Hardware Architecture for AVS Entropy Encoder

Long Xu, Lei Deng, Xiangyang Ji, Xiaoming Peng and Wen Gao, *senior Member*, IEEE

Abstract —*In AVS-P2 video compression standard, similar to MPEG-2, entropy coding firstly assembles two dimensional coefficients of each block into a sequence of (Run, Level) combinations serially. As we know, such the serial run-length method is usually undesirable for hardware accelerator and thus, this paper proposes an efficient parallel algorithm to Run-Length Coding, which can determine the (Run, Level) combinations for one row of coefficients from a block in one clock cycle. In addition, Level-based multiple VLC tables switch mechanism (Context-based VLC) is further introduced in AVS-P2 entropy coding module to identify the big variation of probability distribution of (Run, Level) combinations. As a result, table selection for coding the current Level necessarily depends on the previously coded coefficients. Thus, we propose a parallel Looking-Up Table method, which can select the tables for one row of coefficients from a block in one clock cycle. On the other hand, at RDO stage, the calculation of rate term only needs to get the number of bits for each coded signal without the knowledge of its concrete value. Consequently, in hardware design, the Looking-Up Table in pre-coding can be mapped into a series of logic operations and thus much hardware memory can be saved. At the actual entropy coding, we only need to replace the logic operation of pre-coding with the actual 2D-VLC tables. Using our proposed hardware accelerator of AVS entropy coder, the results of simulation and synthesis demonstrate that the computing complexity and memory requirements are both reduced*[1].

Index Terms —AVS-P2, Entropy Coding, VLC, Context-based, Hardware, Pipeline.

I. INTRODUCTION

Chinese Audio Video Standard is a new national standard for the coding of video and audio, known as AVS [1]. The second part of AVS (AVS-P2) [2], [3] mainly targets at the high definition and high quality digital broadcasting, digital storage media applications. AVS-P2 also adopts the traditional hybrid coding architecture, and its new features includes variable block sizes (16x16 down to 8x8), triangle motion vector prediction [4], direct and symmetric prediction modes [5], quarter precision interpolation [6], 8x8 intra prediction [7] and 8x8 integer DCT transform [8]. Another new feature of AVS-P2 is on the entropy coder named C2DVLC, where multiple tables are designed and used context-adaptively to better match the probability distribution of (Run, Level) combinations.

For high definition (HD) video coding, it is very difficult to implement a real-time encoder/decoder with pure software due to very high date rate requirement. Thus, many hardware architectures for the computation-intensive modules, such as motion estimation [9]-[12] have been provided. Although VLC module is usually less than motion estimation module in terms of complexity, it is still computation-intensive task for the real-time video encoder since it is hard to be processed in parallel. To accelerate VLC coding, in [13], architectures with barrel shifters for VLC and VLD (run-length decoder) are proposed, wherein each codeword is encoded or decoded in one clock cycle. In [14], a solution for high data rate applications by packing the Run amplitude and codeword together is proposed, which can achieve the input data rate of VLC encoder as high as the sampling rate of video/image data. In [15], an reverse VLC encoder/decoder design for MPEG-4 is provided. In [16], a complete VLC encoding system with header packing to form the final bit stream is proposed. In [17], a programmable VLC/VLD core is proposed to perform VLC and VLD within one core by sharing the VLC table memory and shifters. For low power applications, [18] proposes a fine-grained VLC table partitioning method to reduce the power consumption of Looking-up Table. However, all these architectures are dedicated on the technique of hardware implementation, while our architecture is concentrated on the parallel algorithm design. In our architecture, we employ a couple of register arrays to realize the data input pipelined, and the 8-pixel parallel Run-Length Coding and Looking-up Table are proposed to VLC. Meanwhile, we use only logic operations to realize the Looking-Up Table in pre-coding and thus much hardware memory is saved. Moreover, from our view, all the architectures for the specific application cannot achieve the requirement of our proposed AVS HD encoder. Furthermore, due to the explicit difference between C2DVLC of AVS-P2 and those of MPEG and H.26L video coding standards, the hardware architectures on MPEG/H.26L can't been used on C2DVLC of AVS-P2 directly.

The rest of paper is arranged as follows. Section 2 gives an overview on AVS entropy encoder. Section 3 describes the implemented architecture and pipelined design of AVS entropy encoder in detail. Section 4 gives the simulation and synthesis of our hardware design. Finally, Section 5 concludes this paper.

[1] This work was supported in part by the National Science Foundation of China under Grants No. 60672088 and 60333020, and this work has been done while author was with National Source Coding Center (NSCC).

Long Xu, Xiangyang Ji and Wen Gao are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080 China (e-mail: lxu@jdl.ac.cn, xyji@jdl.ac.cn, wgao@jdl.ac.cn).

Lei Deng and Xiaoming Peng are with the Institute of Digital Media, School of Electronic Engineering and Computer Science, Peking University, Beijing 100871 China (e-mail: ldeng@jdl.ac.cn, xmpeng@jdl.ac.cn).

II. C2DVLC IN AVS-P2

AVS entropy coder employs the adaptive variable length coding (VLC). In the process of entropy encoding, the coefficients of each MB are mapped into (Run, Level) combinations, where Level is the magnitude of nonzero coefficient and Run indicates the number of successive zero coefficients before the nonzero coefficient. In the statistical view, Level shows a decreasing tendency with respect to its magnitude while Run shows an increasing tendency. Further, it can be seen that a Level with small magnitude is more likely to be preceded by a large Run. The strong correlation of Run and Level exists, so Run and Level are jointly coded using a single VLC codeword, which is named Two-Dimension VLC coding (2D-VLC) [19]. In addiction, the probability distribution of (Run, Level) combination varies along the (Run, Level) sequence, so we employ multiple 2D-VLC tables for this probability distribution variation. To realize the context-based adaptive switch of these tables in the coding process, a function of the previously coded Level's magnitude is used as an indicator to identify the big distribution variation. Differently to the traditional VLC of MPEG/H.26L, the coefficients are processed in the order of inverse Zig-Zag scan in AVS-P2, which makes it easy to follow the variation based on Level information.

