International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.7, No.3 (2014), pp.257-268
http://dx.doi.org/10.14257/ijsip.2014.7.3.21

A Quick Tables Look-up Optimization Algorithm for CAVLC Decoding

Jianhua Wang[1], Lianglun Cheng[1], Jun Liu[1] and Tao Wang[1]

[1]*Faculty of Automation, Guangdong University of Technology, Guangzhou, China*
123chihua@163.com, llcheng@gdut.edu.cn, liujun7700@163.com,
466586789@qq.com

Abstract

In this paper, a quick table look-up optimization algorithm is presented to solve the problems of long table look-up time for CAVLC decoding in H.264/AVC. The achievement of the new algorithm rests that we make full use of the hash table query and index technology to improve the table look-up speed for CAVLC decoding. The basic idea of the new algorithm is that we take the number of zero in code prefix calculated from input bit-stream as index key of the first level, the value of codeword suffix as index key of the second level, then through index key of the one and second index key above, we can quickly get the decoded codeword located in the third level in a hash table built, which can reduce a lot of table look-up time for CAVLC decoding in H.264/AVC. The simulation results show that our proposed schemes based on hash-index method can reduce about 40% table look up time for CAVLC decoding compared with TLSS method, without degrading video quality.

Keywords: *CAVLC decoding, Table look-up, Hash-index query*

1. Introduction

H.264/AVC is the latest international video coding standard, which has been developed by ITU-T and ISO/IEC [1]. It has greatly improved the compression ratio and video quality and has been widely used in video communication aspects. H.264/AVC has three kinds of levels: Baseline profiles, Main profiles and extended profiles. Baseline profiles are mainly used in a wide range of small size equipments; Main profiles are adopted to improve image quality and compression ratio; extended profiles are applied to networking video streaming transmission.

In the Baseline profiles of H.264/AVC, Context-based Adaptive Variable Length Coding (CAVLC) as an entropy coding tool is used to decode residual blocks. It increases the compression ratio and video quality, at the same time, it also increases power consumption and hardware cost of the decoder due to frequent table look up. During the process of CAVLC decoding, there are five syntax elements to decode, and three in five syntax elements, Coeff_token, Run_before and Total_zeros, need to use variable length tables to decode code, while the rest of them, Level and sign of TrailingOnes (T1s), are decoded by the regular arithmetic operations without using variable length tables look-up. So Tables look-up problem is a very important problem for CAVLC decoding in H.264/AVC. It have been reported that looking-up variable length tables occupy about 96.6% time of the CAVLC decoding in CAVLC decoding and Tables look-up time have became one important performance and power bottleneck in embedded systems especially for multimedia applications [2]. In this paper, tables look up time refers variable length tables look up time.

ISSN: 2005-4254 IJSIP

In order to reduce table look-up time for CAVLC decoding, previous many researchers have proposed various solutions for it. Some general Table Look-up methods, such as Table Look-up by Sequential Search (TLSS), Table Look-up by Binary Search (TLBS), are proposed to improve table look-up speed. Heng, *et al.*, [3] merged all codeword tables into one table and organized the table into sub-tables to save table look-up time, reducing 40% power consumption. Lee, *et al.*, [4] proposed pipelined architecture to save the operation frequency greatly, saving table look-up time. Wang, *et al.*, [5] presented a novel low-cost high-performance CAVLC decoder for H.264/AVC which greatly improved CAVLC decoding speed. Huang, *et al.*, [6] proposed a decoder based on CMOS and FPGA technology which reduces power consumption by 44-48% more than previous low-power CAVLD schemes. Moon, *et al.*, [7] proposed a new VLDs based on integer arithmetic operations for Run_before and Total_zeros to reduce some table look-up time and reduce memory access [7]. Lu, *et al.*, [8] proposed one kind of entropy decode algorithm, which can decrease great amount of time than the original algorithm in the H.264 reference software; Lee, *et al.*, [9] developed new codeword structures, look-up tables and searching methods for the CAVLC syntax elements to reduce the table look up time. Kim, *et al.*, [10] proposed some other integer arithmetic operations to improve greatly table look-up speed. To reduce the time of table matching, a fast method was proposed by using Table Grouping to construct custom CAVLC decoding tables in this paper [11].

