Submit to: Workshop on Multimedia Technologies

Hierarchical Interface Design Methodology: Using Real-Time MP3 codec as a case

Jun-Sheng Zheng , Yeu-Horng Shiau , and Jer-Min Jou

Department of Electrical Engineering, ASIC Lab

National Cheng Kung University

Tainan, Taiwan, R. O. C.

Abstract

In this paper a method for rapidly SoC IP interface design called hierarchical interface design method and models is proposed. The method is an interface design scheme that can be used to integrate different IPs easily. The main concept is to design IP and its interface separately. It introduces a virtual interface concept to simplify interface design. To verify the practicability, we use the hierarchical interface design model to implementation a real-time MP3 codec system. Finally a software /hardware co-simulation is done to verify the entire MP3 real-time codec system. Experiments show that the hierarchical interface design methodology results in minor hardware overhead on the original design. Different IPs can integrate in this scheme to reduce time to market.

Key word: Hierarchical interface design methodology, virtual component, virtual component interface, PVCI

Jun-Sheng Zheng(the contact author):

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: echoiii@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62431-821

Yeu-Horng Shiau,

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: huh@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62431-821

Jer-Min Jou

Current affiliation: Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.

Postal address: ASIC LAB, EE 10F, NCKU, NO.1, Ta-Hsueh Road, Tainan, 701 Taiwan

E-mail address: jou@j92a21.ee.ncku.edu.tw

Telephone number: 06-2757575-62365

1. Introduction

In recent years SoC complexity grows rapidly, different IPs with different functions need to be integrated into a chip. Due to reducing time to market, interface design between different IPs plays an important role. Interface can be a dedicated line or a shared bus. Different IPs communicate with interface, thus interface influences system performance, area, and power.

Hierarchical interface design method and models[2] is an interface design scheme that can be used to integrate different IPs easily. The hierarchical interface design method and models describes interface from system behavior to physical architecture. Without consider the function of components which system connects, designer can integrate different IPs easily into single chip.

The MPEG1 audio layer III (MP3)[1] is a lossy audio compression method, which provides high quality audio under high compression ratio. The MP3 codec system includes different blocks such as subband filter bank (includes analysis and synthesis), MDCT/IMDCT, Huffman coder/decoder, quanitzation/invert quantization. The hierarchical interface design method is used to design interfaces between different blocks. Finally a FPGA verification is done to verify the practicability.

The organization of this paper is as follows. Section 2 describes the hierarchical interface design methodology. MP3 encoder and decoder analysis and their interface design are described in Section 3 and Section 4, respectively. Synthesis and verification results are shown in Section 5. Finally, conclusions are presented in Section 6.

2. Hierarchical interface design methodology

The hierarchical interface design methodology describes IP interface using four levels and 2 domains. The four levels are application level, functional level, virtual component level, and state transition level. The two domains are behavior domain and structure domain.

2.1 Application level

Application level describes data supply and demand between components on system. A data flow

network is adopted to represent the behavior of system, and a node represents data processing component. System designer hasn't to take real transfer behavior into account. Thus system designer can focus on system performance estimation and functional verification. An example of application level is shown in Fig.2-2, which is an invert quantization of MP3 decoding. Input stream pass through a bitstream unpacket to separate it into scalefactor stream and Huffman code stream. The scalefactor stream and Huffman code stream then input the corresponding decoder. After decoding, the decoded scalefactor and decoded code input an invert quantization to re-build original values.

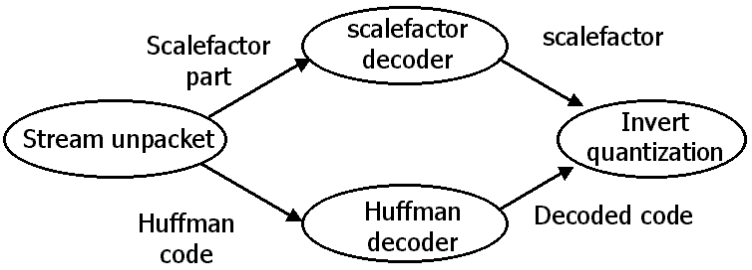

Fig.1 An example of application level

2.2 Functional level:

The functional level describes information of data transfer path. The abstract interface name, I/O port name, interface lifetime are listed to describe the data transfer path. The functional description of data producer and consumer can use high level description language such as C, verilog, or VHDL. Implementation and communication protocol is not sure in this level, thus the interface is still an abstract interface. An example of functional level is shown in Fig.2. Module 1 transfers data by a send procedure through port A, while module 2 gets data by a receive procedure through port B. The send and receive procedures are described in C language.

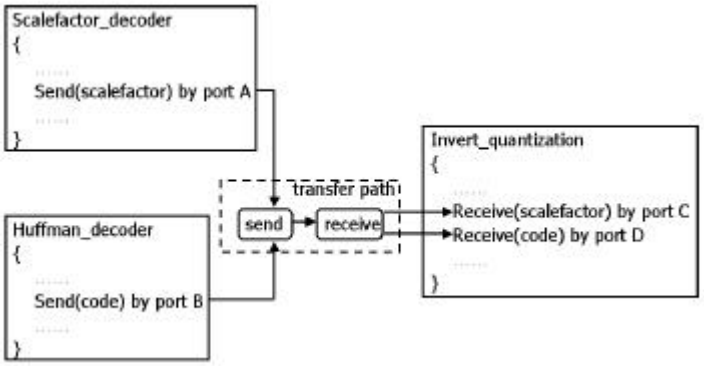

Fig.2 An example of functional level

2.3 Virtual component level

The traditional interface design method design interface of hardware after the definition of functional level is present. Designer designs interface according to the specification between modules described in the functional level. The disadvantage of the traditional method is that if module changes, the interface must be re-design. The hierarchical interface design method maps every module as a virtual component with fixed virtual interface protocol. Thus, when module changes, we only need to adjust virtual interface slightly. The virtual component means a RTL component with a virtual component interface. The virtual component interface converts component original protocol to virtual component interface protocol. The virtual components communicate with virtual component interface. Since they have the same virtual component interface protocol, the wrapper between them can be easily designed. When a module changes, the only part that needed to be adjusted is the wrapper. The relationship of virtual component and virtual interface is shown in Fig. 3.

