Analog Integr Circ Sig Process (2015) 82:135–146
DOI 10.1007/s10470-014-0342-9

Hardware-friendly HEVC motion estimation: new algorithms and efficient VLSI designs targeting high definition videos

Gustavo Sanchez · Bruno Zatt · Marcelo Porto · Luciano Agostini

Received: 31 October 2013 / Accepted: 21 May 2014 / Published online: 14 June 2014

Abstract In this article the spread and iterative search (S&IS) and the low density and iterative search (LD&IS) motion estimation algorithms are presented. The proposed algorithms are hardware-friendly because they have a regular number of cycles to encode a block, they have a regular memory access and the parallelism can be better explored when compared to other published solutions. The proposed algorithms were evaluated under the high efficiency video coding reference software and compared with the state-of-art enhanced predictive zonal search (EPZS) algorithm and with the well know diamond search (DS) algorithm. The designed algorithms presented better BD-rate results than DS but they presented some losses when compared to EPZS. Since EPZS is not focused in a hardware design, it presents a lot of data dependencies, a irregular memory access, a non-deterministic number of operations to encode a block and it must store motion vectors of previously encoded frames. All these features are undesirable for many applications, including those ones focused on battery powered devices. To show the efficiency of the proposed algorithms, two architectures were designed targeting these algorithms and they were synthesized for Altera Stratix 4 and for ASIC using TSMC 90 nm standard-cells technology. Synthesis results show that the architectures are capable to process HD 1080p videos, at real time, by obtaining good results in terms of hardware resources use, power consumption and processing rate when compared to related works. The ASIC implementations are able to guarantee real-time performance for high definition videos consuming 12.5 and 13.5 mW, respectively. These results are possible because S&IS and LD&IS were developed focusing on hardware design.

Keywords Hardware-friendly motion estimation · Hardware design · High efficiency video coding · Fast motion estimation

G. Sanchez (✉) · B. Zatt · M. Porto · L. Agostini
Group of Architectures and Integrated Circuits, Federal University of Pelotas, Pelotas, Brazil
e-mail: gfsanchez@inf.ufpel.edu.br

B. Zatt
e-mail: zatt@inf.ufpel.edu.br

M. Porto
e-mail: porto@inf.ufpel.edu.br

L. Agostini
e-mail: agostini@inf.ufpel.edu.br

1 Introduction

The complexity to encode a digital video has been increasing during the last few years. The main reason for this increase in complexity is explained by the set of novel features brought by the latest video coding standards H.264/AVC [1] and high efficiency video coding (HEVC) [2]. The new features were designed targeting on the increase of coding efficiency without considering the real-time feasibility and hardware design a major design constraint. In current technology, software solutions able to encode high definition (HD) videos in real time employ simplified versions of encoders incurring in coding efficiency losses. Moreover, these solutions spend increased computational resources by demanding powerful processors, which results in high power consumption. Thus, software solutions are not feasible for embedded systems such as a smart-phone or a digital camera. In practice, to encode a HD video in real time, hardware solutions are desired to maximize the results in terms of hardware cost, coding efficiency and power consumption.

On the current video coding standards, the motion estimation (ME) is the major responsible for the compression gains among all encoder tools. However, it is also the encoding tool that spends the most computational effort. In fact, ME represents 80 % of the current encoder complexity [3].

The ME process is based on a search algorithm and a comparison metric (similarity criterion). The ME search algorithm defines how the search for the best matching block will be done (what blocks will be evaluated), considering the candidate blocks inside the search area. For each compared block, the similarity criterion is calculated and the most similar block is selected. The sum of absolute differences (SAD) is the most common criteria used to measure the block similarity [4, 5] [6] and, for this reason, we will refer to a block matching operation in terms of SAD computations. An efficient ME algorithm must be capable to find the best possible matching while comparing the smallest number of candidate blocks.

Current video encoders do not restrict how the ME algorithm is performed. This freedom allows the designers to develop algorithms that better suit to the target application or underlying systems. When real-time systems for high-definition video encoding are considered, hardware acceleration is needed. Therefore, besides providing high efficiency and complexity reduction, the ME algorithms are desired to present hardware-friendly characteristics such as a constant number of operations per coded block and a small internal memory.

The exhaustive ME algorithm is known as full search (FS) [5]. The FS is the most intuitive algorithm; nevertheless it is the most computationally costly in terms of number of SADs computations [7]. The FS compares all candidate blocks of a given search area, always finding the optimal result. This high complexity is undesired due to the difficulty to deliver performance to reach real-time processing for HD videos. As a consequence, FS drastically increases the power consumption. Considering this scenario, the use of fast ME algorithms is mandatory to reduce the complexity even in spite of the possible coding efficiency losses.

Fast algorithms usually explore different search patterns and/or heuristics based on video characteristics to achieve a good trade-off between complexity and objective video quality. A large set of research works discuss and proposed different alternatives of fast ME algorithms by varying the geometric shapes resulting in algorithms such as diamond search (DS) [4, 8], hexagon search [9], three step search (TSS) [10], spiral search [11], among others. However, most of these algorithms present weaknesses when dealing with local minima, especially when encoding HD videos [12]. Local minima are good matching candidate blocks that trap fast algorithms preventing them to find the global minimum.

To improve the geometric shape-based algorithms, some level of intelligence was inserted in the state-of-the-art algorithms, such as the enhanced predictive zonal search (EPZS) [13], by including start point prediction and early termination (see Sect. 2). Still, the predictors employed can be inefficient limiting the algorithm results, especially for high motion and HD videos, due local minima selection. Note, the higher the resolution and the motion intensity, the harder to find the optimal matching block. Therefore, it is desired that an ME algorithm contains characteristic to help the search to avoid local minima by, for instance, inserting some level of random behavior.

In this work we present two ME algorithms, the spread and iterative search (S&IS) [14] and the low density and iterative search (LD&IS) [15], employing a spread-evaluation approach and hardware friendly characteristics. Based on these algorithms, two hardware architectures targeting HD (1,920 × 1,080) videos real-time encoding are presented.

