International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.6, No.5 (2013), pp.179-190
http://dx.doi.org/10.14257/ijsip.2013.6.5.16

A Quick Tables Look-up Algorithm based on Hash Query for CAVLC Decoding

Jianhua Wang[1], Lianglun Cheng[2] and Jun Liu[3]

Faculty of Automation, Guangdong University of Technology, P. R. China
123chihua@163.com[1], llcheng@gdut.edu.cn[2], liujun7700@163.com[3]

Abstract

Aiming to solve the problem of long table look-up time in table look-up of CAVLC (Context-based Adaptive Variable Length Coding) for H.264/ AVC, a quick table look-up algorithm based on hash query (TLHQ) is proposed to reduce table look-up time for CAVLC decoding in this paper. The basic idea of the new algorithm is that it uses query technology of hash table to rapidly determine code length based on the number of zero in code prefix, As a result, it can save a lot of time of table looking-up and code judging. Experimental results show that our supposed new algorithm could reduce about 20% table look-up time and save 1056 byte storage space for CAVLC decoding in H.264/AVC.

Keywords: *Table look-up; Hash query; CAVLC decoding; Decoding time*

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 been widely adopted in many related video communication aspects and has greatly improved the compression ratio and video quality. Context-based adaptive variable length coding (CAVLC) as a basic entropy coding tool of the H.264/AVC baseline profile, has improved the video compression ratio, at the same time, it has also increased the power consumption and hardware cost of the decoder. Thus how to develop a CAVLC decoding method with a low-power design that also meets the demands of real-time CAVLC decoding has become our major issue at present.

During the process of CAVLC decoding, CAVLC decoding needs to decode five syntax elements. Three in five syntax elements, Coeff_token, Run_before and Total_zeros, need to decod by looking up the variable length tables, while the rest of them, Level and sign of TrailingOnes(T1s), are decoded by the regular arithmetic operations without using the looking-up of variable length tables. Since looking-up variable length tables occupies about 96.6% time of the entire CAVLC decoding and consumes a great amount of CAVLC decoder power during the process of CAVLC decoding [2], It become a very important problem for CAVLC decoding in H.264/AVC baseline profile. How to reduce the table look-up time becomes one of our concerns at present. In this paper, tables look up time refers looking up variable length table time, and table storage space refers the variable length table storage space.

In order to reduce the table look-up time, many optimized decoding methods have been developed for the purpose. In the hardware design level, Heng *et al.*, [3] merged all codeword tables into one table and organized the table into sub-tables to reduce table look-up time. Lee *et al.*, [4] proposed pipelined architecture to save the operation frequency greatly, saving table look-up time. Wang *et al.*, [5] presents a novel low-cost high-performance CAVLC decoder for H.264/AVC which can greatly reduces time of CAVLC decoding. In the software design

ISSN: 2005-4254 IJSIP

level, some general table look-up methods, such as Table Look-up by Sequential Search (TLSS), Table Look-up by Binary Search (TLBS), have been required to decode CAVLC. However, the TLSS needs a great lot of table look-up time due to consequent table look-up for the desired codeword, it cannot meet real-time decoding application requirement. The TLBS can improve table look-up speed, but because of its random memory access, it doesn't behave efficiently in some systems. In Moon's method, a new VLDs based on integer arithmetic operations for Run_before and Total_zeros are proposed, which can greatly reduce table look-up time [6]. Lee *et al.*, [7] developed new codeword structures, look-up tables and searching methods for the CAVLC syntax elements to reduce table look-up time. In Kim's method, some integer arithmetic operations are proposed for other syntax elements, which can reduce greatly improve table look-up speed [8]. Uchihara *et al.*, [9] presented a proposal for a efficient software CAVLC decoder architecture in H.264/AVC based on level length extraction (LLE), which achieved 22% faster decoding speed and 38% faster decoder compared with the conventional method.

In this paper, we propose a quick table look-up algorithm based on hash query for CAVLC decoding. We use hash table query to rapidly determine code length in CAVLC decoding, which could reduce much time of table looking-up and code judging. The simulation results show that our proposed scheme can reduce 20% table look-up time and save 1056 byte table storage space for CAVLC decoding without degrading video quality.

The rest of this paper is organized as follows. In Section 2, the principle and complexity analysis of CAVLC decoding are introduced. The proposed decoding method is presented in Section 3. And the simulation result and analysis of proposed scheme compared with TLSS method is presented in Section 4. In Section 5, we give some conclusions.

