Design of Entropy Decoding Module in Dual-Mode Video Decoding Chip for H. 264 and AVS Based on SOPC

Hong-Min Yang, Zhen-Lei Zhang, and Hai-Yan Kong
School of Information Science and Engineering, Shandong University
yanghm900930@163.com
zzlyml123@126.com
sdkhy0808@126.com

Abstract— **This document introduces the hardware design of the entropy decoding module in dual-mode video decoding chip for H. 264 and AVS standard. The entropy decoding module designed in this document can realize the decoding of CA-2D-VLC for AVS standard and CAVLC and CABAC for H.264 standard. The Verilog HDL is used to accomplish RTL design and the FPGA implementation is achieved on an Altera Cyclone II EP2C35F672C6. The result indicates that the design efficiently reduces the circuit area and improves the speed of decoding.**

***Keywords*—H.264; AVS; entropy decoding; FPGA; Verilog HDL**

I. INTRODUCTION

H.264 is currently the most advanced video coding standard，and AVS is China's independent intellectual property rights of audio and video coding standard. Because H.264 standard has grabbed the opportunity of domestic industrialization, but AVS standard is increasingly perfect and gets the vigorous support of the government and already has certain industrial development. So, AVS and H.264 standard will have mutual fusion in our country, and it follows that a requirement is put forward that the video decoding chip must be compatible with the two standards.

The entropy decoding module designed in this document can achieve entropy decoding for AVS and H.264 standard. The design makes a lot of improvement in the circuit structure and look-up algorithm, which improves the decoding efficiency. The stream buffer and shifter, as well as the Exp-colomb decoding module, is reused, and the tables of AVS and H.264 standard are optimized. Combinational logic is used to look up tables, which reduces the use of memory, and the pipelining structure improves the decoding speed. In CAVLC decoding module, the decoding of the coefficient Coffetoken and Trailing_ones_sign_flag are realized in a clock cycle, and the recombinant of residual coefficient is done while decoding Runbefore, which saves the clock cycle. In CABAC module, efficient arithmetic decoding module is adopted, which improves the CABAC decoding speed.

II. ENTROPY DECODING ALGORITHM

The entropy decoding algorithm for AVS standard is adaptive variable length coding based on Exp-colomb code. The syntax elements on macro block layer and above it use fixed-length code or Exp-colomb code of zero order. Residual coefficients adopt context-based adaptive 2-d variable length coding (CA-2D-VLC). The array of (Run, Level) is obtained by doing run-length encoding (RLE) to residual coefficient matrix, and Run presents the number of zeros before every non-zero coefficients, and Level is on behalf of the value of quantization coefficient [1]. Then 19 variable length code tables are looking up according to (Run, Level) to obtain the corresponding codes which are encoded by Exp-colomb code of zero, one, two or three order. So, before variable length decoding for AVS, Exp-colomb decoding of k order is needed, and the formula of the CodeNum and Length are as follows:

$$CodeNum = 2^{leadingZeroBits+k} - 2^k + read\_bits \quad (leadingZeroBits + k) \tag{1}$$

$$Length = 2 \times leadingZeroBits + k + 1 \tag{2}$$

where leadingZeroBits is the number of consecutive zeros in prefix of codeword. Then get syntax elements and (Run, Level) by mapping CodeNum, and finally obtain residual coefficients by inversely scanning (Run, Level).

In the basic and extensional profile of H.264, residual coefficients use context-based adaptive variable length coding (CAVLC), but fixed-length code or Exp-colomb code of zero order, which is the same as that of AVS, is adopted in other profiles. So, it lays a theoretical foundation for the reuse of the Exp-colomb decoding module.