- *S&IS & LD&IS* The S&IS and LD&IS are composed of two main steps: (i) a spread evaluation where many blocks are evaluated far from the co-located block, which is responsible to avoid local minima when encoding high motion/definition videos; and (ii) a central iterative evaluation where a DS is applied in the co-located block, which is important to achieve good results when encoding low motion/definition videos. As the spread evaluation is data-dependencies free and allows the exploration of parallelism; thus, adequate for hardware implementation.
- *ME algorithms evaluation* In this work a complete new strategy is employed to allow the evaluation of the two proposed algorithm under the HEVC encoder environment. All results were obtained through the HEVC reference software—HEVC Model (HM) 5.0 [16]—respecting the recommended common test conditions (CTC) and comparing against state-of-the-art search algorithms.
- *Hardware architectures* Exploiting the parallelism inherent to the S&IS and LD&IS algorithms, two hardware architectures are presented. The architectures are composed of a pseudo-random number generator, a local memory, a memory read/write controller and a set of processing units. Also, the designs have been described in VHDL, and synthesized for an Altera Stratix 4 FPGA and also for TSMC standard cell 90 nm technology focusing on real-time processing HD 1080p and a low power consumption.

This article is divided as follows: Sect. 2 describes the related works. In Sect. 3 the S&IS and the LD&IS algorithms are presented. Details about the experimental setup, our software evaluations and comparisons with DS and EPZS are

presented in Sect. 4. Section 5 presents the VLSI design for the architectural template used in S&IS and LD&IS architectures design. Details about each architecture implementation are also presented in Sect. 5. In Sect. 6 it is presented the synthesis results for both LD&IS and S&IS architectures followed by comparisons with other works in the literature. Finally, Sect. 7 renders the conclusions.

2 Related works

State-of-the-art fast ME algorithms [17] employ heuristics during the prediction process to speed-up the searching for the best matching region in the search area within the reference frame. By doing so, the fast ME algorithms meaningfully reduce the computational complexity when compared to the exhaustive ME algorithm, i.e. the FS algorithm. Another approach, used to accelerate the ME, is the early termination technique [17], that stops the search process when the residue obtained by the ME prediction is lower than a predefined threshold. Although early termination algorithms provide good reduction in the number of compared blocks while sustaining coding efficiency, the number of operations to encode a block is not predictable. A variable non-deterministic number of operations to encode a block is not desirable when designing a hardware solution designed to guarantee real-time processing.

Predictive motion estimation algorithms (PMEA) [17] have been widely used in H.264/AVC and HEVC reference software due to their low complexity and high efficiency. PMEA algorithms store motion vectors (MVs) that were used in past frames and use them to define the initial search points for ME algorithms, resulting in a good video quality at a reduced processing effort. The main idea of PMEA is inspired in Newton's first law which defines that an object either at rest or constant movement should keep the same state unless acted upon by an external force. Bringing this idea to video coding, the MV of two neighbor frames tends to be similar. Another mechanical concept that inspired the PMEA development is that an object that is accelerating should accelerate its motion in the next time instant with a similar acceleration factor. These principles are exploited, respectively, by using as a predictor the same MV of the past frame or based on the acceleration of the past two frames. PMEA-based solutions, however, are not hardware-friendly due irregular number of operations per encoded block, and the need to store MVs of previously encoded frames (resulting in additional memory) to perform the initial evaluation of the future ones. For instance, the memory resources necessary to store the MVs of the past two frames of a HD 1080p video is around 226,800 bits. More details about this memory estimation are presented in Sect. 4.

The EPZS [13], which is a PMEA, decides to best search start point among the following options: the co-located block vector; the median of the co-located block MV and its neighboring MVs; or a vector representing the motion acceleration calculated using the past two co-located blocks. PMEA algorithms, such as EPZS, provide good coding efficiency with reduced complexity. However, these techniques lead to unpredictable number of operations and irregular memory access patterns. Additionally, EPZS may implement early termination further increasing the variation in the number of comparisons required.

There are other works in literature such as [12] and [18–23] that presents a hardware design for ME algorithms and some are capable to encode HD videos in real-time. However, most of them do not presents the ME quality for HEVC and just present the quality evaluation for low definition videos. For these reasons, there is a need for novel ME algorithms that consider hardware–related challenges targeting real-time encoding systems.

3 S&IS and LD&IS algorithms

In this section the S&IS and LD&IS algorithms are presented. The main goal of both algorithms is to facilitate the hardware design, providing a good trade-off between performance, hardware resources utilization and objective quality when encoding HD videos. The quality gain is reached by the use of techniques to reduce the choice of local minima in comparison with other fast algorithms. These techniques are hardware friendly, since present no data dependencies, regular number of operations and regular memory access. Note that by regular memory access we refer to the fact that all accesses are performed within the search window.

Both algorithms can be divided in two steps: a spread evaluation and a diamond-shaped central iterative evaluation. As high motion activity videos tend to have MVs far from the central position of the search area, the spread evaluation is responsible to evaluate blocks far from the co-located block in the search area. Low motion activity videos tend to have MVs near to the central position of the search area and the central evaluation is responsible to evaluate these near blocks.

In the central evaluation, a DS algorithm is applied in the co-located block of the search area. In face to our hardware-friendly objective, the DS main weakness is its non-determinist number of cycles to encode a block. However the DS performs, on average, only 4 iterations and it is possible to limit the iterations without losing much coding efficiency [7]. In the proposed algorithms the DS has been limited to five iterations. The DS is the iterative part of the proposed algorithms and this guarantee that the S&IS and LD&IS will have, at least, the same quality results when compared to the DS.

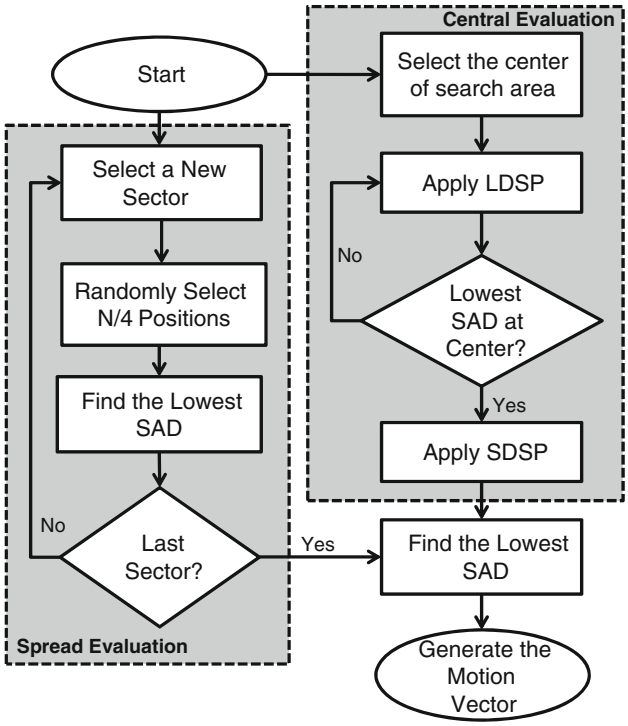

Fig. 1 S&IS algorithm

Combining both spread evaluation and central evaluation in one algorithm makes the proposed algorithms capable to encode videos with different characteristics with a high encoding quality.