2. The Principle and Complexity Analysis of CAVLC Decoding

2.1. Principle of CAVLC Decoding

In baseline profile of H.264/AVC, the CAVLC and Exp-Golomb codes are used as two entropy decoding method. Exp-golomb codes are used to decode indication information and other coding parameters, which are with regular construction, While CAVLC is adopted to decode the quantized transform coefficients for residual blocks. Since the computation complexity of whole entropy decoding time is mainly occupied by the CAVLC decoding, the paper will focus on the CAVLC decoding procedures. In the process of CAVLC decoding, the quantized coefficients are zigzag scanned and then decoded by the five syntax elements. Figure 1 is the decoding order and the definition of five syntax elements above are described as follows.

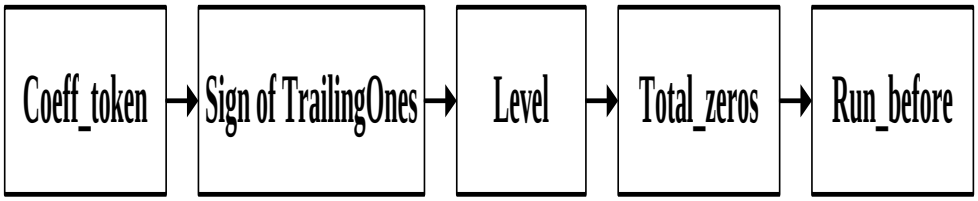


Figure 1. Decoding Order of Five Syntax Elements

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 Table 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. Complexity Analysis of CAVLC Decoding

During the process of CAVLC decoding, many variable length tables are used to decode codeword. However, there are some difficulties in looking up quickly these variable length tables above.

(1) The complexity of code tables storage structure: In H.264/AVC standard, the syntax elements of TotalCoeffs and Tls are stored in the form of 2D-code table .As we all known, For a 2D-code table, it is easy to find the corresponding content by its coordinates, but in turn, it is very difficult to find the corresponding coordinate through the content, because it needs search the whole table. Therefore, to decode the syntax elements of TotalCoeffs and Tls, it needs to spend a great lot of time in looking up 2D-code table and judging codeword for CAVLC decoding, which could greatly adds complexity of CAVLC decoding. The decoding process of TotalCoeffs could occupy up to 52% time in the entire process of CAVLC decoding when QP value is 30 [9].

(2) Selection diversity of codeword tables: In H.264/AVC standard, a codeword has many corresponding code tables, the selection and judgment of code tables needs to make a large number of calculations, which greatly adds the complexity of CAVLC decoding.

(3) Continuity of Code stream: During decoding process of H.264/AVC stream, since input stream is continuous, with no intervening separator in it, this requires CAVLC decoder to make lots of table looking up operations to get different code and do a large number of calculations to accurately judge different codeword from input decoding stream. Because those operations above are greatly time-consuming, they immensely add the complexity of CAVLC decoding.

3. Proposed Scheme

3.1. Variable Length Tables

As is mentioned above, the process of CAVLC decoding needs to decode five syntax elements. Three in five syntax elements above, Coeff_token, Run_before and Total_zeros, need to look up variable length tables. Through analyzing the structure of 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 codeword of 2D-variable length tables for Coeff_token (0≤NC<2). Table 2 is the part codeword of 1D-variable length tables for Total_zeros (Tc=5). In this paper, we just optimize table look-up algorithm to improve table look-up speed for 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 for the input bit-stream of the Coeff_token syntax element (0≤NC<2). The 8-bit Codeword represents the decoded output elements. The front 3 bits of them are for total number of ones (T1) and the other tail 5 bits are for the total number of coefficients (Tc). In table 2, the code represents for the input bit-stream of the syntax elemen of Total_zeros, while the codeword stands for single decoded output directly (Tc=5). 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. 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 some inherent relationships existed between between numbers of 0 in code prefix and code length for Coeff_token and Total_zeros are shown as Table 3 and Table 4 below 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	5, 6	6, 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	3, 4	4	4	5	5

Table 3 and Table 4 mean the inherent relationship between the length of code and proposed Numbers of 0 In Code Prefix and Total_zeros. Through the relationship, we can quickly determine the length of codeword suffix from decoding input bit-stream, which can save lots of time of looking up and judging codeword suffix.

