EFFICIENT IMPLEMENTATION OF VIDEO POST-PROCESSING ALGORITHMS ON THE BOPS PARALLEL ARCHITECTURE

Doina Petrescu

BOPS Inc., 6340 Quadrangle Dr. Suite 210, Chapel Hill, NC 27514

ABSTRACT

Deblocking and deringing are two video post-processing techniques largely used to remove coding artifacts and improve the visual quality when rendering low bit rate coded video. The algorithms used to achieve these tasks are computationally intensive and usually require high speed processors to be able to run in real time. Efficient implementations of signal adaptive filters for video post-processing can be obtained using the specialized features of the parallel BOPS® DSP cores. The performance achieved by deblocking and deringing CIF and SDTV size video sequences on the MANTA™ prototype chip are illustrated. It is shown that such complex tasks may be executed at low clock rates using the BOPS ManArray™ technology.

1. INTRODUCTION

In low bit rate coded video the quantization of DCT coefficients produces annoying artifacts in the decoded sequence. The blocking effect, which is the grid noise along block boundaries mainly visible in smooth areas, with low motion, and the ringing noise which shows along object borders, are such well-known artifacts. Signal adaptive filters are an efficient method to remove these artifacts, while preserving details which belong to the image. Traditionally, deblocking filters try to remove the unwanted boundaries between adjacent blocks by low-pass filtering applied to pixels on both sides of the block borders. However, this type of filtering may introduce undesirable blurring effects when applied to pixels which belong to real image edges. For this reason low-pass filtering is replaced with other types of filtering when local features indicate that a real edge is present. One method used to remove the ringing noise along object borders is to detect the edges in each frame, and apply a smoothing filter along these edges. The decision between edge and non-edge block borders relies on the assumption that real borders have a higher amplitude than edges produced by the quantization of DCT coefficients.

Signal adaptive filters for deblocking and deringing are included in the informative Annex F of the MPEG4 standard[1]. A deblocking filter similar to that in the standard is implemented on BOPS' parallel architecture. For deringing an original method is used, consisting of a 9-tap low-pass filter applied to an adaptive processing window. The filter window is initialized with the values in a 3x3 mask centered on the position whose output is computed. Then all values that are very different from the central one are replaced with the central value. In this way the proposed filter varies between 3x3 low-pass and identity, depending on how much the central value differs from its surrounding ones. The experiments performed over a group of low bit rate coded sequences show that the proposed filter achieves better visual quality than the result from the deringing filter in [1].

2. THE BOPS DSP CORES

BOPS®, Inc. develops and licenses high performance scalable and reusable digital signal processor (DSP) Intellectual Property (IP) cores enabling the shortest time from product concept to high-volume production of System-on-Chip (SOC) products for the Internet, multimedia and wireless communication markets.

The BOPS ManArray™ technology provides several levels of parallelism:

- the data level parallelism, which consists of the simultaneous processing of more than one value by an execution unit,
- the instruction level parallelism, which enables the simultaneous execution of 5 different instructions contained in a Very Long Instruction Word (VLIW) by the different execution units,
- the array level parallelism, which consists in the use of multiple processing elements (PEs) in the architecture, to obtain a linearly scalable performance by sharing the data to be processed between these PEs,
- the functional level parallelism, which relies on the powerful DMA engine that allows data communication and transfers between the DSP local memories and the SDRAM memory to take place simultaneous with the compute operations, thereby providing zero-latency transfers.

The building blocks of the BOPS cores are the Sequence Processor (SP), which achieves most of the control and decision functions, and the Processing Element (PE), which acts as a 'slave' resource to the SP and executes the tasks in parallel, in SIMD mode [2],[3]. Multiple core sizes may be obtained by combining the main building blocks and connecting them by an original Cluster Switch (CS) [2], as shown in Fig.2.