However, arithmetic coding method is also introduced to the main profile of H.264. That is to say, the syntax elements below the slice layer and residual coefficients can adopt context-based adaptive binary arithmetic coding (CABAC). When using CABAC to encode, data in the slice is divided into 399 context models, each of which is identified with ctxIdx, and in each model the probability

of looking up and updating is done. At the start of encoding, binarization towards syntax elements is done, and the syntax elements are transformed into corresponding binary codes. Then at the beginning of each slice, 399 context models are initialized and arithmetic coding can be carried out on the next. After the completion of arithmetic coding, the code engine is normalized and the context models are updated. So when decoding, choose context model firstly according to corresponding syntactic elements, and look up the corresponding probability tables and calculate progressive interval. Then do binary arithmetic decoding, and finally do anti-binary method.

Knowing from the analysis of the above, the difference of variable length coding algorithm of AVS and H.264 mainly lies in the process of the residual coefficient. So in the design of the entropy decoding module, CA-2D-VLC module and CAVLC module share the stream buffer and shifter as well as the Exp-colomb decoding module. And the CABAC module reuses the stream buffer and shifter. Finally, a controller is used to control each module to work orderly.

III. HARDWARE DESIGN

According to the analysis of algorithms, this document designs an entropy decoding module compatible with AVS and H.264 standard. The hardware structure is shown in Fig.1. The module consists of seven parts: decoding controller, Dual-port RAM, stream buffer and shifter, Exp-colomb decoding module, CA-2D-VLC decoding module, CAVLC decoding module and CABAC decoder module.

Dual-port RAM can be regarded as a cache between entropy decoding module and external input, which stores the compressed video input code flow after removing filling bits. It outputs 32-bit corresponding code stream to the stream buffer and shifter according to the reading address coming from the entropy decoding module. Stream buffer and shifter stores 64-bit unsolved code stream, and it removes code words having been decoded and outputs unsolved code stream according to the code length information from the code length accumulator. What's more, it can load new stream from the RAM according to the loading signal from the code length accumulator. Exp-colomb decoding module completes Exp-colomb decoding of the code flow to get the value of CodeNum, and gets corresponding syntax elements by mapping CodeNum. When decoding residual coefficients of AVS, CA-2D-VLC decoding module achieves 2-d variable length decoding according to CodeNum from Exp-colomb decoding module, and outputs (Run, Level), and then gives residual coefficients through inversely scanning. When decoding stream of H.264, if entropy decoding mode is 0, then enable CAVLC decoding module and decode residual coefficients. CAVLC decoding