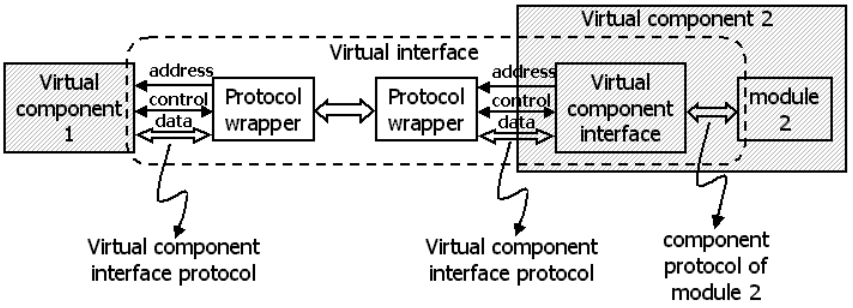

Fig.3 Relation of virtual interface and virtual component

The real implementation and communication protocol is still not sure in this level, but the input/output specification, such as data width, data length, control lines and address lines are known in this level. Thus we use register transfer language to describe data transfer and high level description language such as C, verilog, or VHDL to describe communication between virtual components. The transfer property, such as blocking or non-blocking is also described. Examples of virtual component level on behavior domain and on structure domain are shown in Fig.4(a) and Fig4(b).

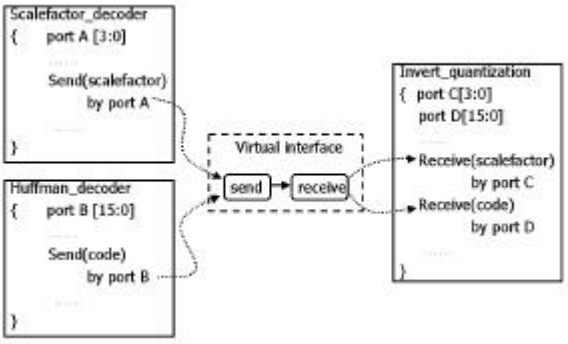

Fig4(a)Examples of virtual component level on behavior domain

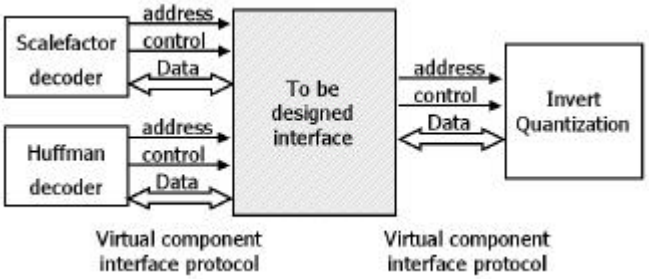

Fig4(b)Examples of virtual component level on structure domain

2.4 State transition level

In this level, we design protocol wrapper from data transfer behavior in the functional level and virtual interface protocol in the virtual component level. Real behavior between virtual component interface protocol such as synchronization and timing constrains are also described in this level. Graphs of state transition level on behavior domain and on structure domain are shown in Fig.5(a) and Fig.5(b).

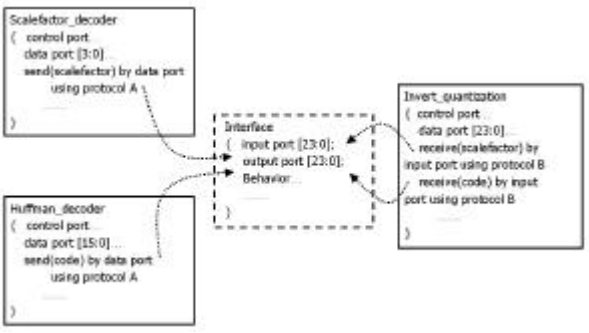

Fig5(a)Examples of state transition level on behavior domain

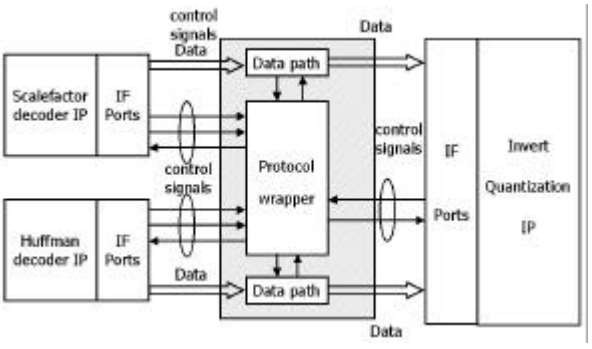

Fig5(b)Examples of state transition level on structure domain

3. MP3 encoder Analysis and Design

The MP3 encoder block diagram is as shown in Fig.6. The input PCM audio samples pass through a poly-phase filter bank, and at the same time the audio samples input a psychoacoustic model. The main function of psychoacoustic model is to calculate the energy transition of input audio samples, and determines block type for MDCT. Finally it finds the signal-to-mask ratio (SMR) for every subband. On the other side, the poly-phase filter bank divides the input audio samples into 32 subbands. The 32 subbands pass through the MDCT and stereo processing. Then we use a loop to find the quantization step. The loop includes quantization and Huffman coding. The main function of the loop is to find a quantization step, such that the quantization distortion is minimum to find the optimized Huffman code. Finally the Huffman codes, quantization scale factors, side information, and header compose a frame. Statistics shows that the subband analysis filter bank and MDCT operation represented 60% of the total encoding time. Thus we use hardware to implement these two parts. The MDCT, reordering and alias reducing are combined into a block.

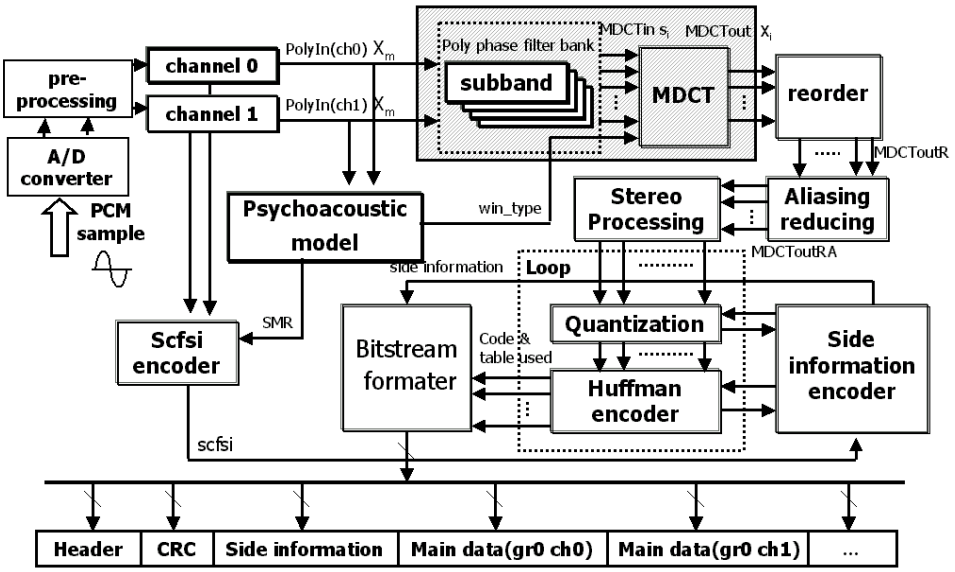

Fig.6 MP3 encoder system block diagram

3.1 Application level analysis

As shown in Fig.7, we use data flow network to describe entire encoder system. The node

