Reuse of Flexible Hardware Modules for Practical Implementation of Intra H.264/SVC Video Encoder

Ronaldo Husemann, Altamiro Amadeu Susin
Depart. Engenharia Eletrica – DELET – UFRGS
Av. Osvaldo Aranha, 103 – Porto Alegre – RS, Brazil
rhusemann@inf.ufrgs.br, altamiro.susin@ufrgs.br

Ricardo Kintschner, José Valdeni, Valter Roesler
Instituto de Informática – II – UFRGS
Av. Bento Gonçalves, 9500 BI IV – Porto Alegre, Brazil
kuronno@gmail.com, {valdeni, roesler}@inf.ufrgs.br

Abstract—The practical implementation of a scalable video encoder requires for very high processing demands. In particular, the H.264/SVC is an emergent standard which combines distinct complex techniques in order to remove redundant data among consecutive layers, impacting in the global encoder complexity increasing. In order to evaluate that, this paper presents a detailed analysis of the required demands of a practical H.264/SVC video encoder. Considering these demands, it is proposed here an innovative approach, which uses flexible computational hardware modules in order to perform iteratively the SVC intra computational coding, for both the base layer and enhancement layers. The proposed solution was implemented in VHDL and compared with other conventional hardware approaches, confirming significant memory and used chip area savings. The aim of this proposal is to contribute to the research community with an innovative and practical solution for the development of scalable video encoders.

I. INTRODUCTION

When H.264/SVC (Scalable Video Coding) standard was finally published in October 2007, after four years of exhaustive efforts of JVT (Joint Video Team) members, the research community knew the most efficient scalable video coding solution. This new standard represents a flexible solution, responsible by breaking the video stream in several layers, in order to provide media adaptability in modern digital multimedia applications, mainly when applied over heterogeneous networks (mobile TV or videoconference) or non-deterministic communication channels (IPTV) [1].

H.264/SVC combines distinct algorithms in order to optimize data reusability among consecutive layers, and consequently improving the global encoding efficiency [2]. Moreover the SVC supports distinct scalability strategies (temporal, spatial and quality or SNR), as alternatives to increase the global encoder flexibility.

Unfortunately, even considering the current state of art in multi-core microprocessors, a complete H.264/SVC encoder implementation, adopting only software approaches, is still impracticable, due to the extremely high computing demands, which are imposed when real-time multi-layer video coding is needed (multiple encoders in the same architecture). In fact, the most adequate solution for this complex codec basically leads to the utilization of some kind of hardware acceleration, as, for instance, dedicated chips like ASIC or programmable logic IP cores [3].

Considering that, this paper presents the analysis of a conceptual SVC encoder model, identifying the most relevant limitations for hardware architectures.

Based on that, we propose an innovative approach which uses flexible hardware modules, specially designed in order to perform the H.264/SVC intra computational coding (forward and inverse transforms and quantization algorithms). The new proposal was implemented in VHDL and compared with a conventional scalable encoder solution in order to illustrate the effective proposal’ gains in terms of chip area and memory utilization.

This paper is organized in the following way: section 2 presents a brief view of the H.264/SVC encoder, section 3 describes the proposed approach, Section 4 shows the results comparing it with some a conventional hardware approach while section 5 presents the authors’ final considerations.

II. BACKGROUND

The emerging H.264/SVC standard has been developed as an addendum of the H.264/AVC codec expanding the conventional single layer to a multi-layer solution. In this environment the first layer (base layer) is responsible for the minimal video resolution, while the other ones (enhancement layers) are responsible for the gradually improvement of the video, adding complementary information [1].

Video sequences are divided in three frame types: frames I (Intra), P (Predicted) and B (Bidirectional). I frames are basically compressed by Transforms (DCT and Hadamard), Quantization and Entropy (CA VLC or CABAC) algorithms and can be used as reference to another predicted frames. Frames P depend temporally of I frames, reusing data from them in order to reduce the total bit stream. B frames are more flexible, using previous and/or posterior I, P or B frames as

This research has been financially supported by the entity FINEP.
references. SVC codec introduces the ability to identify and reuse information from one layer to another, i.e., data from distinct layers within the same temporal frame can be reused in an innovative vertical prediction process [3].