In this paper, we propose a quick table look-up optimization algorithm based on hash-index query for CAVLC decoding. Our algorithm uses hash table and index technology to improve the performance of table look up for CAVLC decoding, which can reduce a lot of tables look up time by decreasing codes looking-up and judging operations. The simulation results show that our proposed scheme can reduce 40% table look-up time compared with TLSS method for CAVLC decoding without degrading video quality.

The rest of this paper is organized as follows. In Section 2, the principle and common optimization method of CAVLC decoding is introduced. The proposed decoding method based on hash-index query is presented in Section 3. The simulation result of proposed scheme compared with general methods is presented in Section 4. In Section 5, we give some conclusions.

2. The Principle and Common Optimization Method of CAVLC Decoding

2.1. Principle of CAVLC Decoding

In baseline profile of H.264/AVC, CAVLC, as one entropy decoding method, is adopted to decode the quantized transform coefficients for residual blocks. The CAVLC decoding decodes residual data into five important syntax elements [9]. These five syntax elements are Coeff_token, Sign of TrailingOnes, Level, Total_zeros and Run_before respectively. Figure 1 is the decoding order and the definition of five syntax elements above are described as follows.

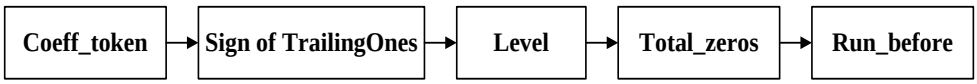


Figure 1. Decoding Order of Five Syntax Elements

- Coeff_token: Both the number of nonzero coefficients (Totalcoeff) and number of coefficients that absolute value is equal to one (TrailingOnes). It is decoded by a Fixed-Length Code Tables and three variable length Tables.

- Sign of TrailingOnes (T1s): Use a single sign bit, which 0 is for positive and 1 is for negative, to represent each T1s in reverse zigzag order. It is decoded by maximum 3-bit reading without any table look-up.
- Level: The values for each nonzero coefficient except for T1s in reverse zigzag order. It is decoded in reverse order and the output of this process is a list of different Level
- Total_zeros: The total number of zero coefficients between the DC and the last nonzero coefficient in zigzag order. It is decoded by looking up some corresponding variable length Tables.
- Run_before: The numbers of zeros preceding each nonzero coefficient in reverse zigzag order. It is decoded by looking up some corresponding variable length tables.

2.2. Common Optimization Method of CAVLC Decoding

Table look-up is a very time-consumption operation in most equipments. As the process of CAVLC decoding involves many table look up operation, it seriously affect the efficiencies of CAVLC decoding. In order to improve efficiency of CAVLC decoding, recently some efficient methods have been adopted to improve table look-up methods. According to analyzing the methods used in at present, they can be summarized into the following classifications.

1) Table Lookup by Sequential Search (TLSS) method: This method is used to look up table by comparing one bit at a time with the decoding table. As it usually needs to research whole code tables to find the desired codeword, this method requires multiple table look-up time.

2) Table Look-up by Binary Search (TLBS) method: This method mainly make full use of advantages of binary tree's high search efficiency and the sub-table's simple storage structure to build a binary tree for each sub-table. This algorithm could greatly improve the search efficiency for code table. Disadvantages of this method lie in its large storage space.

3) Table Look-up by Sub-table (TLST) method: the basic idea of this method is that it mainly changes the original two-dimension table into many small parts. As this method needs to judges key words to determine the node in which block to be searched and to make sequential search or half search in certain pieces, it requires prolonged running time. This method is not suitable for real-time processing.

4) Table Look-up by Code Grouping (TLTG) method: This method gets the number of TC and T1s through calculation based on the number of 0 before 1 in the code stream. As this method cannot get the length of the stream immediately, its defects rests that it needs long time to feedback the length of the code stream.

5) Table Look-up by arithmetic operations (TLAO) method: This method mainly use to arithmetic operations look-up instead of the conventional table look-up method. It can achieve a fast decoding and reduced power consumption by decreasing memory accesses.

