A fast FPGA implementation of the inter-layer deblocking filter for H.264/SVC

Guilherme Corrêa, Luciano Agostini[†], Luís A. Cruz[¥]

[†] Grupo de Arquiteturas e Circuitos Integrados, Universidade Federal de Pelotas, Pelotas, Brasil
e-mail: *{gcorrea_ifm, agostini}@ufpel.edu.br*

[¥] Instituto de Telecomunicações, Universidade de Coimbra, Coimbra, Portugal
e-mail: lcruz@deec.uc.pt

Abstract — **The scalable extension to H.264/AVC defines a coding scheme that organizes data in such a way that starting from a base layer increased spatial, temporal and quality detail can be obtained by the addition of successive information layers. In the case of spatial scalability a fundamental operation is inter-layer prediction which due to the block based nature of the algorithm originates block-edge discontinuities. To alleviate this problem the standard defines a block edge effect filter to smooth block boundaries . This work presents a novel filtering order aimed at speeding-up filter operation in the context of an FPGA-based implementation.**

I. INTRODUCTION

The technological advances of the last decades has brought us a wide range of multimedia-capable devices as well as data networks with varying transmission characteristics At the same time, video coding has attained a development state that makes feasible the transmission of very high resolution (temporal and spatial) video enabling services like HDTV broadcast, SDTV transmission over IP networks and low resolution video communication over wireless links with rendering at portable devices. This plethora of device and network configurations posed a problem to video service providers as they either had to have a multitude of coded representations of each video item in their repository, one for each possible combination of target device and transmission link (in the case of real-time services multiple coders had to be used) or had to employ transcoding devices to convert the original video streams coded at the most inclusive resolution and quality level to a format suitable for transmission and decoding by the target network and receiving device. Since both solutions had serious disadvantages, a new way of achieving Universal Media Access (UMA) was sought in the form of scalable coding techniques which found their way into an extension of the standard H.264/AVC, here forth designated as H.2647SVC.

The general operating principle of scalable coding (as used in H.264/SVC) defines a hierarchy of information layers where a base layer represents the video signal at a given minimum temporal and spatial resolution and quality Additional information layers are used to increase (in discrete steps) temporal and spatial resolution as well as the representation fidelity (quality). The arrangement allows orthogonal refinement along each of the three scalability dimensions, time, space and quality and in the case of H.264/SVC each layer's information is carried as an independent set of bits identified according to the NAL signaling protocol detailed by H.264/SVC normative documents.

Spatial scalability in H.264/SVC is realized using among other operations, inter-layer prediction whereby a higher layer image region is predicted from the corresponding lower layer area by a process of up-sampling. Unfortunately the block-based nature of the entire algorithm causes the appearance of block-edge artifacts at the boundaries between blocks which degrade the quality of the reconstructed video. To reduce the visible effects of this phenomenon, H.264/SVC defines an inter-layer filtering procedure that is carried out across block boundaries (deblocking filter). This filter performs quite well but has a high computational cost and so it is of paramount importance to have efficient implementations of its operations. In the case of implementations using programmable devices (e.g. FPGAs) the major goals are low cycle-count and low gate-count solutions as they translate into respectively low latency operation (possibly also calling for lower speed devices with reduce power consumption) and smaller devices (desirable for portable low-cost coder/decoder devices).

This work presents a new scheduling of the filtering operations of the H.264/SVC interlayer deblocking filter specifically designed for an implementation using FPGA programmable devices. To the best of our knowledge it will be shown that the filtering order proposed achieves a lower operation count than the competing solutions presented the literature.

This communication is structured as follows; in section II we outline the operation of the deblocking filter as specified by H.264/SVC, section III is devoted to the exposition of the best performing filter ordering solutions published in the technical literature as well as our solution, in section IV the architecture of the proposed filter is presented, section V reports on the results and compares them to competing solutions. Section VI concludes with a discussion of the current results and of promising directions for future work.