In order to illustrate that, we consider the example of a simplified three-layer encoder as depicted in Figure 1. We can observe that this solution includes three parallel encoders (one for base layer and two for the enhancement layer). Two scalabilities types are supported: SNR scalability (in enhancement layer 1) and Spatial scalability (enhancement layer 2), when additional upsampling and downsampling modules are required. Information from each layer is stored in a shared video memory, in order to allow reusability by the immediately superior layer encoder. All individually coded data of each layer are sent to multiplexer module, responsible for synchronization and composition of the final stream.

III. PROPOSED SOLUTION

Aiming at the implementation of an intra H.264/SVC encoder we propose an innovative solution which explores the particular data dependency among consecutive layers in order to operate recursively over the same hardware modules, reducing significantly the global encoder complexity [2].

Based on this approach, the computational (forward and inverse) modules are kept running immediately after each macroblock encoding, in order to process the subsequent layer. This process is repeated continuously until the last layer (from the base layer until the highest enhancement layer). This iterative proposal presents two important advantages when compared with conventional SVC solution: (i) reusing of hardware modules, keeping the same modules running continuously and (ii) reduction of memory demands, since data information from each layer after generation can be immediately consumed for the consecutive layer and released for reuse. It is also important to note that in this approach only small quantities of memory are necessary (macroblock) as explained later. In the following sections this proposal is described in more details.

A. Global Architecture

In this work we considered the particular features of data dependency inside a H.264/SVC encoder to propose an innovative architecture that allow hardware and memory reusing. Based on that, the proposed architecture was designed to support the main algorithms for H.264/SVC intra coding, i.e., forward and inverse computational algorithms (DCT, Hadamard and Quantization). In Figure 2 this proposal is depicted.

In order to reach real-time capability traditional hardware scalable approaches incorporate several encoders running in parallel in the same architecture (each one responsible for one layer).

However, even when a complete multi-encoder solution is implemented the overall performance is limited, due to the dependence of data from consecutive layers. i.e. the enhancement layer 1 can be processed only after the base layer coding is finished (the same for the others layers), preventing the ideal parallel computing. It is important to note that the increased complexity of this scalable approach. For instance, considering that a medium grained SVC solution (MGS) should supports up to 16 distinct layers, the implementation of a complete parallel multi-encoder architecture becomes very complex, requiring 16 parallel encoders. Also this layered structure impacts in additional and intensive requirements for the on-board memory, inducing undesirable bottlenecks [4].

III. PROPOSED SOLUTION

Aiming at the implementation of an intra H.264/SVC encoder we propose an innovative solution which explores the particular data dependency among consecutive layers in order to operate recursively over the same hardware modules, reducing significantly the global encoder complexity [2].

Based on this approach, the computational (forward and inverse) modules are kept running immediately after each macroblock encoding, in order to process the subsequent layer. This process is repeated continuously until the last layer (from the base layer until the highest enhancement layer). This iterative proposal presents two important advantages when compared with conventional SVC solution: (i) reusing of hardware modules, keeping the same modules running continuously and (ii) reduction of memory demands, since data information from each layer after generation can be immediately consumed for the consecutive layer and released for reuse. It is also important to note that in this approach only small quantities of memory are necessary (macroblock) as explained later. In the following sections this proposal is described in more details.

A. Global Architecture

In this work we considered the particular features of data dependency inside a H.264/SVC encoder to propose an innovative architecture that allow hardware and memory reusing. Based on that, the proposed architecture was designed to support the main algorithms for H.264/SVC intra coding, i.e., forward and inverse computational algorithms (DCT, Hadamard and Quantization). In Figure 2 this proposal is depicted.

We can note in Figure 2 that a dedicated dual-port FIFO is used as interface for external modules. Also additional DRAM based memories are necessary to store information during iterative operation. The memory “Original MB” is responsible for saving the original input video macroblock information, while the memory “Recovered MB” saves data which are generated after each encoding/decoding iteration. The proposal operates with macroblock granularity, i.e., after receiving each group of 16x16 pixels it computes the base layer and the subsequent layers. After each iteration “Recovered MB” memory is released to be overwritten with new information reducing the memory consumption (only a small part of the whole frame is demand each time).
The key element in this architecture is the control FSM (Finite State Machine) called as SVC Layer Manager. This machine coordinates all other modules, controlling and synchronizing data transfers (defining when and where each memory can be read or written) as explained next.