Each building block contains five execution units which can process independently and simultaneously 32 or 64-bit data. These are: the multiply-accumulate unit (MAC), the arithmetic-logic unit (ALU), a data select unit (DSU), a load unit (LOAD) and a store unit (STORE). The 5 instructions executed in parallel are stored in VLIW Instruction memory (VIM). Each VIM address contains five 32-bit instruction slots. Using Load VLIW (LV) the programmer can load individual instruction slots with 32-bit simplex instructions. The execute VLIW (XV) instruction triggers the execution of a VLIW at a specific VIM address, and the five instructions are executed in parallel.

Each block (PE or SP) has thirty-two 32-bit computation registers, which enable data types of 8, 16, 32 or 64 bits.

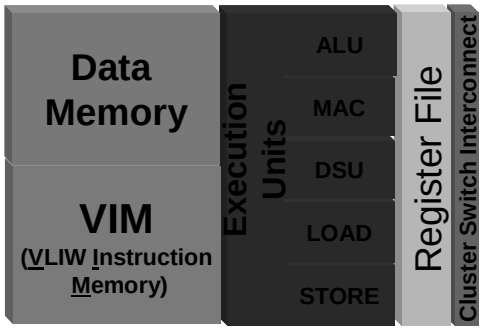

Fig. 1. The PE building block of BOPS cores

Execution units are capable of operating on data from one register or a pair of registers at a time. These 32 or 64 bits may be configured differently, as one 64-bit value, two 32-bit values, four 16-bit values, or eight 8-bit values and operations are performed in parallel on individual data for each type. The term used to refer to this data level parallelism is *packed data.* Special multiplication instructions are designed to enable fast computation for the linear filters. Such is the two-cycle *sum2p* instruction, which calculates the sum of two products as illustrated in Fig. 3. Using packed data 2 or 4 such operations may be performed simultaneously. Special instructions for the computation of local features and fast decisions and data selections needed in signal adaptive filtering are: the absolute difference instruction *absdif* in the ALU, which may calculate in one cycle the absolute differences for eight pairs of 8-bit data, and the copy selective instruction *copys* in the DSU which enables the selection of data using arithmetic flags set previously by comparisons.

The MANTA prototype chip is a proof-of-concept prototype SOC containing a standard implementation of a 2x2 fixed point and floating point core. The data memory on each PE is 16kBytes. Two 32 bit DMA lanes enable the simultaneous transfer of 8 bytes between SDRAM and the on-chip memories. MANTA operating in the 2x2 fixed point mode provides a platform for running the real time code described in this paper.

3. THE DEBLOCKING FILTER

3.1. Filter description

The deblocking filter implemented on the MANTA chip is similar to that in MPEG4[1]. Filtering is performed on both horizontal and vertical block borders. An 8-pixel decision window, perpendicular to the border, including equal number of pixels on both border sides is used to calculate local features and select the filter type and coefficients. The absolute differences between pairs of neighboring pixels are used as features. The feature values are compared against thresholds set as in [1], based on the quantization parameter (QP). High values of the absolute differences indicate the presence of a real image edge which needs to be preserved. When the indicate a smooth region, with no edges, a 7-tap low pass filter is used for calculating six values, three on each side of the border, for 'strong smoothing'. When an edge region is detected, but no abrupt change happens between the two neighboring pixels on each side of the border a 'weak filter' is applied, affecting

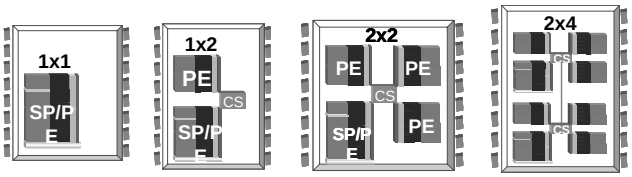

Fig. 2. The 1x1, 1x2, 2x2 and 2x4 BOPS cores building blocks

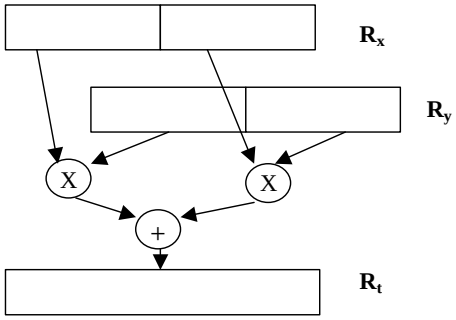

Fig.3 Building block of the *sum2p* instruction

only two border pixels (one for each block). No filtering is performed when a high absolute difference between the two pixels on block borders indicate the presence of an edge on that border.

