HARDWARE ARCHITECTURE DESIGN FOR VARIABLE BLOCK SIZE MOTION ESTIMATION IN MPEG-4 AVC/JVT/ITU-T H.264

Yu-Wen Huang, Tu-Chih Wang, Bing-Yu Hsieh, and Liang-Gee Chen

DSP/IC Design Lab
Graduate Institute of Electronics Engineering and
Department of Electrical Engineering, National Taiwan University
{yuwen, eric, bingyu, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

Variable block size motion estimation is adopted in the new video coding standard, MPEG-4 AVC/JVT/ITU-T H.264, due to its superior performance compared to the advanced prediction mode in MPEG-4 and H.263+. In this paper, we modified the reference software in a hardware-friendly way. Our main idea is to convert the sequential processing of each 8x8 sub-partition of a macro-block into parallel processing without sacrifice of video quality. Based on our algorithm, we proposed a new hardware architecture for variable block size motion estimation with full search at integer-pixel accuracy. The features of our design are 2-D processing element array with 1-D data broadcasting and 1-D partial result reuse, parallel adder tree, memory interleaving scheme, and high utilization. Simulation shows that our chip can achieve real-time applications under the operating frequency of 64.11MHz for 720x480 frame at 30 Hz with search range of [-24, +23] in horizontal direction and [-16, +15] in vertical direction, which requires the computation power of more than 50 GOPS.

1. INTRODUCTION

Video coding experts from ISO MPEG-4 Advanced Video Coding (AVC) and ITU-T H.264 group form the Joint Video Team (JVT). The new techniques include motion estimation (ME) with variable block sizes and multiple reference frames, intra prediction, 2x2 and 4x4 transform, adaptive block size transform, non-uniform quantization, CAVLC, CABAC, in-loop deblocking filter, and more [1]. Compared to MPEG-4 advanced simple profile, up to 50% of bit-rate reduction can be achieved. However, the required computation is more than four times higher. Therefore, hardware acceleration is a must for real-time applications, especially for ME, which is the most computationally intensive part.

Many ME architectures have been proposed for previous standards. Only one 16x16 block and four 8x8 blocks (advanced prediction mode in MPEG-4 simple profile and H.263+) could be used for motion compensation. They cannot fully support the seven kinds of block size (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4) in H.264. In addition, the reference software of H.264 [2] adopts sequential processing of each 8x8 sub-partition. The data dependency between the sub-partitions makes parallel processing impossible. Thus, we must start the architecture design of variable block size ME for H.264 at the algorithmic level.

In this paper, we modify the reference software to let the algorithm more suitable for hardware. In Section 2, we first review the prediction flow in JM4.0d. In Section 3, we present our algorithm with experimental results of coding performance. Next, our architecture, as well as the simulation results and comparison, will be described in Section 4 and Section 5, respectively. Finally, Section 6 gives a conclusion.

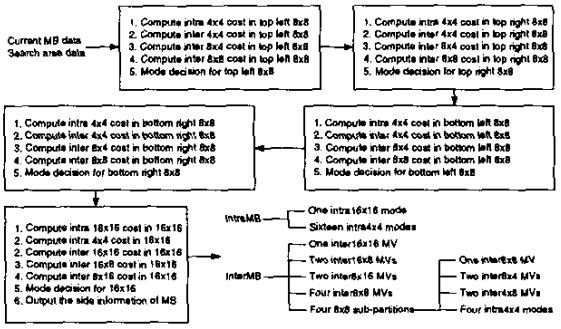

Figure 1: Prediction flow in H.264 software.