II. DEBLOCKING FILTER

Similarly to H.264/SVC so too the non-scalable H.264/AVC standard specifies a deblocking filter to be used in the decoder but in the latter the filter is used to smooth the reference reconstructed block boundaries before using its pixels in the prediction of the current block. The filter is

This work was supported by IT – Coimbra and the Santander Group.

highly adaptive and can discern between naturally occurring edges due to video texture and edges caused by severe quantization and furthermore the filtering operation is applied only to blocks coded in some modes.

The scalable-standard deblocking filter is similar to the non-scalable version in that for the first layer only the intra-coded blocks will be subject to smoothing. This is a result of the single-loop decoding used in the scalable extension of H.264/AVC [7]. Therefore all the blocks that were inter coded will not be filtered and application of the scalable filter will be less expensive than in the case for the non-scalable standard where all the reconstructed blocks will be filtered.

The filtering operation is applied across the horizontal and vertical boundaries of the 4x4 blocks that makeup a macroblock in accordance with the steps 1 to 4 listed bellow with reference to figure 1 [8].

1. Filter across the vertical edges of the luminance macroblock (edges *a*,*b*,*c* and *d* on figure 1).
2. Filter across the horizontal edges of the luminance macroblock (edges *e*,*f*,*g* and *h* on figure 1).
3. Filter across the vertical edges of **each** chrominance block (edges *i* and *j* on figure 1).
4. Filter across the horizontal edges of **each** chrominance block (edges *k* and *l* on figure 1).

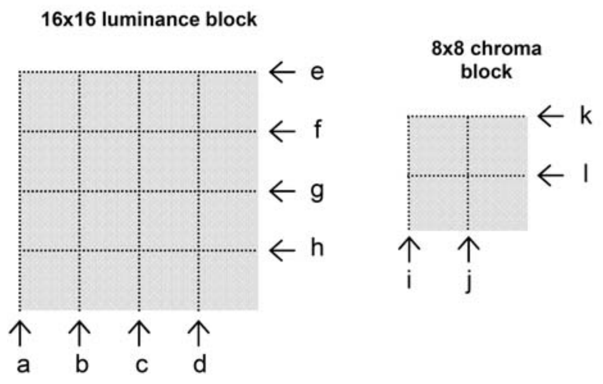

Figure 1 – Block edges to be filtered

Each filtering operation modifies up to three pixels on each side of the edge and involve four pixels of each of the two blocks (**p** and **q**) that meet at the edge across which filtering is taking place. The filter strength is adjustable and depends on the quantization step-size used when the block was coded, on the coding mode of neighboring blocks and the gradient of the values of the pixels computed across the edge being filtered. The standard specifies five different strengths parameterized as bS which takes values 0 through 4, 0 standing for "no filtering" and 4 indicating maximum smoothing. The value of bS is chosen according to the following algorithm:

```
If p or q belong to an intra macroblock (MB)
different from I_BL
  Then bS = 4
If p and q belong to a intra MB of type I_BL
  Then
     If at least one of the transform coefficients
     of the blocks to which samples p0 and q0 belong
     is ≠ 0
            Then bS = 1
            Else bS = 0
If p or q belong to an inter MB
     Then
     If p (or q) belong to a inter MB and the
     residues matrix rSL has at least one value ≠ 0
     in the blocks associated with sample p0 (or
     q0).
            Then bS = 2
            Else bS = 1
```

However for values of bS>0 filtering takes place only if condition (1) is met where p2, p1, p0, q0, q1, and q2 represent the samples on both sides of the edge and occurring in this order

$$|p0 - q0| < \alpha \text{ and } |p1 - p0| < \beta \text{ and } |q1 - q0| \leq \beta \qquad (1)$$

where α and β defined in the standard increase with increasing average value of the quantizer step-size (QP) of blocks **p** and **q** so that if QP is small so will be α and β and the filter will seldom be applied and vice-versa. For more details please consult [3].

III. Filtering Orders

The only restriction imposed by H.264/SVC on the filtering order is that if a pixel is involved in vertical and horizontal filtering, then the latter should precede the former. This is rather loose and offer opportunities for implementation optimization by exploring different filtering scheduling, aiming at faster operation through the use of parallelism or at solutions that use less memory. The filtering order proposed by H.264/AVC [2] is shown in figure 2.

