MEMORY COST VS. CODING EFFICIENCY TRADE-OFFS FOR HEVC MOTION ESTIMATION ENGINE

*Mahmut E. Sinangil**
Anantha P. Chandrakasan
Massachusetts Institute of Technology
Cambridge MA

Vivienne Sze
Minhua Zhou
Texas Instruments Inc.
Dallas TX

ABSTRACT

This paper presents a comparison between various High Efficiency Video Coding (HEVC) motion estimation configurations in terms of coding efficiency and memory cost in hardware. An HEVC motion estimation hardware model that is suitable to implement HEVC reference software (HM) search algorithm is created and memory area and data bandwidth requirements are calculated based on this model. 11 different motion estimation configurations are considered. Supporting smaller block sizes is shown to impose significant memory cost in hardware although the coding gain achieved through supporting them is relatively smaller. Hence, depending on target encoder specifications, the decision can be made not to support certain block sizes. Specifically, supporting only 64x64, 32x32 and 16x16 block sizes provide 3.2X on-chip memory area, 26X on-chip bandwidth and 12.5X off-chip bandwidth savings at the expense of 12% bit-rate increase when compared to the anchor configuration supporting all block sizes.

Index Terms— HEVC, motion estimation, reference buffer, memory area, memory bandwidth

1. INTRODUCTION

High-Efficiency Video Coding (HEVC) is a new video compression standard currently being standardized by the JCT-VC (joint collaborative team on video coding) established by ISO/IEO MPEG and ITU-T [1]. HEVC targets 50% coding gain over AVC/H.264 High Profile. In order to achieve this goal, new tools are being adopted to HEVC. However, it is important to consider the memory cost (in terms of area and data bandwidth) of these tools in hardware.

Larger memory area and bandwidth generally results in higher power consumption and this can be a limiting factor in various applications such as portable multimedia and mobile devices. Moreover, supporting higher bandwidth necessitates more complex circuit solutions at the system level which is not desirable due to increased manufacturing cost. Hence, it provides valuable insight to have a memory size and bandwidth vs. coding efficiency trade-off for different tools and different blocks to make high-level design decisions.

One of the main differences of HEVC from previous video compression standards is the coding tree structure. In this structure, a frame is first divided into LCUs (largest coding units). Then an LCU is further divided into CUs (coding units) in a quad-tree structure. Unless a CU is further divided into four smaller CUs, it is predicted with one of several PU (prediction unit) types. Currently, LCU size can be as large as 64x64 and SCU (smallest CU) can be as small as 8x8.

For inter prediction, there are many different PU types. 2Nx2N, 2NxN, Nx2N and NxN are main PU types where 2Nx2N corresponds to the size of the CU. If asymmetric motion partitions (AMP) are used, non-square PUs for inter prediction also include 2NxnU, 2NxnD, nLx2N and nRx2N. NxN is only allowed at the SCU level.

At the encoder side, decisions can be made to support only a subset of these PU types and CU sizes. This will present a trade-off between hardware complexity and coding efficiency. Since motion estimation is one of the most critical blocks in video encoders, trade-offs in motion estimation (ME) block are very important for the overall encoder design.

This paper presents an analysis that compares various HEVC ME configurations supporting different block sizes based on memory cost in hardware and compression efficiency trade-offs. The rest of this paper is organized as follows: Section 2 presents HEVC ME architecture suitable for HEVC quad-tree structure and HEVC reference software (HM) search algorithm. Section 3 provides details about calculation of on-chip memory area and data bandwidth estimates. Lastly, Section 4 presents comparison between various ME configurations in terms of coding efficiency, memory area and bandwidth.

2. HEVC MOTION ESTIMATION ARCHITECTURE

HM is implemented to achieve highest coding efficiency. Accordingly, in motion estimation, motion cost associated with every possible CU sizes and PU types are calculated to find the best combination that provides the smallest cost. It should

*Funding provided by Texas Instruments Inc.

be emphasized that these searches for different CU sizes and PU types are performed around different and independent centers. Moreover, the processing order is sequential in HM so exact motion information can be used in Advanced motion vector prediction (AMVP) calculations which depend on the motion of neighboring blocks.

In hardware implementation, various simplifications are often considered and implemented [2, 3]. For example, search algorithm are designed to allow search ranges and cost calculations to be shared across different block sizes to reduce hardware complexity [4, 5]. However, these simplifications cause coding loss. In this work, hardware implementation of an architecture suitable for HM's search algorithm is considered to quantify the hardware complexity and cost of this algorithm and the quad-tree coding structure of HEVC. In hardware, this algorithm requires separate and independent engines performing motion search for different block sizes. Block sizes are determined by the corresponding CU sizes and PU types. Fig. 1 shows an HEVC motion estimation engine architecture supporting all block sizes from 64x64 down to 4x4 except AMP partitions. The architecture can be generalized to cover AMP partitions as well.

