DESIGN OF AN INTERLAYER DEBLOCKING FILTER ARCHITECTURE FOR H.264/SVC BASED ON A NOVEL SAMPLE-LEVEL FILTERING ORDER

¹Guilherme Corrêa, ²Thaísa Silva, ³Luis A. Cruz, ¹Luciano Agostini
{gcorrea_ifm, agostini}@ufpel.edu.br, tlsilva@inf.ufrgs.br, lcruz@deec.uc.pt

¹Group of Architectures and Integrated Circuits – Federal University of Pelotas, Brazil
²Microelectronics Group – Federal University of Rio Grande do Sul, Brazil
³Institute for Telecommunications – University of Coimbra, Portugal

ABSTRACT

This paper presents the architectural design for an interlayer Deblocking Filter of the H.264/SVC standard. The architecture described applies a novel and efficient processing order based on sample-level filterings. This order allows a better exploration of the filter parallelism, decreasing in 25% the number of cycles used to filter the videos, when compared to the best related work. Four concurrent filter cores were used in the architecture, which was described in VHDL and synthesized for an Altera Stratix III FPGA device. The timing analysis results showed that this design is able to filter up to 130 HDTV (1920x1080 pixels) frames per second.

Index Terms — H.264/SVC, scalability, deblocking filter, architectural design.

1. INTRODUCTION

The H.264/AVC [1] coder defines, among its modules, a Deblocking Filter [2] which is responsible for increasing the image subjective quality through smoothing operations over the pixels. These operations are performed in order to decrease the perception of some block-shaped artifacts introduced in the frame by a high quantization level. The filter was optional in most of the previous standards, but it was defined as mandatory in the H.264 standard.

The H.264 standard supports scalability through its SVC extension [3], whose operating principle defines a hierarchy of layers where a base layer represents the video signal at a given minimum temporal, spatial and/or quality resolution, and additional layers are used to increase each one of the resolutions. In the scalable version of H.264 (herein called H.264/SVC), a slightly different version of the filter is used between two coding (or decoding) layers in order to smooth the block-shaped artifacts introduced in the reference layer before they are used in the enhancement layer.

Due to its high complexity, a wide research has been done regarding the implementation of the H.264/AVC or SVC Deblocking Filter. The main source of its complexity can be attributed to the fact that each pixel must be read a number of times in different directions to filter a complete macroblock. To deal with this problem, several processing orders were proposed in previous works, all of them aiming at the decrease of the time and the amount of memory used for in the filtering process.

We have proposed in previous works, a new and optimized filtering order for the H.264 Deblocking Filter based on sample level operations. This novel filtering order can be applied to both H.264/AVC and H.264/SVC versions of the standard. It allows a better exploration of the parallelism level, reducing significantly the use of cycles to perform the filtering process in comparison with the previous solutions presented in the literature.

In this paper, we describe the new filtering order to the Deblocking Filter of the SVC Interlayer Intra Prediction and the optimized architectural design for this filtering order. The architecture was described in VHDL and synthesized for Altera Stratix III FPGAs [4].

The paper is structured as follows: in section 2 we outline the operation of the Deblocking Filter. Section 3 is devoted to the exposition of the filter ordering solutions published in the technical literature as well as the novel proposed solution. In section 4 the optimized architecture of the filter is presented. Section 5 reports the results and compares them to related works. Section 6 concludes with a discussion about the current results and shows promising directions for future work.

2. DEBLOCKING FILTER

The Deblocking Filter is an adaptive filter, since it can distinguish a real border in an image (which shall not be filtered) from an artifact generated by a high quantization step (which must be filtered) [2]. The filter is applied to the vertical and horizontal border of each 4x4 block of the luminance and chrominance macroblocks, as shown in Figure 1. At first, the four vertical borders of the luminance component are serially filtered (a, b, c and d, in Figure 1). After that, the four horizontal borders of the luminance component are serially filtered (e, f, g and h, in Figure 1).
Then the same process is done for the chrominance component \((i\) and \(j\) vertical borders and \(k\) and \(l\) horizontal borders in Figure 1).