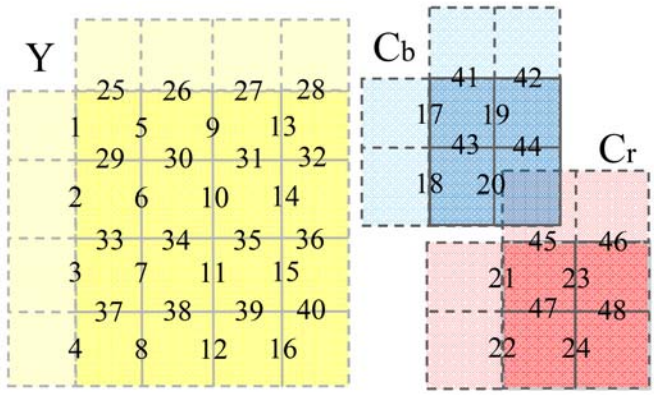

Figure 2 – Filtering order proposed by standard H.264/AVC

It is obvious that all vertical edges are filtered before the horizontal ones and since the results of vertical filtering are needed for the horizontal filtering they have to be kept in memory making this solution quite costly in terms of memory usage. Khurana proposes in [9] a different order where horizontal and vertical filtering alternate as illustrated in figure 3.

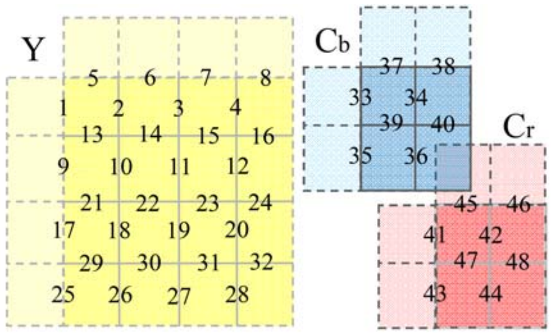

Figure 3 – Alternate order of Khurana [9]

Using this order we have only to keep in memory a line of 4x4 blocks to be used during the next edge filtering and after being filtered in both directions a pixel can be written back to main memory. This principle inspired Shen [10] who proposed the order depicted in figure 4 where a higher frequency of vertical-horizontal filtering direction change is observed

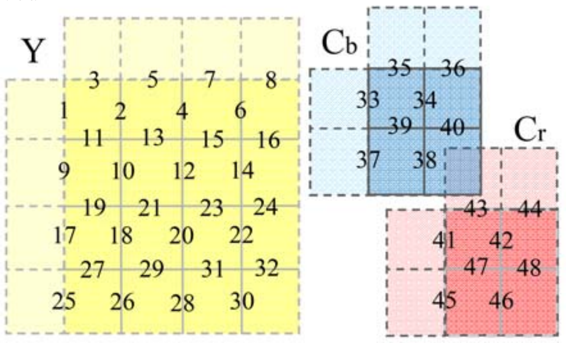

Figure 4 – Shen´s filtering order [10]

resulting in a need for a smaller temporary buffer. Li[11] proposes another solution involving a degree of parallelism with vertical and horizontal filtering occurring at the same time, speeding up the filtering at the cost of having to use two filtering units. The scheduling for this solution is presented in figure 5.

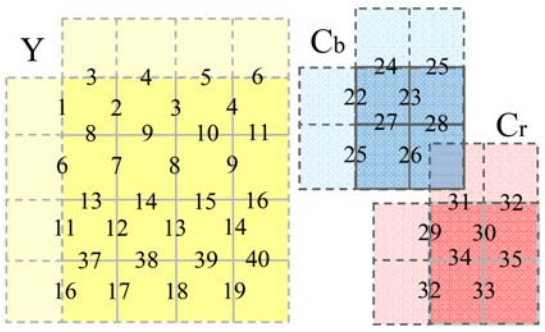

Figure 5 – Filtering order from Li [11]

At the start our work it was decided to take a finer granularity approach and seek filtering orders defined at pixel level instead of at block level as is the case of the solution

presented so far thereby hoping to achieve a higher degree of parallelism while keeping memory use and the number of filtering units low. The filtering order proposed is presented in figure 6 (see last page) were same number labels identify filtering operations that can be executed in parallel on different filtering cores.