3. Proposed Scheme

3.1. Observe Variable Length Tables

As mentioned above, the process of CAVLC decoding needs to decode five syntax elements. Three in five syntax elements, Coeff_token, Run_before and Total_zeros, need to look up variable length Tables. Through analysis the variable length Tables, we can find that Coeff_token has three 2D-variable length tables: 0<=NC<2, 2<=NC<4, 4<=NC<8; while the Run_before and Total_zeros VLDs have one 1D-variable length Tables. Table 1 is the part

codewords of 2D-variable length Tables for Coeff_token(0≤NC<2). Table 2 is the part codewords of 1D-variable length Tables for Total_zeros (Tc=5). In this paper, we mainly need to optimize the Table look-up algorithm for those variable length Tables above.

Table 1. Part of 2D-Variable Length Table for Coeff_token (0≤NC<2)

Code	Codeword	[T1,Tc]
1	0x00	[0,0]
01	0x21	[1,1]
001	0x42	[2,2]
000100	0x22	[1,2]
000101	0x01	[0,1]
00011	0x63	[3,3]
000011	0x64	[3,4]
0000100	0x65	[3,5]
0000101	0x43	[2,3]
…	……	……
000000000000001	0x2d	[1,13]

Table 2. Part of 1D-variable Length Table for Total_zeros (Tc=5)

Code	Codeword
111	3
110	4
101	5
100	6
0101	0
0100	1
011	7
0011	2
0010	8
0001	10
00001	9
00000	11

In Table 1, the code represents the input bit-stream of the Coeff_token syntax element (0≤NC<2). The 8-bit codeword represents the decoded output and the front 3 bits of them are total number of ones (T1) and the other tail 5 bits are the total number of coefficients (Tc). In Table 2, the code stands for the input bit-stream of the syntax elemen of Total_zeros, the codeword stands for single decoded output directly (Tc=5). Note that the other elements of Coeff_token, Run_before and Total_zeros have the same code Table structure as Table 1 and Table 2.

3.2. Find Internal Relationship between Code Length and Numbers of 0 in Code Prefix

By analyzing the codeword structure in Table 1 and Table 2 above, we find that there are some internal relationships between numbers of 0 in code prefix and code length for Coeff_token and Total_zeros, which are shown as in Table 3 and Table 4 respectively.

Table 3. The Relationship Exists between Code Length and Numbers of 0 in Code Prefix Corresponding to Table 1

Numbers of 0 in code prefix	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Code length	1	2	3	5or 6	6 or 7	8	9	10	11	13	14	15	16	16	15

Table 4. The Relationship Exists between Length of Code and Numbers of 0 in Code Prefix Corresponding to Table 2

Numbers of 0 in code prefix	0	1	2	3	4	5
Code length	3	3or 4	4	4	5	5

Table 3 and Table 4 are the internal relationships between the length of code and numbers of 0 in code prefix for Coeff_token and Total_zeros respetively. Through the internal relationship, we can find a new Table look-up method to quickly determine the length of code suffix from input decoded bit-stream, which can save lots of time of looking up and judging code suffix.

3.3. Build Hash-index Table

Based on this principle above, we build hash-index table to express the relationships existed in code and numbers of 0 and length of code prefix and code length. The built hash-index tables for 2D-Coeff_token and the 1D-Total zeros are shown as Table 5 and Table 6 respectively.

Table 5. Part hash-index Table of 2D-Table for Coeff_token (VLCT0, 0≤NC<2)

Code	Hash address	Numbers of 0 in code prefix	Value of code suffix	Index value	Code-word
1	0	0	1	01	0x00
01	1	1	x	1x	0x21
001	2	2	x	2x	0x42
000100	3	3	00	300	0x22
000101			01	301	0x01
00011			1	31	0x63
000011	4	4	1	41	0x64
0000100			00	400	0x65
0000101			01	401	0x43
…	…	……	……	……	……
000000000000001	14	14	x	14x	0x2d

Table 6. Part Hash-index Table for Total_zeros (Tc=5)

Code	Hash address	Numbers of 0 in code prefix	Code suffix	Index value	Codeword
111	0	0	111	0111	3
110			110	0110	4
101			101	0101	5
100			100	0100	6
0101	1	1	01	101	0
0100			00	100	1
011			1	11	7
0011	2	2	1	21	2

0010			0	20	8
0001	3	3	x	3x	10
00001	4	4	x	4x	9
00000	5	5	z	5z	11

In Table 5 and Table 6, the code represents the input bit-stream of the corresponding syntax element. Hash Table addresses are allocated mainly according to numbers of 0 in code prefix of code. Numbers of 0 in code prefix of code could be mapped to hash table addresses by a hash function in hash-index Table. The length of code suffix is determined through the relation existed in code length and numbers of 0 in code prefix, such as Table 3 and Table 4. We takes number of zero in code prefix calculated from input bit-stream as index key of the first level, the value of codeword suffix as index key of the second level, and through the index key of the one and second index key above, we can quickly get the decoded output located in third level in a hash table built.