3.2. Deblocking filter implementation on MANTA

The frame is divided into rectangular slices, which are separately processed by the four PEs working in parallel. The horizontal and vertical deblocking are achieved in two subsequent passes through the filtering procedure. In the first pass, data is filtered for vertical deblocking and stored as transposed w.r.t. the original order. In the second pass, the data is again filtered for vertical deblocking on the transposed order, which is equivalent to horizontal deblocking on the original order. The result is again stored as transposed w.r.t the input, yielding the original order. Instead of selecting a different procedure for each of the three different filtering cases, a decision index is used to select different sets of coefficients from a table. A set containing six groups of 7 coefficients is loaded from the table and six output values, one for each group of coefficients, are calculated for every window perpendicular on block borders. The identity is implemented by several of these coefficients: the full set for the 'no filter' decision and four of the values for the 'weak filter'. According to experiments performed on video sequences of 704 x 480 pixels at 30 frames/s encoded at 384kbits/s using MPEG2, **85% of decisions indicate 'strong filter', 13% 'weak filter' and 2% 'no filter'**. The identity is then used in less than 10% of the output values. Packed data (8 x 8 bits or 4 x 16 bits data in 64 bits register pair) is used for the computation. Filtering is achieved using *sum2p* instructions. The normalization is achieved using shifts to the right. The sequential implementation of the deblocking takes 106 cycles for calculating the decision index and the six output values. By optimizing the code using VLIWs the computation time is reduced by a factor of 2.65, to 40 cycles.

