J Real-Time Image Proc (2010) 5:45–56
DOI 10.1007/s11554-009-0124-2

ORIGINAL RESEARCH PAPER

Real-time compression architecture for efficient coding in autostereoscopic displays

D. P. Chaikalis · N. P. Sgouros · D. E. Maroulis · M. S. Sangriotis

Received: 12 November 2008 / Accepted: 14 May 2009 / Published online: 6 June 2009

Abstract Integral imaging is a promising technique for delivering high-quality three-dimensional content. However, the large amounts of data produced during acquisition prohibits direct transmission of Integral Image data. A number of highly efficient compression architectures are proposed today that outperform standard two-dimensional encoding schemes. However, critical issues regarding real-time compression for quality demanding applications are a primary concern to currently existing Integral Image encoders. In this work we propose a real-time FPGA-based encoder for Integral Image and integral video content transmission. The proposed encoder is based on a highly efficient compression algorithm used in Integral Imaging applications. Real-time performance is achieved by realizing a pipelined architecture, taking into account the specific structure of an Integral Image. The required memory access operations are minimized by adopting a systolic concept of data flow through the core processing elements, further increasing the performance boost. The encoder targets, real-time, broadcast-type high-resolution Integral Image and video sequences and performs three orders of magnitude faster than the analogous software approach.

Keywords Integral imaging · Disparity estimation · 3D image compression · FPGA · Real-time processing

D. P. Chaikalis (✉) · N. P. Sgouros · D. E. Maroulis · M. S. Sangriotis
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Ilisia, 15784 Athens, Greece
e-mail: dhaik@di.uoa.gr
URL: http://www.di.uoa.gr

N. P. Sgouros
e-mail: nsg@di.uoa.gr

D. E. Maroulis
e-mail: dmarou@di.uoa.gr

M. S. Sangriotis
e-mail: msagri@di.uoa.gr

1 Introduction

Depth perception is a highly desired feature in specialized and everyday applications. A large number of stereoscopic goggles and autostereoscopic displays [25] exist in the market nowadays in order to fulfill the needs for high-resolution three-dimensional (3D) viewing. The variety of stereoscopic and autostereoscopic methods is large, ranging from classic two-view to current multi-view systems.

A simple technique for producing high-resolution full parallax 3D images is Integral Imaging (InIm) which was initially proposed by the Nobelist G. Lippman back in 1908. The advances in optics and digital sensors revived the interest in the technique characterized as a near ideal autostereoscopic 3D display solution [12]. The technique allows continuous movement of the point of view in any direction, providing 2D parallax to the viewer, without the need for special glasses, while it allows multiple viewers to experience the 3D effect.

In order to acquire an InIm, a two-dimensional (2D) arrangement of lenses called a Lens Array (LA) is placed at an appropriate distance in front of a Charge Coupled Detector (CCD). In the acquisition stage numerous small elemental images (EIs) are created on the CCD plane by the lenses of the LA as shown in Fig. 1a. In the reproduction stage depicted in Fig. 1b, an appropriate LA is placed in front of the display device (typically a high-resolution LCD monitor) and a 3D representation of the

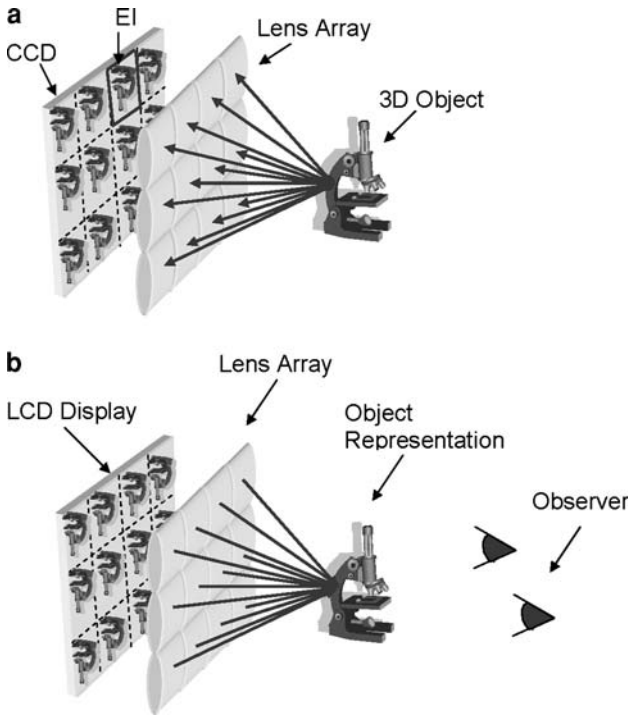

Fig. 1 A 3D capturing and display setup based on the principles of Integral Photography, (**a**) the capturing and (**b**) the display setup

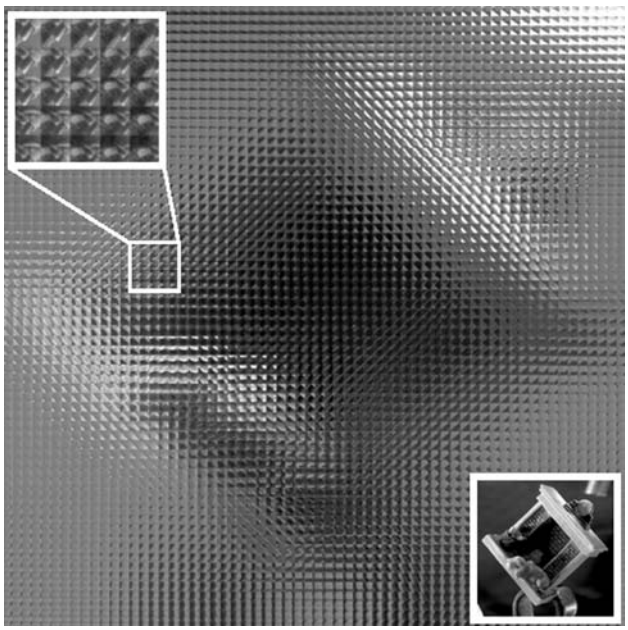

Fig. 2 An InIm of a physical 3D object along with the original object and a magnified portion of a 5 × 5 EIs neighborhood

original object is formed in the space between the observer and the display.