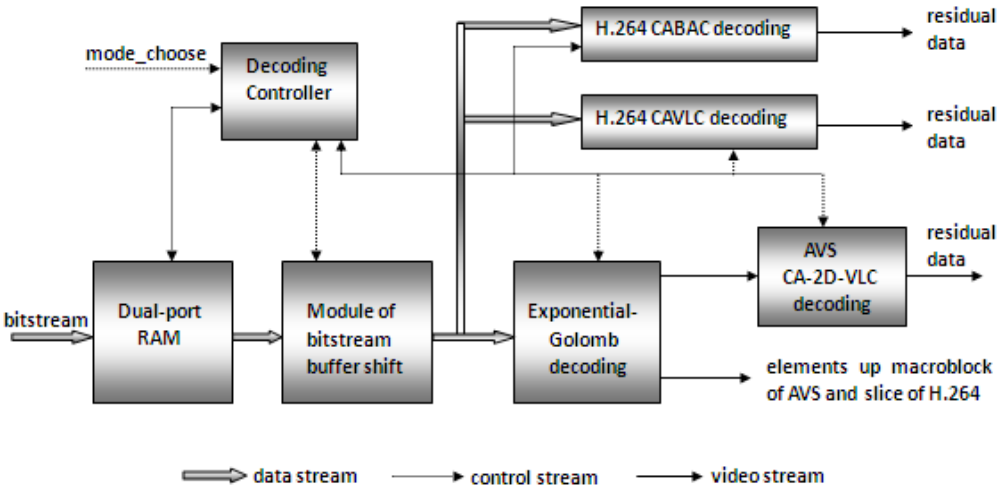

Fig.1 The hardware structure of entropy decoding module

module directly gets unsolved code words from the stream buffer and shifter to work out the syntax elements such as coeff_token, level, runbefore and so on. Then the syntax elements are reordered and the residual coefficients are regained. When decoding stream of H.264, if entropy decoding mode is 1, use the CABAC decoding module to decode the syntax elements in slice layer and below it and residual coefficients. The controller is a Finite State Machine (FSM), which is used to regulate operation of the decoder orderly. According to the difference of the decoding video stream, it is needed to input the signal of *mode_choose* to the controller to choose the mode of the entropy decoding module, and make the entropy decoding module choose the corresponding decoding pathway and hang free decoding module to reduce power consumption.

A. Stream Buffer and Shifter

Stream buffer and shifter is used to be the cache of the code stream loaded from the dual-port RAM, and output unsolved codes for subsequent decoding module through the shift operation[2]. In order to support the mixing length decoding, this document adopts the method of multiple parallel decoding. Through applying the stream buffer and shifter, 32-bit stream is read at a time, and each time one length of the stream is decoded, which improves the speed of the decoding. The structure of the module is shown in Fig.2.

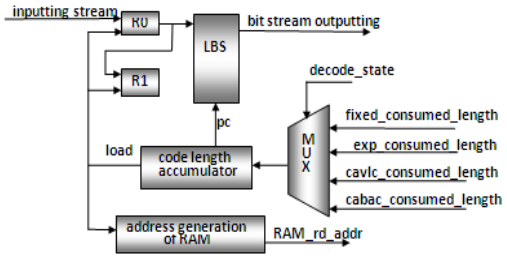

Fig.2 The hardware structure of stream buffer and shifter

This module contains two 32-bit registers R0 and R1, which together make up the 64-bit input of Left Barrel Shifter (LBS). LBS is accomplished by combinational logic, and after it receives the address pc from the code length accumulator, it makes the 64-bit codes input move left pc-bit instantly to generate 32-bit output data new. When the signal load from code length accumulator is high, it indicates that total length of the code words having been decoded is out of 32 bits and 32-bit codes new must be read from the code stream. When the clock edge is effective, R0 stores the 32-bit codes new, and R1 receives the original data from R0 and that is the new data input of LBS. And at the same time, the code length accumulator begins a new round of accumulation, and the internal pc[4:0] accumulates code length consumed again and again. When the pc[4:0] is beyond 31, the code length accumulator will produce carry signal which makes the signal load valid, and that illustrates that more than 31-bit codes have been decoded and new stream must be input. Signal load must be input the module generating RAM address to generate reading address ram_rd_addr for RAM. Whenever the signal load is effective, ram_rd_addr adds 1 and is sent to the dual- port RAM, which makes the RAM output new code flow for updating R0 next time.

In this document, the stream buffer and shifter joins a data selector MUX in order to select the type of the current code length consumed. According to decoding state signal decode_state, it chooses the current code length consumed in the decoding from code length of fixed-length decoding, Exp-colomb decoding, CAVLC decoding and CABAC decoding. So each module can use this module.

B. Exp-colomb Decoding Module

In this document, the reuse of Exp-colomb decoding module, which can be used in decoding of AVS and H.264, is adopted, so the resource consumed of the circuit is reduced. The hardware structure of this module is shown in Fig.3.

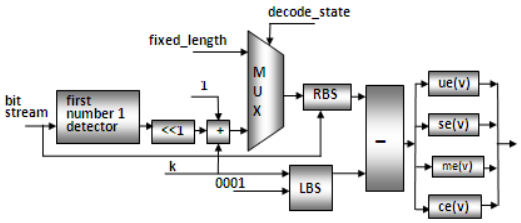

Fig.3 The hardware structure of Exp-colomb decoding module

The first number 1 detector is for detecting the number of consecutive zeros in the prefix of the input stream. It consists of an or gate, and an 8:3 priority encoder and an data selector. It can be used in Exp-colomb decoding module, looking up tables in CAVLC decoding module and normalization in CABAC decoding module.