There are a total of 13 engines in the architecture in Fig. 1: Three engines for each CU size except for the 8x8 CU where there is a fourth engine to support NxN (4x4) partition.

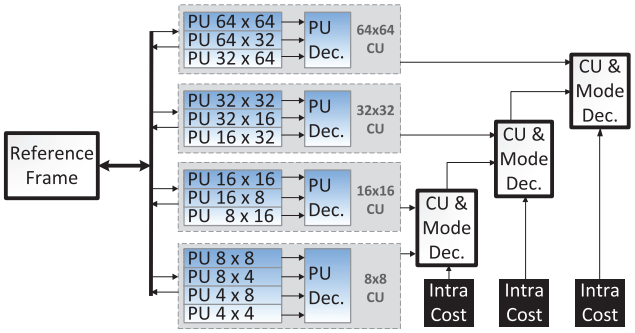

Fig. 1. Architecture of an HEVC motion estimation module supporting all block sizes from 64x64 down to 4x4 except AMP partitions.

The processing order for one LCU is shown in Fig. 2. Motion searches are performed for four 4x4 blocks, two 8x4 and 4x8 blocks and one 8x8 block. Then a PU decision is done to decide what PU type provides the smallest cost for the first 8x8 CU. Similarly, three more 8x8 CUs are processed sequentially and their costs are output to *CU & Mode Decision* block. During this time, PU decision for the first 16x16 CU is also finished and a decision can be done for the first 16x16 CU. This continues until an entire LCU is processed by all engines.

It is important to note that, for a fixed throughput constraint, cycle budget to process a smaller block size is tighter. Hence, data bandwidth requirements can be significantly larger for smaller block sizes compared to larger block sizes.

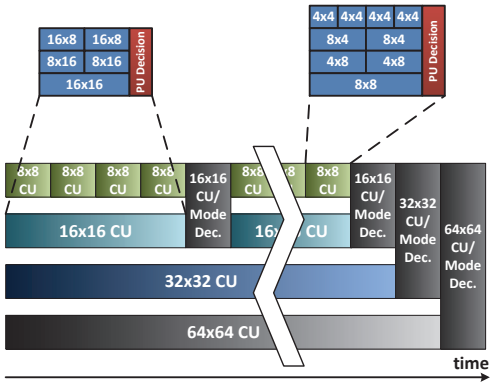

Fig. 2. Operation of the motion estimation module shown in Fig. 1.

Fig. 3 shows the block diagram of each engine. On-chip *Reference Pixel Buffer* holds pixel data for motion search. *AMVP* block calculates the advanced motion vector predictor list and integer motion estimation (IME) is performed around the best AMVP candidate. Fractional motion vector refinement is performed around the best motion vector found in IME. Lastly, *Merge/Skip Estimation* calculate costs associated with the merge and skip modes. At the end, costs and motion vectors are output to higher level control and decision logic.

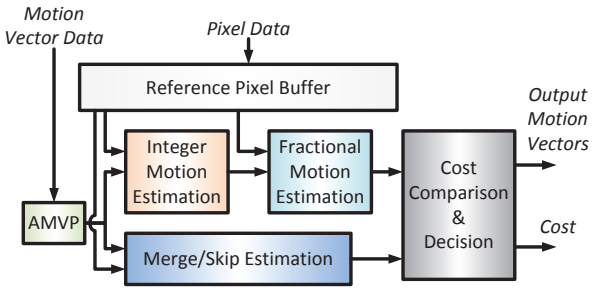

Fig. 3. Block diagram of one of the engines from Fig. 1. Each engine has a separate reference pixel buffer to perform an independent search.

3. HARDWARE COST ESTIMATE OF HEVC MOTION ESTIMATION ENGINES

3.1. On-Chip Reference Buffer Size

As explained in Section 2, each motion estimation engine in Fig. 1 is performing independent searches and for each engine, a separate memory is necessary in each direction (forward and backward) and for each reference frame. Table 1

Block Size	On-Chip Mem. Size	Block Size	On-Chip Mem. Size
64x64	39KB	16x8	21KB
64x32	33KB	8x16	21KB
32x64	33KB	8x8	20KB
32x32	28KB	8x4	20KB
32x16	25KB	4x8	20KB
16x32	25KB	4x4	19KB
16x16	23KB		

Table 1. On-chip reference buffer size needed for each engine to support a search range of ± 64.

shows the size of on-chip memory needed to support ± 64 search range. Extra pixels are necessary for pixel interpolation in fractional motion estimation and they are included in calculations.