represents processing block and the edge represents input/output data. The gray block (poly phase analysis filter bank, MDCT, alias reduction) shows the hardware part of system, and other nodes represent software part. We only list the hardware part edges in Table.1. Because we only use long block of MDCT, psychoacoustic model and reordering are not needed.

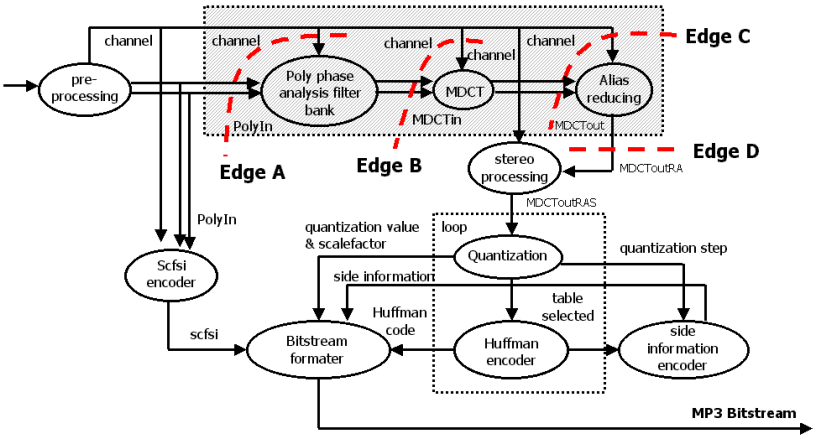

Fig.7 Application level of MP3 encoder

Abstract interface	message	description
Edge A	channel, PolyIn	channel number and input PCM value
Edge B	channel, MDCTin	channel number and MDCT input
Edge C	channel, MDCTout	channel number and MDCT output
Edge D	channel, MDCToutRA	channel number and frequency lines after aliasing reduction

Table.1 Edge properties of poly phase filter bank and MDCT

3.2 Functional level analysis

In this level the simplified algorithms are chosen to process poly phase analysis filter bank and MDCT. According to the symmetry of DCT coefficients[4], the following relation can be derived:

The MDCT cosine coefficient is calculated as:

$$c(i,k) = \cos(\frac{p}{2n}(2i+1+\frac{n}{2})(2k+1))$$

We can derive:

$$\begin{cases} c(\frac{n}{2}-i-1,k) = -c(i,k) & for\ \ 0 \le i \le \frac{n}{4} \\ c(N-i-1,k) = c(\frac{n}{2}+i,k) & for\ \ 0 \le i < \frac{n}{4} \end{cases}$$

The poly phase analysis filter bank coefficient is:

$$M[i][k] = \cos[\frac{(2i+1)\times(k-16)\times \boldsymbol{p}}{64}]$$

It can also be simplified by the DCT coefficients symmetry as:

Symmetry on i direction:

$$\begin{cases} M[i][k] = M[i][32-k] & for\ \ 0 \le k \le 15 \\ M[i][k] = -M[i][16+k] & for\ \ 33 \le k \le 47 \\ M[i][k] = 1 & for\ \ k = 16 \\ M[i][k] = 0 & for\ \ k = 48 \end{cases}$$

for any i

Symmetry on k direction:

$$\begin{cases} M[i][k] = M[31-i][k] = M[15-i][k] = M[16+i][k] & k\%4 = 0 \\ M[i][k] = M[31-i][k] = -M[15-i][k] = -M[16+i][k] & k\%4 = 2 \\ M[i][k] = -M[31-i][k] \qquad M[15-i][k] = -M[16+i][k] & \text{else} \end{cases}$$

for any k

Fig.8 shows the partition of MP3 encoder. We choose bus architecture to implement interface 1 and a point-to-point connect to implement interface 2.

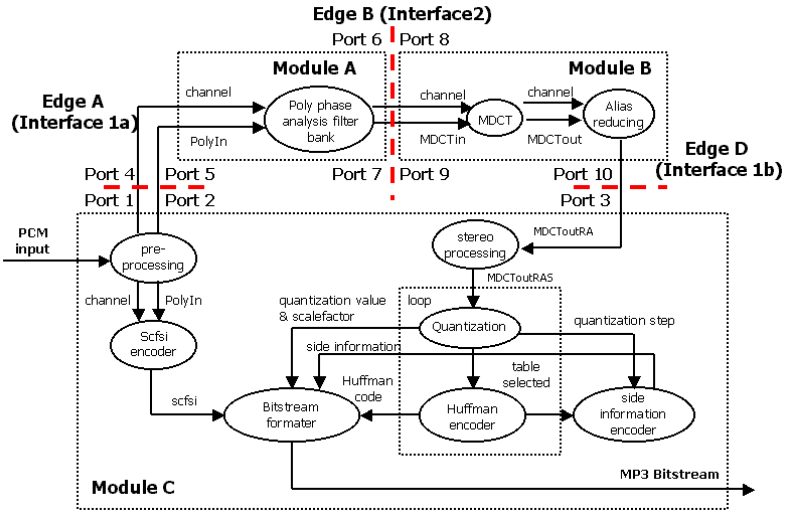

Fig.8 Functional level of MP3 encoder

3.3 Virtual component level analysis

We choose PVCI[3] to complete the interface 1 design, and the interface 2 is point-to-point

connecting. The data IO port properties are valid on this level, the hand shaking signals and control signal will be considered on state transition level, thus we only have to focus on data signals, start, and done signals. A START and DONE signals are added to indicate block start and done, thus we can get IO of poly phase filter bank and MDCT, which is shown in Fig.9.