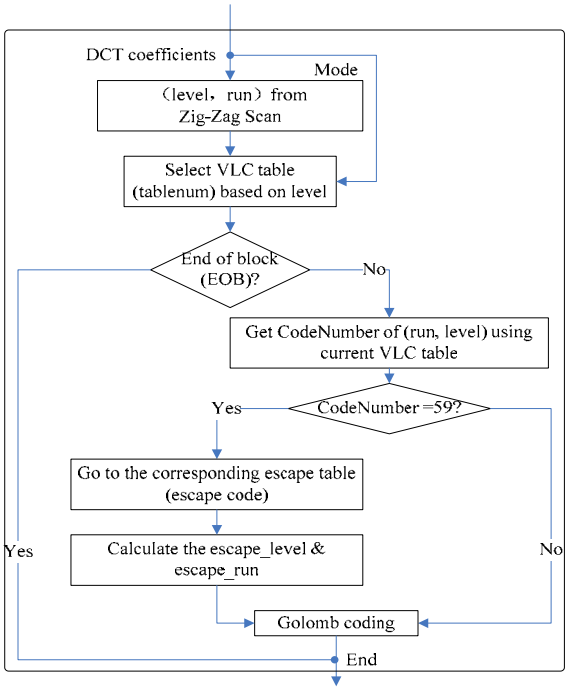


Fig. 1. Flowchart of C2DVLC.

As Fig.1 shown, the residual coefficients after transform (DCT) are firstly mapped into a sequence of (Run, Level) combinations employing Run-Length Coding. Subsequently, a VLC table is selected for each (Run, Level) according to the table switch mechanism. For the first (Run, Level), a default 2D-VLC table is employed. Finally, a unique codeword can be obtained from the selected table, and coded using E-G coder.

III. ARCHITECTURE AND PIPELINE OF AVS ENTROPY ENCODER

In this section, we detail the architecture and pipeline of entropy encoder. Firstly, we propose a parallel algorithm of Run-length Coding, which can process one row of coefficients in one clock cycle. For facilitating the parallel Run-length Coding, we employ a couple of register arrays for Ping-Pong operation of date input, and we believe such processing can realize the parallelization of the traditional VLC in MPEG/H.26L and AVS. Secondly, we propose a parallel Looking-up Table, in which the tables for one row of coefficient can be determined in one clock cycle. While in the previous algorithm, the previously coded Level's magnitude is an indicator of table switch and such serial processing makes the table section can be done for only one coefficient in a clock cycle. However, from our observation, the table switch can be determined by the maximum magnitude of previously coded Levels. Thus, the table selection of one row of coefficients from a block can be performed in one clock cycle by computing multiple maximum values for a row of coefficients. Thirdly, due to the number of bits is only needed in the pre-coding of RDO stage, while the actual codeword of each symbol is not needed, so we simplify the Looking-up Table by logic operations according to the characteristic of E-G coder in the hardware design, which can save about 1k additional memory. If the actual entropy coding is involved, we only need to substitute the logic operation with the VLC tables registered in ROM generally.

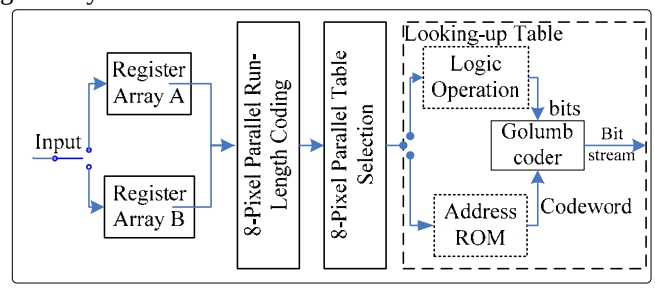


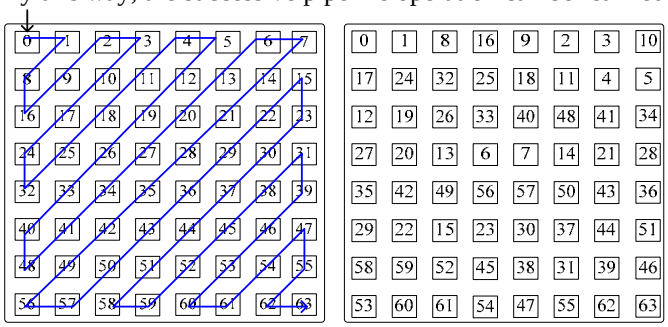
Fig. 2. TOP Architecture of AVS-HD Encoder

In Fig.2, the overview of our proposed hardware architecture of C2DVLC is illuminated. Firstly, two register arrays (A and B) are alternately employed for the data input of a block. Subsequently, the 8-pixel parallel Run-Length Coding is performed as soon as each row of coefficients of a block (named input row in the following content) arrives and thus, 8 (Run, Level) combinations can be obtained. Next, employing 8-pixel parallel Table-Selection, we can determine the tables for the 8 (Run, Level) combinations in one clock cycle. Finally, the Looking-up Table is performed for each (Run, Level) combination by accessing the corresponding ROM of VLC table registered. Especially for Looking-up Table, a simplified architecture with only logic operations for pre-coding is provided which can save the hardware memory and accelerate the hardware processing furthermore.

A. 8-Pixel Parallel Run- Length Coding

In the traditional VLC coding, the (Run, Level) of each nonzero coefficient is calculated one by one in Zig-Zag scanning order. Such serial processing is inefficient in hardware design, so we firstly take the parallel algorithm of Run-Length Coding into consideration in this paper.

Facilitating the parallel processing of VLC, a couple of register arrays are provided to fetch the input blocks. Each register array has 8 rows and 8 column registers for a 8x8 block. In the processing, the coefficients of a block are firstly reordered at the Zig-Zag Scan shown in Fig.3 (a), and registered in one of register arrays at the coding order shown in Fig.3 (b). Then, Run-Length Coding starts, at the same time, another 8x8 block comes into the other register array. By this way, the successive pipeline operation can be realized.



(a) Order of Zig-Zag Scan (b) Order of coding process

Fig. 3. Two register arrays for Ping-Pong operation of data input

As Fig.4 shown, 8 coefficients are processed in parallel and 8 (Run, Level) combinations can be obtained in one pass. Meanwhile, the EOB is set for the first non-zero coefficient. Thus, the process of Run-Length Coding can be detailed as the following steps.

