Design of A Memory-Based VLC Decoder for Portable Video Applications

Wei-Chin Lee

Department of Electronic Engineering
NCTU
Hsinchu, Taiwan
pili@si2lab.org

Yao Li

Department of Electronic Engineering
NCTU
Hsinchu, Taiwan
yaoli@si2lab.org

Chen-Yi Lee

Department of Electronic Engineering
NCTU
Hsinchu, Taiwan
cylee@si2lab.org

Abstract—In this paper, a VLC decoder supporting to decode coefficient data in blocks of MPEG-2 and CAVLC in H.264 is presented. To achieve programmability of the VLC decoder, a memory-based architecture with improved memory efficiency is proposed. Group-based look-up table (LUT) algorithm is extended to multi-table merging (MTM) which extracts redundancy of groups further. With multi-table merging algorithm, all coding tables are integrated into memory more efficiently. While the memory access may lead to much power consumption, a low-power scheme is proposed to reduce memory access. The distributed cache is adopted to save power and improve the decoding throughput as well. Simulation results show that the cache with replacement method can reduce about 60% ~ 95% memory accesses.

I. INTRODUCTION

Variable length coding (VLC) is an important technique to achieve video compression in nowadays video compression standards. The ITU-T/ISO/IEC Joint Video Team established the newest video coding standard known as H.264/AVC [1]. This standard achieves a much more compression rate than the previous standards. There are two entropy coding types in H.264/AVC, one is CAVLC used in the baseline profile and the other is CABAC used in main profile. On the other hand, the MPEG-2 from Motion Pictures Expert Group (MPEG) [2] is widely used in video applications. Therefore, to support these two standards is necessary for a video decoder.

H.264/AVC baseline profile contains CAVLC which is decoded by looking up tables. There are five syntax elements in CAVLC: Coeff-Token, sign of TrailingOnes, Level, Total_Zeros and Run_Before. Sign of TrailingOnes is decoded by reading 3 bits at most. Level can be decoded by detecting first 1 and get certain bits from the bitstream. However, Coeff-Token, Total_Zeros, and Run_Before require to access memory frequently because of looking up tables. The coefficients data in MPEG-2 is composed of run-level pair (RLP) symbol and also decoded by look-up-table

mechanism. From the above, memory-based architecture is used so that different contents can be loaded into memory for MPEG-2 or H.264 standards respectively. Nevertheless, memory access consumes power and thus become overhead for handheld or portable devices. Consequently, a low-power design issue lies in reducing memory access.

Up to now, there has been research efforts reported to reduce memory access. In [3], some short codewords are decoded by arithmetic operation to reduce memory access. But its sequential searching would lead to low throughput. In [4], almost all codewords are decoded by using arithmetic operation while a few codewords are decoded by look-up-table with binary search (TLBS) to reduce memory access. However, the decoding method is not implemented with hardware and the arithmetic operation and condition is complicated. HLLT (Hierarchical Logic for Look-up Table) and PCCF (Partial Combinational Component Freezing) were proposed in [5] to improve speed and reduce power consumption respectively. Finally, the scheme proposed in [6] is to decode short codewords with arithmetic operation while other codewords are decoded by conventional decoding.

From the above, we can find that although those proposed methods can reduce memory access, their decoder cannot support multi-standard video decoder. To reduce the memory space while maintain the programmability for different applications, multi-table merging (MTM) is used to combine all tables into memory [7]. Furthermore, a group partitioning scheme of short VLC codewords is proposed to improve memory efficiency, where cache with Least Recently Used (LRU) replacement is designed to reduce power consumption of the proposed VLC decoder.

II. MULTI-TABLE MERGING ALGORITHM

From the CAVLC decoding process and the group-based VLD algorithm [8], it can be found that storage space is increased if we group all tables and put the information into memory separately. Therefore, we have to develop an algorithm to merge these tables into one single memory. The algorithm is explained as follows:

- (1) Generate group information for multi-table merge:

* Work supported by the NSC of Taiwan, R.O.C under contract NSC96-2221-009-180.

For every coding table, we use group-based algorithm proposed by [8] so we have group information for each table. Each group has four fields: table index, group index, codeword length, pseudo-constant-length codeword (PCLC) and base address. The codeword length of every PCLC must be the same, we make the PCLC codewords have the longest VLC codeword among all coding tables for multi-table merge. For example, the longest length of the codeword of Coeff_Token table is 16-bit, so every codeword is extended to 16 bits by adding 0's.