![Diagram of luminance and chrominance blocks](image)

**Figure 1** – Filtering ordering of the luminance and chrominance borders.

The filter of the scalable standard is similar to the non-scalable version, although the boundary strength \(bS\) [1] calculation performs different operations. Also, only the intra-coded blocks from the reference layers are filtered by this filter. This happens due to the single-loop concept explored in H.264/SVC [3], which defines that only the highest layer can perform motion compensation. Therefore, inter coded blocks are not filtered and the application of this filter is less expensive than the non-scalable filter.

Each filtering operation modifies up to three pixels on each side of the edge and it involves four pixels of each of the two neighboring blocks \((p\) and \(q\)) that are filtered. The filter strength is adjustable and depends on: (1) the quantization step [5] used when the block was coded, (2) the coding mode of neighboring blocks and (3) the pixels gradient values across the edge which is being filtered. The H.264/SVC standard specifies four different strengths parameterized as \(bS\) which takes values 0, 1, 2 or 4 (0 standing for “no filtering” and 4 indicating maximum smoothing). The value of \(bS\) is chosen according to the following algorithm [6]:

If \(p\) or \(q\) belongs to an intra macroblock (MB)
- different from I_BL
  - Then \(bS = 4\)

If \(p\) and \(q\) belong to an intra MB of type I_BL
  - Then
    - If at least one of the transform coefficients of the blocks to which samples \(p_0\) and \(q_0\) belong is \(\neq 0\)
      - Then \(bS = 1\)
      - Else \(bS = 0\)

If \(p\) or \(q\) belongs to an inter MB
  - Then
    - If \(p\) (or \(q\)) belongs to a inter MB and the residues matrix \(rSL\) has at least one value \(\neq 0\) in the blocks associated with sample \(p_0\) (or \(q_0\)).
      - Then \(bS = 2\)
      - Else \(bS = 1\)

However, for values of \(bS > 0\), filtering takes place only if condition (1) is met, where \(p_1, p_0, q_0,\) and \(q_1\) represent the samples on both sides of the edge (in this order). In (1), \(\alpha\) and \(\beta\) are defined in the standard and they increase with the increasing of the quantization step \((QP)\) used in the blocks \(p\) and \(q\). Then, if \(QP\) is small, so will be \(\alpha\) and \(\beta\) and the filter will be applied. The values of \(\alpha\) and \(\beta\) are derived from tables indexed by values which depend on the average \(QP\) used in the blocks which are being filtered.

\[|p_0 - q_0| < \alpha \text{ and } |p_1 - p_0| < \beta \text{ and } |q_1 - q_0| \leq \beta \quad (1)\]

### 3. Filtering Order

To filter a macroblock, the value of a pixel must be read multiple times and the intermediate results of the filterings are stored in a local memory, since they are used in the following steps. In order to improve the use of the local memory, it is possible to reorder the filtering operations in such a way that the intermediate results are used sooner. The only restriction imposed by the standard in relation to the processing order specifies that all the horizontal filterings that use a determined sample must occur before the vertical filterings that use this sample.

The processing order proposed by the H.264/AVC standard [1] is presented in Figure 2. In Figure 2, the vertical borders of the luminance and chrominance blocks are all filtered before the horizontal borders. Since the results of the vertical filterings are used in the horizontal filtering, these intermediate results must be stored. Then, this processing order is expensive in terms of memory use, since it demands the storage of 24 4x4 blocks (16 luminance blocks and 4 blocks for each chrominance) until the horizontal filtering occurs.