One of the main concerns in all 3D systems is the amount of information required for the representation of objects and scenes. The increased storage capacity and bandwidth requirements for transmission of InIm and Integral video content can be sustained for certain types of applications but remain a prohibiting factor for broadcast type applications or everyday person-to-person communications. For this reason, compression of the information in 3D images or 3D video sequences is essential [28] for both storage and transmission purposes. Moreover, the time efficiency of robust compression algorithms is a prerequisite for real-time applications.

Three-dimensional images as well as 3D video sequences involve multiple highly correlated views of objects in a 3D scene. The information redundancy in such representations leads to the need for highly efficient compression schemes in order to reduce the high data volumes produced during acquisition. The overall redundancy is usually system dependent but all stereoscopic systems share inter-pixel redundancy among neighboring pixels and high inter-view redundancy amongst different views contained in the representation. These two types of redundancies are expressed in an InIm as inter-pixel redundancy within each EI and a 2D intra-EI redundancy between neighboring EIs in the 2D array of EIs. An InIm of an object along with the original 3D object and a magnified InIm portion exhibiting high 2D correlation among neighboring EIs is depicted in Fig. 2.

The effectiveness of the video compression algorithms over JPEG in InIm compression is extensively studied in [15, 16, 27, 34]. A number of different scanning topologies for transforming the 2D EI structure to a sequence of images is proposed that further enhance the quality of the compressed InIm. However, this technique is not optimized for real-time applications. The large number of EIs in an InIm indicates that low complexity disparity estimation schemes would favor real-time InIm and Integral video applications over a complete motion estimation scheme.

Sgouros et al. [26] have previously proposed an InIm compression scheme that extends the MPEG-2 [20] functionality and fully exploits the inherent redundancy in an InIm. This is achieved by minimizing the computational complexity of the disparity estimation scheme using a priori knowledge of the geometric characteristics of the lenses of the LA. The low complexity of this disparity estimation scheme allows the use of the exhaustive search method for the disparity vectors thus improving the overall quality of the reconstructed InIm. Additionally, the geometrical constraints, imposed by the LA structure, further simplify the disparity estimation scheme as the required vectors and the search window size can be determined beforehand reducing the overall computational load. The different scanning topologies proposed in [27] along with MPEG-2 specific motion estimation features can be integrated with minimal adaptations to the robust InIm orientated disparity

estimation scheme developed in [26] in order to further enhance the performance of the encoder.

The innovative InIm compression methodology presented in [26] achieves significant results in terms of compression efficiency and image quality. A hardware encoder for InIm can lead to a robust system capable of dealing with real-time delivery of InIm image and video content.

The high-quality demands of InIm encoding require exhaustive Motion Estimation (ME) schemes that correspond to the most computationally intensive task in a video compression algorithm [4, 7]. Accelerating the ME stage of the proposed InIm encoding algorithm will provide a significant performance boost and target real-time InIm applications. An FPGA device is selected over ASIC as the acceleration platform because the former offers increased flexibility, sufficient performance, faster design time, portability and scalability [21, 22].

Recent FPGA implementations of MPEG-2 components offer increased performance, but extended modifications are needed in order to effectively address real-time InIm compression issues. Up to now, hardware optimizations for compression schemes based on motion estimation either demonstrate impractical high bandwidth demands [32, 33] or are insufficient for real-time 3D applications [10]. Currently proposed simplifications on MPEG-2 components are constrained to specific 2D applications [11, 19, 24, 35].

Hardware implementations for 3D applications either address classic stereoscopic video [14] or autostereoscopic systems. In the latter case, they mainly focus on two-camera systems [23], or they address the acceleration of multi-camera rendering [29]. Although InIm coding and representation algorithms are time-consuming because of full parallax compared to other autostereoscopic techniques, there are no hardware implementations that address the acceleration of these processes.

In this paper we propose an integrated novel hardware encoder for real-time compression of InIm data, based on the algorithm presented by Sgouros et al. [26]. The architecture takes into account the bandwidth limitations imposed by a single FPGA implementation, since for completely parallel computations multiple FPGAs are required [32]. Memory access is significantly reduced by adopting a systolic concept of data flow through the core processing elements and computationally intensive calculations are pipelined through several stages in order to maximize speed results.

The proposed implementation is based on a hardware module for the calculation of disparity vector matrices presented in [3]. With respect to our previous work, the disparity vector calculation architecture has undergone extensive pipelining in the adder tree stages, which allows achieving a higher operating frequency although more modules are added on the device. These added modules implement successive coding stages in order to exploit the remaining device surface and further accelerate the InIm coding procedure.

In [26], the problem of compressing single InIms is addressed. In this paper, we show that the proposed hardware system can sustain real-time performance by coding consecutive InIm frames in the case that the above algorithm is used in a manner similar to motion JPEG. Clocked to a maximum frequency of 20 MHz, the system can process high-resolution InIms up to 2,048 × 1,576 pixels suitable for a number of different applications in a rate greater than 30 images per second, which is a practical real-time constraint for high-quality video systems.