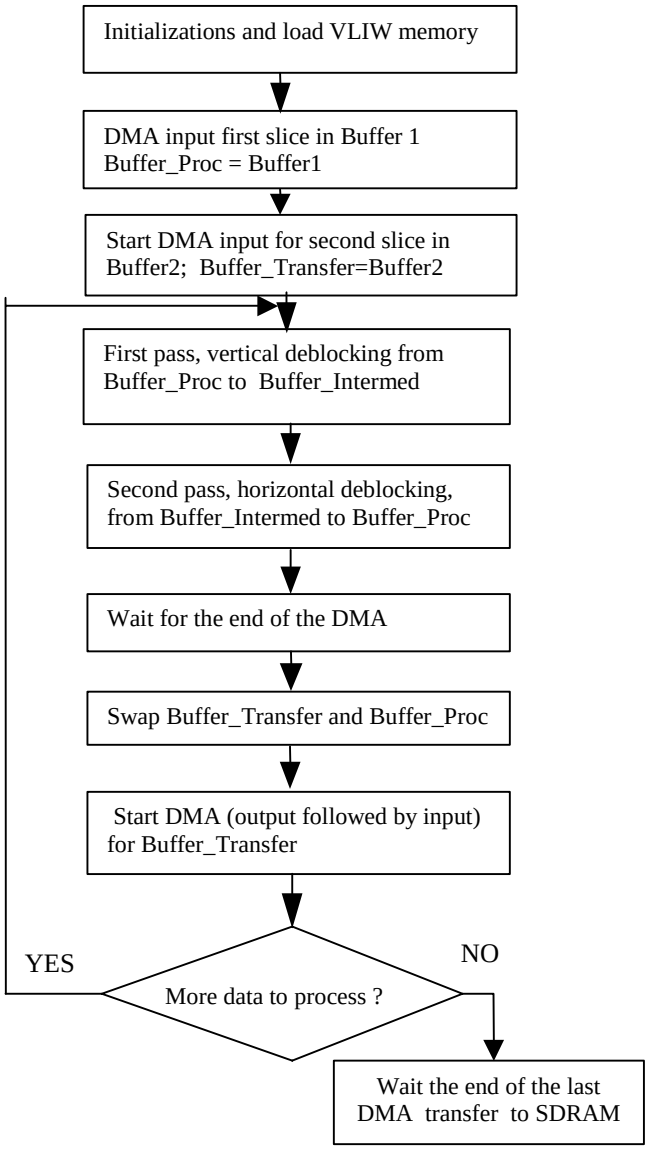

Fig.4. The program flow for deblocking filtering

The design enables the data transfer between the SDRAM memory and a PE memory buffer (Buffer_Transfer) to take place while the computation is performed using two other buffers (Buffer_Proc and Buffer_Intermed). PE data memory is divided into three data buffers. Two of them are alternately used for loading input data, and storing the result. The third buffer, denoted Buffer_Interm, is used for storing the intermediate filtering result after the first pass through the deblocking filter. One DMA channel is used for DMA output and input transfers, the data transfer time taking less than the actual processing. First the filtered data is transferred from Buffer_Transfer to SDRAM, then the buffer is filled with new data from the SDRAM. The only 'wait' states for the DMA to complete correspond to the first DMA transfer from SDRAM and the last DMA transfer to SDRAM. The data transferred from SDRAM into each PE memory include the rectangular slice and the additional

v_0	v_1	v_2
v_3	v_4	v_5
v_6	v_7	v_8

1 2 1

2 4 2

1 2 1

Fig.5. The processing mask and filter coefficients for the deringing filter

boundary rows and columns needed in the computation. In the first pass, the bordering data needed for the second pass is also filtered. The program flow is shown in Fig. 4.

4. THE DERINGING FILTER

4.1. Description

An original adaptive filter is proposed and implemented for deringing. It relies on the assumption that the filtering masks must always include only pixels which are on the same side of an edge that needs to be preserved. Otherwise, undesired blurring of image details occurs. In addition, the procedure targets the ease of implementation on parallel processors, which does not allow the use of data dependent jumps or calls. Experimentally, for the tested data, the visual quality obtained using this deringing filter on very low bit rate sequences is better than that obtained by MPEG4 filter[1] and the filter in [4].

For each pixel of the image a 3x3 mask (v_0-v_8) is processed. Initially, the mask includes the pixel to be computed (denoted v_4) and its 8 neighbors from the original image. The processing mask and filter coefficients are shown in Fig.5. The absolute difference between the pixel and each of its 8 neighbors is compared with a threshold which is equal to QP. If the difference is higher than the threshold, the corresponding neighbor value is replaced in the processing mask by the central value. In this case it is assumed that the neighbor does not belong to the same side of an image edge as the central pixel. Finally, a low pass filter is applied to the values in the processing mask to yield the result. By replacing the values in the processing mask, the filter varies between a low pass filter (when no value is replaced, because no image edge is present in the mask) to the identity (when all differences are larger than the threshold and all values are replaced by the central one).

4.2. Deringing filter implementation on MANTA

The image is divided into rectangular slices, which are separately processed by the 4 PEs. Data transfer between local PE data memories and the SDRAM is performed in the background of the computation, as in the deblocking filter case. Input slices contain, for each PE, the additional bordering rows and columns needed in the computation. This approach increases the amount of transfer but removes data dependencies between the PEs. It works well in such situations when the computation takes longer than the data transfer. The filtering is achieved in a single pass. Eight output values are calculated on each PE in one pass through the computation loop. Packed data (8 x 8 bits or 4 x 16 bits data in 64 bits register pair) is used. In the decision part, where the values of the input vector **v** are selected, the implementation takes advantage of the *absdif* and *copys*

instructions which may be performed on packed 8 x 8 bit data. Additions, multiplies by 2 and shifts for division are used in the computation and the output is calculated as in the equation:

$$y = (((v_4 + 2) * 2 + v_1 + v_3 + v_5 + v_7) * 2 + v_0 + v_2 + v_6 + v_8)/16$$

The code is optimized using VLIWs. The computation takes 96 cycles in the sequential implementation and only 36 in the optimized one, the VLIW efficiency factor being 2.67**.**

