A High Efficient Tables Look-up Algorithm for CAVLC Decoding

Jianhua Wang, Lianglun Cheng, Jun Liu, ShiLiang Luo
Faculty of Automation, Guangdong University of Technology, P. R. China
Email: 123chihua@163.com, llcheng@gdut.edu.cn, liujun7700@163.com, luo2002_88@163.com

***Abstract*—In order to solve the problems of high table memory access and long table look-up time and big table storage space in the process of CAVLC decoding for H.264/AVC, a high efficient table look-up algorithm is presented in this paper. The contribution of this paper lies that we uses a program method to realize fully the no-table looking-up of codeword. Specifically, after finding the relationships existed in code length and numbers of 0 in code prefix and code suffix and code, we uses a program code method to realize entirely UVLCTs(Unstructured Variable Length Coding Tables) in CAVLC decoding, As a result, all decoded codewords in UVLCTS can be decoded and obtained easily through a program execution way instead of TLSS(table look-up by sequential search), which could save a lot of table memory access and reduce amount of table look-up time and save a large number of table storage spaces for CAVLC decoding. The simulation results show that our proposed scheme can save 100% table memory access, reduce about 45% table look-up time and save 2320 byte table storage space of table look-up in CAVLC decoding compared with TLSS method, without degrading video quality.**

***Index Terms*—Memory access; Table look-up; Storage space; CAVLC decoding; Program method**

I. 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. H.264/AVC has three kinds of levels: Baseline profiles, Main profiles and Extended profiles. Main profiles are mainly adopted to improve image quality and compression ratio; Extended profiles are applied to networking video streaming transmission. Baseline profiles are used in a wide range of small size equipments with low complexity and low power consumption.

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, but at the same time, it also increases power consumption and hardware cost of the decoder. As we all know, table look up is a very power-consuming operation, which consumes most power of CAVLC decoder.Since looking-up variable length tables need to occupy about 96.6% time and consume amount of memory access of the entire CAVLC decoding in CAVLC decoding [2], It could seriously affect the efficiency of CAVLC decoding. 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 be decoded 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. So a lot of table look-up time and memory access will be required to find desired codewords from variable length tables and great mount of storage space will be required to store these variable length tables. In this paper, table memory access refers the memory access spent in looking up variable length tables. Table look up time refers the looking up time of variable length tables, and table storage space refers the storage space of storing variable length tables. It is well known that table look-up time and table look-up memory access and table storage space are three important performance and bottleneck in embedded systems especially for small-size multimedia applications [3].In order to reduce the memory access or save the table look-up time, many optimized decoding methods have been developed.

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 memory access and table look-up time, reducing 40% power consumption. Lee et al. [4] proposed pipelined architecture to save the operation frequency greatly, saving memory access and table look-up time. Wang et al. [5] presents a novel low-cost high-performance CAVLC decoder for H.264/AVC, which could greatly improve CAVLC decoding speed. Huang et al. [6] proposed a decoder based on CMOS and FPGA technology which reduced power consumption by 44-48% more than previous low-power CAVLD schemes.

In the software design level, some general table look-up methods, such as TLSS (Table Look-up by Sequential Search),TLBS (Table Look-up by Binary Search), have been required to decode CAVLC. However, the TLSS needs a great amount of memory access and a lot of table

Manuscript received March 2, 2013; revised July 20, 2013; accepted July 28, 2013
Corresponding author: 123chihua@163.com (Jianhua Wang)

doi:10.4304/jsw.9.2.329-335

look-up time to spend in every decoded codeword due to complete table look-up for the desired codeword. 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 reduce some table look-up time and reduce about 65%-88% memory access [7]. Lu et al. [8] proposed a entropy decode algorithm which can decrease about 75.1%-82.7% time than the original algorithm in the H.264 reference software. Lee et al. [9] developed a new codeword structures, a looking-up tables and searching methods for the CAVLC syntax elements and achieved about 90% memory access savings. But Lee's method still need to look up some tables, which caused some memory access consumption. In Kim's method, some other integer arithmetic operations are proposed, which can reduce about 94% memory access, at the same time, it improved greatly table look-up speed [10].In this work[11], Uchihara et al. proposed a fast skip scheme of CAVLC level code, which can reduce 70% of CAVLC level code skip And In the paper[12], in order to reduce decoding time, they presented a proposal for an 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 high efficient table look-up algorithm based on program method for CAVLC decoding. The achievement of our algorithm rests that we uses program code to realize fully the looking-up of UVLCTs (Unstructured Variable Length Coding Tables) in CAVLC decoding, As a result, all decoded codewords in UVLCTs can be obtained easily through a program execution way instead of TLSS (table look-up by sequential search)method, which could save a lot of table memory access, reduce amount of table look-up time and save a large number of table storage spaces in the process of CAVLC decoding. The simulation results show that our proposed scheme can save 100% memory access of table look-up and reduce 45% table look-up time and save 2320 byte table storage space 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 complexity analysis of CAVLC decoding are introduced. The proposed decoding method is presented in Section 3. And the simulation results of proposed scheme compared with existing methods are presented in Section 4. In Section 5, we give some conclusions.