The rest of the paper is divided into four sections. In Sect. 2 we briefly describe the algorithm on which the proposed architecture is based and we introduce some essential terms of the corresponding method. In Sect. 3 we analyze the proposed hardware design for the encoder, the results of which are apposed in Sect. 4. The conclusions and prospects of this study are summarized in the last section.

2 Algorithm description

The InIm compression algorithm described in [26] treats the EIs of the entire image as a spatial sequence of frames with a known disparity pattern due to the LA structure which resembles a 2D array of perfectly calibrated cameras. In this fashion, the algorithm was developed having in mind several modules used in standard video compression schemes like MPEG-2, but differentiates itself in the substitution of all time-dependent quantities with equivalent spatial ones. The main components of the MPEG-2 algorithm as the discrete cosine transform (DCT) and quantization modules are used, while motion estimation is replaced by a disparity estimation scheme proposed in [26]. As opposed to MPEG-2, the search area can be defined beforehand based on the specific InIm characteristics, to a smaller region instead of typical search windows. This fact, along with the prior knowledge of the directionality of the disparity vectors, reduces the computational cost of disparity estimation.

In particular, the significantly reduced size of the search area along with the unidirectionality of the motion vectors imposed by the InIm structure allows for a unidirectional exhaustive block search method to be applied, which retains a relatively low computational cost while targeting high-quality compressed images. The scanning topology used in the technique has either horizontal or vertical parallax, so a unidirectional search is sufficient for producing high-quality

results. Moreover, the computational cost of searching another direction if a different scanning topology is used cannot balance the quality gain of the technique. The computational cost for an exhaustive 2D search used in MPEG-2 is $O(p^2)$, where p is the size of the unidirectional search window. Using an 8 × 8 block size over a 32 × 32 search area, 1,024 block comparisons are needed for the exhaustive MPEG search. In juxtaposition, the search area for the proposed unidirectional technique spawns over one direction and only 32 comparisons are required, reducing the cost to $O(p)$. Moreover, extended optimization is applied to vector calculation precision and coding, in order to maximize quality.

Figure 3a illustrates an InIm segmentation in EIs (I-type and P-type) along with the search method followed. Using a reconstructed I-type EI, an estimation of two neighboring predicted EIs (P-type) of 32 × 32 pixel size is formed by evaluating the proper disparity vectors for each 8 × 8 P block in an I-type EI area of 8 × 32 pixel size. In an InIm, the two P-type EIs are positioned one to the left and the other to the right of the I-type EI, as shown in Fig. 3b. These three EIs form what will hereafter be referred to as a P-I-P EI triplet or P-I-P triplet. The P-I-P triplet maximizes compression efficiency for high-disparity elemental image sequences where correlation between neighboring regions radically deteriorates [30]. The P-I-P scheme ensures best results for the matching process as three contiguous EIs are used. This results in high Peak Signal-to-Noise (PSNR) ratios for the reconstructed image and reduces the variability of the PSNR ratio [27] noticed in the frame sequences of MPEG-2. For an InIm it is crucial to maintain low PSNR variability as the EIs that form an InIm are simultaneously projected to the viewer and large deviations in the PSNR values can lead in diminishing the perceived 3D effect. Finally the transmission sequence has changed to I-P-P in order to maximize decoding efficiency.

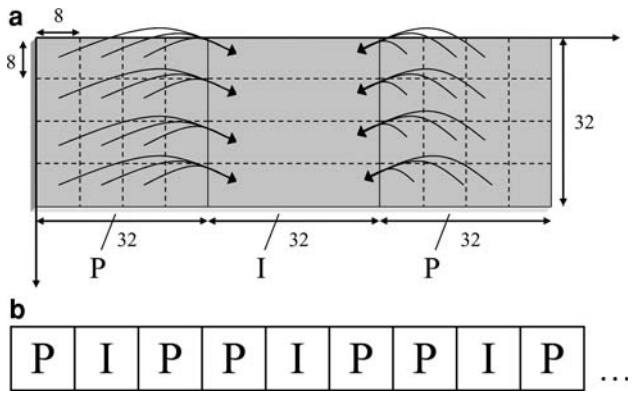

Fig. 3 (**a**) The arrangement of I-type and P-type EIs in an IP image and the P block search area outline, (**b**) Arrangement of EIs in an InIm

In order to determine the "best match" of each P block in the I-type EI, an efficient metric must be used. Several candidates such as the sum of squared differences (SSD), the matching pixel count (MPC), the normalized correlation coefficient (NCC) and the sum of absolute differences (SAD) exist [6, 8, 17]. Although metrics such as SSD and MSE offer higher accuracy with respect to SAD, their computational complexity is also much higher due to the multitude of multiplications involved in the calculations. Hardware implementations of the SAD metric [33] are shown to offer a very good trade-off between computational complexity and accuracy in the determination of the best match. The SAD calculation adds up the absolute differences between corresponding elements in the predetermined macroblocks. For an $m \times n$ pixel block, the SAD value is calculated by the formula:

$$\mathrm{SAD}(U,V) = \sum_{i=0}^{m}\sum_{j=0}^{n} |U(i,j) - V(i,j)|$$

where i, j are spatial coordinates in the pixel domain and U, V represent $m \times n$ pixel blocks in adjacent image blocks. The actual coordinates of these blocks in the corresponding macroblocks are determined by the search algorithm used.

3 Hardware design

The hardware modules are implemented in FPGA using a Celoxica RC1000-PP PCI board [2] based on a Xilinx Virtex XCV-2000E chip, with an equivalent area of two million logic gates. All the design and most of the simulation files are written in VHDL using the Xilinx ISE 5 development software. The memory capacity of the board is 8 MB, while the FPGA device implements 640 KB (80 KB) of dedicated memory. The basic memory modules have a size of 256 × 16 bits and they can be combined in the design process to create bigger modules both in cell and bit-width count. For simulation purposes, the ModelSim simulator software was used. The encoder is implemented using the following constraints for a typical InIm application: EI size 32 × 32 pixels, estimation block size 8 × 8 pixels and DCT block size 8 × 8.

