMDCT algorithm has larger computation. For the direct implementation of N-point IMDCT transformation, we need $N^2/2$ multiplications and $N^2/2-N$ additions. When $N=12$, it needs 72 multiplications and 60 additions. When $N=36$, the number is 648 and 612 correspondingly. The comparison of fast algorithm and original algorithm is shown in Table **2**.

4. EXPERİMENTAL ANALYSİS

4.1. Verification Results of Audio Decoding Module

In order to test various audio modules, this paper randomly selects five candidate MP3 fragments for hardware and software decoding, as shown in Table **1**. Software decoding adopts MP3 and AAC official standard based on C language decoding program, and they extract key module by output functions of C language file.

Since most operations of Huffman decoding are looking up tables without calculation, it is not necessary to perform fixed point transformation of data to verify the correction of look-up results. For decoding speed, the test method in this paper is: when one data decoding with fixed length is completed, the stop count flag bit of one FSM counter is set by Parkinson decoding count register and reading value of counter [18] via ModelSim or Chipscope after decoding. Then the speed of different modules can be compared. For chip resource occupation, this paper adopts comprehensive toolkit in Xilinx ISE Design Suit 13.2, and chooses *xc5vlxll0t* in target device, based on which it generates comprehensive report as occupied data in module resource. With above methods, the verification results of Huffman decoding module using hybrid look-up tables are shown in Table **4**.

The average values between counting results and the test results of other audio fragments are obtained. Based on these results we find that Huffman-decoding module using hybrid look-up table improves 2.38 times and 2.65 times of speed in AAC and MP3, than that of using hybrid look-up table. The decoding speed is raised obviously.

4.2. Performance Test of Fast IMDCT

IMDCT36 module test has two aspects: One is the function test which is used to compare the output results between IMDCT36 and original C language function. The other is performance test which compares the computation time between IMDCT36 and original C language function. IMDCT36 module test adopts similar methods. The function test and performance test are respectively completed by *IMDCT36_Comparison* and *IMDCT36_Performance*. After the tests, the average values of consumed time of original C language function and hardware accelerating in 5000 times are respectively acquired, as is shown in Table **5**.

From Table **5**. we can see, the operation time of IMDCT36 is 30.9% lower than original C language program, so IMDCT36 module can satisfy the design requirement and it helps to accelerate the decoding speed. Based on these results, we find two hardware structures contribute to accelerating speed of MP3 decoder in different degrees. The accelerator can accelerate MP3 decoding more effectively.

The difference between accelerator and coprocessor can be analyzed by the test data: Although the accelerator and coprocessor are signal processing modules which are independent of main processor, accelerator itself can operate and calculate while operating pattern of coprocessor is still serial one. DCT64 module controls the computing process by state

Table 4. Results of Huffman decoding module tests.

Clips Source	Times of Consumed Periods in Decoding		Accuracy of Lookup-Table
	Step by Step	Binary Tree	
001.mp3	117690	254167	100%
002.mp3	104546	229345	100%
003.mp3	92298	236980	100%
004.aac	98834	203977	100%
005.aac	78251	195654	100%

Table 5. Performance test of IMDCT36.

Object	Execution Time
Original C code	1440
IMDCT36 module	994

Table 6. SNR of MP3 hardware decoder

Sampling Rate (kHz)	Bit Rate (kbps)	SNR (dB)
44.1	128	90.01
44.1	192	90.38
44.1	256	91.34

machine. Parallel computing of different quantities will affect the needed time of circuit area and the computation in producing balance between area and time. However, there is only one calculation unit in IMDCT36 module design process.IMDCT36 operation is still based on execution sequence of original function with serial form, after original C language function is rewrited according to new design idea. The specified coprocessor can save decoding step so that the performance can be improved to some extent. In order to further improve the performance of coprocessor, a series of multimedia specified expanded instruction set can be designed. Some common specified instructions in operation design will promote the complicated operations to be completed in one instruction, so it can more effectively improve the effects of hardware acceleration.

4.3. Implementation and Verification of FPGA Prototype Chip

The PCM data from computation and software decoding results of MP3 decoder are directly compared. It is found that most of 16bits PCM values of hardware decoder have difference only on the lowest two bits position. When audio playing software CoolEdit plays the audio values, the software and hardware decoding results can be visually compared by output waveform, as described in Fig. (**8**) and (**9**). The time-domain waveform of PCM value by hardware decoding has better tally with that by software decoding, indicating that the quality of hardware decoding is very high.

We test the hardware decoding results of common MP3 code streams with 44.1kHz sampling rate, and the bit rate is 128kbps, 192bps and 256bps. Their SNR related to software decoding are computed and the results are shown in Table **6**.

Since the audio devices are different, the demands for SNR are also different. From the data in Table **3** we can conclude that the MP3 decoder designed in this paper has higher decoding quality and achieves the desired effect.

CONCLUSION

This paper focuses on the MP3 hardware decoder design. It tries to improve and optimizes the decoder based on algorithms analysis, according to the characteristics of hardware implementation. We discuss the mapping of various modules from algorithm to logic structure of hardware, and introduce the control logic and digital path of its sub-modules, to propose corresponding hardware structure of the algorithm. For Huffman module, it demonstrates the improvement of binary tree look-up table method in speed; for inverse quantization

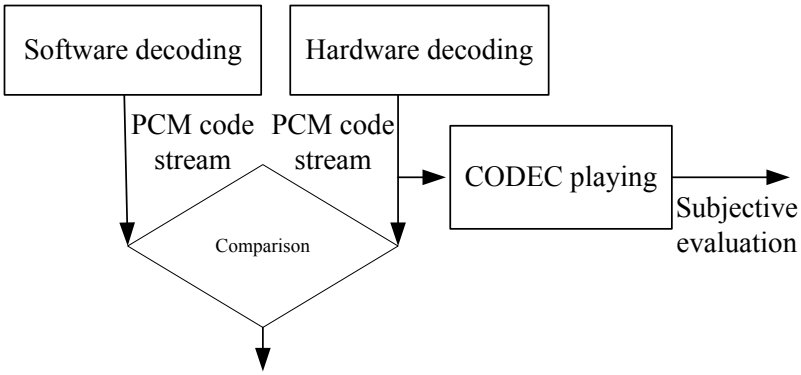

Fig. (8). Test method of decoding results.