![Diagram of H.264/AVC filtering order](image)

**Figure 2** – Original H.264/AVC filtering order [1].

The filtering order proposed by Khurana [7], presented in Figure 3, is based on an alternation between the horizontal and vertical filterings of the blocks. This solution results in a decrease of the local memory size, since just one line of 4x4 blocks must be stored in order to be used by the next filtering steps. When the pixels are completely filtered (i.e., in both directions), they can be written back to the main memory in order to be shown or to be used as reference in the future.
A third processing order, proposed by Sheng [8], is based on the same alternation principle proposed by Khurana [7]. However, in this case, the frequency of change between horizontal and vertical borders (and vice-versa) is higher, occurring after most of the 4x4 border filterings. As a result, there is a higher decrease in the size of the used local memory.

The proposal of Li [9] is based on both data reuse and concurrence principles. The processing order, shown in Figure 4, reduces significantly the number of cycles needed to filter a macroblock through the execution of horizontal and vertical filterings at the same time. For that, two filters are needed, one for each filtering direction. Based on this proposal, a concurrent architecture that uses four filtering cores is presented by Ernst [10].

The number of concurrent filterings is, however, very limited due to the dependences between the data. Using the method proposed by Li [9] would be possible to perform three or more concurrent filterings in the same macroblock without a significant increase in the local memory size.

All the processing orders presented are performed in block level, i.e., the filtering of a 4x4 block edge is performed serially by the same filter and the border of a block can be filtered only after the filtering of the 4 LOPs (Line of Pixels) of the previous (left) block.

The architecture proposed in this paper is based on a new processing order that was developed in a previous project. This new order is based on sample-level filterings, instead of block-level filterings. This way, the filtering of the first LOP of a block can start as soon as the result of the first LOP of the previous block is available.

The processing order oriented to samples allows a better use of the architecture parallelism without increasing significantly the use of local memory and the number of cycles used to filter a macroblock. Figure 6 shows the proposed filtering order. The repeated numbers correspond to the filterings which occur in parallel, in the same cycle, by different filter cores. Considering this order, four filterings occur in parallel, resulting in an architectural design composed by four filtering cores, as detailed in section 4.
4. DESIGNED ARCHITECTURE

The designed architecture was described in VHDL and all its modules were synthesized through the Altera Quartus II software [4]. In order to validate the modules, the deblocking operations were described in C code, based in the reference software, and the execution results were compared to the results obtained by architectural simulations through the Quartus II software [4].

The complete filter architecture is presented in Figure 6. This architecture is composed by one \( bS \) calculator module (\( bS_{\text{calc}} \) in Figure 6), one thresholds calculator module (\( \text{thr}_{\text{calc}} \) in Figure 6), eight transpose matrices (\( T1-T8 \) in Figure 6) and four filtering cores (\( FE1-FE4 \) in Figure 6).

The transpose matrix stores the samples during the period in which the filterings that use these samples are still occurring. Each transpose matrix stores the samples of a full block, in addition to its coding information. The filtering cores perform the filtering operations using the samples and the values of \( bS \), thresholds (\( \alpha \) and \( \beta \)) and \( cl \), which were previously calculated. The \( bS \) calculator defines the filtering strength based on some coding information and the threshold calculator defines the values of \( \alpha \) and \( \beta \) based on the quantization parameters of the two blocks which are being filtered. The \( cl \) calculator is a module that, based on the filtering strength and on the thresholds values, generates a clipping value that is used in the filtering process.

The \( bS \), thresholds and \( cl \) calculators are connected to multiplexers that deliver their results to the corresponding filtering cores. Thus, as these values do not change between the LOPs of the same block, a set of buffers is needed to retain the values of \( bS \), \( \alpha \), \( \beta \) and \( cl \) during the three cycles following the first filtering.

The architecture operates in a pipelined structure that performs four concurrent filterings. Thus, the filtering of each block takes 7 cycles, though a line of blocks (4 4x4 blocks) can be filtered in just 10 cycles. Each filtering core starts its operation one cycle late in relation to the other, so that the LOPs filtered by a core can be reused in the next cycle by the core responsible for the next border (right) filtering.