3.4. Realize Tables Look-up Algorithm based on Hash-index Query

Based on the ideas above, we propose a table look-up algorithm based on hash-index query for CAVLC decoding. The basic idea of new algorithm is that we takes number of zero in code prefix calculated from input bit-stream as index key of the first level, the value of codeword suffix as index key of the second level, and through the determined index value of the one and second above, we can quickly get the decoded output located in third level in a hash Table built, which can save a lot of Tables look-up time. The process of Table look-up algorithm based on hash-index query could be shown as Figure 2.

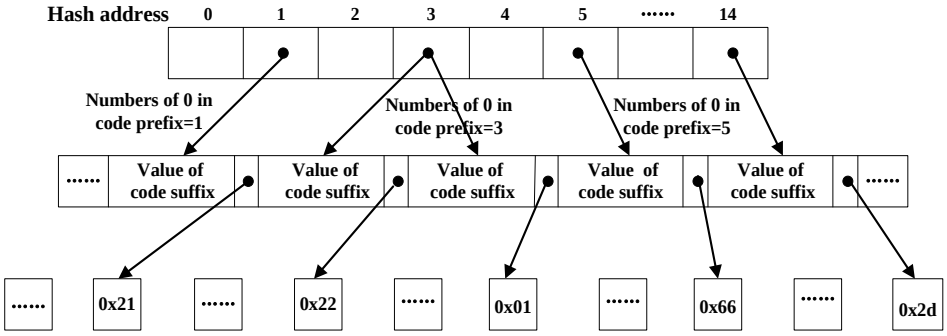


Figure 2. Table Look-Up Algorithm based on Hash-Index Query

The proposed table look-up algorithm based on hash-index query for CAVLC decoding can be summed up some steps as follows.

Step 1: select variable length tables of Coeff_token syntax element through the value of NC.

Step 2: read input decoded bit-stream and calculate numbers of 0 in code prefix as index key of the first level

Step 3: find the length of code suffix in second level by numbers of 0 in code prefix mapped in built hash table.

Step 4: determine the length of code suffix according to length of code suffix or the first value of code suffix. If the length of code suffix found has two possible values, we can get the right length of code suffix after judging the first value of code suffix again.

Step 5: read the value of code suffix as index key of the second level according to the numbers of 0 in code prefix and the length of code suffix above.

Step 6: find decoded codeword in third level through the determined index key of the one and second above.

Table 7. Is pseudo Code of Tables Look-up based on hash-index Algorithm

```
Procedure TableLook up_HashIndex{
    Input: decoding input  bit-stream
    Output: codeword(T1s,TC)
   (1)initialization
   (2)input decoding bit-stream;
   (3)select variable length tables through the value of NC;
   (4)calculate numbers of 0 in code prefix as  index key of the first level ←— ShowBit(x,x,x);
   (5)find hash address ←— Hash (numbers of 0 in code prefix);
   (6)get the length of code suffix ←— Hash (numbers of 0 in code prefix);
   (7)if (the length of code suffix==one possible value){
   (8) get the value of  code Suffix as  index key of the second level ←— the length of code suffix;
   (9) get codeword←—index key of the first second level above;
   (10)  }
   (11)else if (the length of code suffix==two possible value){
   (12)if( the first value of code suffix==1)
   (13)   get the value of  code Suffix as  index key of the second level←— the length of code suffix;
   (14)else
   (15)   get the value of  code Suffix as  index key of the second level←— the length of code suffix;
   (16) get codeword←— index key of the first second level above;
   (17) }
   (18) else{
   (19) codeword =  “get  error codeword”;
}
```

3.5. Demonstrate Method Instance

Take decoding Coeff_token element for an example to illustrate our proposed method based on hash-index method. And supposed that NC value is 0≤NC<2, the bit-stream inputted is 0000100011….Figure 3 is the decoding process of an example with our proposed scheme.