3.1 Spread and iterative search (S&IS)

Figure 1 presents the S&IS flowchart. The S&IS works as following: (1) the search area is divided in four sectors. Each sector represents one quadrant around the origin position centered at the co-located block; (2) *N*/4 spread positions are pseudo-randomly generated for each sector, totalizing *N* spread position; (3) the SAD is computed for each one of these positions and the best one is selected; (4) an iterative process using DS algorithm is started at the center of the search area; finally, (5) the results obtained by step (3) and step (4) are compared and the final MV is generated.

A pseudo-random numbers generator is used to generate *N* spread positions inside the search area and then the SAD is computed for these *N* positions. These SADs are compared and the lowest one is selected. This is the spread block evaluation part of S&IS.

The insertion of the random spread blocks evaluation does not increase much the ME complexity if compared with DS. Combining the random evaluation and DS into the S&IS algorithm, it is possible to achieve quality gains without high losses in terms of complexity. The S&IS complexity is dependent on the *N* parameter. According to previous analysis [14], when targeting HD videos, the best *N* that maximizes the tradeoff among complexity and video quality is 96.

It is possible to observe in Fig. 1 that there are no data dependencies between the spread block evaluation and the central iterative evaluation. This characteristic is important when designing a hardware solution to achieve the maximum possible efficiency because the central evaluation and the spread evaluation can be calculated in a parallel or in a sequential fashion according to the designer interest. In our hardware solution (presented in Sect. 5) the parallelism was explored in the hardware design to increase the processing rate. The DS is an iterative algorithm with data dependencies among the iterations. When the parallelism is explored in the hardware design of S&IS algorithm, the iterative DS step is the limiting factor of the overall throughput, since all the SAD calculations of the spread evaluation can be done in parallel. Then the hardware design of spread evaluation can be constructed to reach the same processing rate than that obtained by a single DS solution.

3.2 Low density and iterative search (LD&IS)

The LD&IS follows the same idea of S&IS to perform a spread evaluation and an iterative evaluation around the co-located block in the search area. The difference between these algorithms is in the step responsible to generate the positions that will be evaluated. In S&IS these positions are generated by a pseudo-random generator. In LD&IS these positions are pre-fixed before the start of the encoding process. In LD&IS it was used 100 fixed positions covering the whole search area. The search positions used by the multi-point step are 5, 10, 20, 30 and 40 samples away from the center to each side, in both x and y axis. The proposed search area for this algorithm is 96 × 96 pixels, which represents a density of 0.011 computed blocks per pixel in the search area for a 16 × 16 block size. Moreover, this density has been used in our evaluation because it presented the best trade-off between complexity and efficiency [15]. However the proposed algorithm can be easily adapted to a higher density number of compared blocks.

One of the main advantage on using LD&IS compared to S&IS is the perfectly regular memory access. LD&IS memory accesses are pre-defined in the beginning of the encoding process, while in S&IS those positions are randomly generated.

4 Experimental setup and evaluations

4.1 Experimental setup

The S&IS and LD&IS algorithms were evaluated on the following test video sequences defined at CTC: Traffic,

Table 1 Simulation results using BD-rate measurement compared to EPZS algorithm

Algorithm	DS				LD&IS				S&IS			
Videos	Y (%)	Cb (%)	Cr (%)	Enc. time (%)	Y (%)	Cb (%)	Cr (%)	Enc. time (%)	Y (%)	Cb (%)	Cr (%)	Enc. time (%)
Traffic (2048 × 1152)	7.53	7.13	7.93	97.04	5.88	5.16	5.80	111.70	4.54	3.80	4.32	119.30
BQTerrace (1080p)	3.46	3.05	2.80	94.89	2.34	1.15	1.55	107.26	2.09	1.17	1.73	114.06
ChinaSpeed (1024 × 768)	9.20	10.05	10.90	86.33	7.50	7.48	7.92	99.08	7.37	7.18	7.58	104.38
RaceHorses (832 × 480)	18.67	23.94	25.75	85.37	14.75	16.81	17.80	95.87	12.60	14.13	15.90	99.96
BQSquare (416 × 240)	0.24	0.07	0.82	97.84	0.23	0.19	0.92	111.19	0.39	−0.11	0.45	116.68
Average	7.82	8.85	9.64	94.26	6.14	6.16	6.80	107.69	5.40	5.23	6.00	114.48

BQTerrace, ChinaSpeed, RaceHorses, and BQSquare. The video resolution of each video sequence is presented in Table 1. These videos have been chosen for these experiments because it covers many different resolutions and video characteristics, such as motion activity, light and texture (high, medium and low) which are important to demonstrate the robustness of the algorithms. Four different quantization parameters (QP) have been evaluated: 22, 27, 32 and 37. The default parameters of HEVC reference software were used for Random Access mode according to CTCs.

The DS, S&IS and LD&IS have been implemented in HM 5.0. The S&IS and LD&IS central evaluation are limited to five iterations and the N used is 96 points (24 per sector) and 100 points (25 per sector), respectively.

In the implementation of DS, S&IS and LD&IS were not optimized focusing on performance. Those optimizations were not performed because it was not the focus of this work to show how fast our software solution can compute compared to EPZS but our intention is to show that the quality obtained with S&IS and LD&IS algorithms and the possible time results related to our architecture design. The complexity of S&IS and LD&IS are explained in our hardware development in Sects. 5 and 6.