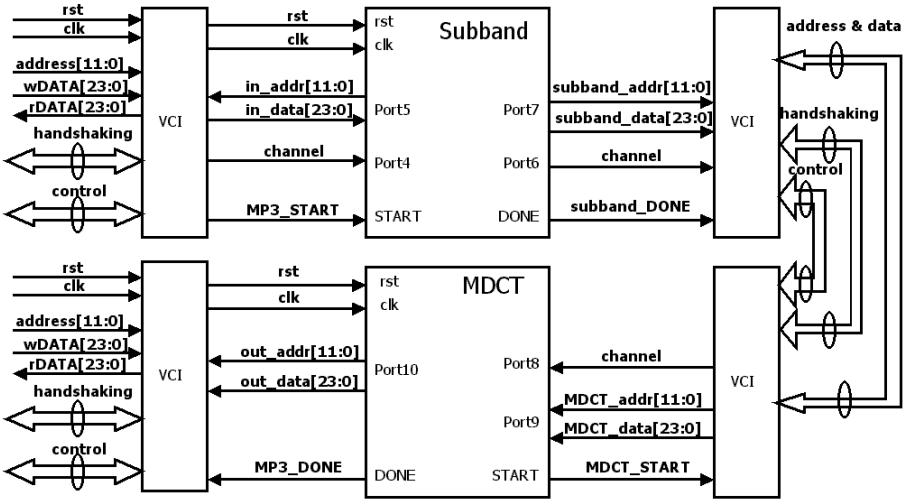

Fig.9 Interface between poly phase filter bank, MDCT and system PVCI

3.4 State transition level design

The architecture of the interface 1 is shown in Fig.10. The PVCI has a read port *rDATA* and a write port *wData*, thus we need input buffer and output buffer to store temporary input and output until the store data reaches the processing unit (576 samples in this example).

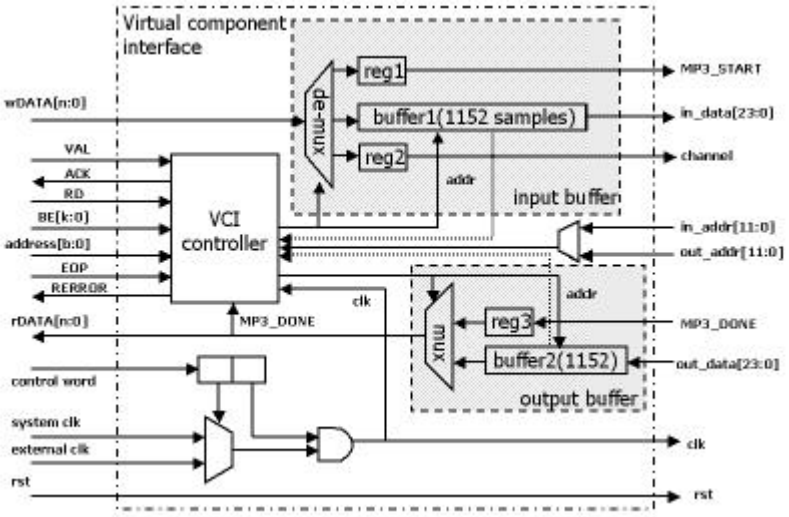

Fig.10 The architecture of the interface 1

The VCI controller is a simple FSM that controls data read and write when PVCI data valid signal *VAL* is high. We use a flexible clock architecture that can be select from system clock or custom external clock.

Because a static architecture is chosen for interface 2, and the packet length of poly phase analysis filter bank output is 32, packet length of MDCT input is 36 from 18 current and 18 previous outputs of poly phase analysis filter bank, a buffer is required for data format converting. The architecture of interface 2 is shown in Fig.12.

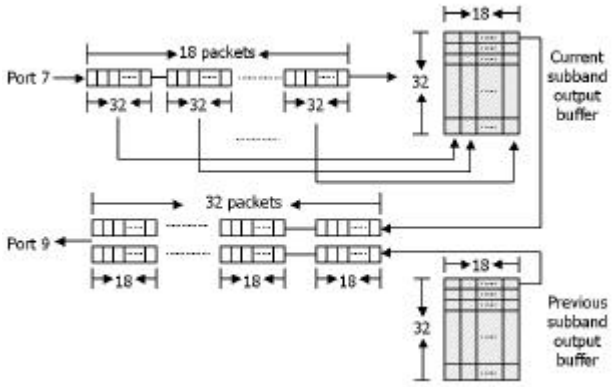

Fig.11 Data format converting between port 7 and port 9

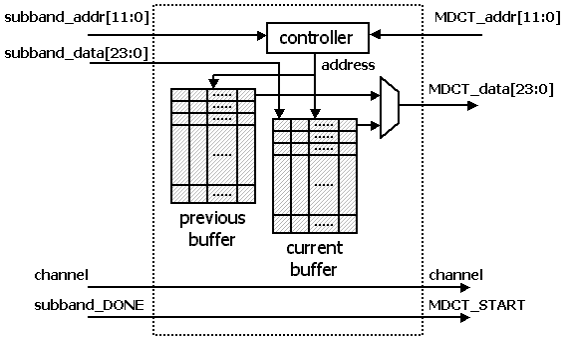

Fig.12 Architecture of interface 2

4. MP3 decoder Analysis and Design

The block diagram of MP3 decoder is as shown in Fig.13. The input MP3 bitstream passes the synchronization and CRC check block, header decoder, and side information decoder. These blocks get CRC, header, and side information and store them to buffers for later use. The main data then pass through a scalefactor decoder to decode scalefactor for inverting quantization. Then the scalefactor and the values from Huffman decoder input the invert quantizer together to re-build the original spectrums. Finally the stereo processing, reordering, alias reduction, IMDCT, and poly phase synthesis filter bank is synthesized to the PCM samples. Same as encoder, the software/hardware interface and poly phase synthesis filter bank/IMDCT interface are designed by hierarchical interface design method and models. The same PVCI protocol is also used as the decoder IP interface protocol, and the decode IP can be easily integrated into any type of bus.