II. The Principle and Complexity Analysis of CAVLC Decoding

A. 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 computation complexity of whole entropy decoding is mainly occupied by the CAVLC decoding, this 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. The decoding order and definition of five syntax elements above are described as follows.

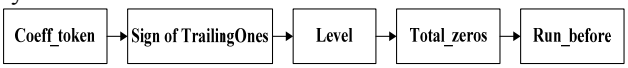

Figure1. Decoding order of five syntax elements

- Coeff_token: Both the number of nonzero coefficients (Totalcoeff) and number of coefficients that absolute value is equal to one (TrailingOnes).
- 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.
- Level: The values for each nonzero coefficient except for T1s in reverse zigzag order.
- Total_zeros: The total number of zero coefficients between the DC and the last nonzero coefficient in zigzag order.
- Run_before: The numbers of zeros preceding each nonzero coefficient in reverse zigzag order.

B. 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 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, decoding the syntax elements of TotalCoeffs and Tls needs to spend a great lot of time and memory access in looking up 2D-code table, matching 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 [13].

(2) Selection diversity of code 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.

III. PROPOSED SCHEME

A. 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 analysis 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 has one 1D-variable length tables. Table 1 is the Part codeword of 2D-variable length tables for Coeff_token (2≤NC<4). Table 2 is the Part codeword of 1D-variable length tables for TOTAL_ZEROS (TC=6). In this paper, we just need to optimize the lookup algorithm for those variable length tables above.

TABLE 1.
PART OF 2D-VARIABLE LENGTH TABLE FOR COEFF_TOKEN (2≤NC<4)

Code	Codeword	[T1,Tc]
10	0x21	[1,1]
11	0x00	[0,0]
011	0x42	[2,2]
0100	0x64	[3,4]
0101	0x63	[3,3]
00110	0x65	[3,5]
00111	0x22	[1,2]
001000	0x66	[3,6]
000111	0x02	[0,2]
…	……	……
0000000000001	0x6f	[3,5]

TABLE 2.
PART OF 1D-VARIABLE LENGTH TABLE FOR TOTAL_ZEROS (TC=6)

Code	Codeword
111	2
110	3
101	4
100	5
010	6
011	7
001	9
0001	8
00001	1
000001	0
000000	10

In table 1, the code represents for the input bit-stream of the Coeff_token syntax element (2≤NC<4). 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**, the codeword stands for single decoded output directly (TC=6). The other elements of Coeff_token, Run_before and Total_zeros have the same code table structure as Table 1 and Table 2.

B. 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 corresponding relationships in code length and numbers of 0 in code prefix for Coeff_token and Total_zeros. Table 3 and Table4 are the corresponding relationships respectively.

TABLE 3.
THE RELATIONSHIP EXISTS BETWEEN LENGTH OF CODE AND NUMBERS OF 0 IN CODE PREFIX CORRESPONDING TO TABLE 1

Numbers of 0 in code prefix	Code length
0	2
1	3 or 4
2	4 or 5
3	6
4	7
5	8
6	9
7	11
8	12
9	13
10	13 or14
11	14
12	13

TABLE 4.
THE RELATIONSHIP EXISTS BETWEEN COD LENGTH AND NUMBERS OF 0 IN CODE PREFIX CORRESPONDING TO TABLE 2

Numbers of 0 in code prefix	Code length
0	3
1	3
2	3
3	4
4	5
5	6
6	6

Table 3 and Table 4 are the relationship conditions between the code length and numbers of 0 in code prefix for Coeff_token (2≤NC<4) and Total_zeros (Tc=6). Through the relationships above, we can find a quick way to determine the length of code suffix through making use of the relationships between the code length and numbers of 0 in code prefix, which can save lots of time and memory access of looking up tables and matching, judging and processing code suffix.