4.2 Simulation results

The well-accepted BD-rate measurement [24] is used to evaluate the coding performance. Table 1 presents the BD-rate and time results for DS, S&IS and LD&IS in comparison to EPZS. Analyzing the time results, the DS was capable to obtain a complexity reduction of 5.74 % in comparison to EPZS. Considering the LD&IS and the S&IS, the proposed algorithms increased the complexity by 7.69 and 14.48 % when compared to EPZS, respectively. However, the proposed algorithms implementations were not focusing on performance optimization.

Figure 2 presents a PSNR–bitrate curve plotting the results obtained in traffic sequence for the four evaluated ME algorithms. The behavior of this curve is similar for the others video sequences. As previously expected, the EPZS obtained the best results among the evaluated algorithms. In comparison to DS, LD&IS and S&IS in average the EPZS obtained a bitrate reduction of 7.82, 6.16 and 5.40 % considering the same quality (considering the luminance component). The best case results occurred in BQSquare and the worst case results occurred in RaceHorses.

In BQSquare, which is a low resolution and low motion activity video, the bitrate reduction obtained by EPZS is inexpressive when compared to any algorithm. This happens because for low motion activity videos a good MV tends to be obtained around the center of the co-located block. The DS exploration around the center obtains good results in the kind of videos, which makes the central evaluation of the proposed algorithms obtain good MV. Adding some evaluations far from the co-located block in this case only increases the encoding effort. The BD-rate of the proposed algorithms are very close to EPZS in this case.

RaceHorses and ChinaSpeed are medium resolution and high motion activities videos. In this case, the bitrate reduction obtained by EPZS is meaningful. In this case, good MVs can be found far from the co-located block and the DS cannot transpose local minima around the central position, which makes the DS efficiency much lower than EPZS. The EPZS reduction in bitrate when compared to S&IS and LD&IS is explained because the EPZS uses a MV predictor, which guides immediately the EPZS search to an area that contains good MVs. The RaceHorses video, in which it was obtained the worst case results, is the video with higher motion activities characteristics, which is the video that the ME should face the highest problems during the encode process.

In Traffic and BQTerrace, which are high resolution and low motion activities videos, the EPZS efficiency is still

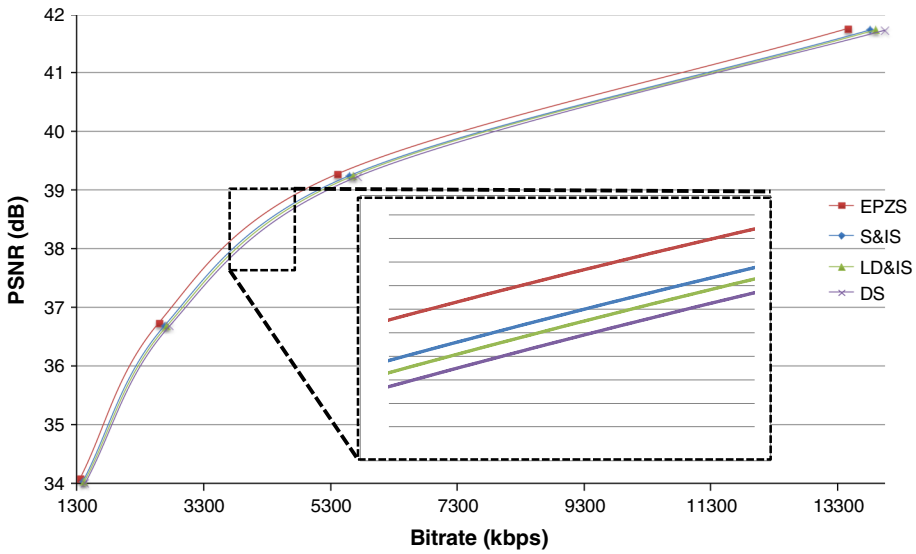

Fig. 2 PSNR-bitrate curve comparing EPZS, DS, S&IS and LD&IS

considerable because of the high resolution of the videos. In this case, good MV should be found around the co-located block. However, the main problem is the high resolution of these videos, which makes the DS very susceptible to local minima falls. In this case, the efficiency of the proposed algorithms is evident, because of the gains obtained evaluating spread positions in the search area when compared to DS. The S&IS and LD&IS also suffer with the increase of resolution due to the lower density of spread points in the search area; however, the decrease in coding efficiency happens smoother in relation to DS.

The EPZS gains were already expected and occur because the EPZS MV predictor is very sophisticated guiding the search process to areas with a good block match. Still, given the hardware-oriented goals of our development, it is not a good option. In our project decisions, some hardware-friendly characteristics are desired: low variation in the number of operations per encoded block, regular memory access (so data reuse is possible) and reduced internal memory. The EPZS vector predictor has, as the main disadvantage, the need to store all MVs of past two frames. Moreover, once the search window is centered around the best predictor, it is not possible to prefetch this data from external memory nor guarantee the reuse of data between neighboring blocks.

In relation to the MVs memory required to implement EPZS, considering that each MV is represented using seven bits for each axis (which allows a search area range of [−64, +64]), and the block size is 16 × 16 samples, it is possible to determine the minimum size of internal memory necessary to store these vectors. Table 2 presents the estimated memory usage only for the EPZS vector predictor for three resolutions: enhanced-definition television (EDTV) 480p, HD 720p and HD 1080p. For an HD 1080p video (for the configuration described before) it would be necessary 226.8 kbits only for the EPZS MV predictor, for example. Summing up the memory size for MV predictors, the irregular memory access, and the EPZS irregular number of operations per encoded block, a hardware architecture for the EPZS algorithm tends to perform inefficient in terms of the hardware usage and performance.

Table 2 Estimation of memory usage for the EPZS algorithm

Video (resolution)	Memory (bits)
EDTV 480p	37,800
HDTV 720p	100,800
HDTV 1080p	226,800

5 Architectural design of S&IS and LD&IS

Both S&IS and LD&IS architectures use the same architectural template, presented in Fig. 3, that will be described in the first subsection of this section. The next subsection will describe the modifications in the architecture template to design the S&IS architecture and the last subsection will describe the modifications in the template to design the LD&IS architecture.

5.1 Architectural template

Figure 3 presents the architectural template with the main features of both S&IS and LD&IS architectures. This architecture has been designed considering a fixed block size 16 × 16 samples, with a 4:1 sub-sampling rate. A maximum of five iterations was defined for the DS algorithm implemented in the central evaluation and a search area used is 96 × 96 samples.