Because the number of zeros in front of the first '1' is not more than 15, first 16 bits of the 32-bit stream are only needed to be input to the first number 1 detector. The first number 1 detector can detect the number of zeros leadingZeroBits in the prefix. Then according to the formula (2), leadingZeroBits, which is moved left one bit, plus the order k of Exp-colomb code and 1 is the length of the Exp-colomb decoding length. But, the analysis found that the relationship among the binary size of Exp-colomb code CodeValue and leadingZerosBits and k is as follows.

$$CodeValue = 2^{leadingZeroBits+k} + read\_bits(leadingZeros + k) \quad (3)$$

Improving formula (1), the relationship between CodeNum and CodeValue is as follows.

$$CodeNum = CodeValue - 2^k \quad (4)$$

Formula (4) illustrates that the binary size of Exp-colomb code minus 2^k is equal to CodeNum, so we simplify the calculation of CodeNum according to this relationship so as to simplify the design. Deliver the 32-bit input code flow into right barrel shifter(RBS) and move it right 32-length bits, and then codevalue is obtained. The codevalue minus 2^k is equal to CodeNum. Also the design joins a code length data selector which makes this structure can support fixed-length code decoding. That's to say, when decode_state is in the condition of fixed_decoding, the length of fixed-length code is input to RBS and then the input stream is moved directly, and fixed-length code can be decoded.

After decoding the codenum, the syntax elements of

ue(v), se(v), me(v) and ce(v) is obtained by mapping codenum .

C. CA-2D-VLC Decoding Module

CA-2D-VLC decoding module can decode residual coefficients of AVS. And it gets residual data (run level) according to the code value codenum after Exp-colomb decoding. In this document, the index of look-up tables and the tables are optimized and integrated, and adaptive pipelining technology is introduced to improve the decoding speed. The hardware structure of this module is shown in Fig.4.

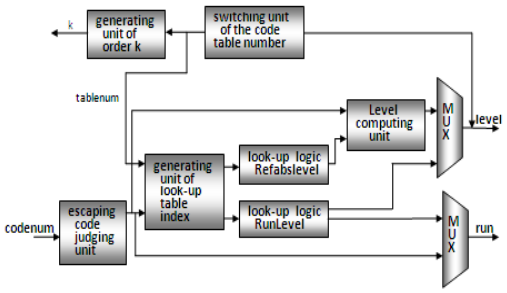

Fig.4 The hardware structure of CA-2D-VLC decoding module

In this structure, first input codenum into escaping code judging unit. If codenum is less than 59, there is no need to decode escaping code, and the syntax element trans_coefficient equals codenum. And then current used table number tablenum generated by trans_coefficient and switching unit of the code table number is input into the generating units of look-up table index, and a look-up table index is generated to look up the look-up logic RunLevel and the value of run and level is obtained. If codenum is greater than or equal to 59, the escaping code decoding is needed to be introduced. The first codenum is assigned to the syntax elements trans_coefficient. In escaping code judging unit, calculate run = (trans_coefficient-59) / 2 as the output run. And then run and tablenum are input to generating unit of look-up table index to generate the index of look-up logic Refabslevel, and look-up logic Refabslevel is searched to get refabslevel. The next codenum is assigned to the syntax elements escape_level_diff and then level is worked out by the refabslevel and escape_level_diff. If trans_coefficient is odd, then the level is equal to -(refabslevel +escape_level_diff), otherwise the level is equal to (refabslevel+ escape_level_diff).

When decoding run and level, there are 19 variable length code tables that you need search. In the document, the index of the look-up code tables is optimized, and combinational logic is used to search the look-up code tables. That's to say, the look-up logic of Refabslevel and RunLevel is designed to combinational logic circuit, and look-up index composed by the trans_coefficient and tablenum is input to the combinational logic circuit to get the value of run, level and refabslevel.