IV. Global Architecture

Our architecture uses four filter cores who share a bS computing unit as well as a α and β computing unit. Since filtering occurs along a line-of-pixels (LOP) which can be vertically or horizontally aligned we would need a transposition operation to filter orthogonally the pixel storage direction. Instead we use transposition matrices to be able to access pixel memory in either direction. The filter architecture is composed of the modules shown in figure 7: eight transposition matrices (T1 to T8), four filter kernels (F1 to F3), one bS calculation unit (bS_calc), a threshold calculator (thr_deriv) a c1 calculator (c1_calc) and a control unit. Filtering is performed in a pipelined fashion as follows:

- Cycle 1. First LOP from first blocks of macroblock are loaded into T1,T2,T3,T4,
- Cycle 2. Second LOP are loaded into T1,T2,T3,T4.
- Cycle 3. Third LOP are loaded into T1,T2,T3,T4.
- Cycle 4. Fourth LOP are loaded into T1,T2,T3,T4.
- Cycle 5. First filter core starts filtering || First LOP of second set of blocks are loaded into T5,T6,T7,T8.
- Cycle 6. From this cycle on, one more filter core is activated until all are busy.

From this point on, for each cycle two filtering operations are performed per filtering core. If vertical filtering has been done LOP can be sent to main memory otherwise it is stored in its matrix and later read in filtered in the vertical direction.

When all the filtering operations using pixels from a transposition matrix have been done the matrix can be filled with data from then next set of blocks. This way the matrix

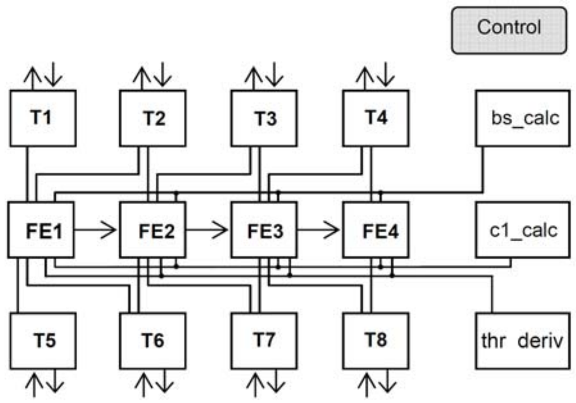

Figure 7 – Filter architecture

that receives the samples from the first block will also store the samples from the ninth block and so on.

V. RESULTS AND DISCUSSION

The architecture proposed in this article takes 53 cycles to filter one macroblock which is about 24% less than the best of the competing proposals with the same number of filtering cores [12]. Table 1 shows other solutions data concerning number of cycles necessary for filtering one macroblock and size of working memory needed.

	Cycles per MB	**Filter Cores**	**Memory Bytes**
H.264/AVC [2]	192	1	512
Khurana [9]	192	1	128
Sheng [10]	192	1	80
Li [11]	140	2	112
Ernst [12]	70	4	224
This work	**53**	**4**	**128**

Table 1 – Comparison with other solutions

Even though only one of the other methods uses four filter cores, our proposal is faster and uses less memory. The architecture was described using VHDL and synthesized for Altera Stratix IV devices. The core filter used 737 ALUTs and was able to run at 166,56 MHz which considering operation of the four cores allows filtering of all blocks of a video with VGA resolution (640x480) at a rate of 5,780 frames per second.

VI. CONCLUSIONS

This work presented a competitive architecture for the implementation of the deblocking filter of H.264/SVC. Results show its advantages over similar published designs, outperforming the best one by 24%. In the near future the architecture will undergo further validation, synthesis and prototyping using Altera Stratix IV devices.