A total of 0.65MB of on-chip memories are necessary to support a single reference frame in forward and backward directions for the entire motion estimation module in Fig. 1. This number heavily depends on the selected search range size. To quantify the effect of search range size on the compression efficiency, simulations in HM-3.0 are performed under the test conditions defined in [1] and results are provided in Table 2. From ± 64 to ± 16, bit-rate increase is 0.1%, 0.1% and 3.5% in low delay, low delay with P and random access test conditions. It should be noted that HM searches through all possible combinations during encoding and also implements a highly-complex search algorithm. In a practical hardware implementation, the coding loss due to reduced search range size can be expected to be significantly larger.

It should also be noted that on-chip memory size for small block sizes is not significantly lower than the size for larger block sizes (39KB for 64x64 and 19KB for 4x4) and smaller block sizes do not provide a significant advantage in terms of memory size.

Additional on-chip storage (e.g. line buffers for motion information) can be necessary for *AMVP* and *Merge/Skip* but the size heavily depends on the specific implementation and the target resolution. Moreover, these buffers can be shared across parallel engines. For this work, on-chip line buffers are considered for motion information of the top line in forward and backward directions. For a 4Kx2K video encoder, the amount of storage is estimated to be 0.03MB.

Search Range	Low Delay	Low Delay & P	Random Access
± 64	0%	0%	0%
± 32	0%	0%	0.8%
± 16	0.1%	0.1%	3.5%

Table 2. Effect of search range window size on coding efficiency. Increases in bit-rate are given with respect to HM-3.0. Single reference frames in both directions are used.

3.2. On- and Off-Chip Bandwidth

The second consideration in this section is about on- and off-chip bandwidth as these numbers can be a limiting factor in practical implementations. On-chip bandwidth is determined by the size of reference buffer for each engine and how frequently it is accessed. For the search algorithm in HM, during IME, entire search range is accessed if the result of the initial search is not good enough. This occurs in the case of complex motion. To capture the worst-case upper limit, it can be assumed that the entire search range in the reference buffer is accessed for every block. On-chip bandwidth for FME is significantly smaller as only a refinement is done at this stage. Lastly, bandwidth for motion information of neighboring blocks that is necessary for *AMVP* and *Merge/Skip* candidate calculations is small compared to the on-chip bandwidth of the integer and fractional motion estimation.

Block Size	On-Chip BW	Off-Chip BW	Block Size	On-Chip BW	Off-Chip BW
64x64	2.2	1.49	16x8	39.6	13.72
64x32	3.8	1.86	8x16	39.6	10.33
32x64	3.8	1.48	8x8	75.6	17.47
32x32	6.4	3.64	8x4	145.9	30.21
32x16	11.5	6.05	4x8	145.9	22.94
16x32	11.5	5.20	4x4	283.8	36.92
16x16	20.9	7.62			

Table 3. On- and off-chip bandwidth requirement for each engine in Fig. 1 with a search range of ± 64 for 4Kx2K at 30fps. All numbers are in GB/s.

Off-chip bandwidth considered here is the off-chip memory's read bandwidth to bring reference pixel data from off-chip to the on-chip buffers for motion estimation. Similarly, off-chip bandwidth is determined by the size of the reference buffer and how frequently reference buffers for each engine need to be updated. Because of the correlation of motion between neighboring blocks, in the ideal case, data re-use between consecutive blocks can be close to 100%. However, it should be noted that the processing order of blocks in an LCU does not allow 100% data re-use and hence causes the same part of the reference window to be read multiple times. Increasing size of the on-chip buffer can improve the data re-use at the expense of larger on-chip memory area. In this work, minimum buffer sizes given in the previous sub-section are assumed in the bandwidth calculations. Pixel data to evaluate *AMVP* and *Merge/Skip* candidates might require additional bandwidth if these candidates are spatially far away from each other and do not fall into the search range in reference buffer.

Table 3 shows on- and off-chip bandwidth requirement for each engine. It should be noted that small block sizes such as 4x4 requires a very large on-chip and off-chip bandwidth compared to larger block sizes and imposes a higher cost for hardware implementation.

	Configuration #										
	1	2	3	4	5	6	7	8	9	10	11
64x64	Y	Y	Y	Y	Y	Y	Y	N	N	N	N
64x32	Y	Y	N	Y	N	Y	N	N	N	N	N
32x64	Y	Y	N	Y	N	Y	N	N	N	N	N
32x32	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
32x16	Y	Y	N	Y	N	N	N	Y	N	N	N
16x32	Y	Y	N	Y	N	N	N	Y	N	N	N
16x16	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
16x8	Y	Y	N	N	N	N	N	Y	N	Y	N
8x16	Y	Y	N	N	N	N	N	Y	N	Y	N
8x8	Y	Y	Y	N	N	N	N	Y	Y	Y	Y
8x4	Y	N	N	N	N	N	N	N	N	N	N
4x8	Y	N	N	N	N	N	N	N	N	N	N
4x4	Y	N	N	N	N	N	N	N	N	N	N
Ref. Buffer Size (KB)	680	565	248	439	208	234	163	356	170	201	115
On-Chip BW (GB/s)	1581	429	209	121	59	32.5	17.3	409	205	351	192
Off-Chip BW (GB/s)	159	69	30.2	27.4	12.7	8.5	5.1	64	28.7	49.1	25.1
Bit-Rate Increase (%)	0	2	3	12	12	34	34	3	4	7	11

Fig. 4. Hardware cost vs. coding efficiency comparison table for 11 different motion estimation configurations. "Y" and "N" represents if a block size is supported or not respectively.