Figure 7 illustrates the operation flow of the filter in its first ten cycles. In the first four cycles, the first LOPs of the four first blocks in a macroblock are written in the transpose matrices. In the second cycle, the reading of the second LOPs of the four first blocks occurs in parallel with the threshold and \( bS \) calculations corresponding to the first border (between the left macroblock and the first block of the current macroblock). In the third cycle, the reading of the third LOPs occurs in parallel with the threshold and \( bS \) calculations corresponding to the second border and with the \( cl \) calculation corresponding to the first border. From the fourth cycle on, the filterings start to happen in the first core, since all the parameters are available. A new core is enabled in each of the following three cycles, so that all of them work concurrently from the seventh cycle on.

5. RESULTS AND COMPARISONS

As explained before, the architecture proposed in this paper considered a new filter ordering and its consequent new algorithm. This new filtering order was initially proposed in one of our previous works. In order to decide whether it is the best option of use or not, an analysis considering the number of cycles and the local memory size needed to filter a complete macroblock in each filtering order was done. Table 1 shows a comparison between the used filtering order and five other orders found in the literature.
As can be seen in Figure 5 and in the last line of Table 1, the filtering order used in this project performs the whole filtering of a macroblock in 53 cycles (35 cycles for luminance and 9 cycles for each chrominance type). The best number of cycles reached in the related works was of 70 cycles that was obtained by Ernst [10]. The Ernst work is based on Li's processing order [9] and it uses four parallel filtering cores. That means that our new filtering order decreases in at least 25% the number of cycles used in the deblocking filter when compared to the previously published solutions. In the best case, our solution reaches a gain of 72% in relation with the related works [1], [7] and [8]. Our solution also requires 128 bytes for storage, 16 bytes for each transpose matrix. Such matrices store the pixels of the current macroblock which are under filtering (i.e., the pixels which were not filtered in both directions yet). The pixels from the neighbor macroblocks (left and above) are also stored in the transpose matrices when necessary. When all the pixels of the transpose matrices are filtered in both directions, they are written back to the main memory. This way, the last line and the last column of blocks in a macroblock must be read again from the main memory when they are used as neighbor blocks.

Table 1 – Comparison between processing orders

<table>
<thead>
<tr>
<th>Filtering Order</th>
<th>Cycles per MB</th>
<th>Filter Cores</th>
<th>Memory Needed (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264/AVC [1]</td>
<td>192</td>
<td>1</td>
<td>512</td>
</tr>
<tr>
<td>Khurana [7]</td>
<td>192</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>Sheng [8]</td>
<td>192</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>Li [9]</td>
<td>140</td>
<td>2</td>
<td>112</td>
</tr>
<tr>
<td>Ernst [10]</td>
<td>70</td>
<td>4</td>
<td>224</td>
</tr>
<tr>
<td><strong>Our order</strong></td>
<td><strong>53</strong></td>
<td><strong>4</strong></td>
<td><strong>128</strong></td>
</tr>
</tbody>
</table>

The architecture presented was synthesized for the EP3SL50F484C2 Stratix III Altera FPGA. Table 2 shows the synthesis results in terms of combinational ALUTs and dedicated registers used by each module and by the complete architecture (last line of Table 2). As expected, the filtering core is the module which used the biggest amount of logical elements and which showed the highest architecture critical delay (approximately 10.5 ns).

Considering the Quartus II Slow 1100 mV 85C Model for timing analysis, the synthesis results showed that the architecture is capable to work at 265.67 MHz. On the other hand, if the Slow 1100 mV 0C Model is considered, the architecture is able to work at 288.52 MHz. Taking, as reference, a frequency of 270 MHz (value between the two presented by the timing models) and considering a HDTV (1920 x 1080 pixels) resolution, the processing rate of the designed filter is around 130 frames per second. This frame rate satisfies and outperforms the minimum requirements to reach real time.