1) Level is the magnitude of input coefficient, which can be obtained immediately from the input coefficients {*a0, a1, ⋯, a7*} as Fig.4 shown.
2) Run is the number of zero coefficients before the current coefficient. As Fig.3 shown, we have reordered the coefficients as Zig-Zag scanning order, and stored in one of register arrays as the coding order.
3) We define *base0* to be the Run of the first coefficient (*a0*) of each input row. Thus, the Run of *a0* will equal *base0* if *a0* is nonzero. When *base0* of each input row is obtained, the Runs of other coefficients in the current row can be obtained immediately without the knowledge of other rows. Meanwhile, we define *base1* to be the number of zero coefficients preceded by the last nonzero coefficient of each row. For example, if the first row of current 8x8 block is "0,5,6,3,0,0,4,0,0", then *base1* equals 2, and *base0* equals 1. Besides, we employ a counter reset for each 8 clocks (i.e., counter=0-7) to check whether the coefficients are from a same 8x8 block.
4) *base0* is determined as: When counter starts (counter=0), *base0* equals 0. When counter is nonzero, there are two cases in our processing. One is "*base0=base0+base1*" if "*base1=8*" occurs, the other is "*base0=base1*" if "*base1* ≠*8*" happens.

The principle of EOB:

1) Checking *nonzero*{*i*, *i*∈[0,7]}, for the first *nonzero*{*i*}=1, the corresponding *eob*0{*i*}, *i*∈[0,7] equals 1, otherwise, *eob*0{*i*} is 0.
2) For the first input row with nonzero *eob*0{*i*, *i*∈[0,7]}, the corresponding *eob*1{*i*}, *i* ∈ [0,7] is 1, otherwise, *eob*1{*i*} is 0. Thus, the coefficient with *eob*1{*i*} equaling 1 just is attached EOB information for a 8x8 block.

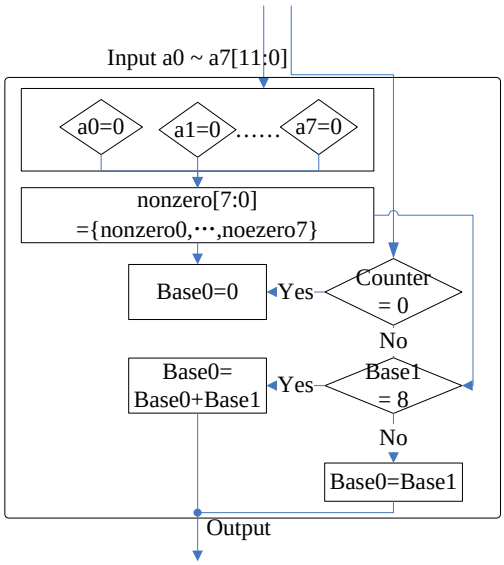


Fig. 4. Flowchart of Run-Length Coding

As Fig.5 shown, the pipeline of Run-Length Coding can be divided into 4 levels as follows.

1) Including 8 comparers and a counter, each comparer checks whether a coefficient from the current input row is nonzero (denoted as *nonzero*{*i*} *i*∈[0,7]), and the counter checks whether the coefficients belong to the same 8x8 block. In detail, *nonzero*{*i*} equals 0 if *i*-th coefficient is zero, and the counter is reset for each 8 clocks which just is the processing period of a 8x8 block.
2) This level consists of 2 parallel logics. One is used to find the first nonzero coefficient by checking *nonzero*{*i*, *i*∈[0,7]}, and the other is used to record *base1* of each input row which is used to compute *base0* of next row. We use *eob*0{*i*}(*i*∈[0~7]) being 1 to represent the first *nonzero*{*i*} being 1, Obviously, there is only one "1" in *eob*0{*i*}(*i* ∈ [0~7]) for each input row. Meanwhile, *base1* as the number of zeros preceded by the last nonzero coefficient of each input row is recorded for computing *base0* of next input row.
3) This level consists of two multiplexers (MUX). One is used to get the EOB of a block from *eob*0{*i*, *i*∈[0,7]} of each input row, where *eob1*{*i*} equals 1 for the first input row with nonzero *eob0*{*i*}; the other is used to get Run of the first nonzero coefficients of each input row, and set it to *base0*. Both of them employ a counter as the controlling signal.
4) This level consists of 8 comparers, which are used to get the (Run, Level) of each coefficient respectively. As

soon as we get Run of the first nonzero coefficient of a input row, the Runs of other coefficients from the current input row can be obtained with the help of *nonzero*{*i*, *i* ∈ [0,7]} obtained from 1). Finally, the output of this module for a coefficient is formatted as *Couple_data*[19:0] = {*absLevel*[11:0], *run*[5:0], *sign_level*, *eob*1}, where *absLevel* is the magnitude of the considered Level, *sign_level* is the sign of Level, *eob1* represents the EOB information.

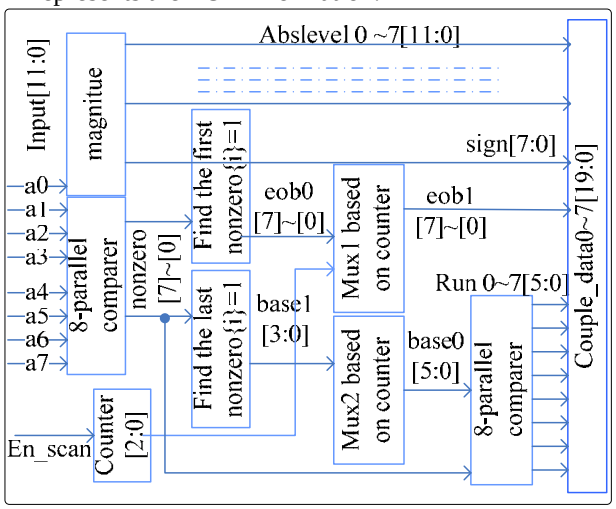


Fig. 5. Pipeline of Run-Length Coding

B. 8-Pixel Parallel Table Selection

In AVS-P2, the table of current coefficient is determined by the table of the neighboring coefficient previously coded, and the maximum magnitude of Levels coded previously. Assumed that *maxAbsLevel* is the magnitude of maximum Level coded, *abslevel* is the magnitude of current Level, the table selection (table switch) is performed as follows. Initially, *maxAbsLevel* is 0 and VLC0 is employed for the first nonzero coefficient. Then, if *absLevel* of the current coefficient is larger than *maxAbsLevel*, a new table will be employed; otherwise, the previous one is still used. When a new table is selected, *absMaxLevel* is set to *absLevlel*. However, such serial processing is not efficient for the hardware design. From our analysis, the maximum magnitude of Levels coded previously can absolutely determine the table of current coefficient. Thus, we can propose the following 8-pixel Parallel Table Selection shown in Fig.6.