2. PREDICTION FLOW IN H.264 SOFTWARE

Figure 1 shows the prediction flow of a 16x16 macro-block (MB) in H.264 software. The top left 8x8 block is first processed and followed by the top right 8x8, bottom left 8x8, bottom right 8x8, and 16x16. The mode decision considers not only the sum of absolute difference (SAD) (2-D Hadamard transformed for intra modes and sub-pixel ME) but also the exact cost of side information. The entropy coding of intra modes depends on the context produced by the left and top neighbors. Besides, the intra prediction of a block cannot get the correct predictor until the neighboring blocks are quantized and then reconstructed. Moreover, the motion vectors (MVs) are medium predicted by the left, top, and top right neighbors. The cost function can be computed only after the modes of neighboring blocks are determined. Obviously, the methods in H.264 software cannot be used in hardware implementation because of the inevitable sequential processing resulted from the data dependency of neighboring blocks. The main problem comes from the "exact" cost of side information. In fact, it is the SAD, not the cost of side information, that dominates the total cost. It is not necessary to exactly calculate the cost of side information.

0-7803-7761-3/03/$17.00 ©2003 IEEE

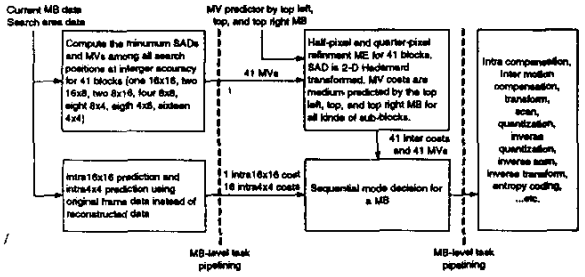

Figure 2: Modified prediction flow.

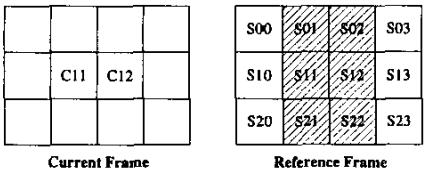

Figure 3: Overlapped search range of adjacent MBs. Each square is 16x16 and assume search range is [-16, +16].

3. PROPOSED HARDWARE-ORIENTED ALGORITHM

Figure 2 shows the proposed hardware-oriented prediction flow. Full search ME is performed at integer search positions and sub-pixel refinement is next executed. The search range center is determined by the MV predictor in H.264 software, but we use (0, 0) as center in order to share the overlapped search area of adjacent MBs and reduce the memory transfer from external RAM to on-chip SRAM, as shown in Fig. 3. During integer-pixel ME, we compute the SAD of 41 blocks without MV cost. Next, the sub-pixel refinement is performed around the best integer search position of 41 blocks. At the refinement stage, the MV cost is considered. However, we do not use the exact cost but an approximation. The exact MV predictor is replaced by the medium of the MVs of the top left, top, and top right MB for all kinds of sub-blocks, as shown in Fig. 4. For example, the exact MV cost of the 4x4 C22 block is related to the MVs of C12, C13, and C21. During the ME phase, we change the MV predictors of all the 41 blocks to the medium of MV0, MV1, and MV2 in order to facilitate the parallel processing of the 41 blocks. Of course after the mode decision is finished, the entropy coding module must calculate the exact MV predictors in the sequential order defined by standard, but this will not cause the processing bottleneck. As for the intra prediction, we use original frame data, instead of the reconstructed pixels of neighboring blocks, as predictors. At high bit-rates, the original frame pixels are very close to the reconstructed pixels, so the mode decision is still correct. At low bit-rates, the differences may become significant. We proposed an error term to model the differences so that the mode decision will not go wrong. The readers can refer to [3] for more details.

The experimental results of our modifications are shown in Fig. 5 The test conditions are I-P-P-P-P-P-P..., one reference frame, CAVLC, low-complexity mode decision, search range [-16, +16],

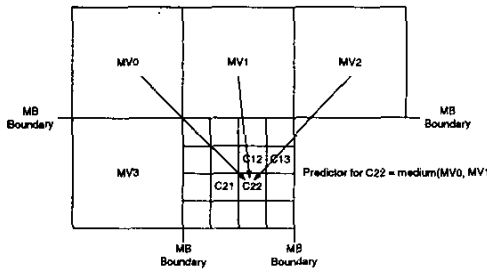

Figure 4: MV predictor used for the 41 blocks in current MB during the ME phase.