C. Relationship between Length of Code Suffix and Numbers of 0 in Code Prefix

Table 5 and Table 6 are the relationship between length of code suffix and numbers of 0 in code prefix for 2D-Coeff_token and 1D-Total_zeros based on the relationship between numbers of in code prefix and Code length respectively.

TABLE 5 .
PART RELATIONSHIP BETWEEN NUMBERS OF 0 IN CODE PREFIX AND LENGTH
CODE SUFFIX FOR COEFF_TOKEN (VLCT1, 2≤NC<4)

code	Numbers of 0 in code prefix	Length of Code suffix
10	0	2
11		

011	1	1or 2
0100		
0101		
00110	2	2 or 3
00111		
001000		
000111	3	2
…	……	……
0000000000001	12	0

TABLE 6
PART RELATIONSHIP BETWEEN NUMBERS OF 0 IN CODE PREFIX AND LENGTH
CODE SUFFIX FOR TOTAL_ZEROS (TC=6)

Code	Numbers of 0 in code prefix	Length of Code suffix
111	0	3
110		
101		
100		
010	1	3
011		
001	2	0
0001	3	0
00001	4	0
000001	5	0
000000	6	0

In table5 and table6, the numbers of 0 in code prefix of code is obtained by calculating consequent zero from input bit-stream. The length of code suffix can be determined by the relation existing between code length and numbers of 0 in code prefix, such as table3 and table4. Since Code is made up of numbers of 0 in code prefix and code suffix. After determining numbers of 0 in code prefix of code and code suffix, we can determine the code. Based on the basic idea above, we suppose a table look-up algorithm based on program method for CAVLC decoding.

D. Tables Look-up Algorithm based on Program Method

In this paper, based on analysis above, we propose a new table look-up algorithm based on program method 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 the first progress entry of code judging, the value of code suffix gotten according to the length of code suffix from input bit-stream as the second progress entry of code judging , whose length is determined by the relation existing between numbers of 0 in code prefix and code length, then through the first and second progress execution above, we can quickly get the decoded output. The process of table look-up algorithm based on program method could be shown as Figure 2.

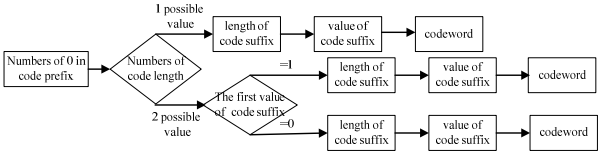

Figure 2. Process of tables look-up based on program method

The proposed table look-up algorithm based on program method for CAVLC decoding can be summed up as following some 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 as the first progress entry of code judging.

Step 3: get the numbers of code length according to numbers of 0 in code prefix above.

Step 4: determine the length of code suffix according to the relationship existing between code length and numbers of 0 in code prefix. If the code length has two possible values, we will determine the numbers of code length after judging the first value of code suffix again.

Step 5: read the value of code suffix as the second progress entry of code judging

Step 6: find decoded codeword according to through the first and second progress execution of code judging above

Now take decoding Coeff_token syntax element for example to illustrate decoding process with our supposed table look-up algorithm. And suppose that NC value is $0 \le NC < 2$, the bit-stream inputted is 0000100011…, Figure 3 is a decoding example process with our supposed algorithm based on program method**.** Table 7 is pseudo code of tables look-up based on program method.

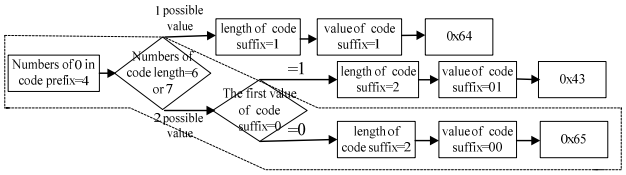

Figure 3. Process of tables look-up based on program method

Table 7.
Part pseudo code of tables look-up based on program method

```
Procedure TableLook up_Program{
  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 ←ShowBit(x,x,x);
  (5)select the first program code entry ←numbers of 0 in code prefix;
  (6)get code length ← numbers of 0 in code prefix;
  (7)get the length of code suffix ← code length-Numbers of 0 in code prefix+1;
  (8)if (the length of code suffix==one possible value){
  (9)   get the value of code suffix ←the length of code suffix;
  (10)  select the second program code entry ← the value of code suffix;
  (11)  get codeword;
  (12) }
  (13)if (the length of code suffix==two possible value){
  (13)if( the first value of code suffix==1){
  (14)  get the value of code suffix;
  (15)  select the second program code entry← the value of code suffix;
  (16)  get codeword;
  (17) }
  (18)else{
  (19)  get the value of code suffix ←the length of code suffix;
  (20)  select the second program code entry←the value of code suffix;
  (21)  get codeword;
  (22) }
  (23) }
  (24) else{
  (25) codeword = "decoding error"
  (26)}
}
```