1) Computing 8 maximum values for the Levels of a input row (level{*i*, *i*∈[0,7]}):
   *max*0 = **Max**{*level*0},
   *max*1 = **Max**{*level*0,*level*1},
   ……
   *max*6 = **Max**{*level*0, *level*1,…, *level*6},
   *maxlevel*1 = **Max**{*level*0, *level*1,…, *level*7}.
2) Given the maximum Level of a input row coded previously *maxlevel*0, it equals 0 when the coding of current block starts; otherwise, it will be **Max**{*maxlevel*0, *maxlevel*1}.
3) Determining the tables for 8 (Run, Level) combinations at most from an input row in one pass. Given 8 table numbers *tab_value*{*i*, *i*∈[0,7]}, the table of *level*0 is determined by *tab_value*0 = *maxlevel*0; whereas the table of *level*{*i*} (*i* ∈ [1,7]) is determined by *tab_value*{*i*}= **Max**{*maxlevel0*, *max*{*i*-1}} (*i*∈[1,7]).
4) Selecting corresponding VLC table according to the AVS Standard. Taken Intra-coded block for example, there are 7 tables (VLC{*i*}_Intra, *i*∈[0,6]) for it, and table of each coefficient is selected according to its *tab_value* as follows.

```
if( tab_value{i}>10)
        VLC6_Intra;
else if( table_value{i}>7)
        VLC5_intra;
else if( table_value{i}>4)
        VLC4_Intra;
else if( table_value{i}>2)
        VLC3_Intra;
else if (tab_value{i}>1)
        VLC2_Intra;
else if (tab_value{i}>0)
        VLC1_Intra;
else
        VLC0_Intra
```

For example, the coefficients "0, 5, 6, 3, 0, 0, 4, 0, 0; 0, 8, 9, 0, 0, 0, 0, 0;…" is for a intra-coded block. Then *maxlevel*0 equals 0 and *maxlevel*1 equals 6 after the coefficients of the first row are processed. From the second row, we can get

"*maxlevel*0=**Max** {*maxlevel*0, *maxlevel*1}
=Max {0, 6} =6,
*max*0 = 0,
*max*1= **Max**{0,8}=8,
*max*2= **Max**{0,8,9}=9,
*max*3= *max*4 = *max*5= *max*6=9".

Furthermore, *tab_value* for each coefficient of second row can be obtained as

"*tab_value0=maxlevel0*=6,
tab_value1=**Max**{*max0,maxlevel0*}=6,
tab_value2=**Max**{*max1,maxlevel0*}=8,
tab_value3= tab_value4= tab_value5= tab_value6= tab_value7 =9".

Then, we can determine VLC4_Intra and VLC5_Intra for the nonzero coefficients 8 and 9 respectively as step 4).

The pipeline design of table selection shown in Fig.7 is implemented with 4 levels as follows.

1) This level consists of 56 comparers and a counter. 56 comparers are used to getting *max*{0-6} and *maxlevel*1 for each input row, and the counter is used to distinguish the different blocks. In a clock cycle, 8 coefficients can be processed in parallel.
2) Computing *maxlevel*0: *maxlevel*0 is 0 if counter equals 0; otherwise, *maxlevel*0 = {*maxlevel*0, *maxlevel*1}.
3) Including 7 comparers. *Max*{0-6} defined above are compared with *maxlevel*0 respectively to get *tab_value*{1-7}. *tab_value*0 is determined by *maxlevel*0 individually, i.e., *table_value* = *maxlevel*0.
4) Consisting of 8 parallel MUX for determining *Tab_num*.