In addition, CA-2D-VLC decoding module uses the adaptive pipelining technique. Considering the principle of balance and the need of decoding process, the variable length decoding pipeline is divided into the following four stages: the output of stream buffer and shifter, Exp-colomb decoding, escaping code decoding and RunLevel decoding output. Main arithmetic unit of the output of stream buffer and shifter is the stream buffer and shifter, completing buffering the output stream from RAM and moving the output stream to be decoded. Main arithmetic unit of Exp-colomb decoding pipelinig segment is Exp-colomb decoding modules, achieving the function of decoding Exp-colomb codes. Main arithmetic unit of escaping code decoding pipelining segment is also Exp-colomb decoding module, working out escaping code escape_level_diff. Main arithmetic unit of RunLevel decoding output pipelining segment is RunLevel calculation module, calculating run and level by looking up variable length code tables of AVS. These three arithmetic units cascade to form a complete pipeline through register and each arithmetic unit can work in parallel in each clock cycle, which improves the decoding speed. The adaptive pipeline can adjust the pipelining stages according to different code words. The workflow of the pipeline is shown in Fig.5.

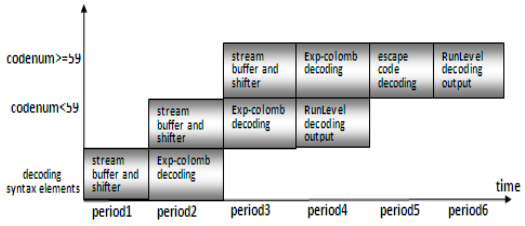

Fig.5 The workflow of the pipeline

D. CAVLC Decoding Module

CAVLC decoding module is for decoding the residual data encoded by CAVLC in video stream of H.264. CAVLC decoding module achieves the decoding of six kinds of syntax elements including TotalCoeffs, TrailingOnes, the sign trailing coefficients T1, Levels, TotalZeros and RunBefore from the compressed bit stream from the stream buffer and shifter. And then these syntax elements are recombined to get the residuals coefficients. The structure of CAVLC decoding module is shown in Fig.6.

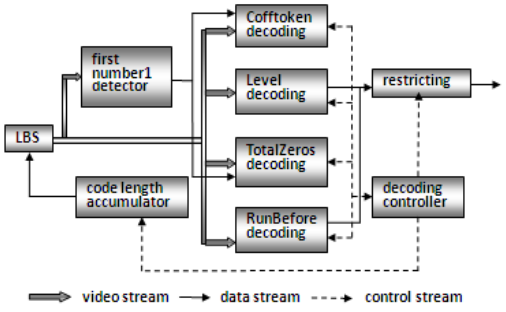

Fig. 6 The structure of CAVLC decoding module

CAVLC decoding module and other decoding module reuse stream buffer and shifter, so the Left Barrel Shifter (LBS) and code length accumulator in Fig.6 are not needed to be designed again. That's because the structure of coefftoken, level and totalzeros in the compressed code stream of CAVLC is similar with that of Exp-colomb code, both adopting structure of [m consecutive 0] 1 [suffix]. So when decoding such syntax elements, the number of zeros in the prefix is detected by first number 1 detector, and then shorter suffix syntax elements are decoded to get code value. Thus in this structure, first number 1 detector in Exp-colomb decoding module is reused. A part of the input stream is sent to first number 1 detector to get the number of zeros in the prefix, and another part is sent to syntax elements decoding unit, which unites the number of zeros in prefix to get code value. In syntax elements decoding unit, firstly, Coefftoken decoding unit and the sign of trailing coefficients decoding unit are used to work out the total number of non-zero coefficients totalcoeffs and the number of trailing coefficients trailingones, which are sent to the subsequent decoding unit, and subsequently Level decoding unit repeatedly works (totalcoeffs- trailingones) times to have the value of non-zero coefficients except the trailing coefficients Level. Then TotalZeros decoding unit is enabled to work out the total number of zeros before the last non-zero coefficient totalzeros with entrance parameter totalcoeff. Finally RunBefore decoding unit is motivated for the number of zeros before non-zero coefficients. Restructuring module works out the residual coefficients according to the syntax elements above. CAVLC decoding controller consists of a FSM, and distributes decoding task for each unit in different decoding cycles and makes each unit run orderly to rebuild residual coefficients.