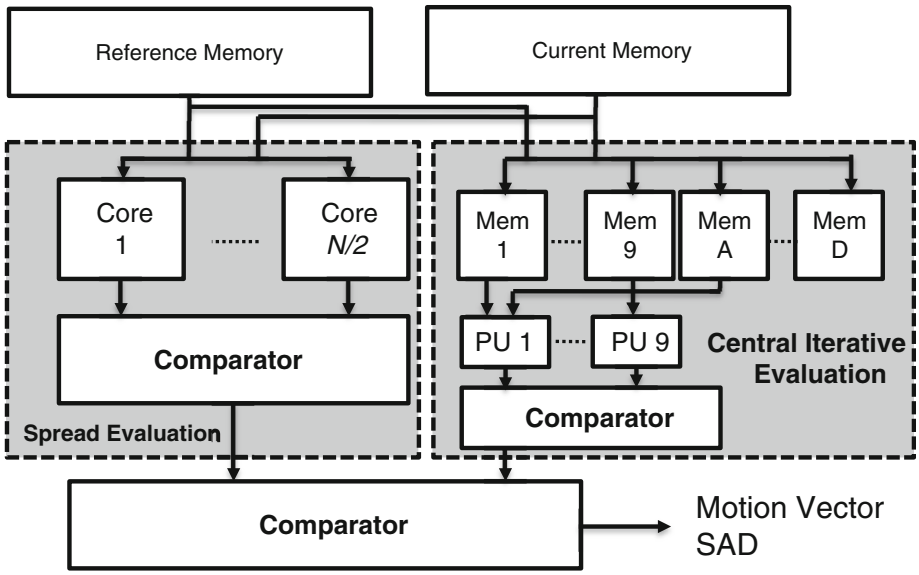

Fig. 3 Spread architectures template

The use of 16 × 16 block size with 4:1 sub-sampling rate is function of the focused resolution. In HD videos the use of small block sizes does not significantly contribute with the encoding efficiency [12], then 16 × 16 or bigger blocks are a good option if only one block size is supported. The sub-sampling is a good option for hardware design since it causes an expressive complexity reduction with an acceptable quality loss [12].

Hardware solutions are usually designed focusing on real-time applications. As the computation time of DS is non-deterministic, because it depends on how many times the DS will iterate, an iteration restriction has been inserted in the central evaluation to guarantee, even in worst case, a high processing rate. Through software evaluation it was possible to determine that by using five iterations it is possible to achieve a good trade-off between video quality and computational complexity.

Figure 4 presents the cycle diagram of the proposed architecture template where it is possible to notice, cycle-by-cycle, the step in which the central iterative evaluation or the spread evaluation are processing. The priority of the proposed hardware solution is to finish the central iterative evaluation before starting to process the spread evaluation. To achieve a better processing rate, the reference memory starts to be written in the central positions, necessary for the first large DS Pattern (LDSP). During this period, the DS local memories start to read data from the Reference Memory and the local memories are filled with the data of the nine positions of the LDSP (Mem 1 to Mem 9 in Fig. 3) and data for the refinement (Mem A to Mem D in Fig. 3). When the local memories are filled with the data to perform a DS iteration, all local memories start to be read line-by-line and the data are sent to the processing units (PU1 to PU9 in Fig. 3), where the SAD computation is performed for each LDSP position. The comparator is responsible to find the lowest SAD among the LDSP results and the position with this lowest SAD is defined as the center for the next LDSP iteration.

The SAD processing unity (PU) receives eight samples per cycle, where each samples is represented with one byte. These eight samples are one line of the subsampled 16 × 16 block. The PU processes this line of samples in four pipeline stages. The comparator receives the SADs calculated by the nine PUs and it is able to define the lowest SAD in five cycles.

Each Core in Fig. 3 is composed by a local memory (8 × 8 bytes) and a controller to select the position that will be written in the local memory and a PU. A half of the N fixed positions are evaluated in parallel, then, the hardware design uses $N/2$ cores.

When the DS is in the processing step during a LDSP iteration, i.e., when it is in PU or in comparison step, the reference memory starts to be read by the spread evaluation. In this case, the complete reference memory positions need to be read. The controller of each core is responsible to select if its local memory should or should not store the information read from the reference memory according if the current core has selected that position to evaluate.

When the DS finishes the first LDSP calculations, the reference memory stops to be read by the spread evaluation and the DS step starts to read the reference memory again to get the necessary data for the next LDSP iteration step. When the DS finishes this reference memory reading (i.e. when it is again in PU or in comparison step), the spread evaluation continues reading the remaining lines of the Reference Memory, finishing the reading of both superior quadrants. In the next DS LDSP iteration, the SADs of these first half positions are computed in parallel with the

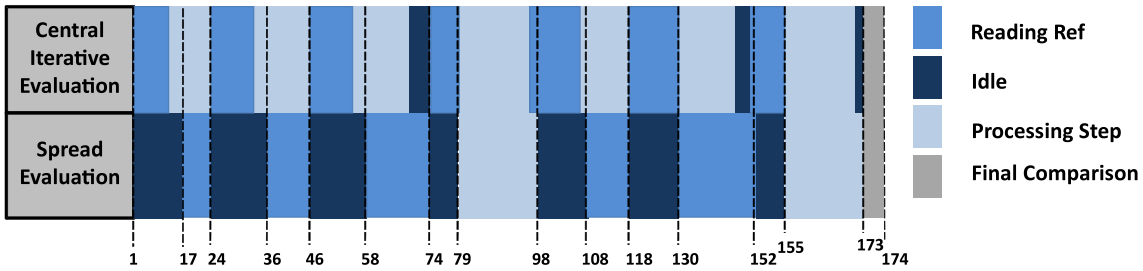

Fig. 4 LD&IS and S&IS cycle diagram

LDSP SAD calculation. Then, the spread evaluation SADs are sent to the comparator of the spread evaluation and the best SAD of these *N*/2 positions is stored. The spread evaluation comparator receives *N*/2 SAD values and it is able to find the best SAD in seven clock cycles.

This interlaced way to read the reference memory is repeated for the fourth and fifth LDSP iterations when the inferior sectors of the reference memory are read. When the small diamond search pattern (SDSP) process is evaluated, the last *N*/2 positions of the spread evaluation are computed. The SADs calculated in this stage are compared with the best SAD of the top reference memory positions and with the best DS SAD and the MV is generated.