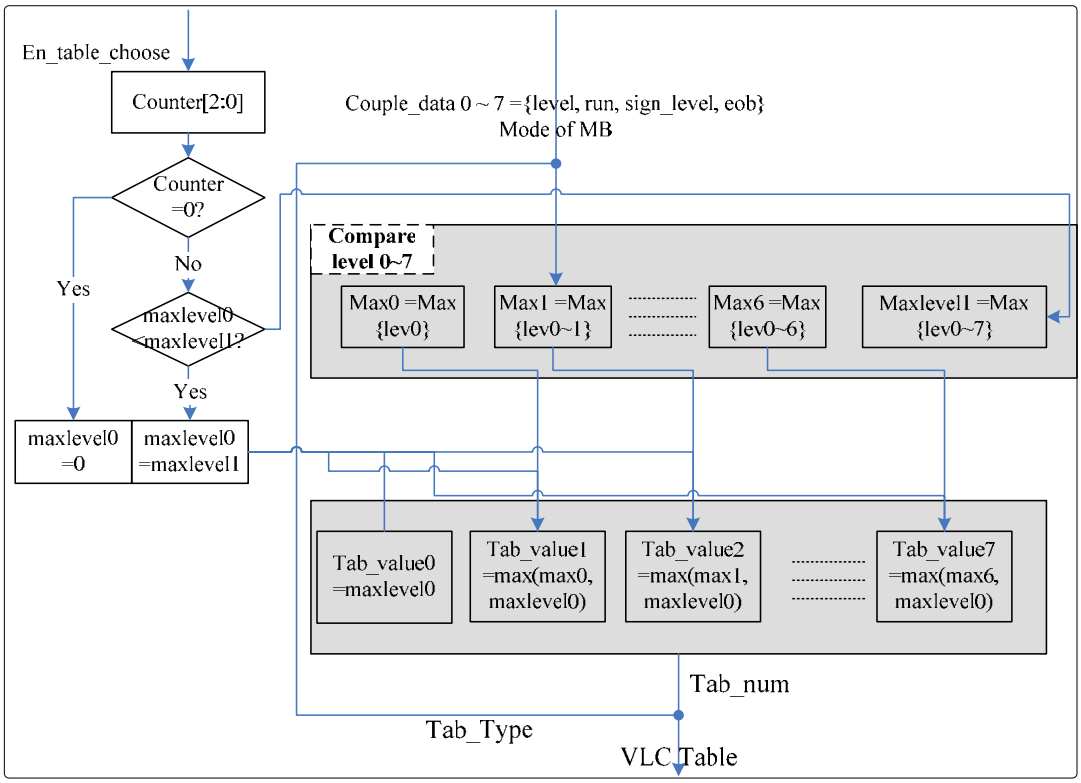


Fig. 6. Flowchart of 8-Pixel Parallel Table Selection

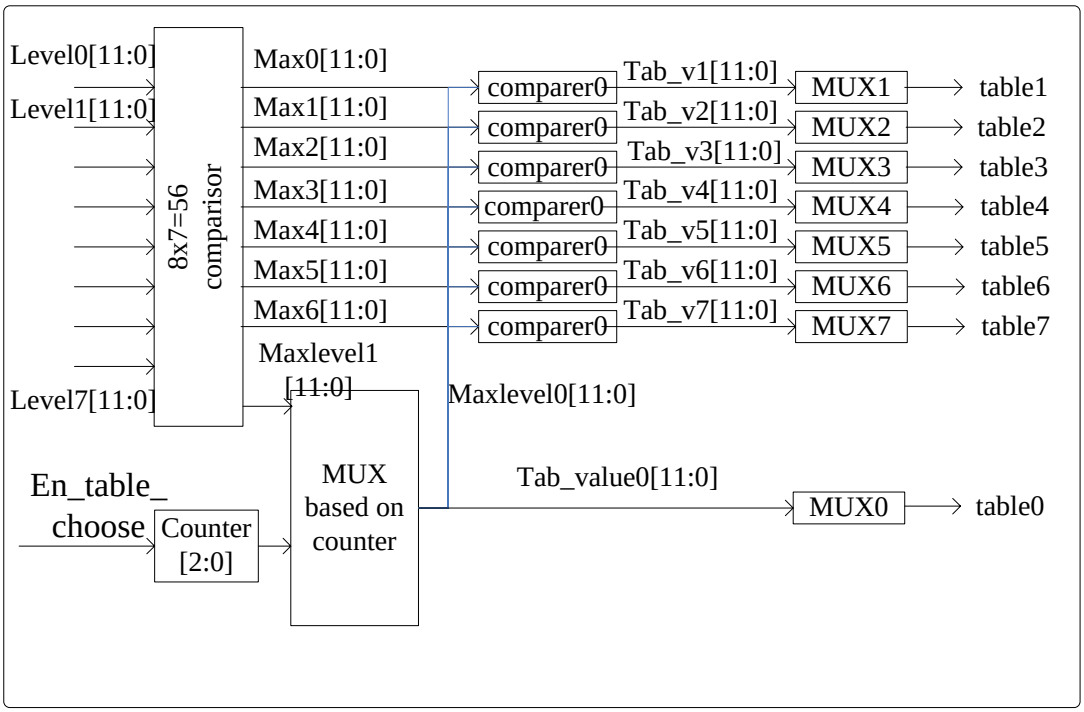


Fig. 7. Pipeline of 8-Pixel Parallel Table Selection

C. Looking-up Table with Logic Operation

For each (Run, Level), we can get its *CodeNumber* by looking up corresponding VLC table. In hardware, we usually register the VLC tables in ROM, and the looking-up table just is to address the corresponding ROM. However in pre-coding, only the number of bits of each symbol is needed for calculating rate term of R-D function. Therefore, we proposed a simplified entropy coder which is realized only with logic operations in hardware design. Based on the characteristic of E-G coder, we can get the width of each codeword and thus, each VLC table can be converted into the corresponding bit width (bandwidth) table. For example, Considered VLC2_Intra of AVS tabulated in Table.1, its bandwidth table tabulated in Table.2 can be deduced.