3.1 System overview

Figure 4 presents a block diagram of the encoder as it is implemented on the development board.

The block RAM is organized into three separate banks: one for the I-type EIs and two for the two P-type EIs that are adjacent to each I-type EI. A total of 16 P-I-P EI triplets are stored in FPGA memory, according to the FPGA's maximum dedicated memory capacity. The disparity estimation

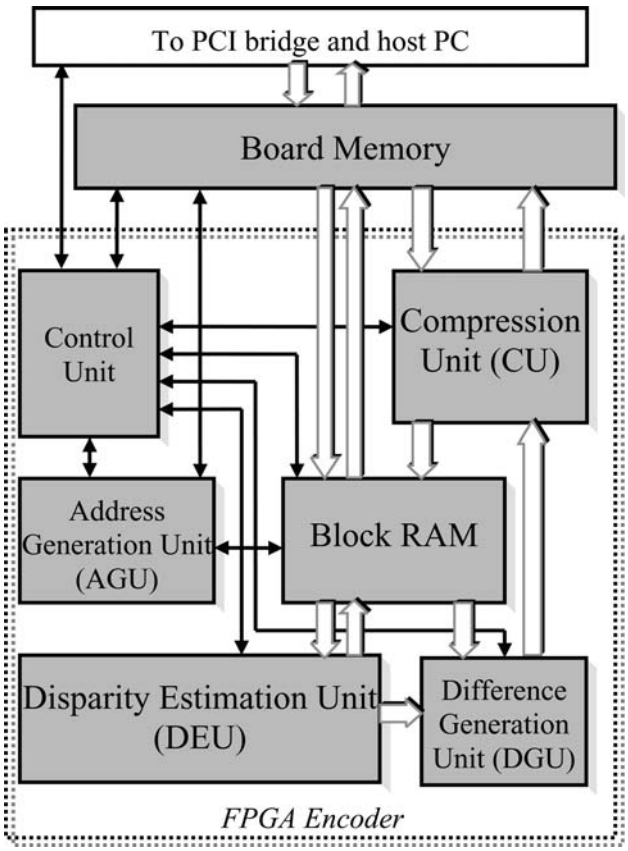

Fig. 4 An overview of the FPGA encoder

unit (DEU) uses the image data from the Block RAM to generate the disparity vectors. The difference generation unit (DGU) calculates the differences between the blocks in P-type EIs and the estimated blocks in the I-type EIs, using the results from the DEU. The compression unit (CU) generates the final compressed blocks that will be used at the reconstruction stage. It is also responsible for compressing and decompressing the I-type EIs used in the disparity estimation procedure. The CU consists of 3 two-dimensional Discrete Cosine Transform (2D-DCT) modules, one 2D Inverse-DCT (2D-IDCT) module and the appropriate Quantization and Inverse Quantization modules (see also Sect. 3.4). The Address Generation Unit (AGU) and the control unit are necessary for synchronizing the data transfers; the control unit exchanges control signals with every other unit in the FPGA and also with the board components and the host PC, while the AGU provides the address words to the FPGA and board memory modules. On the development board, the four board memory modules are used to store the initial image data and the final results and the FPGA system communicates with them with address and data buses and other control signals.

In the next paragraphs, we present the functional details of the aforementioned units that comprise the encoder.

3.2 Disparity estimation unit

The disparity vectors, which form the disparity vector matrices of the P-type EIs, are calculated in the DEU. The values that describe these vectors are deduced by determining the minimum SAD value for each 8 × 8 P-type block, according to the method illustrated in Fig. 3. In this section, we describe the hardware architecture of the DEU and focus on the method by which the disparity vectors are created.

Systolic arrays are shown to maximally exploit block match operations' regularity in an exhaustive search strategy [13]. Parameters like search range and block size are used to decide on the number of Processing Elements (PEs) in the systolic structure [5]. In our implementation, the PEs (SAD Units) are arranged in the form of a one-dimensional systolic array [31] in order to accommodate for the unidirectional exhaustive search and minimize memory access.

In the proposed encoder, the SAD Units are composed of eight AD modules, an adder tree and an accumulator [3]. Each block comparison is completed in eight clock cycles, one for each pixel column comparison. The "best match" search method proposed in [26], which is from left to right, favors the pixel column comparison method.

The unidirectional search method is implemented by partitioning the P-type and I-type EI into 8 × 32 pixel areas and traversing these areas column-wise in a left-to-right direction, as shown in Fig. 5. In what follows, an 8 × 32 partition of an EI will be referred to as a four-block area. For all of the horizontally adjacent 8 × 8 P blocks in the P-type EI, I blocks are used from the corresponding I-type EI four-block area. It is clear that many I blocks are overlapped and it is efficient to try enhancing the parallelism by minimizing the pixel reads in each four-block area. Fig. 6a illustrates the P blocks in a P-type EI four-block area. In Fig. 6b, the numbers of reads N_i for each pixel column are drawn, in case the SAD value calculation was performed by sequentially comparing an 8 × 8 P block with the I blocks. These numbers (N_i) are easily derived by estimating of how many I blocks each pixel column is a part and how many times each I block is read. For a four-block area, it holds that $\sum_{i=0}^{31} N_i = 408$.