5. PERFORMANCE OF THE IMPLEMENTATION

5.1. Theoretical lower bounds for the computation cycles

On each PE, the deblocking filter loop takes 40 cycles to calculate six output values in each group of eight pixels perpendicular on block borders. For one frame having horizontal and vertical dimensions H and V, the loop runs V*(H/8-1) for vertical deblocking and H*(V/8-1) for horizontal deblocking of luminance. This makes a theoretical lower bound of (V*H/4-V-H)*40 cycles, without including the initializations. The deringing filter loop takes 36 cycles for calculating eight output values. For one frame having horizontal and vertical dimensions H and V, the loop runs V*H/8 times for the deringing of luminance. The frame is divided and processed on four PEs and the performance scales linearly with the number of PEs. If FPS denotes the frame rate, the theoretical lower bounds of the computation cycles for filtering the luminance on four PEs are:

Deblocking: ((V*H/4-V-H)*40*FPS)/4 cycles/s

Deringing: ((V*H/8)*36*FPS)/4 cycles/s

These figures include only the loop computation cycles, and do not add the cycles needed for setting up the loops and computing the parameters for the DMA transfers.

5.2. Experimental performance

Deblocking and deringing were applied to the luminance component, on 4:2:0 video frames, at two different sizes. The cycle counts shown in Table 1 were obtained on the MANTA chip. MANTA does not have byte multiplies and it only allows word aligned (1 word = 32 bits) access to data memory. One frame from the sequence Susi (704x480 pixels, 30 frames/s) decoded from an MPEG2 video stream at 384 kbits/s is displayed before and after deblocking and deringing in Fig. 6 and Fig.7.

6. CONCLUSIONS

The implementation of deblocking and deringing algorithms on BOPS parallel DSP cores is presented. A new type of deringing filter is proposed and implemented to preserve real image edges while smoothing out the inside of objects. The main features of the architecture used in the implementation are the multiple levels of parallelism (data level, instruction level and system level parallelism), the powerful instruction set, the scalability of the architecture and the strong DMA engine which enables data transfers to happen with almost zero overhead. The experimental performance illustrates that such complex tasks may be developed and performed in real time, at low clock rates. MANTA running at 100 MHz (100 Mcycles/s) may achieve both these filtering tasks using only half of its computational bandwidth.

	Frame size	Cycles/ Frame	Cycles/s for 30 frames/s
Deblocking	704 x 480	1,200,000	36 Mcycles/s
	352 x 240	300,000	9 Mcycles/s
Deringing	704 x 480	600,000	18 Mcycles/s
	352 x 240	150,000	4.5Mcycles/s

Table 1. Cycle counts for deblocking and deringing filters running on MANTA.

7. REFERENCES

1. ISO/IEC 14496-2:1999 Information Technology – Coding of audio-visual objects – Part 2: Visual, Annex F, “Deblocking Filter”, pp.290-292, “Deringing Filter” , pp.292-293.
2. Gerald G. Pechanek, Stamatis Vassiliadis and Nikos P. Pitsianis, "ManArray Interconnection Network: An Introduction," *Proceedings of EuroPar '99 Parallel Processing*, Aug. 31-Sept. 3, 1999, Toulouse, France, Lecture Notes In Computer Science Vol 1685, pp 761-765.
3. Gerald G. Pechanek, Charles Kurak and Bruce Schulman, "Design of MPEG-2 Function with Embedded ManArray Cores," *Proceedings of DesignCon 2000*, Jan. 31- Feb. 3, 2000.
4. H.W.Park and Y.L.Lee, “A Postprocessing Method for Reducing Quantization Effects in Low Bit-Rate Moving Picture Coding”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.9, No.1, Feb.1999, pp.161-171.
5. S.D.Kim, J.Yi, H.M.Kim and J.B.Ra, “A Deblocking Filter with Two Separate Modes in Block-Based Video Coding”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.9, No.1, Feb.1999, pp.156-160.

Fig.6. Frame from sequence Susi (704x480 pixels) at 384 kbits/s with noticeable artifacts

Fig.7. Frame from Fig. 6 after deblocking and deringing