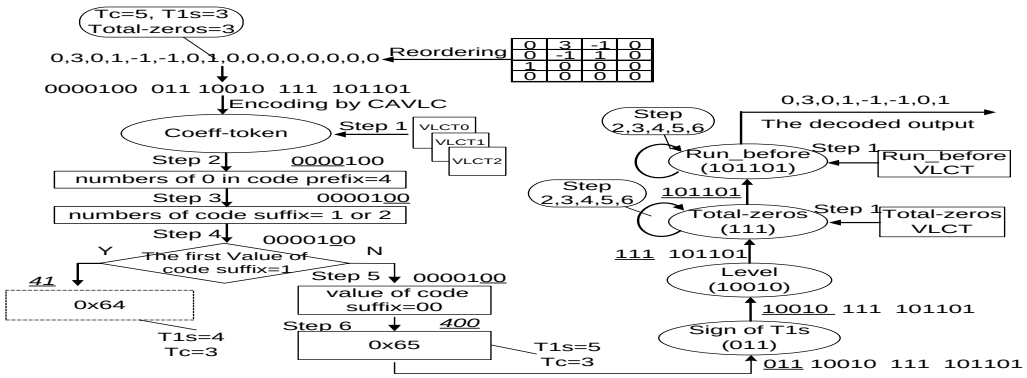


Figure 3. A Decoding Example with our Proposed Scheme

From Figure 3, we can clearly see the CAVLC decoding process of an example with our proposed scheme. The decoding process for our proposed scheme is shown as follows:

Step 1, variable length table 0 of Coeff_token is selected as Table entry because of NC value (0≤NC<2);

Step 2, read input decoding bit-stream and take the numbers of 0 in code prefix (4)calculated from input decoded bit-stream as index key of the first level(4);

Step 3, look up hash Table address (4) in Table 4 because of length of code suffix (4), and get the length of code suffix for 1 or 2.

Step 4, as there are two possible values (1 or 2) in length of code suffix, we can get the right length of code suffix (2) after judging the first value of code suffix (0) again.

Step 5, read code suffix (00) from the input decoded bit-stream according to length of code suffix(2) and take it as index key of the second level(00)

Step 6, through index key of the one and second level (400), we can find the decoded codeword (0x64) in table 4, at last, transform 0x64 into T1s=3, Tc=5.

After decoding the Coeff_token, level, sign of T1s, Total_zeros and Run_before syntax elements are decoded in sequence. As to Total_zeros and Run_before, the decoding steps are the same as the syntax elements of Coeff_token. At last, we could get the decoded output

3. Simulation Results and Analysis

In order to verify the effectiveness of our proposed method above, we take some experiments. In this section, our designed experiments mainly include two parts: experimental environment and reduce Table look-up time. In our experiments, we compared our proposed method compared with conventional TLSS, TLBS methods in the performance of Table look-up time.

4.1. Simulation Environment

The simulation environment was conducted on a Intel 2GHz processor, 1GB memory capacity, Intel Windows XP operating system. Table 8 shows some parameters of test sequences, including the name, resolutions, frame rate and frame number of test sequences in our simulation experience. Some common encoding parameters are shown at Table 9.

Table 8. Parameters of Test Sequences

Sequence(SEQ)	resolution	Frame rate	#frames
Forman(FM)	CIF(176×144)	25	30, 90
Crew(C)	CIF(176×144)	25	30, 90
Paris(P)	CIF(176×144)	25	30, 90
Wal(W)	CIF(176×144)	25	30, 90
Football(FB)	QCIF(352×288)	25	30, 90
Harbour(H)	QCIF(352×288)	25	30, 90
Mobile(M)	QCIF(352×288)	25	30, 90
Soccer(S)	QCIF(352×288)	25	30, 90

Table 9. Encoding Parameters

Profile	Baseline
SATD (Hadamard)	On
RDOptimization	1
RDO	On (fast algorithm)
MV search range	±32 pixels
Reference frame	5 frames
QP	24,28, 32
Motion search	Fast search
Intra interval	0
Motion search	Fast search
SymbolMode	0 (CAVLC is used)
QPPrimeYZeroTransform BypassFlag	0 (lossless)
File	Vlc.c
Encoder	JM 16.2 [12]

4.3. Reduce Table Look-up Time

In this subsection, we mainly evaluate the performance of Table look-up time with our proposed algorithm, which was compared with TLSS and TLBS methods in different sequences, different frames and QP value. The experimental results are shown in Table 10 as follows.