(2) Sort all groups:

According to the value of PCLCs in descending order, we sort all groups which belong to different tables. The part of the group information of Coeff_Token table which contains 13 items needed to be stored in the memory is shown in Figure 1.

	Valid bit	PCLC_mincode	CL	base_addr
VLC0.G8	1	16'h0200	9	45
VLC1.G8	1	16'h0200	9	35
VLC2.G7	1	16'h0200	10	7
VLC4.G6	1	16'h0200	8	1
VLC0.G9	1	16'h0100	10	41
VLC1.G9	1	16'h0100	11	27
VLC2.G8	1	16'h0100	10	3
VLC0.G10	1	16'h0080	11	37
VLC1.G10	1	16'h0080	12	19
VLC2.G9	1	16'h0080	10	1
VLC0.G11	1	16'h0040	13	29
VLC1.G11	1	16'h0040	13	11
VLC2.G10	1	16'h0040	10	0

Figure 1. One part of group information of Coeff_Token tables

(3)Apply codeword merging to PCLC_mincodes:

From the result of the above steps, several values of PCLC_mincodes which belong to different tables and different groups are the same; therefore, we can merge them into one group so that those codeword groups became a merged group. The result is shown in Figure 2. There are four groups and the differences in every group are also needed to be stored, such as base_addr and CL. Thus, only 4 items needed to be stored, which is less 13 items in Fig.1. There are $17+16+11+7+8 = 59$ items after step1. After this process, there are only 23 groups, which is much less than the sum of group number of individual tables.

	Valid bit	merged PCLC_mincode	CL	base_addr
VLC0.G8	1	16'h0200	9	45
VLC1.G8	1		9	35
VLC2.G7	1		10	7
VLC4.G6	1	16'h0100	8	1
VLC0.G9	1		10	41
VLC1.G9	1		11	27
VLC2.G8	1	16'h0080	10	3
VLC0.G10	1		11	37
VLC1.G10	1		12	19
VLC2.G9	1	16'h0040	10	1
VLC0.G11	1		13	29
VLC1.G11	1		13	11
VLC2.G10	1		10	0

Figure 2. One part of group information of Coeff_Token tables

(4)Set table information:

After the codeword is merged, the merged groups and PCLC_mincodes are MTM groups and MTM_PCLC_mincodes respectively. The table information of a coding table includes the valid-bit and the length of codewords. Because the shortest length of codeword is 1 bit and the length is from 1 bit to 16 bits, we just store (length-1), i.e. 0~15 in the memory to save memory space. After this shifting operation, the smallest (length-1) in all groups is defined as MTM_CL-1 and stored in the group information memory. Therefore, the difference between the larger (length-1) and (MTM_CL-1) which is defined as CL_diff is stored in the table information memory. The memory space is further saved because the data redundancy among the lengths in a MTM group is exploited. The table information and group information are shown in the Figure 3.

When the MTM algorithm is applied, the symbol addresses are designed to have different offsets so that we can correctly allocate symbols in the memory. Take Coeff_Token as an example, because VLC 0~3 has 62 symbols, symbol address offset of VLC0 is 0 and symbol address offset is 64, 128 and 192 for VLC1, VLC2 and VLC3, respectively.

For simplification, the Figures shown above are just for Coeff_Token tables (VLC0~4). In this paper, we can further merge all tables in one memory to achieve programmability by using the MTM algorithm. This is a different design as compared to the VLC decoder proposed by [3~6] which focused on Coeff_Token tables to achieve memory-efficient decoding while use hardwired circuit to decode Level, Total_Zeros, and Run_Before respectively.

The paper proposed a symbol memory allocation so that one memory is used and almost supports enough memory space for symbols of TB14 and TB15 in MPEG-2 and Coeff_Token, Total_Zeros and Run_Before tables in H.264. The allocation is shown in Figure 4. We can see that for TB14 and TB15 in MPEG-2, the run and level pair can be aligned in one entry and for H.264, there is a mask to extract different symbols for two-symbol tables (Coeff_Token tables) and one-symbol tables (Total_Zeros and Run_Before tables). For H.264, there are $248 + 14 + 135 + 9 + 42 = 448$ symbols, and thus we have to use 512-entry memory with $512 - 445 = 67$ entries unused. However, with this allocation method, we can just use 256-entry memory instead of 512-entry memory to improve memory efficiency and also reduce the critical path for hardware implementation.