Combinational logic is adopted to look up tables in this module, which reduces the access to the memory and saves resources. Factor of priority is took into account when combinational mapping circuit is designed. So code tables are divided into several sub-tables according to the frequency of the codes and the similarity of the suffix of the codes. The smaller the number of sub-tables is, the higher the priority is. When designing combinational circuit, give higher priority to these tables. So when combinational logic circuit searches tables, in most cases tables having higher priority will only be searched, having not to search all of the mapping tables. Consequently, the efficiency of looking up tables is improved, and the power consumption is reduced. Level decoding module is implemented with arithmetic operation, and achieves the decoding of the prefix and suffix in a decoding clock cycles, and the value of a Level can be got each clock cycle. Restructuring module starts to work after working out first runbefore, and finally (i_level-i_run) clock cycles are only consumed to rebuild residual data, which saves a lot of decoding clock cycles.

E. CABAC Decoding Module

When entropy decoding mode of H.264 is 1, syntax elements in slice layer and below it and residual data apply CABAC decoding module. The module receives the compressed code stream from the stream buffer and shifter, and does arithmetic decoding to get every bit of the syntax elements, and finally applies the anti-binary method to get the value of the syntax elements. The hardware structure of the module is shown in Fig.7.

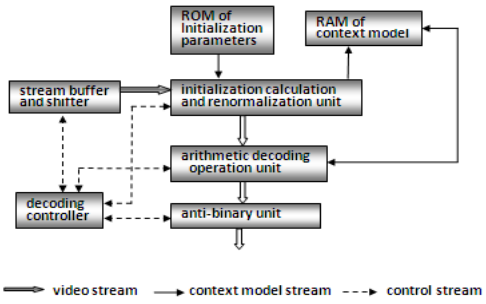

Fig.7 The hardware structure of CABAC decoding module

The CABAC decoding module consists of ROM of initialization parameters, RAM of context model, the initialization calculation and renormalization unit, arithmetic decoding operation module, anti-binary unit and CABAC decoding controller. Stream buffer and shifter is shared with the CA-2D-VLC decoding module and CAVLC decoding module, which is not redesigned.

In the process of CABAC decoding, the context model will be updated after the completion of decoding each bit. For the realization of decoding one bit per clock cycle, this article hopes that the context model of next bit to be decoded is read while writing a new context model to RAM. Therefore, different addresses are needed to be read and written in the same clock cycle. Based on the reason above, dual-port RAM is used, which meets the needs of both reading and writing RAM.

Combinational logic circuit is used in renormalization module in order to avoid occupying clock cycles. Context modeling, arithmetic decoding operation and anti-binary module can be compressed into pipelining mode, so that the three stages can run in parallel and a bit of syntax elements can be decoded each clock cycle, which greatly improves the decoding speed.

IV. Result And Verification

In this document, we used Verilog HDL to realize RTL design of each functional module in entropy decoding module in dual-mode video decoder chip, and did synthesis and simulation on Altera's FPGA. The results of synthesis indicated that the logic resource of entropy decoding module we designed was small. Report of synthesis is shown in Fig.8. After achieving synthesis, simulation was

done respectively. The results of simulation of CA-2D-VLC, CAVLC and CABAC decoding module are shown in Fig.9, Fig.11 and Fig.12.