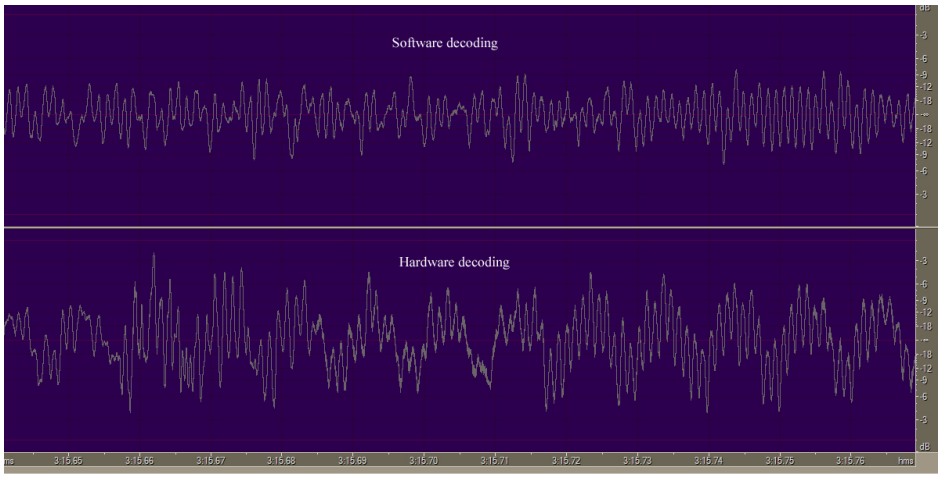

Fig. (9). Time-domain waveform of encoded PCM signals by software and hardware

module, it mainly discusses its advantages compared to current methods, during the computation of *is* value with 4/3 power. It does not need multiplier or multiplication and solves some key problems in design of compensated word code table. For IMDCT module, it explains the unified structure of decoding operation for MP3 and AAC, and the calibration in computation. When introducing the modules design, we verify the characteristics and advantages from functional simulation perspective and provide practical effect based on the prototype chip.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENTS

The research work was supported by the Natural Science Foundation of Education Department of Henan Province (Grant No. 2011B510020; 2011B510022) and the Key Technologies R & D Program of Henan Province (Grant No. 132102210101, 142102210580).

REFERENCES

[1] H. Hedberg, "A complete MP3 decoder on a chip", *Proceedings of IEEE International Conference on Microelectronic Systems* Education, Anaheim, CA, United states, pp. 103-114, 2005.

[2] B. Wang, M. Rong, and W. Liu, "Implementation of MP3 decoder in Openrisc developing system", *Computer Engineering*, vol. 31, no. 11, pp. 205-207, 2005.

[3] K. H. Bang, N. H. Jeong, J. S. Kim, and Y. C. Park, "Design and VLSI implementation of a digital audio specific DSP core for MP3/AAC", *IEEE Transactions on Consumer Electronics*, vol. 1, no. 48, pp. 790-795, 2002.

[4] W. Li, X. Chen, and X. He, "*Design and Implementation of Extracurricular Experiment Board in Digital Electronic Technology*", *Modern Computer*, vol. 12, no. 1, pp. 54-59, 2010.

[5] C. L. Chang, and K.S. Hsu, "Design and implementation of MP3-music on demand system using streaming technology", *Journal of Network and Computer Applications*, vol. 26, no. 4, pp. 291-321, 2003.

[6] P. MalK, "Highly scalable IP core to accelerate the forward/backward modified discrete cosine transform in MP3 implemented to FPGA and low-power ASIC", *IET Circuits, Devices and Systems*, vol.5, no.5, pp.351-359, 2011.

[7] H.S. Kim, S.H. Kim, and K. S. Chung, "FPGA implementation of unified kernel structure for MDCT/IMDCT in audio coding schemes", *Proceedings of International SoC Design Conference*,, Jeju, Korea, pp.100-103, 2011.

[8] V. Nikolajevic, "Fettweis gerhard, computation of forward and inverse MDCT using Clenshaw's recurrence formula", *IEEE Transactions on Signal Processing*, vol. 51, no. 5, pp. 1439-1444, 2003.

[9] B. Wang, M. Rong, and W. Liu, "Implementation of MP3 decoder in Openrisc developing system", *Computer Engineering*, vol. 31, no. 11, pp. 205-207, 2005.

[10] H. Lıu, W. Lıu, and W. Xıng, "Huffman decoding module based on the hardware and software co-design", *Electronic Design Engineering*, vol. 19, no. 16, pp. 174-177, 2011.

[11] H. Li, P. Li, and Y. Wang, "A new decomposition algorithm of DCT-IV/DST-IV for realizing fast IMDCT computation", *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 735-738, 2009.

[12] S.V. Narasimhan, and M. Harish, "Spectral estimation based on discrete cosine transform and modified group delay", *Signal Processing*, vol. 86, no. 7, pp. 1586-1596, 2006.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014