To compute a whole 16 × 16 block it is necessary 174 clock cycles as shown in Fig. 4. It is important to notice that the comparator design (in spread evaluation) used by this work allows the usage of 33–64 cores in spread evaluation without changing the number of cycles necessary to process a 16 × 16 block. This allows a freedom for the designer to select the best number of cores that best fits in the proposed application.

Next subsections present the details about the S&IS and LD&IS hardware architectures. Both architectures use the same architectural template, presented in Fig. 3, however, each algorithm needs a specific design for the Spread, and Central Iterative evaluations.

5.2 S&IS architecture

The S&IS architecture has been designed using 96 spread positions, which means that it uses 48 cores. The S&IS core is presented in Fig. 5 and contains a 32 bits linear feedback shift register (LFSR) to generate a pseudo-random number for each core.

In the beginning of the process, the LFRS is signalized to generate a pseudo-random number. When the reference memory is read, if the line that is been read is inside the randomized position, the enable store control signalize the local memory to store the line position of the line that was randomized.

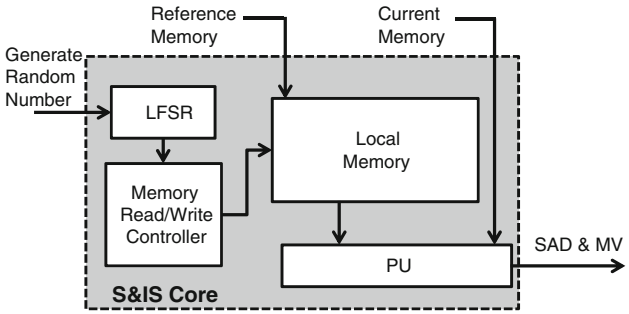

Fig. 5 S&IS core architecture

5.3 LD&IS architecture

The LD&IS architecture has been designed using 100 spread positions, since it was desired that both solutions use similar hardware resources and the core architecture used in LD&IS is simpler than S&IS architecture. The LD&IS core architecture is presented in Fig. 6. The difference between S&IS and LD&IS Cores is that the LFSR are removed in LD&IS architecture and the values are substituted by VCC or GND pins. Comparing to S&IS Core, it was removed a 32 bits register and some logic elements to create the LFSR.

6 Synthesis results and comparisons with related works

The S&IS and LD&IS architectures have been described in VHDL and synthesized for an Altera Stratix 4 EP4S40G 2F40I2 FPGA [25] and for the TSMC standard cell 90 nm technology. Synthesis results are presented in Table 3. For the S&IS architecture, the FPGA results indicate the use of 10 % of the total available ALUTs, 8 % of the total available registers and <1 % of the available memory bits. For the LD&IS architecture, the FPGA results indicate the use of 16 % of the total available ALUTs, 10 % of the total available registers and <1 % of the available memory bits. Both architectures need 174 clock cycles to encode a 16 × 16 block. Since the architectures reached a maximum operation frequency of 254.8 and 210.0 MHz, respectively, it is

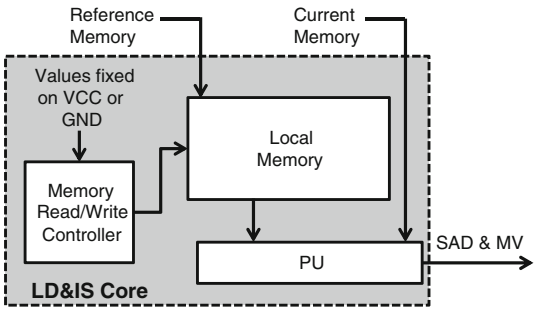

Fig. 6 LD&IS core architecture

possible to process 180 and 149 HD 1080p frames per second for the LD&IS and S&IS architectures, respectively.

The standard cell synthesis for TSMC 90 nm technology was focused on obtaining real-time processing (at 30 frames per second) for HD 1080p videos. Both architectures can reach this performance with a low operational frequency of 42.3 MHz. The low required frequency has a positive impact on the power consumption of both architectures. In fact, LD&IS and S&IS architectures are capable to process HD 1080p videos in real time with a power consumption of 12.5 and 13.5 mW, respectively.

There are many published ME algorithms in the literature, however, only a few of them are focused on hardware encoding of HD videos. Some of them do not consider the coding efficiency degradation at HD due and only evaluate the algorithms for CIF and QCIF video resolutions. Based on this fact, Table 3 presents comparisons with some related works such as [12] and [18–23] that allow a fair comparison. Table 3 brings the following parameters: (1) the synthesized technology, (2) the maximum operation frequency reached, (3) area resources utilization, (4) memory bits usage, (5) cycles to encode a block, (6) performance in frames per second for HD 1080p video and (7) power consumption.

The work [12] proposes a new algorithm and its respective architecture called multi-point diamond search (MPDS) algorithm. MPDS is also designed intending to avoid local minima in HD videos. The block size used is also 16 × 16 with a 4:1 sub-sampling rate. The work [12] uses a smaller area but 46 % more memory bits than the proposed architectures. If on the one hand, [12] can process a block in fewer cycles resulting in lower frequency to encode a HD 1080p video in real time. On the other hand, comparing the quality results, the S&IS and LD&IS are able to provide superior quality results.

A ME architecture for the fast top-winners (FTW) algorithm is proposed in [18]. That work uses the level C data reuse scheme [26], which allows a reduced use of memory bits. Quality results for HD videos are not presented. This architecture is not able to process HD 1080p videos in real time, since its performance for this video resolution is only of eight frames per second. Our work achieves almost the double of operation frequency and uses 7 times less cycles to encode a block than [18], which results in a better processing rate for the proposed architectures. Our architecture consumes about a quarter of the power demanded by [18], the difference of technological node must be taken into consideration though.