REFERENCES

[1] C. A. Segall e G. J. Sullivan, “Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, N. 9, pp. 1121-1135, Set. 2007.

[2] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG. Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding), 2003.

[3] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz and M. Wien, “Joint Draft 11 of SVC Amendment”, Joint Video Team, Doc. JVT-X201, Jul. 2007.

[4] H. Huang, W. Peng, T. Chiang e H. Hang, “Advances in the Scalable Amendment of H.264/AVC”, Communications Magazine, IEEE. Vol. 45, No. 1, pp. 68-76, Jan. 2007.

[5] H. Schwarz, D. Marpe, T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp. 1103-1120, Sept. 2007, invited paper.

[6] M. Wien, H. Schwarz, T. Oelbaum, “Performance Analysis of SVC”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp.1194-1203, Sept. 2007.

[7] C. A. Segall, G. Sullivan, “Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp. 1121-1135. Set. 2007.

[8] I. Richardson, “H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia”, John Wiley & Sons Publishers, USA, 2003.

[9] G. Khurana, A. A. Kassim, T. P. Chua e M. B. Mi, “A pipelined hardware implementation of In-loop Deblocking Filter in H.264/AVC”. IEEE Transactions on Consumer Electronics, 52(2):536 – 540, 2006.

[10] B. Sheng, W. Gao, e D. Wu, “An Implemented Architecture of Deblocking Filter for H.264/AVC”. Proceedings - International Conference on Image Processing, ICIP, 1:665 – 668, 2004.

[11] L. Li, S. Goto, e T. Ikenaga, “A highly parallel architecture for deblocking filter in H.264/AVC”. IEICE Transactions on Information and Systems, E88(7):1623 – 1628, 2005.

[12] Ernst Eric, “Architecture Design of a Scalable Adaptive Deblocking Filter for H.264/AVC”. MSc thesis. Rochester, New York, 2007.

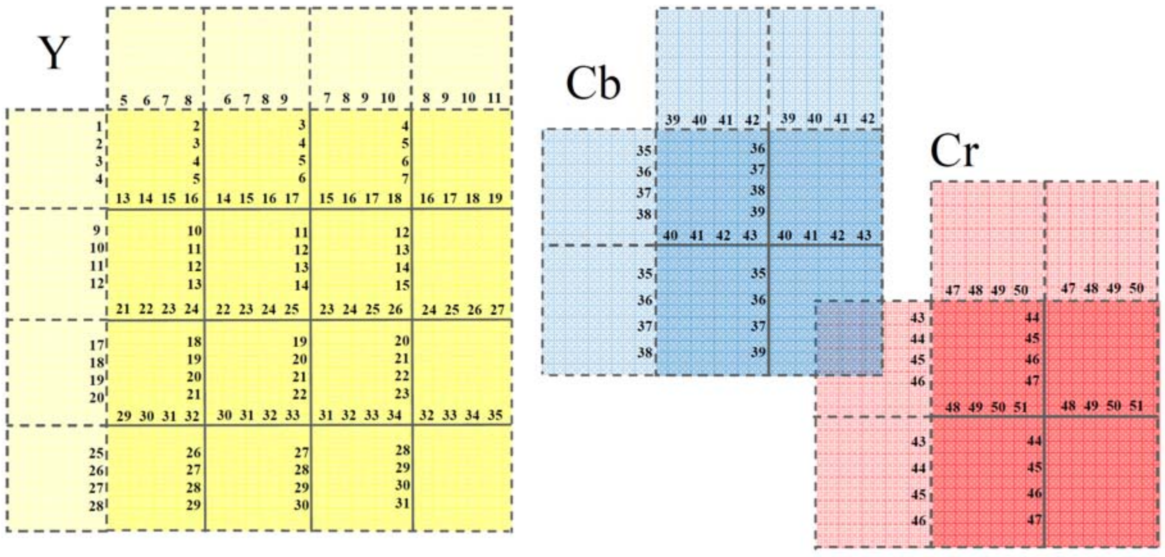

Figure 6 – Proposed filtering order.