Flow Status	Successful - Wed Jul 17 08:51:45 2013
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	arsvld
Top-level Entity Name	dual_mode_entropy_decoding
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	No
Total logic elements	2,850 / 33,216 (9 %)
Total combinational functions	2,639 / 33,216 (8 %)
Dedicated logic registers	626 / 33,216 (2 %)
Total registers	626
Total pins	373 / 475 (79 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Fig.8. Report of synthesis

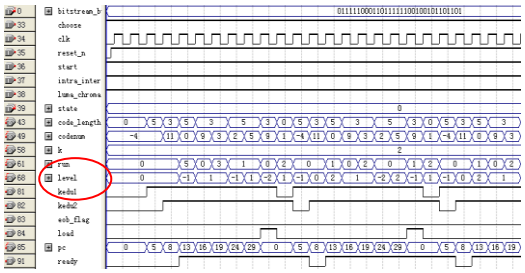

Fig.9 Simulation of CA-2D-VLC module

We then added the entropy decoding module designed as a peripheral component to the SOPC. The structure designed in SOPC is shown in Fig.10. Then, generate the hardware of the design in SOPC. The hardware platform portion of the work is shown in Fig.13.

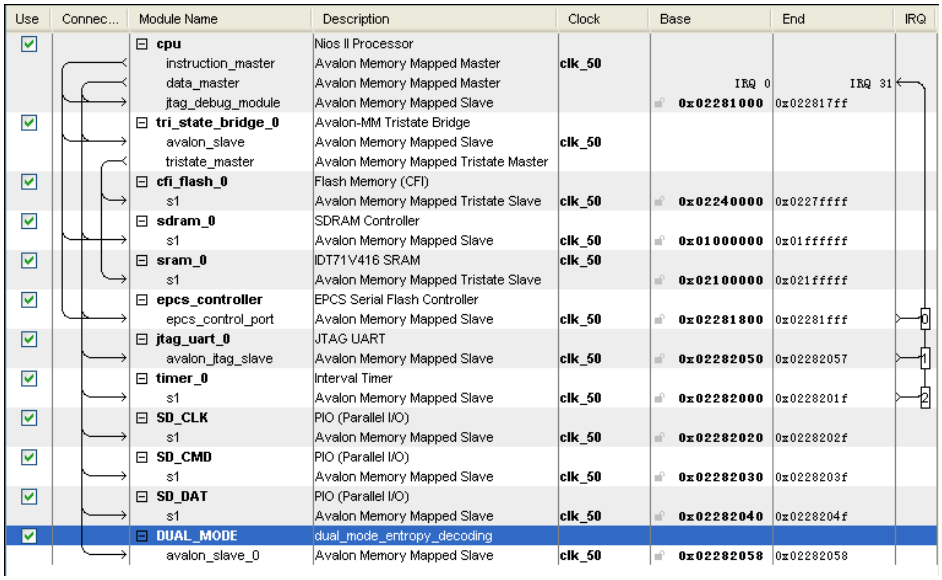

Fig .10 SOPC Builder interface

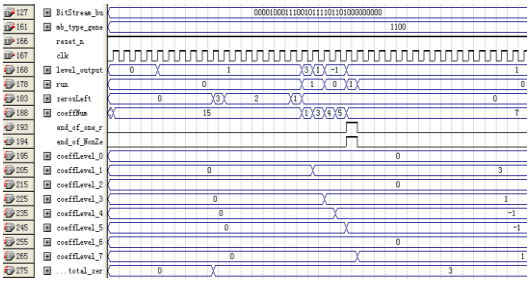

Fig.11 Simulation of CAVLC module

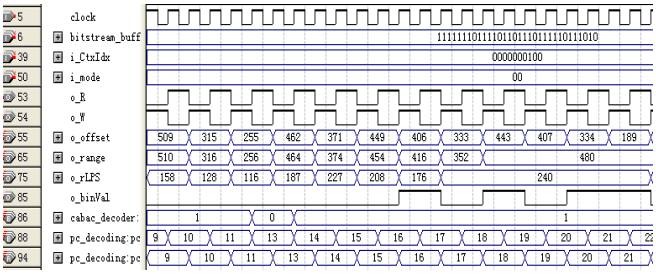

Fig.12 Simulation of CAVLC module

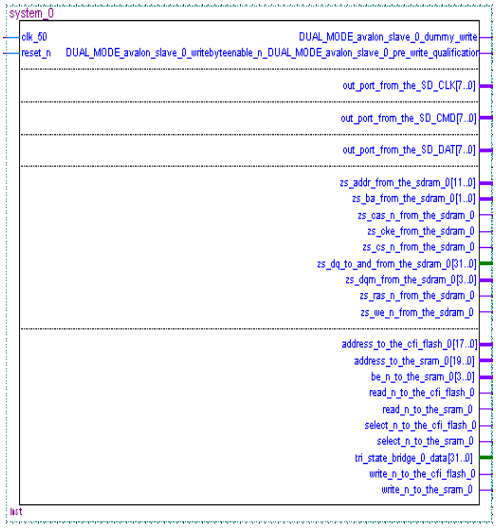

Fig.13 Hardware platform portion generated in Quartus II

Combination of hardware and software was applied to complete the video encoding and decoding. Finally, after assigning pins and generating .sof file, the overall structure of Fig.13 was downloaded to the Altera Cyclone II EP2C35F672C6, and the stream foreman.yuv was input to this structure. The system implemented based on the EP2C35F672C6 of DE-2 development board is in Fig.14. Compared with the output results of reference software RM52j of AVS and JM9.4 of H.264, the result of this overall structure shows that the entropy decoding module in this document accomplishes decoding the video stream. The results are shown in Fig.15 and Fig.16.