From Figure 3 and Table 7, we can see clearly that our

supposed algorithm can use program code to realize code-tables and look up codeword according to the relationship existing in code length and numbers of 0 in code prefix and code suffix and code, it can save a lot of table memory access and reduce table look-up time and save a large number of table storage spaces in the process of CAVLC decoding. Figure 4 is the decoding process of Coeff_token for an example with our proposed scheme.

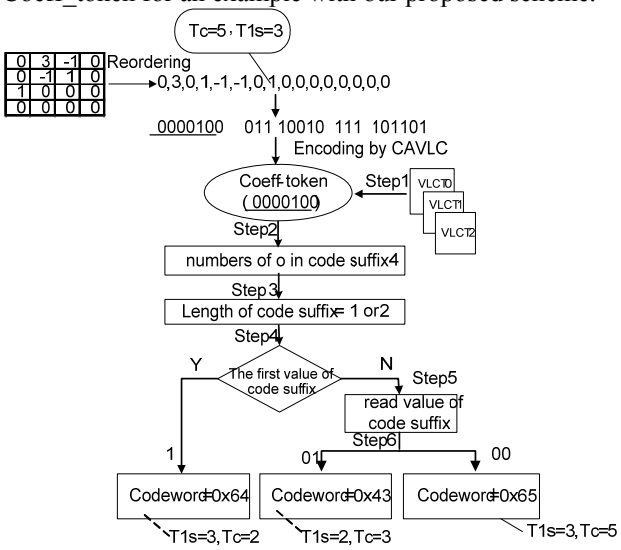

Figure 4. Decoding process of Coeff_token with our proposed scheme

From Figure 4, we can clearly see the Coeff_token decoding process with our proposed scheme. The decoding process is as following:

Step 1, select variable length table0 of Coeff_token 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 decoding bit-stream as the first progress entry of code judging;

Step 3, get numbers of code length (1 or 2) according to length of code suffix (4)

Step 4, get two possible value (1 or 2) for length of code suffix according to the relationship existing between code length and numbers of 0 in code prefix. After judging the first value of code suffix (0) again, we can get the length of code suffix (2)

Step 5, read value of code suffix (00) from decoding bit-stream according to length of code suffix (2) and take it as the second program entry of code judging

Step 6, find the decoded Codeword (0x64) according to through the first and second progress execution of code judging above, 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.

IV. Experimental Results and Analysis

In order to verify the effectiveness of our proposed method above, we take some experiments. Our designed experiments mainly include four parts: experimental environment, save table memory access, reduce table look-up time and save table storage space.

. A. Experimental 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
Mobile (M)	CIF(176×144)	25	60, 120
Walk(W)	CIF(176×144)	25	60, 120
Paris(P)	CIF(176×144)	25	60, 120
Football (F)	QCIF(352×288)	25	60, 120
Soccer(S)	QCIF(352×288)	25	60, 120
walk(W)	QCIF(352×288)	25	60, 120

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 [15]

B. Save Table Memory Access

In this subsection, We evaluate the memory access savings of our supposed table look-up compared with other four methods, TLSS, TLBS, Moon's [7], and Kim's [10], and Figure 5, Figure 6 and Figure7 are the saving results

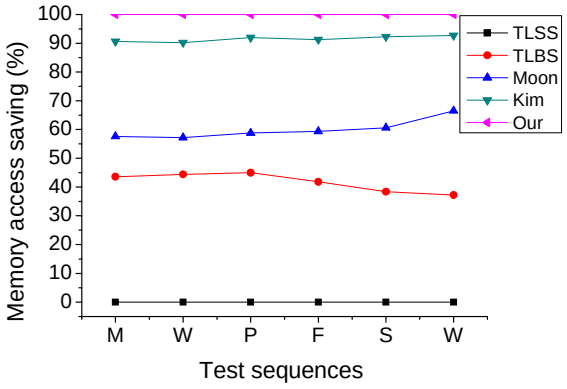

Figure 5. Memory access savings comparison in table look-up for 60 frames of test sequence (QP=24)