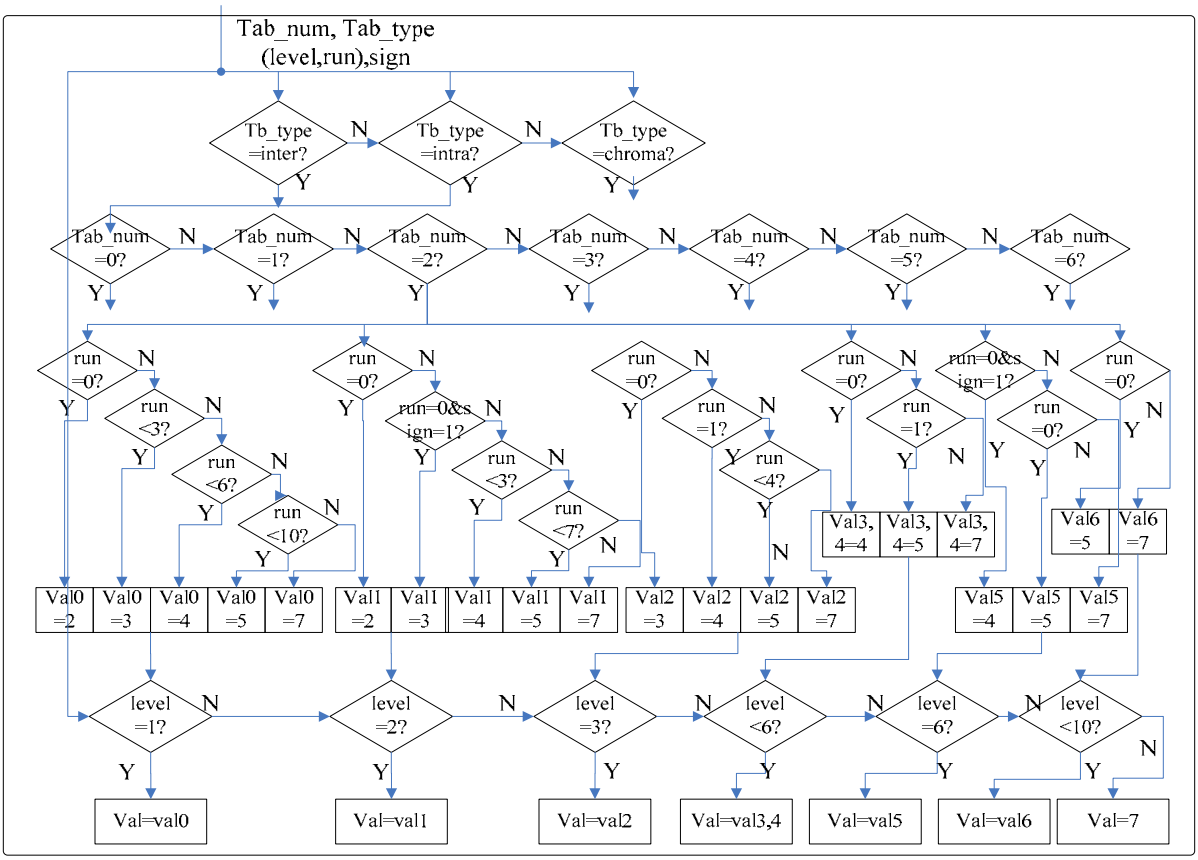


Fig. 8. Flowchart of Looking-up Table

TABLE I
VLC2_INTRA

Run	EOB	absLevel > 0									RefAbsLevel
		1	2	3	4	5	6	7	8	9	
	8	-	-	-	-	-	-	-	-	-	
0	-	0	2	6	13	17	27	35	45	55	10
1	-	4	11	21	33	49	-	-	-	-	6
2	-	9	23	37	-	-	-	-	-	-	4
3	-	15	29	51	-	-	-	-	-	-	4
4	-	19	39	-	-	-	-	-	-	-	3
5	-	25	43	-	-	-	-	-	-	-	3
6	-	31	53	-	-	-	-	-	-	-	3
7	-	41	-	-	-	-	-	-	-	-	2
8	-	47	-	-	-	-	-	-	-	-	2
9	-	57	-	-	-	-	-	-	-	-	2

TABLE II
THE BANDWIDTH TABLE OF VLC2_INTRA

Run	EOB	absLevel > 0									RefAbsLevel
		1	2	3	4	5	6	7	8	9	
	3	-	-	-	-	-	-	-	-	-	
0	-	2	2	3	4	4	**4, 5**	5	5	5	10
1	-	3	**3, 4**	4	5	5	7	7	7	7	6
2	-	3	4	5	7	7	7	7	7	7	4
3	-	4	5	5	7	7	7	7	7	7	4
4	-	4	5	7	7	7	7	7	7	7	3
5	-	4	5	7	7	7	7	7	7	7	3
6	-	5	5	7	7	7	7	7	7	7	3
7	-	5	7	7	7	7	7	7	7	7	2
8	-	5	7	7	7	7	7	7	7	7	2
9	-	5	7	7	7	7	7	7	7	7	2

According to the standard, VLC_intra2 uses 2-order Golomb coder, thus, the *codeNumber* in Table.1 can be mapped into the bandwidth of the *codeNumber* said in Table.2. And, escape event is represented by 7 in Table2. There are two special cases in Table.1: (run=1, level=2) and (run=0, level=6), which are mapped into two optional values in Table.2. It's due to the processing of "*CodeNumber =CodeNumber* +1" performed if the Level is negative. Fig.8 is the flowchart of transition of VLC tables and bandwidth tables.

Based on the analysis above, we can give the hardware

design with 3 levels as Fig.9 shown

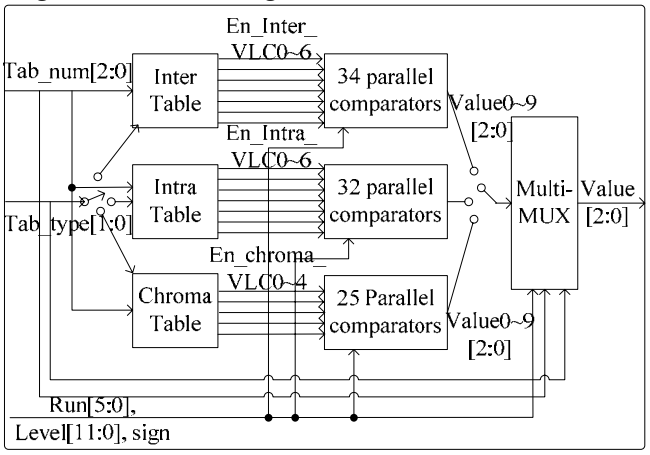


Fig. 9. Pipeline of Looking-up Table

1) Consisting of a switch of table type followed by another switch of table number. With table type and table number, we can address a unique table, i.e., a unique codeword can be selected for each (Run, Level). There is only one "1" in the 19 selecting signals including en_inter_VLC{0-6}, en_intra_VLC{0-6}, en_chroma_ VLC{0-4}.
2) Consisting of 91 comparators, 34 of them are for Inter-coded block, 32 of them are for Intra-coded block and the others are for chroma block. It should be pointed that the comparing criteria for looking up table maybe is different due to the different characteristic of each table. Inter_VLC0, Inter_VLC1, Inter_VLC2, Inter_VLC3, Intra_VLC0, Intra_VLC1, Intra_VLC2, Intra_VLC3, chroma_VLC0, chroma_VLC1, chroma_VLC2 use Run as comparing criteria, others use Level as the comparing criteria. And such method can reduce the number of comparison in processing.
3) Consisting of one switch and one Multi-MUX (multi-pass selection). The signal from Multi-MUX alternate with comparison signal from the second level, so that a pass from the second level is selected.

After 3 level pipelines above, we can get the bandwidth of each (Run, Level) combination. And, the escape event occurs when bandwidth equals to 7.