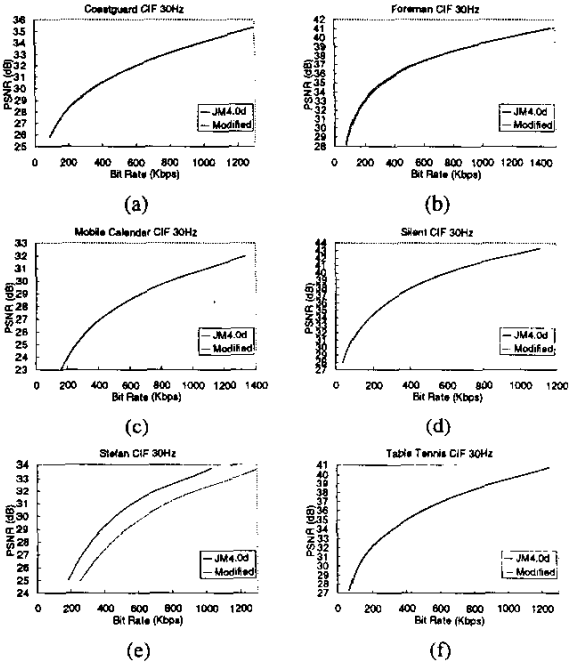

Figure 5: The rate distortion curves of various standard sequences generated by JM4.0d and our modified software.

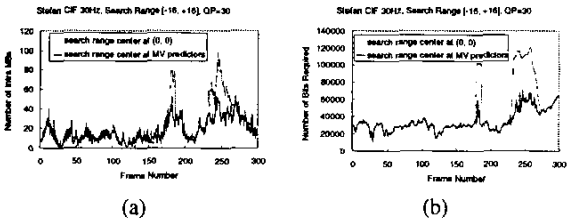

Figure 6: (a) Number of intra MBs v.s. frame; (b) Number of bits required v.s. frame.

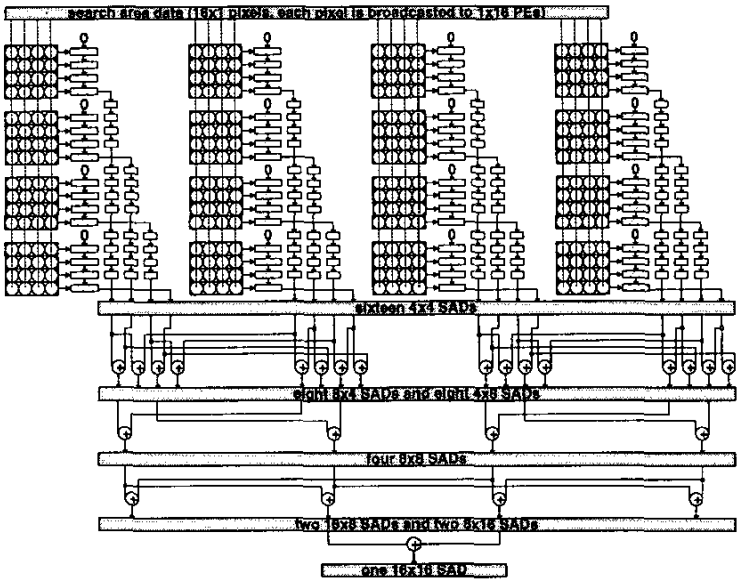

Figure 7: Proposed architecture for integer ME.