The work [19] presents an algorithm for multi-resolution ME (MMEA). That work considers two reference frames in the ME process which is an interesting option to enhance the coding efficiency. The designed architecture is able to process HD 1080p videos in real time at 200 MHz. That work also does not present quality results for HD videos. The work [19] uses less memory bits than our solutions, however the area used in [19] is higher than LD&IS and S&IS architectures. Also, the number of cycles to encode a

Table 3 Synthesis results and comparisons

Architecture	Technology	Frequency (MHz)	Area	Memory (kbits)	Cycles per block	1080p fps	Power (mW)
Porto et al. [12]	Stratix 4	199.2	34.5 KALUTs	53.4	170	174	n.a
Lai et al. [18]	180 nm	83.3	26 KGates	28.7	1282	8	60.8
Yin et al. [19]	180 nm	200	260 KGates	11.3	872	28	n.a
Cetin et al. [20]	FPGA 90 nm	63	33 KLUTS	8 dual-port block RAM	104 (average)	74.7	n.a
Kao et al. [21]	180 nm	154	321 KGates	9.7	631	30	374
Tasdizen et al. [22]	Virtex 5	130	18,2 KLUTs	0.5	467	8.5	n.a
Vanne et al. [23]	130 nm	200	14 KGates	2.5	390/437/680	56	59
LD&IS	Stratix 4	254.8 (max.)	18.5 KALUTs	46	174	180	n.a
LD&IS	90 nm	42.3 (target 1080p)	150.4 KGates	57.4	174	30	12.5
S&IS	Stratix 4	210.0 (max.)	18.4 KALUTs	33.9	174	149	n.a
S&IS	90 nm	42.3 (target 1080p)	151.2 KGates	56.4	174	30	13.5

block of LD&IS and S&IS architectures are lower than [19].

Adaptive true motion estimation (ATME) algorithm is presented in [20]. It also evaluates the algorithm for HD 1080p videos. In that work only 104 cycles are necessary, in average, to process a 16 × 16 block. The architecture presented in [20] requires only 63 MHz and is able to process HD 1080p videos in real time. Comparing the area results, the LD&IS architecture uses less hardware resources. The proposed architectures are capable to achieve more than the double of [20] processing rate.

In [21] a ME capable to encode 30 HD 1080p frames per second is presented and it was synthesized for 180 nm technology. In comparison to our work, [21] uses a higher number of cycles to encode a block and also larger hardware resources. Comparing power results, [21] presents a higher power consumption because the proposed architectures need a much lower frequency to encode at the same processing rate, this way it is possible to drastically reduce the architectures power consumption.

An architecture for the fast algorithm dynamic variable step search (DVSS) is presented in [22]. This architecture also uses a 16 × 16 block size and presents synthesis results for a distinct FPGA technology. The DVSS architecture is capable to process 34 HD 1080p frames per second, which is much lower than the processing rate of our FPGA solution. This is explained by the fact that our solutions are capable to process a 16 × 16 block in a fewer number of cycles. The main advantage in [22] is that the architecture uses only 0.51 K memory bits, which is much lower than the memory proposed in this work. This also explain the need of much more cycles than LD&IS and S&IS architectures because DVSS needs to asks for data in an external memory in a higher rate.

The work [23] proposes three different ME algorithms in the same architecture: hexagon based search (HEXBS), block based gradient descent search (BBGDS) and TSS. This architecture needs 390, 437 and 680 cycles per coded block for each algorithm, respectively. In Table 3 it is presented only the processing rate results for BBGDS because it is the average case of this architecture, however, the three algorithms are capable to achieve real-time results for HD 1080p videos. All algorithms proposed in [23] need a higher number of cycles per encoded block than the architectures proposed in this work. The architecture proposed by [23] uses less hardware resources than our architecture, however, the LD&IS and S&IS architectures dissipate less power than [23], which is explained by the high frequency that the architecture proposed in [23] requires to achieve real-time performance.

In comparison to related works, our solutions presented a high number of memory bits, however, if compared to EPZS algorithm, the memory necessary just for the MV predictor should be around four times the memory usage in our solutions. Therefore, our solutions present a good balance in terms of coding efficiency, hardware usage, power consumption and overall performance.

7 Conclusions

This article presented the S&IS and the LD&IS ME algorithms, which are fast and hardware-friendly ME algorithms. The hardware designs were also presented in this article for both algorithms. Those VLSI designs were focused on obtaining a low number of cycles to encode a block, which results mainly in a high performance with low operational frequency, with a good impact on the power consumption reduction, when compared to other works in literature.

The algorithms were evaluated using the HEVC reference software and compared to EPZS and DS using the BD-rate measurement. The proposed algorithms were capable to reach better results than DS, which is a classical algorithm. In turn, the EPZS obtained a higher quality results than our solution, which has occurred mainly because of EPZS MV predictor. However, implementing the EPZS MV predictor uses around 226,800 memory bits and goes in the opposite direction of our goals. Note that our main objective was to define a hardware-friendly ME algorithm able to be implemented in Hardware using reduced memory/hardware usage, low energy consumption, and high performance.

The S&IS and the LD&IS architectures were described in VHDL and synthesized to an Altera Stratix 4 FPGA and also for standard cells on TSMC 90 nm technology. FPGA synthesis results show that the architecture is capable to process 149 (S&IS) and 180 (LD&IS) HD 1080p frames per second. Standard cell 90 nm technology synthesis results show that our proposed solutions are capable to process HD 1080p videos in real time, at 30 frames per second, consuming only 12.5 and 13.5 mW for the LD&IS and S&IS, respectively.

References

1. JVT, Wiegand, T., Sullivan, G., & Luthra, A. (Ed.). (2003). Draft ITU-TRecommendation and final draft international standard of joint video specification (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC).
2. JCT. (2011). Working Draft 3 of high-efficiency video coding. JCTVC-E603.
3. Cheng, Y., et al. (2009). An H.264 spatio-temporal hierarchical fast motion estimation algorithm for high-definition video. *IEEE International Symposium on Circuits and Systems (ISCAS), 2009*, 880–883.

