|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  7th Meeting: Geneva, CH, 21-30 November, 2011 | Document: JCTVC-G659 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Study of Entropy Coding Methods Complexity** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Information | | |
| *Author(s) or Contact(s):* | Marta Karczewicz  In Suk Chong  Xianglin Wang Rajan Joshi  10160 Pacific Mesa Blvd San Diego, CA 92121 USA | Tel: Email: | 1-858-658-5673 [martak@qualcomm.com](mailto:martak@qualcomm.com)  1-858-845-5477 [ichong@qualcomm.com](mailto:ichong@qualcomm.com)  1-858-651-5135  [xianglin@qualcomm.com](mailto:xianglin@qualcomm.com)  1-858-658-4511  [rajanj@qualcomm.com](mailto:rajanj@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document summarizes the key issues in hardware implementation of CABAC and CAVLC decoding. Throughput numbers obtained for these method for H.264/AVC are quoted. Statistics of number of symbols and bins decoded per frame for the current test conditions are given.

# CABAC and CAVLC Implementation Challenges

While other processes in video decoding can be parallelized, the serial nature of entropy coders makes them a bottleneck for overall codec performance. For example, the bin-rate requirements for the H.264/AVC standard are: max bins per picture (Mbins) 9.2 (Level 4.0), 17.6 (Level 5.1). Peak bin rate (Mbins/sec) is 275 (Level 4.0), and 2107 (Level 5.1) [1]. Since even optimized CABAC decoders can usually decode ~1 bin per cycle, the hardware implementation have to run at high frequency to ensure real time decoding.

When implemented in hardware, the bin decoding procedure is realized by a series of elementary operations. In the first step, the context model is selected (CS) based on the context model index (ctxIdx) which may be a constant or determined by the value of previously decoded bins. After loading (CL) the context model (state and MPS value) assigned to ctxIdx from the context model memory the binary arithmetic decoder (BAD) computes the bin value and a new context model. The binarization matching operation evaluates the bin string constructed so far and decides whether the syntax element was completely decoded or not. At the same time, updated context model is stored to the memory.

To optimize performance of CABAC decoding, the most effective way is to apply pipeline scheme. During the decoding, four sequential memory accesses are required: (1) reading neighboring information necessary to calculate context model index, (2) reading context model, (3) reading the value of codIRangeLPS from table rangeTabLPS, and (4) writing updated context model into context model memory. Since every memory access need a clock edge, four pipeline stages are necessary. The 4-stage pipeline architecture of CABAC decoder is shown in Fig 1, in which each stage contains at least one memory access.



Figure 1: Pipeline stages.

One of the problems with the initial pipeline arrangement in Fig 1 is the structural hazard caused by CL and CU accessing the same memory. To resolve this conflict a context model reservoir (CMR) can be used in which cluster of contexts models used for current syntax element (SE )is pre-fetched from SRAM and placed in the register bank. The CS and CL operations can now be processed at the same time. Similarly BAD, CU and BM can be also take place in a single cycle (Figure 2). The stalls are now reduced to CMR update during SE switching.



Figure 2: Pipeline stages with CMR and data forwarding.

The remaining issue is the data dependency which occurs when the value of CM for one bin dependents on the previous bin value, e.g.:

* SE is represented by variable length codeword. E.g., in case of last\_significant\_coefficient\_pos for 32x32 blocks, the length of the unary part of the codeword ranges from 1 to 16 bins. Hence, while decoding a bin of last\_significant\_coefficient\_pos, whether the bin is the last one of last\_significant\_coefficient\_pos or not is unknown until we obtain this bin value.
* The decoded bin value decides which SE is decoded next, e.g., cbf\_luma.
* Two consecutive bins share same context, e.g., bins representing significant\_coeff\_flag, last\_significant\_coefficient\_pos or coeff\_abs\_level\_greater1\_flag.

To solve this problem data forwarding can be used. During context selection for the bin for which the bin bini+1 contexts indices are estimated for the case when the decoded value of the previous bin bini is 0 and for the case when the decoded value of this bin is 1. Then when the bin bini is decoded one of the two context models is selected.

To decode two bins per cycle two CABAC engines can be concatenated. In this case up to 12 contexts models may have to be pre-fetched for bins bini and bini+1:

* For bin bini, 4 combinations for possible values of the previous bins bini-2, bini-1.
* For bin bini+1, 8 combinations for possible values of the previous bins bini-2, bini-1 and bini.