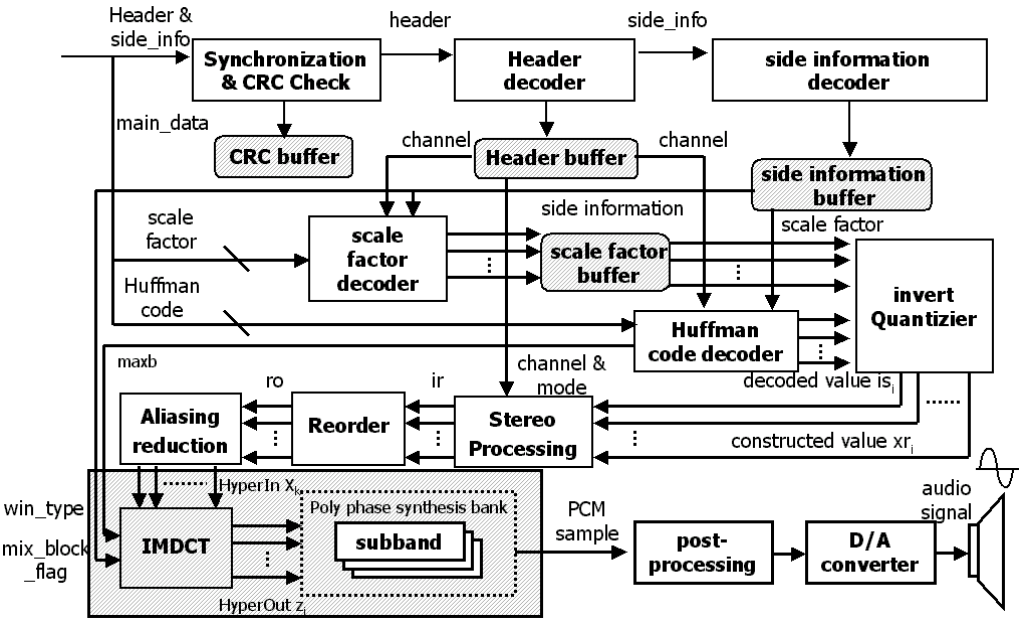

Fig.13 MP3 decoder system

4.1 Application level analysis

As shown in Fig.14, we use data flow network to describe entire decoder system. Gray block (IMDCT, frequency inverse, and poly phase synthesis filter bank) shows the hardware part of system, and other nodes represent software part. We only show the edge between hardware/software, and hardware/hardware.

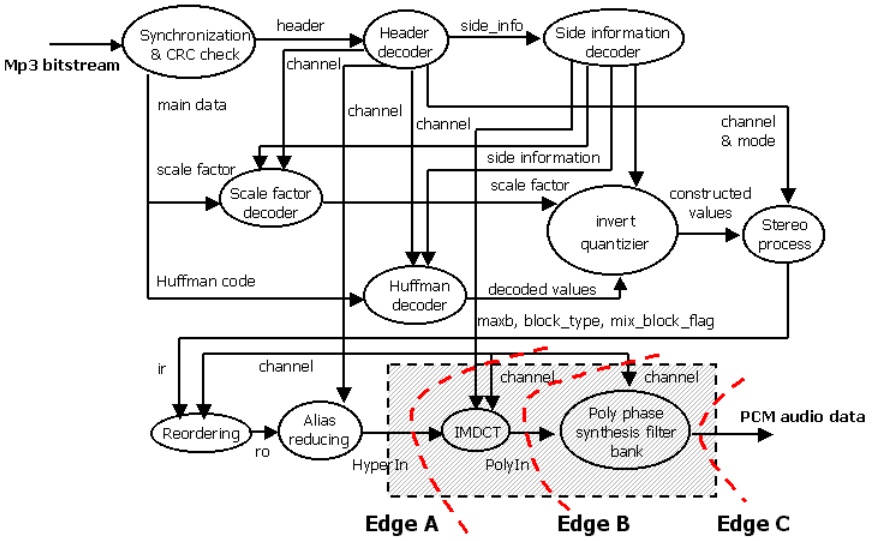

Fig.14 Application level of MP3 decoder

Abstract interface	message	description
Edge A	channel, maxb, mix_block_flag, block_type, HyperIn	channel number, number of non-zero band, block type, mix block flag and IMDCT input
Edge B	channel, PolyIn	channel number and subband input
Edge C	PolyOut	PCM output

Table.2 Edge properties of IMDCT and poly phase filter bank

4.2 Functional level analysis

The bottleneck of computing complexity still lies on the poly phase synthesis filter bank and IMDCT, thus we use hardware to implement them. The same symmetry simplification of DCT is done for poly phase synthesis filter bank and IMDCT cosine coefficients. The Functional level of MP3 decoder is shown in Fig.14.

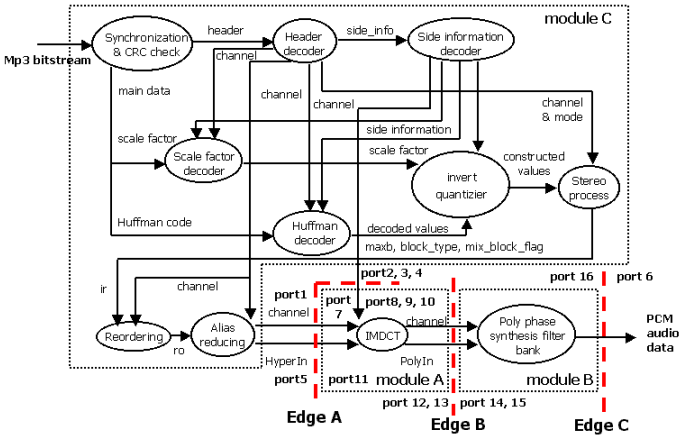

Fig.14 Functional level of MP3 decoder

4.3 Virtual component level analysis

We also choose PVCI to complete the interface 1 design, and the interface 2 is point-to-point connecting. Like the encoder, the data IO port properties are valid on this level, and we only have to focus on data signals, start, and done signals. A START and DONE signals are added to indicate block start and done, thus we can get IO of poly phase filter bank and IMDCT, which is shown in Fig.15.