Instead of reading each pixel column one time whenever it is needed for a SAD calculation, the DEU reuses the pixel values of the columns that are already fetched from the FPGA memory modules to calculate successive SAD values. Thus, three minimum SAD values and the corresponding disparity vectors for a four-block area are computed with a single simultaneous read of the four-block I-type EI and the four-block P-type EI area. Thus, in this case, the EIs are only transferred once from the FPGA memory to the DE Unit for each four-block comparison and the number N_i' of pixel column reads is $N_i' = 1, 0 \leq i < 32$,

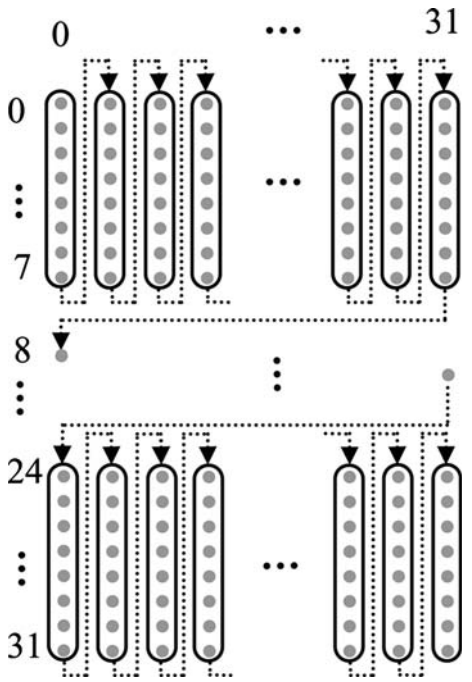

Fig. 5 The four-block pixel areas forming the EIs and their read sequence

summing up to $\sum_{i=0}^{31} N_i' = 32$ pixel column reads for a four-block area. This results to $\sum_{i=0}^{31} N_i - \sum_{i=0}^{31} N_i' = 376$ less memory read cycles for each four-block area, compared to the simplest approach of comparing the I blocks sequentially to each P block, which translates to 92% less memory accesses needed for performing the SAD calculations.

In total, 51 SAD values have to be computed for the 3 P blocks of the P-type four-block area, due to the unidirectional exhaustive search method used [9]. Aiming to minimise memory access and the time during which each SAD unit stays idle, it was concluded that we can use only 25 units in the one-dimensional systolic array, by inserting a "clear" cycle each 8 clock cycles, during which a synchronous clear signal is rippled through the SAD units, starting with SAD unit #0. The minimum SAD value for each P block derives from the comparison of the SAD values calculated for it and takes place in the Comparison module, which is illustrated in Fig. 7. The outputs of the 25 SAD units are the inputs of the Comparison module and the outputs of the Comparison module are the disparity vectors for the 3 P blocks of the four-block area. The disparity vectors virtually represent the relative distances between the position of the P block in the P-type EI and the "best match" of this block in the I-type EI search window.

On each clock cycle, the Comparison module receives one SAD value for each P block, compares it with the one previously stored and keeps the smaller of the two values. The comparison between two values requires only one clock cycle and it can be performed simultaneously with the SAD calculations, as each SAD unit provides as a result one clock cycle after the previous one.

3.3 Difference Generation Unit

The disparity vector matrices are essential to the decoding process, so they are stored to the board memory. They are

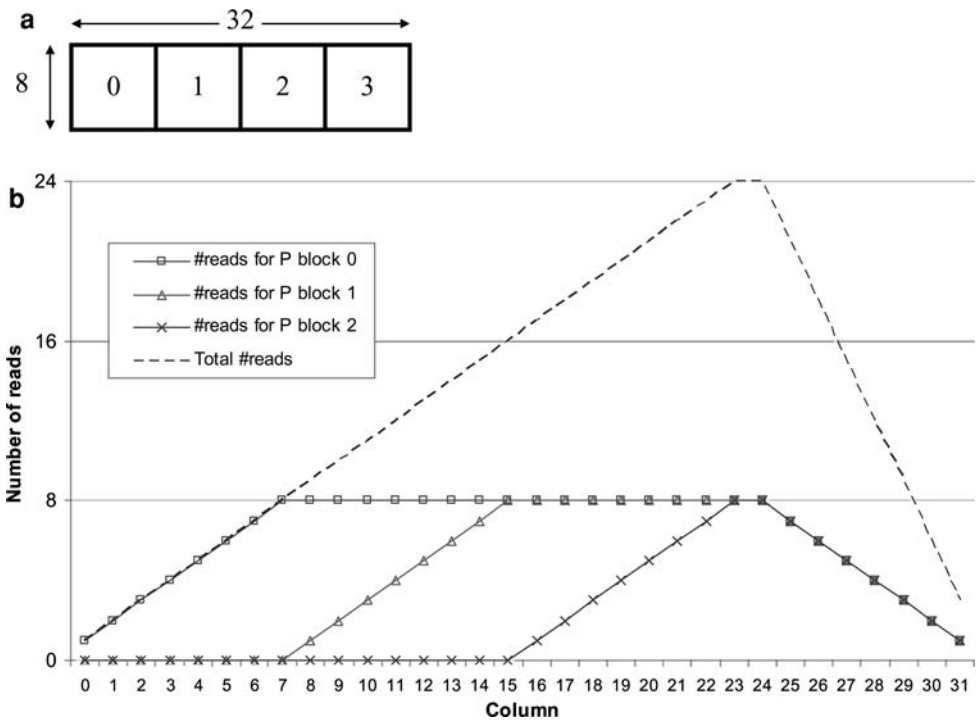

Fig. 6 (**a**) The P blocks in a P-type EI four-block area, (**b**) diagram depicting the number of reads (N_i) for each pixel column in an I-type four-block area and the total number of reads