B. Algorithm of the SVC Layer Manager Module

Figure 3 presents the algorithm supported by the SVC Layer Manager in order to implement the proposed iterative H.264/SVC multi-layer encoding.

After the first initialization, when global information is read (number of layers, type of data, and values of each layer QP) the module starts transferring data from the input FIFO to the “Original MB” memory and indicate the presence of new valid data on it. In the first iteration data from this memory is directly used for base layer encoding. As soon as the first data in the Computational Inverse module is detected the “Recovered MB” memory is rewritten, preparing the second iteration. When the base layer data counter reaches the size of a macroblock the first enhancement layer can be processed. Then the residual information is computed (original – recovered data) and signaled as valid input data in the computational forward module.

For each new iteration a distinct QP value must be computed (enhancement layer), resulting in a different quality level. When new data is detected in the computational inverse module, it is again overwritten in the Recover MB Memory, reducing even more the residual information. This iterative process is repeated until the last enhancement layer.

C. Hardware Computational Modules

In order to reach the desired performance for a practical SVC implementation, it was necessary to use dedicated hardware modules, which were especially designed to perform the intra computational algorithm with increased data throughput [5]. Basically six modules are needed to implement a complete H.264/SVC intra coding, where three are used for direct coding (DCT, Hadamard and Quantization), while the others ones are responsible by the inverse operation (iDCT, iHadamard and Dequantization). The efforts for speeding up this module data throughput are particularly important for the adopted recursive approach, where the global performance is directly proportional to the main loop timing efficiency.

All modules are designed to operate adopting pipelined structures, in order to reduce the overall work periods. Another important feature of these modules is the alignment of the internal data bus with external memory path width, reducing delays related with memory accesses. Considering that all modules were designed to support 64-bits data path, which was imposed by the necessity to process simultaneously eight samples per clock (8x8 = 64bits).

The first module (DCT) was designed to operate in a three-step approach: (i) linear transformation, (ii) data transposition and (iii) another linear transformation [6]. Each DCT linear transform is implemented by a two-stage pipeline, while the transposition module requires a four-stage pipeline. Based on that, our complete DCT solution has a total latency of eight stages. Moreover, considering that H.264/SVC specifies the DCT to operate over 4x4 pixel blocks we duplicate internally this module, consuming in parallel two pixel lines of 4 samples or 8 pixels per clock.

The next module (Hadamard) adopts a similar approach to implement the effective transformation. However the Hadamard algorithm must do be applied only over a particular matrix composed by the first coefficients of each 4x4 block after DCT encoding. Two internal memory buffers were included in order to support 8-samples input without delays: the first one (MB buffer) is used to store the original macroblock data, while the other (HAD buffer) stores the values really updated by the Hadamard transformation. A final module (Hadamard Composer) reads both buffers in parallel, synchronizing the Hadamard module output.

The same methodology here presented was adopted for the inverse transforms: DCT (iDCT) and Hadamard (iHAD), e.g. a duplicated approach is proposed to support simultaneously 8-samples vectors [7].

Finally, our Quantization module was designed in a three-stage pipeline processing simultaneously four samples per clock. In the first stage QP-based parameters are determined by specific lookup tables (QP Tables) while four multipliers are used to rescale the input samples [3]. In the second stage, adequate offset values are added to the first stage multiplications results (round operation) while the quantity of quantization bits (qbits) is determined. The final stage composes the output data based on a qbits-based shifting operation (integer division). Due to the complex operations of multiplication (first stage), quantization is the slowest module...
of our architecture, which can limit significantly the overall encoder performance. In order to avoid that constraint we designed a fast solution with four Quantization modules, working in a master-slave approach, i.e., in the first clock two modules are feed (processing simultaneously eight pixels), while in second clock the remainder modules are feed. This parallel approach increased the occupied area, but in compensation it duplicated the global encoder efficiency.

The Dequantization module adopts a similar approach, but occupying only two stages (round operation is not necessary here). Dequantization module replication is also performed in order to support parallel processing of eight samples per clock.