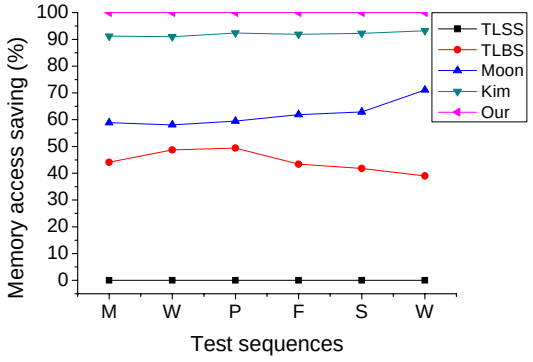

Figure 6. Memory access savings comparison in table look-up for 60 frames of test sequence (QP=28)

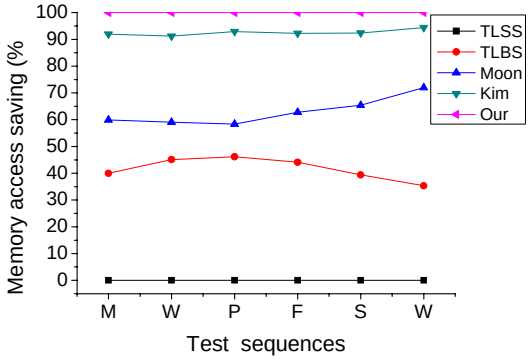

Figure 7. Memory access savings comparison in table look-up for 120 frames of test sequence (QP=32)

From Figure 5, Figure 6 and Figure 7, we can see that our proposed algorithm has the best table memory access saving in five methods above, reaching 100% memory access saving. Kim's method follows. The reason for it is that, we use program method to instead of all tables looking-up, which can save a lot of table memory access spent in looking up table. In this experience, we also find that TLBS and Moon's method have the worse save result, while TLBS method shows better saving results in the higher QP and Moon's method in the lower QP. The similar conclusion can be found in reference [10].

C. Reduce Table Look-upTime

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

From Table 10, we can clearly find that our proposed algorithm has superior results in five methods above and shows about 45% saving compared to the standard TLSS method. The main reason for reducing table look-up time in CAVLC decoding for H.264/AVC lies that the use of program method. Because our proposed algorithm uses a program method to instead of table look-up in CAVLC decoding, which can reduce the numbers of codetable looking-up and save the time of code searching, matching, judging and process operation.

TABLE 10.
DECODING TABLE LOOK-UP TIME IN CAVLC FOR H.264/ AVC (MS)

QP		24					28					32				
SEQ		TLSS	TLBS	Moon	Kim	Ours	TLSS	TLBS	Moon	Kim	Ours	TLSS	TLBS	Moon	Kim	Ours
M	60	73545	48646	50738	44564	40940	61793	39653	41674	35302	32326	53787	32664	35983	30990	28899
	120	157372	110231	115682	103547	89856	113086	69035	61825	64542	59456	87642	57543	59709	54675	50862
W	60	54676	34683	36782	312881	28509	45890	26747	28673	25783	24090	38853	27689	29654	24661	20613
	120	115432	74573	76577	72675	66906	83896	56784	57672	53661	48301	62004	40672	44367	38603	35468
P	60	64676	44676	51879	41715	35168	53985	32005	34800	309878	28540	46567	28007	29182	27823	25004
	120	155424	83204	88990	83408	81328	109934	53896	67897	60992	58305	85674	49538	51076	48631	44456
F	60	40295	28116	30675	25707	22590	33554	25465	23543	206004	18092	27665	20567	21770	17784	15118
	120	81887	55435	57564	50672	47263	61734	37564	39683	35697	32870	51764	30668	31896	28201	27625
S	60	43351	27044	2962	25061	23025	29990	18856	20999	17657	16781	16479	12537	13768	9847	8565
	120	84451	48785	50647	47366	45610	47894	28885	30872	26577	24966	33144	19361	20625	19946	18197
W	60	61824	39674	31670	35675	33633	51698	30668	33645	29667	28152	43246	29789	30230	27162	25031
	120	147354	95455	99875	90054	81631	113542	706754	796004	626594	59400	78564	55679	57651	49654	44189

D. Save Table Storage Space

As our algorithm uses a program way to realize the table look-up for CAVLC decoding, it can also completely save data codeword storage space for UVLCTs. Table 11 is the consumption condition of storage space for our method compared with TLSS

TABLE 11.
CONSUMPTION CONDITION OF TABLE STORAGE SPACE (BYTE)

method	syntax elements	TLSS	ours
Storage space	Coeff_token	408	0
	Coeff_tokenDC	408	0
	Total_zeros	480	0
	Total_zerosDC	800	0
	Run_before	224	0

From table 11, we can see that our proposed scheme can save about 2320 byte table storage space compared with TLSS method. The main reason for saving space lies in the use of table look–up base on program method, which can completely save all table storage space for CAVLC decoding.