III. LOWER POWER SCHEME

After merging all tables in the memory, we can see that the symbol memory is large and thus the access time and power consumption. Therefore, we have to reduce memory access to reduce power consumption. In [9], several LUTs implemented with ROM are separated according to group probability distribution and then one cache implemented with register files are used to reduce memory access. However, the separated LUTs lead to unequal throughput performance, that is, if the cache is missed, LUT1 is also missed, and the decoder would search until the LUTk is hit.

Table Information					MTM Groups		

Figure 3. MTM group information and table information.

Figure 4. 256-entry symbol memory allocation in VLD for MPEG-2 and H.264 standards

Take throughput into account, we use two-level memory hierarchy concept. Individual register file is used as cache for the corresponding table. Thus, we can just look up cache first and if it is missed, table memory, group memory and symbol memory would be accessed. Initially, the cache update scheme is simple. The codeword and the corresponding symbol are recorded in the cache until the cache is full. And it won't be updated if all registers are written. The format of one entry in the register file is shown in Figure 5. For Coeff_Token ($nC \geq 0$) tables, $m=3$, $n=6$, $x=4$ and $y=7$ are the width. For Coeff_Token ($nC = -1$) table, $m=0$, $n=3$, $x=4$ and $y=7$ are the width. For Total_Zeros or Run_Before tables, $m=4$, $n=4$, $x=4$ and $y=4$ are set as the width. Finally, $m=2$, $n=1$, $x=2$ and $y=2$ for Total_Zeros (chroma 2x2).

Figure 5. Format of the register file

Figure 6. shows the simulation results of 300 frames from the first 100 frames of akiyo, the first 100 frames of foreman and the last 100 frames of foreman sequence with 3 different cases for QP = 20, QP=24 and QP=28.

The x-axis is frame number and d y-axis is the cache hit rate. From this figure, as QP is lower, hit rate is lower than the hit rate under higher QP. That is, for lower QP of a video sequence, the decoder will access memory more frequently. The average hit rate is 54.8% for QP = 20, 64.2% for QP = 24 and 76.9% for QP = 28.

Figure 6. Probability of finding codeword in register file for Coeff_Token tables under QP=20,24 and 28, respectively

Because total number of codewords in VLC0~3 is $62 \times 4 = 248$, we choose an appropriate size of register file, 64, to store codeword length, codeword value, table index, and symbols. TABLE I. shows the size of register file for all tables.

TABLE I. REGISTER FILE INFORMATION FOR INDIVIDUAL TABLES (A=NUMBER OF CODEWOS,B=NUMBER OF ENTRIES OF CACHE)

Table	A	B	Average hit rate		
			QP=20	QP=24	QP=28
Coeff_Token ($0 \leq nC$)	248	64	54.8%	64.2%	76.9%
Coeff_Token ($nC = -1$)	14	8	68.0%	74.5%	82.6%
Total_Zeros (for 4x4)	135	64	69.7%	72.2%	73.8%
Total_Zeros (for 2x2)	9	8	90.7%	90.3%	89.9%
Run_Before	42	16	75.3%	71.9%	68.4%

To reduce cache size, Least Recently Used (LRU) replacement method is used. The LRU replacement is achieved by the counter of each cache entry. The counter represents how recently the entry is used. If the entry is used (cache is hit), the counter for the entry is incremented. Otherwise, the counter is decremented. If the cache is full and cache is missed, we replace the entry where the count is minimal. That is, the entry is least recently used so it is replaced by new content. TABLE II. is the result of the cache

using LRU replacement. As a result, the memory access can be reduced greatly and hence the power consumption.

TABLE II. REGISTER FILE INFORMATION FOR INDIVIDUAL TABLES
(A=NUMBER OF CODEWORDS,B=NUMBER OF ENTRIES OF CACHE)

Table	A	B	Average hit rate		
			QP=20	QP=24	QP=28
Coeff_Token (0<=nC)	248	32	57.4%	66.9%	77.1%
Coeff_Token (nC = -1)	14	8	70.5%	73.7%	78.4%
Total_Zeros (for 4x4)	135	32	95.4%	95.6%	95.7%
Total_Zeros (for 2x2)	9	8	93.2%	93.0%	92.3%
Run_Before	42	16	96.2%	95.5%	93.4%

IV. ARCHITECTURE

As shown in Figure 7, the input shifter provides 2 bytes of bitstream to the VLC decoder input and the symbol address is calculated. Finally, the symbol memory is accessed to decode the symbol.