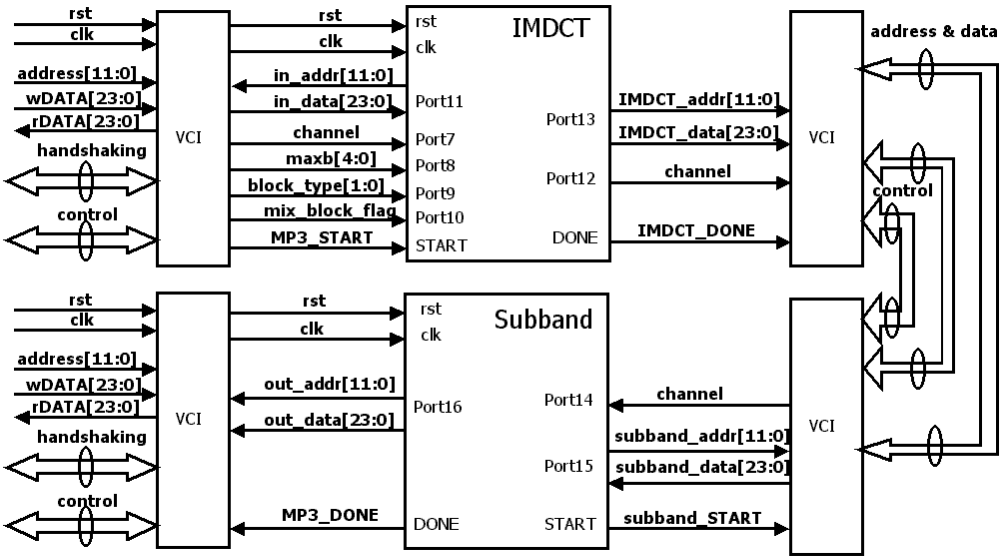

Fig.15 Interface between IMDCT, poly phase synthesis filter bank and system PVCI

4.4 State transition level design

The architecture of the interface 1 is shown in Fig.16. The decoder needs several parameters to decode, such as *maxb*, *block_type*, *mix_block_flag*, because PVCI protocol is chosen, these parameters are transferred like normal data on write data bus *wData* and are stored in the same buffer for later use.

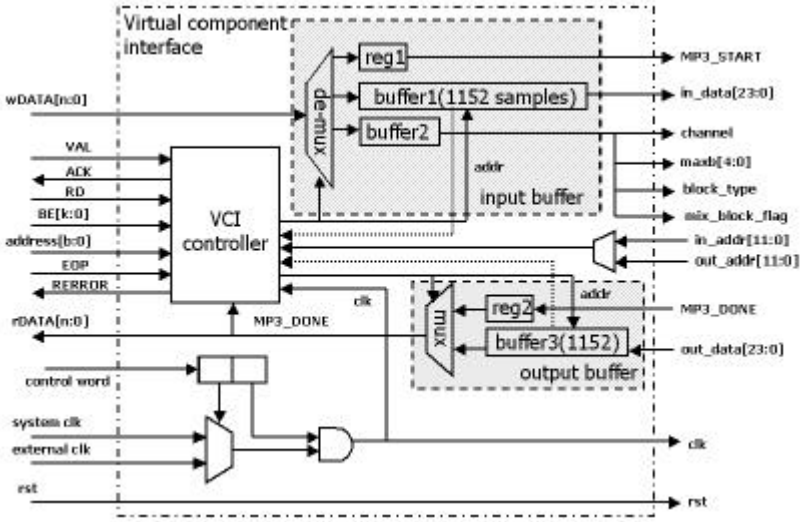

Fig.16 Architecture of interface 1

As shown in Fig.17, packet length of the IMDCT output is 18, while packet length of the poly phase synthesis filter bank input is 32, a buffer is needed to convert different data format. Thus the

architecture of interface 2 is shown in Fig.18. Because the parameters ***maxb[4:0]***, ***block_type[1:0]***, ***mix_block_flag*** are not required for synthesis filter bank processing, only channel is transferred to poly phase synthesis filter bank.

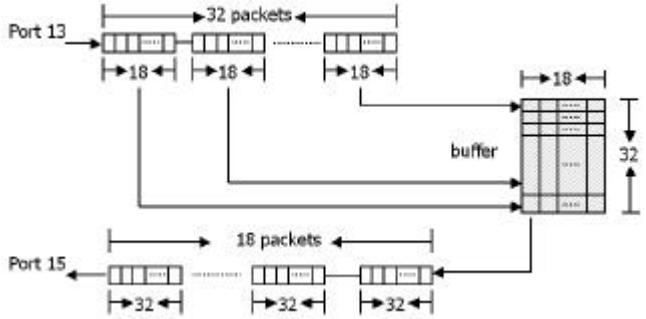

Fig.17 Data format converting between port 13 and port 15

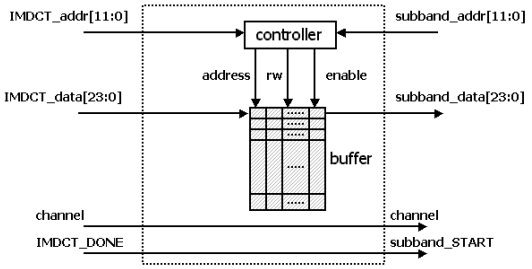

Fig.18 Architecture of interface 2

5. Verification and synthesis results

5-1 Verification strategy

As shown in Fig.19, the verification strategy of encoder and decoder is done by a bottom-to-top scheme.

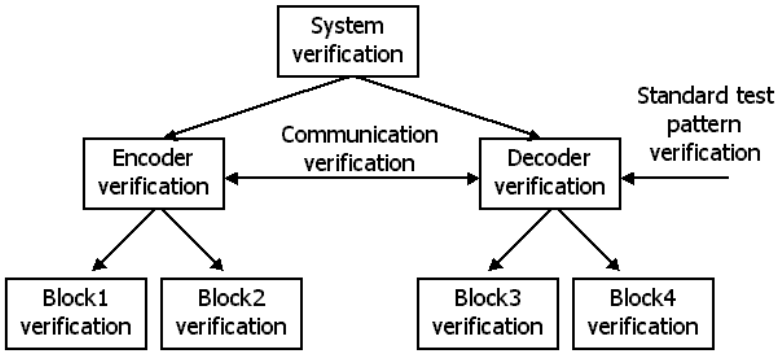

Fig.19 Verification strategy

Functional block verification: The basic functional blocks (poly phase analysis/synthesis filter bank, MDCT/IMDCT) are verified with bit-by-bit comparing to software result.

Encoder/decoder verification: After functional block verification, we can integrate functional blocks to encoder or decoder. The decoder is then tested by standard test pattern (Layer III test bitstream package v2.2 M.Dietz FhG/IIS/Inel 24.04.1994). The encoder and

decoder are verified using software/hardware co-simulation scheme. The verification module of encoder is shown in Fig.20. The software part of encoder verification platform is Pentium II-300 with 128 MB RAM, while the hardware target is Xilinx VirtexE-2000 FG680. The real environment of encoder platform is shown in Fig.21.