More details about these hardware modules can be found in [6].

IV. RESULTS

Before starting the validation of this proposal it is crucial to identify the real demands for a real-time intra H.264/SVC encoding. In terms of data throughput worst case we can consider a scalable application of HDTV video with 16 layers (limit for SNR scalability). If the video sequence, in this case, is configured in 4:2:0 format at 30 fps, it will represent a theoretical demand of 995Mpixels/s or 1493MSamples/s (considering luma and chroma components) for the encoder.

This extremely high demand (almost 1.5GSamples/s) was considered during the design of our flexible computational modules, mainly in order to allow reuse of them for multiple layers. In Table I our proposal is compared with related works [7][8][9], which also implement complete intra computational modules dedicated for HD videos and synthesized to XC2VP30 FPGA.

TABLE I. COMPUTATIONAL MODULES COMPARISON

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Computational Forward Modules</th>
<th>Solution</th>
<th>LUTs</th>
<th>Period (ns)</th>
<th>MSamples/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (x16)</td>
<td>Logashanmugam [7]</td>
<td>546</td>
<td>12.54</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>B (x16)</td>
<td>Korah [8]</td>
<td>3924</td>
<td>12.9</td>
<td>330</td>
<td></td>
</tr>
<tr>
<td>Our Forward Proposal</td>
<td>5343</td>
<td>4.64</td>
<td></td>
<td>1724</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Computational Inverse Modules</th>
<th>Solution</th>
<th>LUTs</th>
<th>Period (ns)</th>
<th>MSamples/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Agostini [9]</td>
<td>3249</td>
<td>7.54</td>
<td>132</td>
<td></td>
</tr>
<tr>
<td>Our Inverse Proposal</td>
<td>3359</td>
<td>4.64</td>
<td></td>
<td>1724</td>
<td></td>
</tr>
</tbody>
</table>

Our proposal occupies more area, but, on the other hand, it presents a data throughput compatible with SVC demands. The main reason for these great differences is that all related works (scenarios A, B and C) were originally designed for single-layer operation. However, as already commented in Section 2, these modules can be rearranged in a multi-encoder approach to support complex scalable applications. So, generally speaking, a technical solution for that worst scalable case can be obtained replicating 16 times (number of layers) these single layer computational modules (Table 2).

TABLE II. SCALABLE HARDWARE SOLUTIONS COMPARISON

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Computational Forward Modules</th>
<th>Solution</th>
<th>LUTs</th>
<th>Period (ns)</th>
<th>MSamples/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (x16)</td>
<td>Logashanmugam [7]</td>
<td>8736</td>
<td>12.54</td>
<td>1264</td>
<td></td>
</tr>
<tr>
<td>B (x16)</td>
<td>Korah [8]</td>
<td>62784</td>
<td>12.09</td>
<td>5280</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Computational Inverse Modules</th>
<th>Solution</th>
<th>LUTs</th>
<th>Period (ns)</th>
<th>MSamples/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Agostini [9]</td>
<td>51984</td>
<td>5.74</td>
<td>2112</td>
<td></td>
</tr>
<tr>
<td>Our Complete Proposal</td>
<td>11485</td>
<td>4.64</td>
<td></td>
<td>1724</td>
<td></td>
</tr>
</tbody>
</table>

We can note that the “A scenario” solution, even when 16x replicated, does not support the demand of 1493MSamples/s. Considering that, only scenarios B and C were used to compose a complete scalable computational architecture. This solution, called as “B+C scenario”, supports more than 2Gsamples/s, guaranteeing the worst case SVC demand. However the total area occupied for this multi-encoder is very high. Compared with that, our complete proposal, which includes computational modules, MB memories and SVC Layer Manager (Figure 2), occupies much less area, also supporting the maximal required scalable data throughput.

V. CONCLUSIONS

In this paper we propose an innovative approach, which explore reusing of flexible computational hardware modules to implement the SVC intra computational coding, including forward and inverse algorithms of DCT, Hadamard and Quantization. Our proposal supports the processing of more than 1.7Gsamples/s, which is enough to support the theoretical worst case for SVC intra encoding, consuming a chip area approximately ten times smaller than a conventional multi-encoder scalable architecture.

REFERENCES