3.3. Build Hash Table

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

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

Code	Hash table address	Numbers of 0 in code prefix	Length of code suffix	Code length
1	0	0	1	1
01	1	1	0	2
001	2	2	0	3
000100	3	3	1 or 2	5 or 6
000101				
00011				
000011	4	4	1 or 2	6 or 7
0000100				
0000101				
…	……	……	……	……
000000000000001	14	14	0	15

Table 6. Part Hash Table of 1D-table for Total_zeros (Tc=5)

Code	Hash table address	Numbers of 0 in code prefix	Length of code suffix	Code length
111	0	0	3	3
110				
101				
100				
0101	1	1	1 or 2	3 or 4

0100				
011				
0011	2	2	1	4
0010				
0001	3	3	0	4
00001	4	4	0	5
00000	5	5	0	6

In Table 5 and Table 6, the code represents for the input bit-stream of the corresponding syntax element. Hash table addresses in the table are allocated according to numbers of 0 in code prefix of code. Numbers of 0 in code prefix of code are mapped to Hash table addresses by hash function. The length of code suffix is determined through the relation existing between code length and numbers of 0 in code prefix. We make use of the number of zero in code prefix Calculated from input bit-stream to rapidly find the length of code suffix by a hash table built, which can reduce much time of code looking-up and judging.

3.4. Tables Look-up Algorithm Based on Hash Query

Based on the analysis above, we propose a quick table look-up algorithm for CAVLC decoding based on hash table query. The basic idea of new algorithm rests that we make use of the numbers of consecutive zero Calculated in input bit-stream to find the code length by a hash table built. The Table look-up algorithm based on hash query is shown as Figure 2.

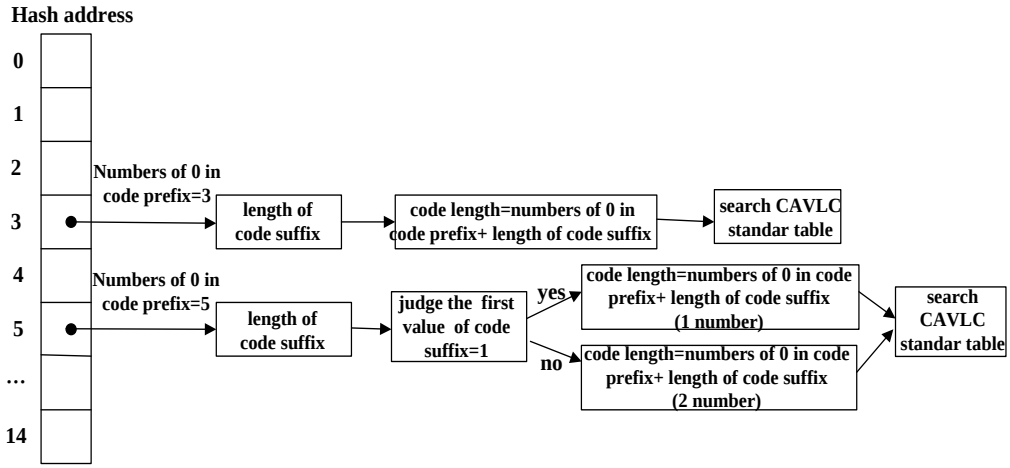


Figure 2. Table Look-up Algorithm based on Hash Query

The detained process of proposed table look-up algorithm base on hash table query for CAVLC decoding can be summed up as following steps.

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

Step 2: read input decoding bit-stream and calculate numbers of 0 in code prefix

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

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

Step 5: get code length according to the numbers of 0 in code prefix and length of code suffix above.

Step 6: search codtab to get decoded codeword according to code length determined above.

Take decoding Coeff_token syntax element for example to illustrate decoding process for our supposed table look-up algorithm. And suppose that NC value is 0≤NC<2, the bit-stream inputted is 0000100011…The detained decoding process of an example with TLHQ algorithm are shown as Figure 3