Table 10. The Comparison Results in Table Look Up Time for Three Methods

\	QP	24			28			32		
SE-Q	Fra-mes	TLSS	TLBS	Ours	TLSS	TLBS	Ours	TLSS	TLBS	Ours
FM	30fs	91452424	703654562	64022671	67444596	54798689	4868503	46882240	36675651	33249835
	90fs	306339376	239356324	219563835	208683480	167782033	149423914	160228410	128135788	112836908
C	30fs	106440956	72655340	76533631	87563480	69727275	63915094	66303640	53675677	47594250
	90fs	303817596	240763544	217477176	231906120	176903672	151872774	166929040	130367557	118132188
P	30fs	67843250	53654235	48463850	56537290	52066743	50470973	49427280	37890456	34947884
	90fs	179095793	13445360	12485000	154580870	12867546	11120958	130896338	98677102	92201640
W	30fs	127718726	935691723	88219806	112815480	83016734	76294810	83498650	64667543	57197705
	90fs	361732130	27456978	254670549	311956514	245780426	218501055	251010610	19534780	174243486
FB	30fs	121012110	90335461	86228407	10646260	84654087	7859623	74306817	6067833	5375746
	90fs	291077420	239347533	210998130	210894200	178905664	151461901	166631660	70257890	52469854
H	30fs	110447540	88673652	81960188	81513646	65489076	59499007	42399550	3371096	3057059
	90fs	340546073	25808654	239962010	241007480	189056468	168122731	206009120	167441435	144991633
M	30fs	125115200	91547043	86816152	85790423	65140457	58401646	37523680	29436734	26425126
	90fs	390783324	293654484	267386109	28507620	21975458	19900433	125915116	94698965	88743036
S	30fs	48479094	38635442	35457044	33094810	25673322	23996231	19407230	15265582	13728886
	90fs	161562740	135564536	113706161	110695448	84675530	78683115	67301960	53068420	48054257

From Table 10, we could find that our proposed algorithm has superior results than other algorithms and shows about 40% saving time compared to the standard TLSS method. TLBS follows. The main reason for reducing Table look-up time for CAVLC decoding in H.264/AVC is that because our proposed algorithm uses a hash-index query method to look-up Table for CAVLC decoding, which could reduce lots of the operations of looking up and judging code length and code prefix and codeword by a hash-index Table built, therefore saving a lot of Table look-up time.

5. Conclusion

Table look-up is a very important operation for CAVLC decoding in H.264/AVC, which could seriously affect the efficiency of CAVLC decoding. In this paper, a quick a Table look-up optimization scheme based on hash-index query is proposed to solve the problem of long table look-up time for CAVLC decoding in H.264/AVC. Specifically, in our schemes, after finding the existing relationships between code length and numbers of 0 in code prefix, we uses hash Table and index technology to reduce Table look-up time. The simulation results show that our proposed schemes based on hash-index query can reduce about 40% Table look-up time for CAVLC decoding compared with TLSS method, without degrading video quality.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive opinions in improving this paper. The work was supported by the Joint Funds of the National Natural Science Foundation of China (No.U2012A002D01);The Strategic Emerging Industries Special of Guangdong Province (No.2012A09100013-2012BAF11B04–5150).Foundation for Distinguished Young Talents in Higher Education of Guangdong (No.LYM11057); Doctoral Project for Natural Science Foundation of Guangdong Province (No.S2012040006666).