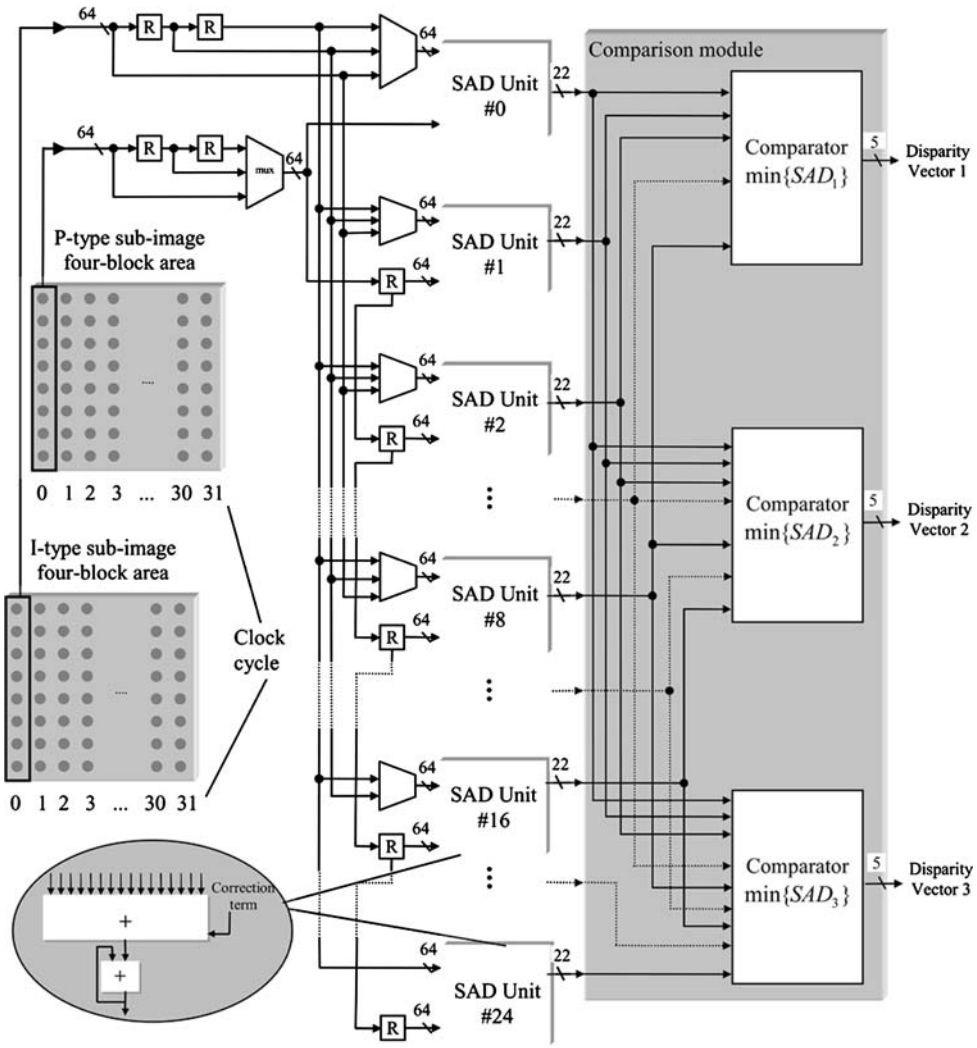

Fig. 7 Block diagram of the DVC Unit showing the additional data buses and the delay registers in order to feed each SAD unit with the appropriate image data

also forwarded to the Difference Generation Unit (DGU) where they are used for creating the difference blocks. These blocks derive by subtracting the I-type EI estimated blocks by the P-type EI blocks, using the disparity vectors in order to determine the location of the former. The difference blocks, along with the disparity vector matrices, will be used at the decoder for the reconstruction of the P-type EIs using the I-type EIs.

The calculation of the difference blocks is performed by subtracting one P-type pixel column from one I-type column on each clock cycle. This way, it is performed in the DGU right after the disparity vector calculation for a four-block area by accessing the same image data using the same addressing scheme. In total, each I-type four-block area must be swept-read three times: the first one for the P(left) disparity vector calculation, the second one for the P(right) disparity vector calculation and the P(left) difference generation, and the third one for the P(right) difference generation. A scheduling diagram of the DEU and DGU operation is depicted in Fig. 10 in the result's section.

3.4 Compression unit

In order to achieve more efficient compression rates, the I-type EIs undergo a compression and decompression process at the first part of the encoder's operation phase by passing through the Compression Unit, which is illustrated in the block diagram of Fig. 8 using dashed arrows. This is done in order to assure that the image used as reference frame at the encoding procedure is actually the same image as that used at the decoding procedure, which evidently produces the optimal results. Doing so, the disparity vectors are calculated by comparison of the P-type EIs to the I-type reconstructed EIs, that will eventually be available at the decoder. The I-type EI pixels are read from the board memory and are transmitted to the I-type EI Compression Unit, where they are initially coded using a