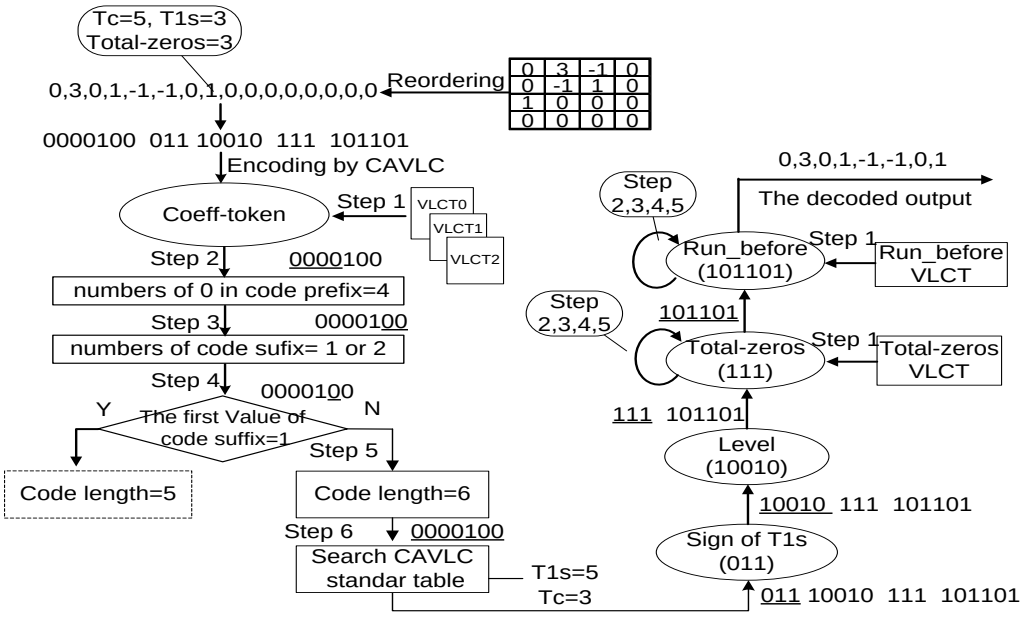


Figure 3. A Decoding Example with our Proposed Scheme

From Figure 3, we can clearly see the decoding process of an example with our proposed scheme.

In step 1, select variable length table0 of Coeff_token as table entry, because NC value is 0≤NC<2;

In step 2, read input decoding bit-stream and the numbers of 0 in code prefix calculated from input decoding bit-stream is 4;

In step 3, look up hash table 4 to get the length of code suffix being 1 or 2.

In step 4, as length of code suffix has two possible values, so we need to judge the first value of code suffix further and then we can directly get the length of code suffix being 2, and obtain the value of code suffix being 00.

In step 5, we can get the code length being 7 and the code being 0000100;

In step 6, as to 0000100, look up CAVLC table of Coeff_token, and the decoded Codeword is T1s=2 and Tc=5. After decoding the Coeff_token, level, sign of T1s, Total_zeros and Run_before syntax elements in sequence, we can get the last decoded output. Same steps can be taken for the syntax elements of Total_zeros and Run_before.

4. Simulation Results and Analysis

In order to verify the validity of the proposed method above, we take some experiments and simulation. This section mainly includes three parts: simulation environment, reduce table look up time and save table storage space.

4.1. Simulation Environment

The simulation Environment was conducted on an Intel 2GHz processor, 1GB memory capacity, Intel Windows XP operating system. Some common encoding parameters are shown in Table 7. Table 8 shows some parameters of test sequences, including the name, resolutions, frame rate and frame number of test sequences in our simulation experience.

Table 7. 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	On
RDOptimization	Fast search
Intra interval	0
Motion search	Fast search
SymbolMode	0 (CAVLC is used)
QPPrimeYZeroTransformBypassFlag	0 (lossless)
Encoder	JM 16.2 [10]

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

4.2. Reduce Table Look-up Time

In this subsection, we will evaluate the performance of table look-up time of our proposed algorithm for CAVLC decoding, which was compared with the conventional TLSS algorithms in different sequences with different frames and QP. The results are shown as follows in Table 9.

Table 9. Table Look-up Time for CAVLC Decoding in H.264/ AVC (us)