V. CONCLUSION

In this paper, a high efficient table look-up scheme abased on program method is proposed for CAVLC decoding in H.264/AVC. In our scheme, we use program method to realize fully the UVLCTS and decode codeword without any table look-up, which can resolve problem of high table memory access and long table look-up time and large table storage spaces. Simulation results show that our proposed scheme not only can save 100% memory access in table look-up, but also reduce about 45% time in decoding table look-up than the conventional CAVLC decoding and save 2320 byte table storage space, without degrading video quality.

ACKNOWLEDGMENT

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);Key Projects of National Natural Science Foundation of China (No.U2012A002D01);The Strategic Emerging Industries Special of Guangdong Province (No.2012A09100013-2012BAF11B04–5150). Project of Ministry of Science and Technology (NO.2012BAF11B04).

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] Junghee Lee, Chanik Park, and Soonhoi Ha, “Memory Access Pattern Analysis and Stream Cache Design for Multimedia Applications,” in Proc. of Asia and South Pacific Design Automation Conf (DAC), Jan,2003,pp.22-27.

[3] Heng-Yao Lin, Ying-Hong Lu, Bin-Da Liu, and Jar-Ferr Yang, “A highly efficient VLSI architecture for H.264/AVC CAVLC decoder,” IEEE Transactions on multimedia, vol. 10, no. 1, pp.31-34 , 2008.

[4] Byung-Yup Lee, and Kwang-Ki 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, pp.2781-2789, 2010.

[5] Wang, Kyu-Yeul, Kim, Byung-Soo; Lee, Sang-Seol; Kim, Dong-Sun; Chung, Duck-Jin. “A novel low-cost high-throughput CAVLC decoder for H.264/AVC,” IEICE Transactions on Information and Systems, Vol. E94-D, No. 4, pp. 895-904, 2011

[6] Fang, C.-H. ,Fan, C.-P. “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, pp.104-114, 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, No3, pp.933-938, 2005.

[8] Lu, Da,Liu, Guofan Zhu and Lingli,“An optimization for CAVLC code table lookup algorithm in H. 264 decoder,” The 2th International symposium on Intelligence Information Processing and Trusted Computing, China, 2011,pp.79-83.

[9] Jun Young Lee, Jae Jin Lee, and SeongMo Park, “New lookup tables and searching algorithms for fast H.264/AVC CAVLC decoding” , IEEE Trans. Circuits and Syst. Video Technol. vol. 20, no.7 pp.1007-1017, 2010.

[10] Yong-Hwan Kim, Yoon-jong Yoo, Jeongho Shin, Byeongho Choi, and Joonki Paik, “Memory-efficient H.264/AVC CAVLC for fast decoding,” IEEE Trans On Consumer Electronics, Vol.52, No.3, pp.943-952, 2006.

[11] Naofumi Uchihara, Hiroki Hayakawa, Hiroyuki Kasai. “Fast CAVLC Level Code Skip scheme for H.264/AVC stream manipulation,” Consumer Electronics (ICCE), 2012 IEEE International Conference. Pp.267 - 268 , Jan. 2012

[12] Naofumi Uchihara, Hiroki Hayakawa, and Hiroyuki Kasai. “Efficient H.264/AVC software CAVLC decoder based on level length extraction,” IEEE Transactions on Consumer Electronics, Vol. 58, No. 1,pp.146-153, 2012

[13] Jun-Young Lee, Jae-Jin Lee, and SeongMo Park, “New lookup tables and searching algorithms for fast H.264/AVC CAVLC decoding,” IEEE Trans. On Circuits and Systems for Video Technology, vol.20, no.7, pp.1007-1017, 2010

[14] 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.

[15] K. Suhring, “JM 16.2 software,” http://iphome.hhi.de/suehring/tml/.

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, IoT, CPS and wireless sensor networks.

Lianglun Cheng was born on August 22, 1964 in Hubei, China. 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, 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 wireless transmission, CPS and wireless sensor networks.

Shiliang Luo was born on October, 1978 in JiangXi, China. He received his M.S degree in automation from Guangdong University of Technology, Guangdong, China, in 2005. Currently he is pursuing Ph.D degree in Control Science and Engineering at Guangdong University of Technology. His research interests include CPS and wireless sensor networks.