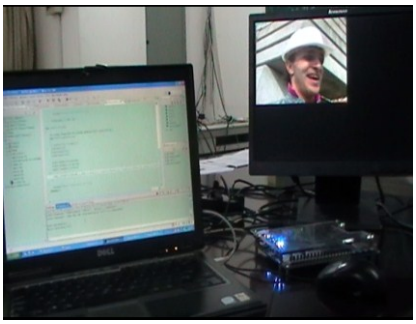

Fig.14 The system implemented based on the EP2C35F672C6 of DE-2 development board

Fig.15 The output results of the reference software

Fig.16 The output results of the overall structure designed in this document

V. CONCLUSIONS

In this document, we designed the entropy decoding module in dual-mode video decoder chip using FPGA, and then embed it into SOPC. Use hardware and software to accomplish the decoding of the video stream. The entropy decoding module designed in hardware can achieve decoding of video stream of 4CIF and 720p HD. It can be used as an accelerator in decoding, which can greatly improve the decoding speed.

REFERENCES

[1] Bin Sheng, Wen Gao, Don Xie and Di Wu. An Efficient VLSI Architecture of VLD for AVS HDTV Decoder[J].IEEE Transactions on Consumer Electronics, May.2006:696-701.

[2] Liu Wei, Yong-en Chen. VLD Design for AVS Video Decoder[C].Second International Workshop on Knowledge Discovery and Data Mining.Jan.,2009:648-655.

[3] GB/T20090.2-2006.Information technology Advanced decoding of audio and video Part2: video.

[4] Bi Houjie, Wang Jian. A new generation of video compression coding standard:H.264/AVC

[5] Tingan Lin, Shengzen Wang, TSUMING Liu. An H.264/AVCdecoder with 4*4 block level pipeline[C]. IEEE Int Symp Circuit Syst, 2005: 1810-1813.

[6] Iain E. G. Richardson. H.264/MPEG-4 Part 10 White Paper: Context Adaptive Binary Arithmetic Coding. 2002.

[7] Ke Zhang, XiaoYang Wu, Lu Yu. An Area-efficient VLSI Implementation of CA-2D-VLC Decoder for AVS [J]. ISCAS 2007, May 2007.

[8] Wu Di, Gao Wen, Hu Mingzeng, Ji Zhenzhou. A VLSI architecture design of CAVLC decoder. Proc. IEEE int. Conf. ASIC. Oct. 2003:692-695.