Table 2 – Logical elements used by the architecture

<table>
<thead>
<tr>
<th>Module</th>
<th>#ALUTs</th>
<th>#registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>transpose matrix</td>
<td>199</td>
<td>206</td>
</tr>
<tr>
<td>threshold calculator</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>bS calculator</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>c1 calculator</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>filter core</td>
<td>737</td>
<td>0</td>
</tr>
<tr>
<td><strong>complete architecture</strong></td>
<td><strong>7868</strong></td>
<td><strong>206</strong></td>
</tr>
</tbody>
</table>

Table 3 shows the characteristics of other architectures for the Deblocking Filter found in the literature. Even though all the architectures found are designed targeting the filter of the H.264/AVC standard, they can be compared to the SVC Interlayer Deblocking Filter due to the slight difference between their operations.

As can be seen in Table 3, the architecture designed in this paper presents the highest operation frequencies and the highest frame rates among all related works. Our architecture also uses the lower number of cycles per macroblock. These good results are function of the new filtering order and it algorithm, combined with the optimized architectural solution that was designed.

The comparison about the use of hardware resources is not easy, since the technologies of the presented designs are different. But it is possible to compare the number of filter cores used in each design. Only our design and the Ernst...
design [10] used four cores and the other solutions used only one filter core. This parallelism of four cores allows the high processing rates achieved and the parallelism is possible in function of the efficient filtering order adopted in this work.

### Table 3 – Logical elements used by the architecture

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>Stratix III</td>
<td>Virtex 5</td>
<td>Virtex - 2P</td>
<td>0.18µm</td>
<td>0.18µm</td>
<td>0.18µm</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>270</td>
<td>43.40</td>
<td>135</td>
<td>200</td>
<td>36.45</td>
<td>100</td>
</tr>
<tr>
<td>ALUTs or Gates</td>
<td>7,868</td>
<td>16,594</td>
<td>4,331</td>
<td>18,700</td>
<td>12,600</td>
<td>20,900</td>
</tr>
<tr>
<td>Memory Size</td>
<td>206 dedicated registers</td>
<td>32 BRAMS</td>
<td>65 BRAMS</td>
<td>384 bytes</td>
<td>256 bytes</td>
<td>12,160 bytes</td>
</tr>
<tr>
<td>Cycles per MB</td>
<td>53</td>
<td>70</td>
<td>256</td>
<td>222</td>
<td>110</td>
<td>246</td>
</tr>
<tr>
<td>FPS @ HDTV</td>
<td>130</td>
<td>77</td>
<td>71</td>
<td>116</td>
<td>30</td>
<td>57</td>
</tr>
</tbody>
</table>

Comparing our design with Ernst design [10], which use the same number of filters core, it is possible to notice that our design make a better use of the parallelism, since our design uses 53 cycles per macroblock while Ernst solution uses 70 cycles per macroblock.

### 6. CONCLUSIONS AND FUTURE WORK

This paper reported the implementation of the Interlayer Deblocking Filter for the H.264/SVC coder/decoder. The presented architecture is based on a new processing order based on sample-level filterings, on the opposite of the works found in the literature, which are based on block-level filterings. The related solution joins an efficient filtering order and the respective algorithm with an optimized architecture, reaching the best results among all compared works.

The architecture was designed in VHDL, synthesized and validated for an Altera Stratix III FPGA. It reached a maximum frequency between 265 and 288 MHz for the two timing models used. Considering a frequency around 270 MHz, the architecture is able to process up to 130 frames per second (HDTV 1080p resolution), satisfying and outperforming the minimum requirements for real timing processing.

Among the other compared solutions, the architecture designed in this paper presented the best results in terms of frame rate, frequency and use of cycles per macroblock.

Concerning the next steps of this work, it is intended to prototype the filter architecture and to integrate it to an upsampling module developed in our group, in order to create the Interlayer Intra Prediction block of H.264/SVC. Also, it is intended to apply the new filtering order to the H.264/AVC Deblocking Filter.

### 7. REFERENCES