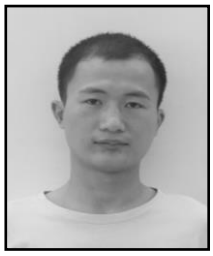
References

[1] Joint Video Specification (ITU-T Rec. H.264|ISO/IEC 14496-10)-Joint Committee Draft Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Doc. JVT-G050r1.doc, **(2002)**.
[2] J. Lee, C. Park and S. Ha, "Memory Access Pattern Analysis and Stream Cache Design for Multimedia Applications", Proc. Asia and South Pacific, Design Automation Conf (DAC), Kitakyushu, Japan, **(2003)** January 21-24.
[3] H.-Y. Lin, Y.-H. Lu, B.-D. Liu and J.-F. Yang, "A highly Efficient VLSI architecture for H.264/AVC CAVLC Decoder, IEEE Transactions on multimedia, vol. 10, no. 1, **(2008)**.
[4] B.-Y. Lee and K.-K. Ryoo, "A Design of High-Performance Pipelined Architecture for H.264/AVC CAVLC Decoder and Low-Power Implementation", IEEE Transactions on Consumer Electronics, vol. 56, no. 4, **(2010)**.
[5] K.-Y. Wang, B.-S. Kim, S.-S. Lee, D.-S. Kim and D.-J. Chung, "A novel low-cost high-throughput CAVLC decoder for H.264/AVC", IEICE Transactions on Information and Systems, E94-D, vol. 4, **(2011)**.
[6] C.-H. Fang and C.-P. Fan, "Very-large-scale integration design of a low-power and cost-effective context-based adaptive variable length coding decoder for H.264/AVC portable applications", IET Image Processing, vol. 6, no. 2, **(2012)**.
[7] Y. H. Moon, G. Y. Kim and J. H. Kim, "An Efficient Decoding of CAVLC in H.264/AVC Video Coding Standard", IEEE Trans. on Consumer Electronics, vol. 51, no. 3, **(2005)**.
[8] D. Lu, G. Liu and Z. Lingli, "An optimization for CAVLC code Table lookup algorithm in H. 264 decoder", 2011 2th International Symposium on Intelligence Information Processing and Trusted Computing, Hubei, China, **(2011)** October, pp. 79-82.
[9] J. Y. Lee, J. J. Lee and S. M. Park, "New Lookup Tables and Searching Algorithms for Fast H.264/AVC CAVLC Decoding", IEEE Trans Circuits and System Video Technology, vol. 20, no. 7, **(2010)**.
[10] Y. H. Kim, Y. J. Yoo and J. Shin, "Memory-Efficient H.264/AVC CAVLC for Fast Decoding", IEEE Trans On Consumer Electronics, vol. 52, no. 3, **(2006)**.
[11] G. Sullivan and G. Bjontegaard, "Recommended simulation common conditions for H.26L coding efficiency experiments on low-resolution progressive-scan source material", ITU-T VCEG, Doc. VCEG-N81, **(2001)**.
[12] K. Suhring, "JM 16.2 software", http://iphome.hhi.de/suehring/tml/.

Authors

JianHua Wang, he was born on February 6, 1982 in Guangdong, China. He received his B.S degree in Electronic Information Science and Technology from Shaoguan University, Guangdong, China, in 2006. Currently he is pursuing Ph.D degree in Control Science and Engineering at Guangdong University of Technology. His research interests include 3G wireless video transmission, cyber-physical systems, IoT and wireless sensor networks.

LiangLun Cheng, he was born on August 22, 1964 in Hubei. He received his M.S and Ph.D degrees from Huazhong University of Science and Technology, Hubei, China in 1992 and Chinese academy of Sciences Jilin, china in 1999 respectively. He is a Prof and doctoral supervisor of Guangdong University of Technology. His research interests include RFID and WSN, IoT and CPS, production equipment and automation of the production process, embedded system, the complex system modeling and its optimization control, software of automation and information, *etc.*,



Jun Liu, he was born on October 11, 1986 in Hubei, China. He received his M.S degree in Control Science and Engineering from Guangdong University of Technology, Guangdong, China, in 2012. Currently he is pursuing Ph.D degree in Control Science and Engineering at Guangdong University of Technology. His research interests include 3G wireless video transmissions, cyber-physical systems and wireless sensor networks.

Tao Wang, he received the PHD Degree in network security from Sun Yat-Sen University, Guangzhou, China, in 2010. His current research interest includes context-aware computing, service composition, protocol optimization in wireless sensor network and cyber-physical system.