4. COMPARISON OF MOTION ESTIMATION CONFIGURATIONS

Fig. 4 shows memory cost and coding efficiency results for 11 different configurations. Each column corresponds to a different configuration supporting all or some of the available block sizes. Configuration #1 supports all block sizes and is the anchor configuration for this work. HM-3.2 simulations are performed to quantify coding loss for each configuration. The bit-rate increase in Fig. 4 is given as the average of the numbers from all-intra, low-delay, low-delay P and random-access common test conditions defined by JCT-VC [1].

Fig. 5 plots off-chip bandwidth savings vs. bit-rate increase, with respect to the anchor. Each configuration is denoted by a dot on this figure except for the anchor configuration as it would be at the bottom left corner of the plot. The slope of the lines connecting each configuration to the bottom-left corner provides a graphical method to compare how efficient each configuration is. A smaller slope means that more area savings can be achieved with smaller bit-rate increase (coding loss).

It can be observed from Fig. 4 and Fig. 5 that configurations supporting smaller block sizes such as 4x4 require largest area and bandwidth although the coding gain achieved through supporting them is relatively smaller. In other words, not supporting smaller partitions has a smaller effect on coding efficiency although these engines contribute significantly to bandwidth and area. For example, by removing 4x4, 4x8 and 8x4 block sizes in configuration #2, 17% memory area, 3.7X on-chip bandwidth and 2.3X off-chip bandwidth can be saved at the expense of only 2% coding loss.

On-chip reference buffer size mainly depends on the search range and block size. However, from smaller to larger

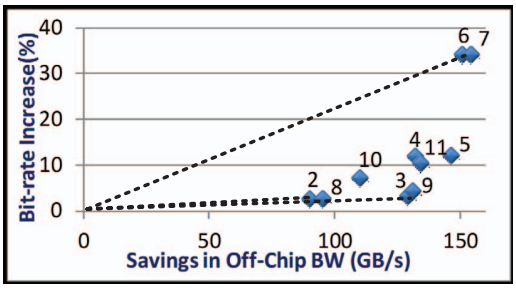

Fig. 5. Scatter plots showing off-chip bandwidth savings vs. bit-rate increase, with respect to the anchor configuration.

block sizes, the increase in memory size is not very significant. In terms of memory bandwidth, small block sizes, especially smaller than 8x8, impose very high bandwidth requirements. If savings are necessary due to system level restrictions for bandwidth, small block sizes can be chosen not to be supported.

5. CONCLUSION

This paper presents a comparison for various HEVC ME configurations in terms of memory cost in hardware and coding efficiency. On-chip buffer size, on-chip bandwidth and off-chip bandwidth estimations are calculated based on the reference implementation provided in HM.

Final decision on supported block sizes depends on the area and bandwidth limitations as well as coding efficiency specifications of the target encoder. Since larger area and higher bandwidth often results in higher power consumption, battery-powered applications might trade-off some of the coding efficiency for lower power consumption. If coding efficiency has the highest priority, all block sizes can be supported (configuration #1) although this might lead to a significant area and power consumption. If area and power are critical, configuration #5 and configuration #7 can be efficient solutions.

6. REFERENCES

[1] "Joint Call for Proposals on Video Compression Technology," ITU-T SG16/Q6, 39th VCEG Meeting: Kyoto, 17-22 Jan. 2010, Doc. VCEG-AM91.

[2] Yu-Kun Lin et al., "A 242mW $10mm^2$ 1080p H.264/AVC High-Profile Encoder Chip," in *IEEE ISSCC*, Feb. 2008, pp. 314 –615.

[3] Vivienne Sze et al., "A 0.7-V 1.8-mW H.264/AVC 720p Video Decoder," *IEEE JSSCC*, vol. 44, no. 11, pp. 2943 –2956, Nov. 2009.

[4] Tung-Chien Chen et al., "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder," *IEEE TCSVT*, vol. 16, no. 6, pp. 673 – 688, June 2006.

[5] Tsung-Han Tsai et al., "High Efficiency Architecture Design of Real-Time QFHD for H.264/AVC Fast Block Motion Estimation," *IEEE TCSVT*, vol. 21, no. 11, pp. 1646 –1658, Nov. 2011.