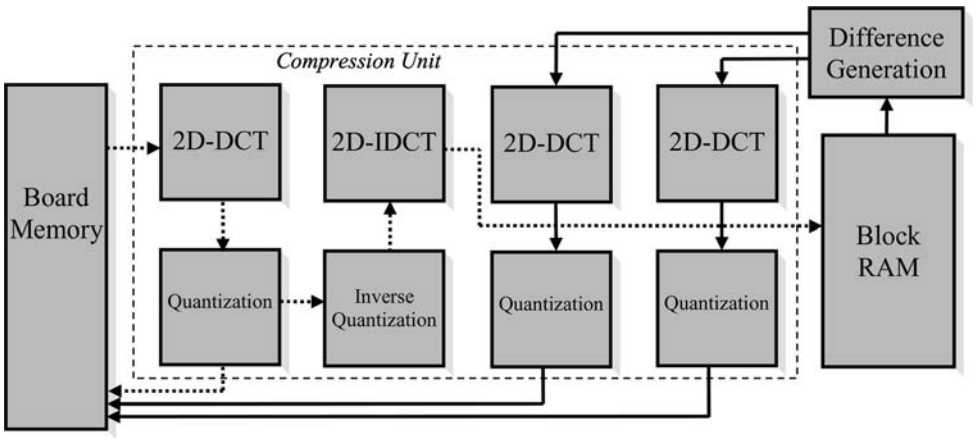

Fig. 8 A block diagram representing the Compression Unit for the I-type EIs along with the required transfer of the EIs from the board to the FPGA memory

forward 8 × 8 2D-DCT transform and quantized according to the standard JPEG luminance table [18]. At the decoding stage, the values are inverse-quantized followed by an 8 × 8 inverse 2D-DCT. After the inverse transform, the decoded I-type EI pixels are stored in block RAM.

The 2D-DCT and 2D-IDCT cores are composed of 1D-DCT cores and a transpose memory, using a row–column decomposition approach [1]. The implemented core outputs one sample per clock cycle, with an initial 87 clock cycles latency. The internal calculations are performed with an accuracy of 15 bits, providing an efficient trade-off between image quality and area coverage.

At the quantization stage, each DCT coefficient is divided by a defined quantization value. The function that describes this process is

$$Q(i,j) = \left\lfloor \frac{\mathrm{DCT}(i,j)}{Q_{\mathrm{LUT}}(i,j) \times q_{\mathrm{factor}}} \right\rfloor$$

where $Q(i, j)$ denotes the quantized value, $\mathrm{DCT}(i, j)$ the compressed value, $Q_{\mathrm{LUT}}(i, j)$ the quantization value and q_{factor} the quantization factor.

The quantization values are stored in a look-up table (LUT). Several LUTs are implemented in order to obtain different quality levels, according to the quantization factor. In order to overcome the complexity of hardware division in the quantization stage, the values are instead multiplied by the reciprocal of the quantization value of the selected JPEG quantization table. The values are represented in the quantization look-up tables (LUTs) with 12 bit precision, leading to an accuracy of 2^{-12}, and the decimal bits are truncated. In the inverse-quantize stage, the quantized values are multiplied with the integer values of the standard JPEG inverse quantization tables, in order to produce the final reconstructed DCT values. Two more 2D-DCT and Quantization modules are used for the compression of the difference blocks before they are stored to the board memory. The compression process of the difference blocks is depicted in Fig. 8 using solid arrows.

3.5 Encoder operation

Before the encoder starts its operation phase, 6 MB of image data are transferred from the host PC to the board memory. Using three memory modules for the image data, the FPGA system can access one pixel of each of the EIs forming the P-I-P EI triplets on each clock cycle, thus increasing the parallelization of the overall procedure.

The hardware's operation phase can be divided into two main parts. The first part involves the transfer of the image data from the board memory to the FPGA block RAM. This transfer is necessary for rearranging the image data in a way that the DEU can process them quickly. Moving of the data is also needed in order to exploit the speed and flexibility advantages of the block RAM and its dual-ported ability. The first part also includes the processing of the I-type EIs by the Compression Unit, as explained in Sect. 3.4.

The second part of the hardware's operation phase includes the creation of the disparity vectors by the DEU and the calculation of the difference blocks by the DGU. The image data are read from the block RAM and are used to compare image blocks, calculate SAD values and determine the I-type EI blocks for which the SAD value for each P-type EI block is minimum. The results of these calculations are the disparity vectors, which form the disparity vector matrices for each P-type EI. The matrices are simultaneously stored in the board memory and forwarded to the DGU. In the DGU, the difference of the image blocks is calculated according to the vectors' information. The resulting difference blocks are compressed by the CU before storing to the board memory.

When the last block is stored to the board memory, the FPGA system returns to its initial state, waiting for a start signal from the host PC. This final transfer completes the operation phase of the encoder, and the host PC retrieves the data from the board memory for further processing.

4 Results

The implementation results of the Xilinx ISE 5 development software reveal that the encoder occupies 18 187 FPGA slices, which correspond to 95% of the Virtex XCV-2000E programmable area. Table 1 presents the distribution of the FPGA resources over the main system components. Timing results show that the encoder can be clocked with a maximum frequency of 20.3 MHz. This value is justified by the almost complete area usage and the specifications of the device. In total, 8,405 clock cycles are need for the processing of 16 P-I-P triplets as shown in the simplified timing chart for the main units comprising the hardware processor, presented in Fig. 9.

The timing of the units is selected so as to minimize idle clock cycles. The board-to-FPGA memory transfer is accelerated by increasing the Compression Unit's clock rate to double the clock rate of the remainder system. The DEU is enabled when there will be valid image data in the FPGA memory modules to process without stall until all disparity vectors are computed. The DGU begins its operation when the first disparity vectors are available. Finally, the CU is capable of processing the image data as soon as they are available by the DGU. Appropriate buffering is implemented for efficient transfer of the compressed data to the board memory. A simplified circuit scheduling diagram for the above units is presented in Fig. 10.

For actual operation, the FPGA clock is set to 20 MHz and approximately 0.42 ms are required for the creation of 32 disparity vector matrices. This timing easily derives by dividing the 8,405 clock cycles for the 16 P-I-P triplets by the operational frequency of 20 MHz. These disparity vector matrices correspond to 32 P-type EIs of the InIm. The overall time for processing an entire image is proportional to the image size.