For the integrality of our hardware design, we give the algorithm and pipeline of Golomb Coding in this paper, which are illustrated in Fig.10 and Fig.11 respectively. In AVS-P2, a (Run, Level) combination is mapped into a *CodeNumber* firstly, and then, according to E-G coder, the bit width is calculated as

$$M = floor\log_2(CodeNumber + 2^k), \qquad (1)$$

where k is the level of Golomb, *CodeNumber* is the value to be coded, M represents the number of bits of information bits of Golomb. Thus the number of bits for coding the *CodeNumber* is $M_stream = 2M+1-k$. For escape event, current table is employed to code Run, and Level is coded according to the prescription of standard. This module can be divided into 6-level pipeline as follows.

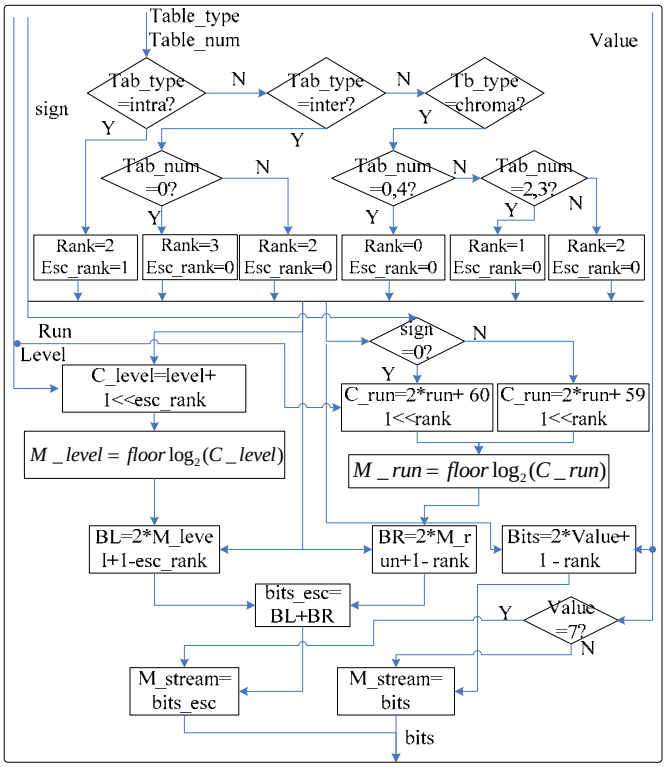


Fig. 10. Flowchart of Golomb Coding

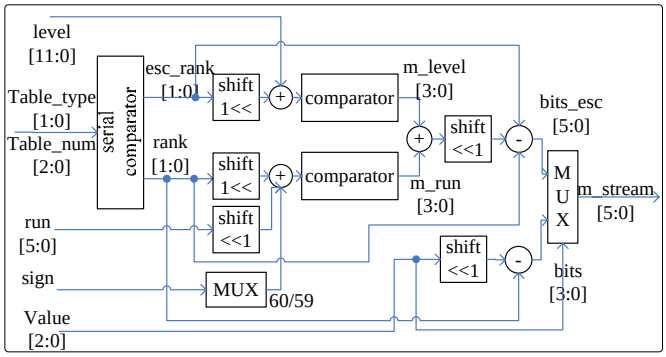


Fig. 11. Pipeline of Golomb Coding

1) Including one comparer, 3 shifters and one MUX. The comparer gets the rank of Golomb coder for coding Run and escape event. Two of shifters are used to bit shift of the rank of Golomb coder for coding Run and Level respectively. The other is used to bit shift of Run. The MUX is to select the constant parameter for Run of escape event from 59 and 60. When the Level of escape event is negative, 59 is used; otherwise, 60 is selected.
2) Two adders, perform the *C_level* = *level*+1<<*esc_rank*, and *c_run*=2**run*+59/60 +1<<*rank*.
3) Two comparers are used to get the information bandwidth of Level and Run respectively.
4) A adder used to perform *m_level*[3:0]+*m_run*[3:0].
5) Two shifters and two adders are used to get the number of bits for coding both escape and non-escape event.
6) A selector determines which is outputted between *bits*[3:0] and *bits_esc*[5:0]. When *value*[2:0]=7, output *bits_esc*[5:0]; else output *bits*[3:0].

IV. SIMULATION AND SYNTHESIS

The proposed 2D-VLC entropy encoder is implemented using Verilog language. It is simulated on the large amount of data using Modesim, and the simulating results are absolutely consistent with the AVS standard. Meanwhile, it is synthesized with ISE 9.1.03.i using Xc4vlx160 of Virtex4, and the synthesis results is illustrated in the following text box, where the frequency of synthesis is 238.72MHz which is enough to satisfy the real-time processing requirement of HD video. According to the AVS-P2 verification model, a C-code model of whole encoder is also developed to generate simulation vectors. The simulation results show that our Verilog code is functionally identical with the C-code model on AVS-P2 standard.

```
Target Device:
  xc4vlx160-11ff1148
Product Version:
  ISE 9.1.03i
Device Utilization Summary:
  Number of BUFGs:           1 out of 32   3%
  Number of External IOBs: 116 out of 768 15%
  Number of LOCed IOBs:      0 out of 116  0%
  Number of Slices:     7464 out of 67584 11%
  Number of SLICEMs:      75 out of 33792 1%
Design statistics:
  Minimum period:       4.189ns
  (Maximum frequency: 238.720MHz)
```

V. CONCLUSIONS

In this paper, we detail the hardware design of C2DVLC for AVS-HD video encoder. The previous hardware accelerators of VLC based on serial algorithm are somewhat low efficient, and unable to fulfill our requirements. Our provided architecture is derived from the parallel realization of C2DVLC. Meanwhile, the optional architecture of C2DVLC in pre-coding is realized with only logic operations. Thus, the higher speed of hardware implementation and less memory requirement can be both expected. Moreover, our proposed architecture can be easily migrated to other VLC hardware design of MPEG/H.264.

ACKNOWLEDGMENT

This study was supported by the National Source Coding Center (NSCC). Tremendous thanks and appreciations go to the entire FPGA group of NSCC and the video coding group of JDL.