\	QP	24			28			32		
SEQ	Frames	TLSS	Ours	Improve (%)	TLSS	Ours	Improve (%)	TLSS	Ours	Improve (%)
Forman	30fs	91452424	739308398	23.7	67444596	55831619	20.8	46882240	38302483	22.4
	90fs	306339376	246077523	24.5	208683480	17465327	19.5	160228410	133523675	20.0
Crew	30fs	106440956	89975448	18.3	87563480	74332728	17.8	66303640	55811144	18.8
	90fs	303817596	254880533	19.2	231906120	196032223	18.3	166929040	141465288	18.0
Paris	30fs	67843250	55976298	21.2	56537290	47193063	19.8	49427280	41361740	19.5
	90fs	179095793	149996476	19.4	154580870	130558167	18.4	130896338	109353665	19.7
Wal	30fs	127718726	107416926	18.9	112815480	95850025	17.7	83498650	70344271	18.7
	90fs	361732130	299943723	20.6	311956514	262368890	18.9	251010610	107962394	20.7
Football	30fs	121012110	99353124	21.8	10646260	8894118	19.7	74306817	61466187	20.9
	90fs	291077420	243375769	19.6	210894200	178572715	18.1	166631660	140617434	18.5
Harbour	30fs	110447540	92891118	18.9	81513646	69196643	17.8	42399550	35659840	18.9
	90fs	340546073	284499643	19.7	241007480	202697628	18.9	206009120	172971553	19.1
Mobile	30fs	125115200	101885342	22.8	85790423	71492019	20.0	37523680	30960132	21.2
	90fs	390783324	320314200	22.0	28507620	23386070	21.9	125915116	105016777	19.9
Soccer	30fs	48479094	40602256	19.4	33094810	27351082	21.0	19407230	16118961	20.4
	90fs	161562740	13573047	18.8	110695448	92943281	19.1	67301960	56319631	19.5

From Table 9, we can clearly see that our proposed algorithm can shows about 20% saving time than standard TLSS method. The main reason for it is that our proposed algorithm uses a TLHQ method to reduce the operation of table look-up and code judging in CAVLC decoding, which can save a lot of table look-up time.

4.3. Save Table Storage Space

As our algorithm uses hash query to improve the table looking-up speed for CAVLC decoding, at the same time, it can also save table storage space. Table 10 is the consumption condition of storage space for our method compared with TLSS.

Table 10. Consumption Condition of Storage Space for our Method Compared with TLSS(byte)

method	Syntax elements	TLSS	Ours	Save space	save percent(%)
syntax elements value	Coeff_token	408	234	174	42.7
	Coeff_tokenDC	408	234	174	42.7
	Total_zeros	480	252	228	47.5
	Total_zerosDC	800	410	390	48.8
	Run_before	224	134	90	67.2

From Table 10, we can clearly see that our proposed scheme could save about 1056 byte table storage space. The main reason for it lies in the use of hash table query. With hash table query, our method could save the space of lentab storage in TLSS method, which can reduce total table storage space for CAVLC decoding.

5. Conclusion

In this paper, an efficient table look-up scheme is proposed for CAVLC decoding in H.264/AVC baseline profile. Since we use hash table query to quickly determine code length, as a result, it could reduce lots of operation of table look-up and code judging, which could reduce a lot of table look-up time. Simulation results show that our proposed scheme not only can reduce about 20% table look-up time than TLSS in CAVLC decoding, but also can save 1056 byte table storage space without degrading video quality.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive opinions in improving this letter. The work was supported by the Joint Funds of the National Natural Science Foundation of China and Guangdong Natural Science Foundation (No.U2012A002D01); Key Projects of National Natural Science Foundation of China (No.U2012A002D01);Foundation for Distinguished Young Talents in Higher Education of Guangdong (No. LYM11057).

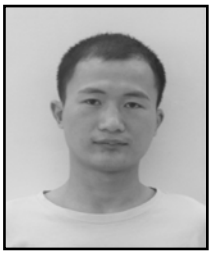
References

[1] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC). Doc. G050r1, **(2003)**.
[2] D. Lu, G. Z. Liu and 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 79-82.
[3] H.-Y. Lin, Y.-H. Lu and B.-D. Liu, "A highly Efficient VLSI architecture for H.264/AVC CAVLC Decoder", IEEE Transactions on multimedia, vol. 10, no. 1, **(2008)**.
[4] K.-Y. Wang, B.-S. Kim and S.-S. Lee, "A novel low-cost high-throughput CAVLC decoder for H.264/AVC", IEICE Transactions on Information and Systems.E94-D, vol. 4, **(2011)**.
[5] 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)**.
[6] 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)**.
[7] J. Young Lee, J. Jin Lee and S. Mo 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)**.
[8] Y. Hwan 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)**.
[9] N. Uchihara, H. Hayakawa and H. Kasai, "Efficient H.264/AVC Software CAVLC Decoder based on Level Length Extraction", IEEE Transactions on Consumer Electronics, vol. 58, no. 1, **(2012)**.
[10] K. Suhring, JM 16.2 software, http://iphome.hhi.de/suehring/tml/.

Authors

JianHua Wang 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 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 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 transmission, cyber-physical systems and wireless sensor networks.