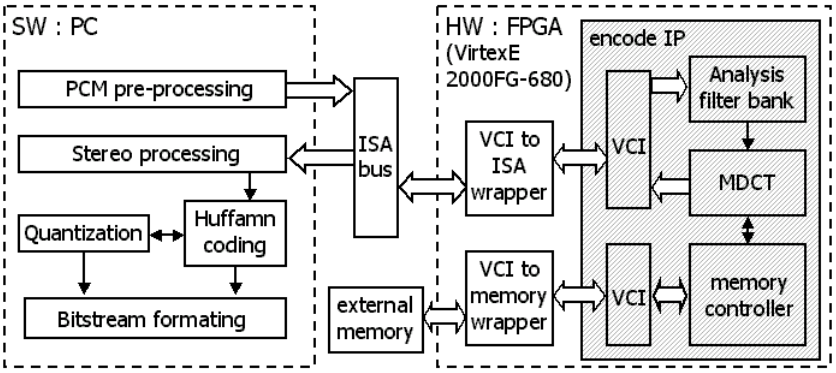

Fig.20 Verification module of encoder

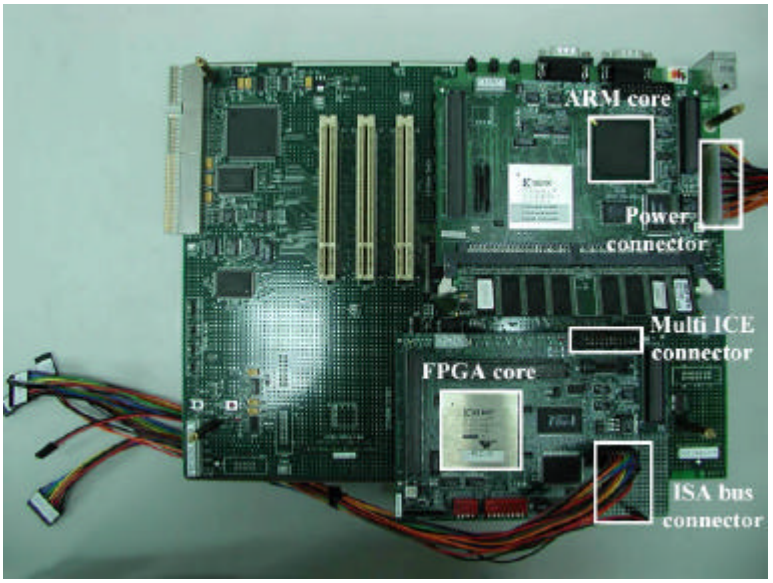

Fig.21 Verification module of encoder

The verification module of decoder is shown in Fig.22. The software part of decoder verification platform is Pentium III-450 with 256MB RAM, while the hardware target is Xilinx Virtex-1000 BG560. The interface between software and hardware is ISA bus. The real environment of decoder platform is shown in Fig.23.

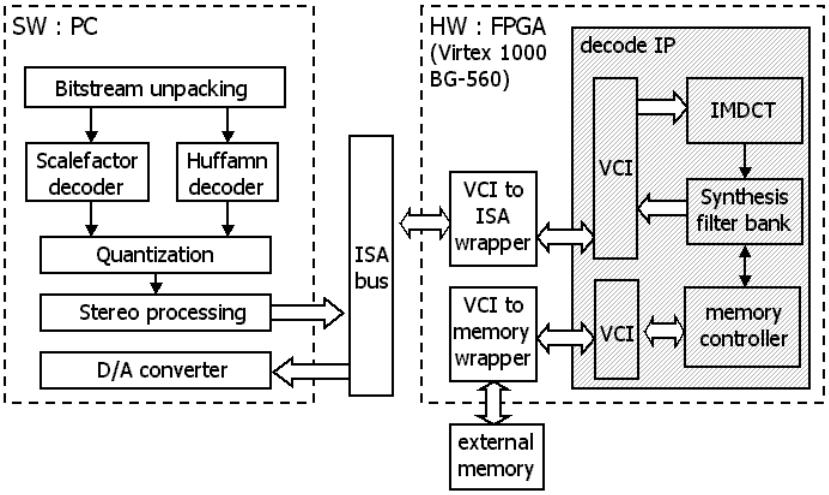

Fig.22 Verification module of decoder

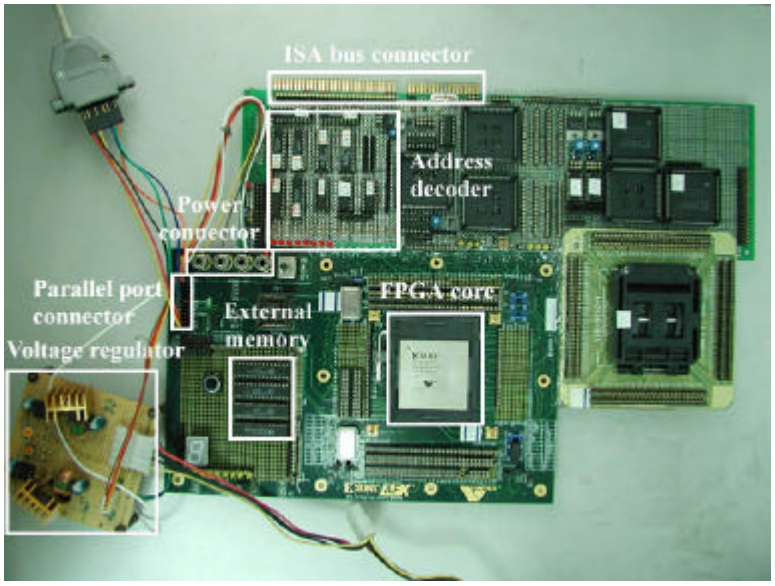

Fig.23 Real FPGA environment of decoder

System verification: The system verification is done by using human listening tests. The verification on this level can test coding and decoding distortion.

5-2 Synthesis result of encoder

The synthesis result and resource using is shown in Table.3, we can find that the maximum frequency can reach about 20MHz. For an audio with maximum available sampling frequency 48kHz and dual channels, the encoder must processing 41.67 frames in a second. In other words, one frame must been encoded in about 24ms. In out design, one frame can be processed in

68653*20=3.43ms without interface communication time, thus real time coding can be reached. The FPGA layout of encoder is shown in Fig.24.