4. Kuhn, P. (1999). *Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation* (Vol. 2). Amsterdam: Kluwer Academic Publishers.
5. Bhaskaran, V., & Konstantinides, K. (1999). *Image and video compression standards: Algorithms and architectures* (2nd ed.). Boston: Kluwer Academic Publishers.
6. Walter, F., Diniz, C., & Bampi, S. (2012). Synthesis and comparison of low-power high-throughput architectures for SAD calculation. *Analog Integrated Circuits and Signal Processing, 73*, 873–884.
7. Sanchez, G., Correa, M., Noble, D., Porto, M., Bampi, S., & Agostini, L. (2012). Hardware design focusing in the tradeoff cost versus quality for the H.264/AVC fractional motion estimation targeting high definition videos. *Analog Integrated Circuits and Signal Processing, 73*, 931–944.
8. Zhu, S., & Ma, K. (2000). A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing, 9*(2), 287–290.
9. Zhu, C., Lin, X., & Chau, L. (2002). Hexagon-based search pattern for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(5), 349–355.
10. Jing, X., & Chau, L. (2004). An efficient three-step search algorithm for Block motion estimation. *IEEE Transactions on Multimedia, 6*(3), 435–438.
11. Chok-Kwan, C., & Lai-Man, P. (2000). Normalized partial distortion search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology, 10*(3), 417–422.
12. Porto, M., et al. (2011). An efficient ME architecture for high definition videos using the new MPDS algorithm. In *ACM Symposium on Integrated Circuits and Systems Design (SBCCI)* (pp. 119–124).
13. Tourapis, A. (2002). Enhanced predictive zonal search for single and multiple frame motion estimation. In *Proceedings of Visual Communications and Image Processing (VCIP)* (pp. 1069–1079).
14. Sanchez, G., et al. (2012). Spread and iterative search: A high quality motion estimation algorithm for high definition videos and its VLSI design. In *IEEE International Conference on Multimedia and Expo (ICME)* (pp. 1079–1084).
15. Sanchez, G., et al. (2013). A fast hardware-friendly motion estimation algorithm and its VLSI design for real time ultra high definition applications. In *IEEE Latin American Conference on Circuits and Systems (LASCAS)*.
16. Joint Collaborative Team on Video Coding (JCT-VC) (2011). High Efficiency Video Coding (HEVC) Test Model 5 (HM 5) Reference Software [Online]. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-5.0-dev/. Accessed May 2013.
17. Lu, J., et al. (2007). An epipolar geometry-based fast disparity estimation algorithm for multiview image and video coding. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 17*(6), 737–750.
18. Lai, Y., et al. (2010). Hybrid parallel motion estimation architecture based on fast top-winners search algorithm. *IEEE Transactions on Consumer Electronics (TCE), 56*(3), 1837–1842.
19. Yin, H., et al. (2010). A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 20*(9), 1242–1254.
20. Cetin, M., et al. (2011). An adaptive true motion estimation algorithm for frame rate conversion of high definition video and its hardware implementations. *IEEE Transactions on Consumer Electronics (TCE), 57*(2), 923–931.
21. Kao, C., Wu, C., & Lin, Y. (2010). A high-performance three-engine architecture for H.264/AVC fractional motion estimation. *IEEE Transactions on Very Large Scale Integration Systems, 18*(4), 662–666.
22. Tasdizen, O., et al. (2009). Dynamically variable step search motion estimation algorithm and a dynamically reconfigurable hardware for its implementation. *IEEE Transactions on Consumer Electronics, 55*(3), 1645–1653.
23. Vanne, J., et al. (2009). A configurable motion estimation architecture for block-matching algorithms. *IEEE Transactions on Circuits and Systems for Video Technology, 19*(4), 446–476.
24. Bjontegaard, G., (2001, April). Calculation of average PSNR differences between RD curves. In *VCEG-M33, ITU-T SG16/Q6 VCEG, 13th VCEG Meeting, Austin, USA*.
25. Altera Corporation. Altera: The Programmable Solutions Company. Available at www.altera.com. Accessed May 2013.
26. Chen, C.-H., et al. (2006). Level C+ data reuse scheme for motion estimation with corresponding coding orders. *IEEE TCSVT, 16*(4), 553–558.

Gustavo Sanchez received the Electrical Engineer degree from Sul-Rio-Grandense Federal Institute of Education, Science and Technology (2013) and B.S. degree in Computer Science from Federal University of Pelotas (2012). Sanchez is currently pursuing his M.S. degree in Computer Science at Federal University of Pelotas and is planning to end this course by March, 2014. His research interests include video coding, memory solutions for digital systems, motion estimation algorithms, FPGA based design, low power design for motion estimation architectures and 3D video coding.

Bruno Zatt received his B.E. and M.Sc. in Computer Engineering from the Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil in 2006 and 2008, respectively. Mr. Zatt received his Ph.D. degree on Microelectronics from the PGMICRO (Graduate Program on Microelectronics) at the same university in 2012 with "summa cum laude" distinction. Currently Bruno Zatt is a Professor at the Federal University of Pelotas (UFPel), Pelotas, RS, Brazil, and a member of the Group of Architectures and Integrated Circuits (GACI). He has 9+ years research experience on algorithms and hardware architectures for video processing including 3 years researching on low power embedded realization for the Multiview Video Coding as intern researcher at the

Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. His main research interests are fast algorithms and low-power hardware architectures for video processing and 2D/3D video coding for mobile applications.

Marcelo Porto received the B.S. degree in Computer Science from Federal University of Pelotas, Pelotas, RS, Brazil, in 2006 and the M.S. and Ph.D. degrees, also in Computer Science, from Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, in 2008 and 2012, respectively. He is Professor since 2009 at the Center of Technological Development (CDTEC) of Federal University of Pelotas, Pelotas, Brazil, and member of the Group of Architectures and Integrated Circuits (GACI). His research interests include video coding, motion estimation algorithms, FPGA based design and VLSI design for video coding. He is a member of IEEE and also member of the IEEE Circuits and System Society.

Luciano Agostini received the B.S. degree in Computer Science from the Federal University of Pelotas, Pelotas, RS, Brazil, in 1998 and the M.S. and Ph.D. degrees from the Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, in 2002 and 2007 respectively. He is a Professor since 2002 at the Center of Technological Development (CDTEC) of Federal University of Pelotas, Pelotas, Brazil, where he leads the Group of Architectures and Integrated Circuits (GACI). Since 2014 he is the Executive Vice President for Research and Graduation Studies of UFPel. He has more than 100 published papers in journals and conference proceedings. His research interests include video coding, arithmetic circuits, FPGA based design and microelectronics. Dr. Agostini is a Senior Member of IEEE and he is a member of the IEEE Circuits and System Society, of the IEEE Signal Processing Society, of the Brazilian Computer Society (SBC) and of the Brazilian Microelectronics Society (SBMicro). He participated on several organizing and program committees of international events in his research area.