Take efficiency into account, 30 coding tables in CAVLC are separated into two categories. We use one memory for the 16 coding tables and one for the other 14 coding tables to reduce unused locations.

The core part of the VLC decoder was implemented by Verilog HDL and the memory is generated by Artisan TSMC 0.13-um memory compiler. The design was synthesized with TSMC 0.13um cell library. The gate count is 40067 (excluded cache) at 100MHz.

Figure 7. Hardware architecture of the memory-based VLC decoder

The memory space of the proposed design is shown in TABLE III.

TABLE III. MEMORY SPACE FOR TABLE INFORMATION, GROUP INFORMATION AND SYMBOL MEMORY

	Storage content	Number of bits
Table information memory	Valid bit and CL_diff	16 x 23 x 4 + 14 x 5 x 3
Group information memory	Base address	16 x 23 x 8 + 14 x 5 x 8
Symbol memory	C.T and TZ/RB Run-Level	256 x 11
	C.T for CAVLC	16 x 7
Total		8114 bits

TABLE IV. shows the comparisons of several VLC decoder designs. We use larger space for supporting MPEG2 and H.264 which are quite different standards instead of supporting only one standard. Besides, the space from [10] didn't include symbol memory space which is reduced in the proposal.

TABLE IV. COMPARISON OF VLC DECODERS

	[3]	[5]	[6]	[10]	Proposed
Mechanism	Hardwire & Memory	ROM	Hardwire	Memory	Memory
Memory space (bits)	N/A	N/A	N/A	1014	8114
Reduced memory access	95%	N/A	65%~88%	N/A	50%~95%
Multi-mode	H.264	H.264	H.264	MPEG4	H.264 MPEG2

V. CONCLUSION

With the MTM algorithm, all symbols in coding tables of CAVLC and MPEG-2 (TB14 and TB15) can be put into one 256-entry memory. By using the 2-level memory hierarchy method, we can save memory access at least 50~60% and most 80~95%. Also the VLD decoder can achieve programmability so that it can be used in multi-mode video decoding applications.

REFERENCES

- [1] ITU-T Rec.H.264/ISO/EC 11496-10, "Advanced Video Coding", Final Committee Draft, Document JVT-E022, September 2002.
- [2] International Standards Organizations, ISO/IEC 13818-2, MPEG-2 Standard Document.
- [3] Y. H. Moon, "A New Coeff_Token Decoding Method with Efficient Memory Access in H.264/AVC Video Coding Standard," IEEE Trans. Circuit and System for Video Technology, Vol. 17, no. 6, pp.729-736, June 2007.
- [4] Y. H. Kim, Y. J. Yoo, J. Shin, B. Choi, and J. Paik, "Memory Efficient H.264/AVC CAVLC for Fast Decoding," IEEE Trans. Consumer Electron. , Vol. 52, No. 3, pp.943 - 952, Aug. 2006.
- [5] H. C. Chang, C. C. Lin, and J. I. Guo, "A Novel Low-cost High-performance VLSI Architecture for MPEG-4 AVC/H.264 CAVLC Decoding," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), 2005, pp. 6110-6113.
- [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. Consumer Electron. , vol. 51, no. 3, pp. 933-938, Aug. 2005.
- [7] B. J. Shieh, "Area Efficient and High Throughput Memory-Based VLC Codec Designs", Ph. D. dissertation, 2001
- [8] B. J. Shieh, Y. S. Lee, and C. Y. Lee, "A New Approach of Group-based VLC Codec System with Full Table Programmable," IEEE Trnas. Circuits and System for Video Technology, Vol. 11, no. 2, pp.210-221, Feb. 2001.
- [9] C.H. Liu, B. J. Shieh, and C.Y. Lee, "A Low-Power Group-Based VLD Design," . Proceedings of the 2004 International Symposium on Circuits and Systems, vol.2, pp.II 337- II 340, May 20, 2004
- [10] P.Y. Chen, Y.M. Lin, "A Low-Cost Implementation for MPEG-4", IEEE Trans. Circuits Syst. II, Express Briefs, vol. 54, no. 6, pp. 507-511, June 2007 (SCI).