Resource	USED	MAX available	%USED
Number of Slices	4181	19200	21%
Total Number Slice Registers	853	38400	2%
Total Number 4 input LUTs	7560	38400	19%
Number used as LUTs	7530		
Number used as a route-thru	30		
Number of bonded IOBs	155	512	30%
Number of Block RAMs	42	160	26%
Number of GCLKs	1	4	25%
Number of GCLKIOBs	1	4	25%
Total equivalent gate count	609415		
Additional JTAG gate count for IOBs	7488		
Minimum period	43.653ns (Maximum frequency: 22.908MHz)		
Maximum net delay	24.334ns		

Table.3 Resource using of encoder

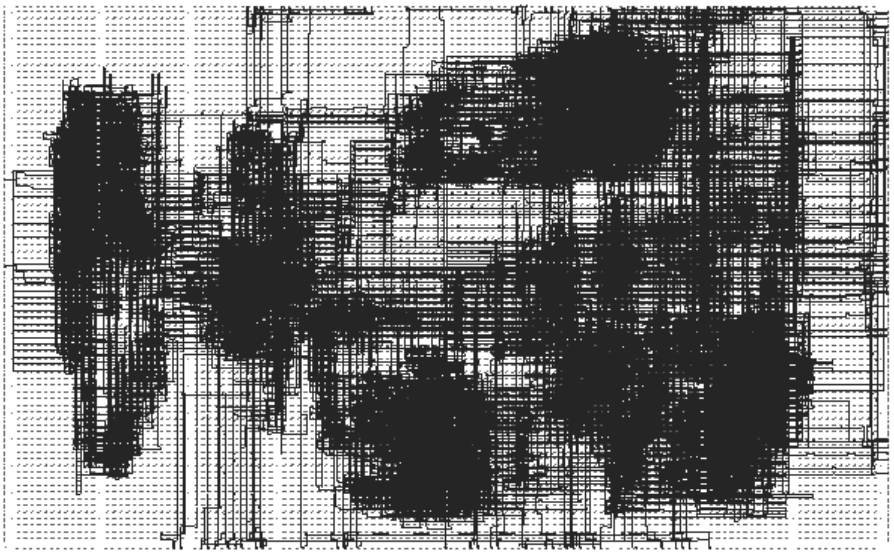

Fig.24 FPGA layout of encoder(Target: VirtexE-2000 FG680)

5-3 Synthesis result of decoder

The synthesis result and resource using is shown in Table.4, we can find that the maximum frequency can reach about 20MHz, thus real time decoding can be reached. The FPGA layout of encoder is shown in Fig.25.

Resource	USED	MAX available	%USED
Number of Slices	4231	12288	34%
Number of Slice Flip Flops	1922	24576	7%
Total Number 4 input LUTs	6723	24576	27%
Number used as LUTs	6686		
Number used as a route-thru	37		
Number of bonded IOBs	103	404	25%
Number of Block RAMs	32	32	100%
Number of GCLKs	2	4	50%
Number of GCLKIOBs	2	4	50%
Total equivalent gate count	583704		
Additional JTAG gate count for IOBs	5040		
Minimum period	44.779ns (Maximum frequency: 22.332MHz)		
Maximum net delay	19.159ns		

Table.4. Resource using of decoder

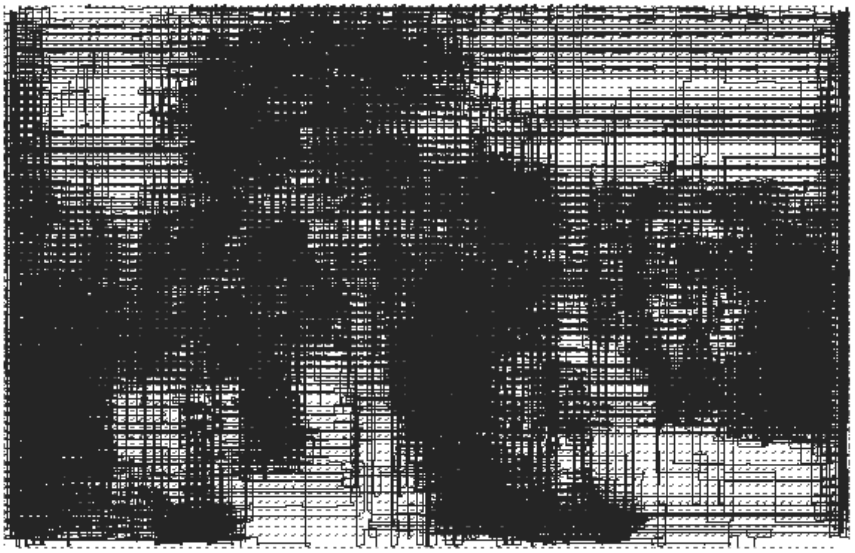

Fig.25 FPGA layout of decoder (Target: Virtex-1000 BG560)

6. Conclusions

In this paper we presents a method for rapidly SoC IP interface design. A real-time MP3 codec system is used as an example to verify the practicability. Since a standard PVCI is used to implement the interface of MP3 codec system, system designer can easily integrate the MP3 codec into any bus architecture. Experiments show that the hierarchical interface design methodology results in minor performance overhead on the original design. Different IPs can be integrated by this scheme to reduce time to market.

7. Reference

[1] ISO/IEC 11172-3. "Coding of moving picture and associated audio for digital storage media at up to about 1.5Mbit/s —Part 3 audio", November 1991.

[2] Jer-Ming Jou, Kuang-Ming Wu. "Research and development of a hierarchical interface design methodology and models for SoC IP integration", Department of Electrical Engineering NCKU, Tainan, Taiwan, R.O.C. June, 2001.

[3] Virtual Socket Interface Alliance™, "Virtual Component Interface Standard Version 2", April 2001.

[4] Y. H. Fan, V. K. Madisetti, and R. M. Mersereau. "On fast algorithm for computing the inverse modified discrete cosine transform", IEEE 1999.

[5] D. Y. Chan, J. F. Yamg and C.C. Fang. "Fast implementation of MPEG audio coder using recursive formula with fast discrete cosine transforms", IEEE Trans. speech and audio processing, Vol.4, No.2, pp144-148, March 1996.

[6] D. Pan. "A tutorial on MPEG/audio compression", IEEE Multimedia, Vol. 2, No. 2, Summer 1995.

[7] J. Enerstam and J. Peman. "Hardware implementation of MPEG audio real-time encoder", Lulea University technology, September 1998.

[8] T. Sakamoto, M. Taruki, and T. Hase. "A fast MPEG audio layer III algorithm for 32 bit MCU", IEEE Trans. on Consumer Electronics, Vol. 45, No. 3, August 1999.

[9] VSIA. http://www.vsi.org/