and Hadamard transform opened. Six standard sequences are tested from low bit-rates to high bit-rates. Our modifications result in almost no PSNR degradation except for Stefan. This is because the true motion vectors of Stefan from frame #175 to #195 and from frame #230 to #280 are larger than the search range. Here, we can see the significant gain when the search range center of a MB is located at the MV predictor. Nevertheless, as we stated before, if we located the search range center at (0, 0), take search range of [-16, +16] as an example, the memory transfer of the overlapped 2/3 search area from the external RAM to on-chip SRAM can be saved for adjacent MBs. Therefore, in the view point of hardware engineers, we suggest to increase the search range to [-24, +24] or larger if the extra cost of on-chip SRAM is affordable. Note that in this case, the saving of system bus bandwidth is 75% because there are 3/4 overlapped search area for two adjacent MBs. If the design target is low cost, we suggest to solve this problem at the algorithmic level. That is, we can adjust the search range center by the number of intra MBs in the previous frame. If the true MVs are beyond the search range, many MBs will be intra-coded. We can observe this situation in Fig. 6. If the number of intra MBs in previous frame is smaller than a threshold, we can use (0, 0) as search range center and reuse the overlapped search area. Otherwise, we can reload the whole search area that are centered at the MV predictor if the real-time requirements still can be achieved.

4. HARDWARE ARCHITECTURE

In this section, we will describe our architecture for the integer ME module, which is the most computationally intensive part in the encoder. Full search scheme is adopted because of its simplicity and regularity. The SADs of the 41 blocks in a search position are computed in parallel in one clock cycle. Our architecture is illustrated in Fig. 7. Each circle stands for a processing element (PE). Each 8-bit pixel in the current MB is stored in the corresponding PE. In each cycle, 16x1 8-bit search area pixels are inputted, and each pixel is broadcasted to 1x16 PEs. Every PE computes the absolute difference between the current MB pixel and the search area pixel. Each rectangle that contains 4x1 PEs is responsible for calculating the sum of 4x1 absolute differences. The 4x1 sum is then passed to the right. The rectangle shaded with slash lines adds the 4x1 sum from its left side and the result from top. The added result is latched in the register. The rest small rectangles are registers and are used as delay lines. In this way, the sixteen 4x4 SADs can be computed in one cycle. The eight 8x4 SADs, eight 4x8 SADs, four 8x8 SADs, two 16x8 SADs, two 8x16 SADs, and one 16x16 SAD can also be easily computed in the same cycle by an adder tree.

An example of the detailed data flow is shown in Fig. 8. Assume the search range is [-16, +16], the search area is 48x48, and each square stands for 16x16 search area. After the current MB are loaded into each PE, 16x1 search area pixels are inputted in each cycle. At cycle 0, the left 16x1 pixels on the row 0 in the search area are inputted. Then, the left 16x1 pixels on row 1 are inputted at cycle 1, and go on. At cycle 15, all the candidate block data of search position (-16, -16) have been broadcasted to the PE array, and the SADs of 41 blocks at search position (-16, -16) are calculated in parallel. The SADs of search position (-16, -15) - (-16, +16) will be available at cycle 16 to 47, respectively. The rest search positions can be traced by analogy. In order to output the 16x1 pixels in parallel, we have to store the search area data in different 16 on-chip SRAM modules.

As you can see, the utilization of the previous data flow cannot achieve 100%. When the search position is changed in the horizontal direction, extra 15 cycles are required to load the candidate block data. The advanced data flow is shown in Fig. 9. Now the PE array needs two ports of 16x1 search area data. The top 48x32 search area data are stored in 16 on-chip SRAM modules, and the bottom 48x16 search area are stored in another 16 on-chip SRAM modules so that 32 search area pixels can be outputted in one cycle.

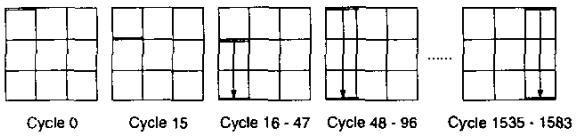

Figure 8: Basic data flow.

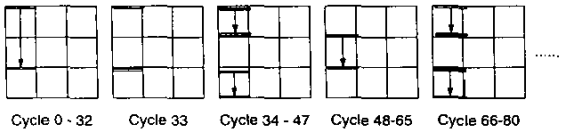

Figure 9: Advanced data flow.