REFERENCES

[1] AVS working group official website. http://www.avs.org.cn.
[2] Information technology Advanced coding of audio and video Part2: Video. AVS-P2 Standard draft, Mar 2005.
[3] Audio Video Coding Standard Workgroup of China (AVS) Video Coding Standard FCD1.0, Nov, 2003.
[4] Junhao Zheng, Di Wu, Lei Deng, Don Xie, and Wen Gao. A Motion Vector Predictor Architecture for AVS and MPEG-2 HDTV Decoder. 7th Pacific-Rim Conference on Multimedia (PCM 2006), Hangzhou, China, pp424-431, Nov.2-4, 2006.
[5] Xiangyang Ji, Debin Zhao, Wen Gao, Qingming Huang, Siwei Ma, Yan Lu. New Bi-Prediction Techniques for B Pictures Coding. The 2004 IEEE International Conference on Multimedia and Expo (ICME'2004), Taibei, Taiwan, Jun.27-30,2004.
[6] Ronggang Wang, Chao Huang, Jintao Li,Yanfei Shen Sub-pixel Motion Compensation Interpolation Filter in AVS The 2004 IEEE International Conference on Multimedia and Expo (ICME'2004), Taibei, Taiwan, Jun.93-96,2004.
[7] Peng Zhang, Debin Zhao, Siwei Ma, Yan Lu, Wen Gao. Multiple Modes Intra-Prediction in Intra Coding. The 2004 IEEE International Conference on Multimedia and Expo (ICME'2004), Taibei, Taiwan, Jun.27-30,2004.
[8] Si-Wei Ma and Wen Gao. Low Complexity Integer Transform and Adaptive Quantization Optimization. J. comput. Sci. & Technol. May 2006, Vol.21, No.3, pp.354-359.
[9] Lei Deng, Gao Wen, Hu Mingzeng and Ji Zhenzhou, A High Efficient Architecture for Motion Estimation Based on AVC/AVS Coding Standard. Journal of Computer Research and Development, Nov.2006, Vol.43, No.11, pp.1972-1979.
[10] Li Zhang, Don Xie, Di Wu: Improved FFSBM Algorithm and Its VLSI Architecture for AVS Video Standard. J. Comput. Sci. Technol. 21(3): 378-382 (2006).
[11] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matching algorithm," IEEE Trans. Circuits Syst., vol. 36, no. 2, pp. 1317-1325, Oct. 1989.
[12] Y. S. Jehng, L. G. Chen, and T. D. Chiueh, "An efficient and simple VLSI tree architecture for motion estimation algorithms," IEEE Trans. Signal Process., vol. 41, no. 2, pp. 889-900, Feb. 1993.
[13] S. M. Lei and M. T. Sun, "An entropy coding system for digital HDTV applications," IEEE Trans. Circuits Syst. Video Technol., vol. 1, no. 1, pp. 147-155, Mar. 1991.
[14] H. C. Chang, L. G. Chen, Y. C. Chang, and S. C. Huang, "A VLSI architecture design of VLC encoder for high data rate video/image coding," in Proc. IEEE Int. Symp. Circuits and Systems, vol. 4, 1999, pp. 398-401.
[15] M. Novell and S. Molloy, "VLSI implementation of a reversible variable length encoder/decoder," in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, vol. 4, 1999, pp. 1969-1972.
[16] J. Y. Yang, Y. Lee, H. Lee, and J. Kim, "A variable length coding ASIC chip for HDTV video encoders," IEEE Trans. Consum. Electron., vol. 43, no. 3, pp. 633-638, Aug. 1997.
[17] Y. Fukuzawa, K. Hasegawa, H. Hanaki, E. Iwata, and T. Yamazaki, "A programmable VLC core architecture for video compression DSP," in Proc. IEEE Workshop Signal Processing Systems, 1997, pp. 469-478.
[18] S. H. Cho, T. Xanthopoulos, and A. P. Chandrakasan, "A low power variable length decoder for MPEG-2 based on nonuniform fine-grain table partitioning," IEEE Trans. Very Large Scale (VLSI) Syst., vol. 7, no. 2, pp. 249-257, Jun. 1999.
[19] Qiang Wang, De-Bin Zhao and Wen Gao. Context-Based 2D-VLC Entropy Coder in AVS Video Coding Standard. J.Comput. Sci. & Technol. May 2006, Vol.21, No.3, pp.315-322.

Long Xu received his MSc degree in Applied Mathematics from Xidian University, China, in 2002. And now, he is working for his PhD degree in Computer Science at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include image/video coding, wavelet-based image/video coding and computer vision.

Lei Deng received his MSc degree in Computer Science and Engineering from Harbin Institute of Technology, China, in 2002, and his PhD degree in Computer Architecture and Video Signal Processing from Harbin Institute of Technology. And now, he is working for a Post PhD at the Institute of Digital Media, School of Electronic Engineering and Computer Science, Peking

University. His research interests include computer architecture, digital signal processing and video compression.

Xiangyang Ji Received the MSc degree in Harbin Institute of Technology, China, 2001. He is a PhD candidate at the Institute of Computing technology, Chinese Academy of Science. His research interests include video coding and video transmission.
.

Xiaoming Peng is a MSc candidate at the Institute of Digital Media, School of Electronic Engineering and Computer Science, Peking University. His research interests lie in the video coding, VLSI design.

Wen Gao (M'92-SM'05) received his BSc and MSc degrees in computer science from Harbin University of Science and Technology and Harbin Institute of Technology, China, in 1982 and 1985 respectively, and PhD in electronics engineering from the University of Tokyo, Japan, in 1991. He was with the Harbin Institute of Technology from 1985, served as lecturer, professor, and head of department of computer science until 1995. He was with Institute of Computing Technology, Chinese Academy of Sciences, from 1996 to 2005. During his professor career in Chinese cademy of Sciences, he was also appointed as the director of Institute of Computing Technology, executive vice president of Graduate School, as well as the vice president of University of Science and Technology of China. He is currently a professor at the School of Electronics Engineering and Computer Science, Peking University, China. He is the editor-in-chief of Journal of Computer (in Chinese), associate editor of IEEE Trans. on Circuit System forVideo Technology, and editor of Journal of Visual Communication and Image Representation. He published four books and over 300 technical articles in refereed journals and proceedings in the areas of multimedia, video compression, face recognition, sign language recognition and synthesis, image retrieval, multimodal interface, and bioinformatics. He earned Chinese National Award for Science and Technology Achievement in 2000, 2002, 2003 and 2005 respectively.