Hence the multi-bin processing increases the complexity of the architecture and increases the critical path delay.

When using pipeline approach the achieved throughput can be close to 1 bin/cycle [2]. When multi-bin processing is used in addition the reported throughput is over 1 bin/cycle for regular bins and 2 bins/cycle for bypass bins.

VLC codeword can be decoded by parallel pattern matching based on logical functions, which guarantees decoding at least one codeword per cycle. Regular-constructed VLCs can be decoded using with arithmetic operations utilizing the properties of codeword table, e.g., leading 0s. For short codewords multiple-symbol decoding scheme can be employed. Hence the average throughput reported for CAVLC decoder can be over 1 SE/cycle [3].

# HEVC Statistics

Since the throughput for CAVLC is measure in SE/cycle, and in bin/cycle in the case of CABAC, to have an approximation of expected complexity we measured average numbers of bins (CABAC) and symbols per frame (CAVLC) decoded for 1080p sequences. Low complexity intra and low delay (P) configurations were used with entropy coding method set to CAVLC or CABAC. The results are summarized in table below. In case of CAVLC coding sign is counted as a symbol when coefficients are coded in run mode. In level mode, level and its sign are treated as one symbol. The ration of regular bins to symbols is on average over 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Intra | | | Inter | | |
| Sequence | QP | CAVLC symbols/fr. | CABAC bins/fr. | | CAVLC symbols/fr. | CABAC bins/fr. | |
| Regular | Bypass | Regular | Bypass |
| Kimono | 22 | 271121 | 642902 | 285283 | 114802 | 268063 | 82724 |
| 27 | 160868 | 367151 | 153246 | 59345 | 132905 | 36349 |
| 32 | 103131 | 230068 | 81044 | 31993 | 68389 | 15941 |
| 37 | 67412 | 144688 | 41521 | 17446 | 36152 | 7049 |
| Cactus | 22 | 593858 | 1420536 | 347116 | 181118 | 515191 | 102978 |
| 27 | 252704 | 658060 | 127757 | 66590 | 179668 | 32778 |
| 32 | 134549 | 355699 | 59050 | 32952 | 80454 | 14089 |
| 37 | 79375 | 201297 | 29073 | 17966 | 39574 | 6657 |
| ParkScene | 22 | 1185128 | 2726517 | 979889 | 438365 | 1176039 | 255736 |
| 27 | 623004 | 1422879 | 409113 | 92087 | 245289 | 39066 |
| 32 | 363571 | 809531 | 184078 | 27413 | 65280 | 9089 |
| 37 | 215397 | 489084 | 85163 | 10699 | 23104 | 2924 |
| Basketball | 22 | 913242 | 2130234 | 628704 | 212811 | 554641 | 107531 |
| 27 | 458075 | 961656 | 268529 | 65462 | 142495 | 27584 |
| 32 | 264551 | 557817 | 124216 | 31082 | 64498 | 10828 |
| 37 | 152051 | 323713 | 56975 | 16168 | 32870 | 4757 |
| BQSquare | 22 | 930674 | 1981494 | 684912 | 184776 | 411752 | 83899 |
| 27 | 540617 | 1149231 | 323101 | 77353 | 163917 | 30119 |
| 32 | 297857 | 648935 | 146646 | 34411 | 70499 | 11453 |
| 37 | 151700 | 341241 | 64788 | 15723 | 31546 | 4615 |

# References

[1] Sze, V., and A.P. Chandrakasan. “A High Throughput CABAC Algorithm Using Syntax Element Partitioning,” in *Proceedings IEEE International Conference on Image Processing (ICIP)*, pp. 773-776, Nov 2009.

[2] Y. Jan and L. Jozwiak, **“**CABAC Accelerator Architectures for Video Compression in Future Multimedia : A Survey,” in *Proceedings 9th international Workshop on Embedded Computer Systems : Architectures, Modeling, and Simulation,* SAMOS 2009, July 2009.

[3] Xiaohua Tian, Thinh M. Le, Yong Lian, "Entropy Coders of the H.264/AVC Standard: Algorithms and VLSI Architectures," Springer 2010.