At cycle 0 to 32, the new data flow is the same as the previous one. At cycle 33, not only the original 16x1 pixels in Fig. 8 but also the additional 16x1 pixels (at cycle 48 of the previous flow) are inputted to the PE array. At this time, the 16x1 PEs on row 0 (the top 16x1 PEs in Fig. 7) should choose the new additional 16x1 pixels while the rest 16x15 PEs on row 1-15 should choose the original 16x1 data. At cycle 34 to 47, the SADs of search positions (-16, +2) - (-16, +16) are computed, and in the mean time, the search area data required for search position (-15, -16) are also inputted to the PE array. During cycle 33-47, the 16x1 PEs on each row should select the proper 16x1 search area data in order to get the correct results. In this way, when the search position is changed in the horizontal position, no bubble cycles exist so that the throughput can be increased. When ping-pong mode current MB buffers are implemented, the utilization is 100%, and the required cycles for each MB is 1089. In sum, our architecture is a 2-D systolic array with 1-D data broadcasting and 1-D partial result reuse (4x1 SADs). Sixteen 4x4 SADs are generated by the PE array, and a parallel adder tree computes the rest SADs of larger block sizes.

5. IMPLEMENTATION

Our design goal is listed as follows: 720x480 frame size, 30 frames per second, search range [-24, +23] in the horizontal direction and [-16, +15] in the vertical direction. Our design is described by Verilog HDL and synthesized by SYNOPSYS Design Analyzer with AVANT! 0.35um 1P4M cell library. The backend tool we used is CADENCE Silicon Ensemble.

Figure 10 shows our chip layout. We did not use ping-pong mode current MB buffers in order to save chip area. Therefore, some extra cycles are needed to load the current MB data. The utilization of the PE array is 97%, and the required frequency is 64.11MHz. The critical path constraint is set to 15ns. We use sixteen 128x8 SRAMs to store the upper 64x32 search area data and another sixteen 64x8 SRAMs to store the bottom 64x16 search area data. The chip specifications are shown in Table 1. The total gate count is about 106K. The PE array requires 64K gates, the current MB buffer requires 15K gates, and the rest gates are spent on the 41 comparators to find the minimum SADs, registers to hold the minimum SADs and corresponding MVs, and control circuits.

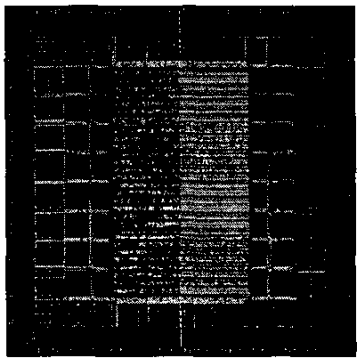

Figure 10: Chip layout.

Table 1: Chip Specifications.

Process	TSMC 1P4M 0.35um
Area	5,056 um × 5,056 um
Package	128 CQFP
On-chip memory	24,576 bits
Gate count	105,575
Maximum frequency	66.67 MHz
Power consumption	737.32 mW @ 66.67 MHz
Search range	horizontal [-24, +23], vertical [-16, +15]
Processing capability	720x480 @ 30 Hz

6. CONCLUSION

We proposed a hardware architecture of variable block size motion estimation dedicated for H.264. The architecture design begins with the analysis of the reference software. We removed the data dependencies that prevents parallel processing and MB pipelining. The architecture contains a PE array, which is a 2-D systolic array with 1-D data broadcasting and 1-D partial result reuse, and a parallel adder tree to generate the SAD of larger block sizes based on 4x4 SADs. Our memory access scheme can result in 100% utilization of PE array. Real-time applications of 720x480 frame at 30Hz can be achieved only under 64.11MHz.

7. FUTURE WORK

We are now establishing the cell-based design flow of 0.25um technology. We plan to integrate the current design together with quarter-pixel ME and intra prediction using 0.25 um technology.

8. REFERENCES

[1] *Committee Draft of Joint Video Specification (ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC)*, July, 2002.

[2] *Joint Video Team (JVT) software JM4.0d*, August, 2002.

[3] T.C. Wang, Y.W. Huang, H.C. Fang, and L.G. Chen, "Performance analysis of hardware oriented algorithm modifications in H.264," submitted to *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.