4.1 FPGA encoder performance

Complete evaluation of the encoder's performance is achieved using different resolution InIms. This is done in order to evaluate the encoder for both everyday applications like entertainment or educational environments that require 3D content delivery and high-resolution demanding applications like medical intra-operative environments or simulators with strict quality and robust performance considerations. Hence, InIms with different resolutions were downloaded to the board memory and the encoder has processed them iteratively for several seconds. The mean processing time for each image is considered as the processing performance of the encoder. Figure 11 depicts the total number of processed InIms per second of operation versus InIm resolution.

As shown in Fig. 11, the encoder achieves real-time performance in all standard cases while real-time performance is also maintained for high-resolution InIms. The high throughput of the encoder for low-resolution InIms makes it suitable for processing a large number of Integral video streams in real-time, while rates above the threshold of 30 InIm images per second of operation for high-resolution InIms show that it is a robust solution in cases where quality cannot be compromised to achieve real-time performance.

Table 1 Distribution of the FPGA resources over the main architectural components

Component	Slices	BRAMs
DEU	9,552	0
DGU	85	2
CU	6,036	16
Control	1,943	0
AGU	563	0
Block RAM	8	102
Total	18,187	120

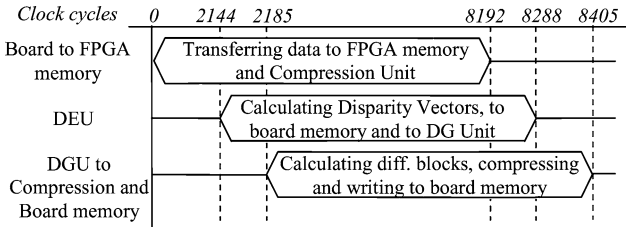

Fig. 9 Timing chart of the main units comprising the hardware processor. The clock cycle count corresponds to a processing phase of 16 P-I-P triplets

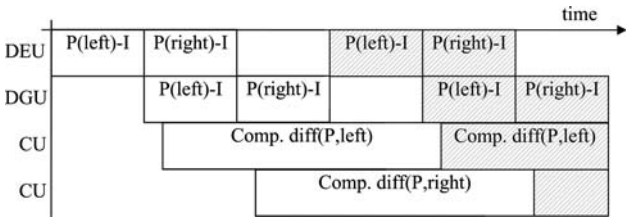

Fig. 10 Circuit scheduling for the DEU, DGU and CU

4.2 Hardware/software comparison

For further evaluation of the performance of the encoder, a comparison with the homologous software approach is also presented. It is worth noting that the present architecture was developed as a hardware implementation of a software method already proposed for robust InIm compression [26]. Accordingly, a direct timing performance comparison with the respective software parts can clearly demonstrate

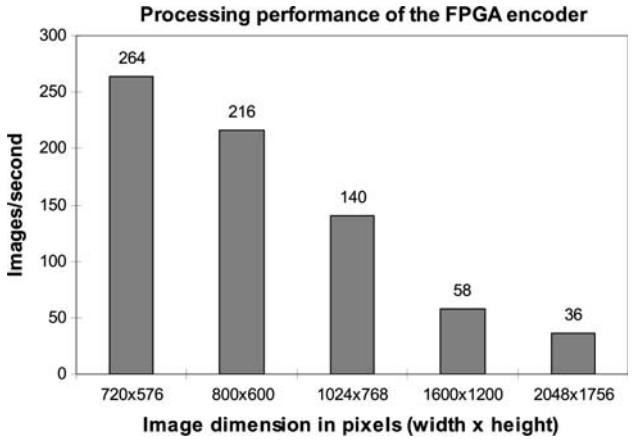

Fig. 11 The processing performance (in InIm images/s) of the encoder measured as a function of the image resolution in pixels

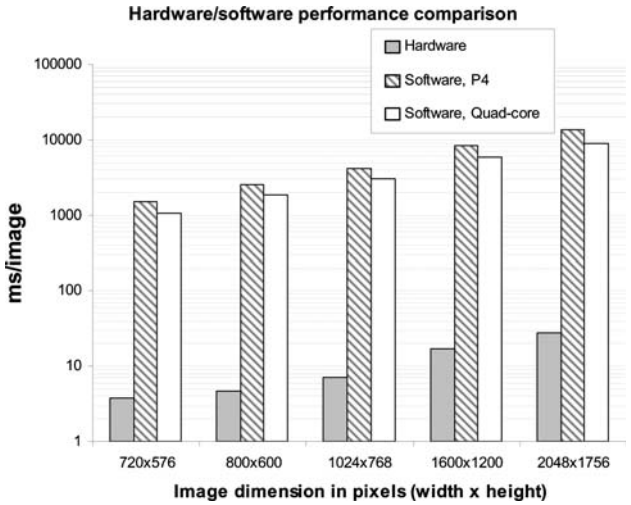

Fig. 12 FPGA encoder versus software performance comparison measured in processing time per InIm for several image dimensions

the performance gain of the hardware approach adopted in our encoder.

The timing performance of the software parts that produce the same results as the encoder has been measured using an Intel-based 2.4 GHz desktop computer, equipped with 512 MB of RAM, which is of the same technological era as the FPGA development board, and a Quad-Core 2.33 GHz desktop computer, equipped with 2 GB of RAM, which represents a robust contemporary PC solution. The software was created in C programming language and compiled using the Visual Studio standard compilation libraries. With this software, the same measuring method as the one used for the encoder was applied. In other words, InIms of various dimensions were processed for a fixed number of iterations and the mean processing time for each InIm group was derived.

The results have shown that software processing requires several seconds, even for images at the low end of the tested dimensions. More than a second is needed for a typical resolution of 720 × 576 pixels and more than 10 s for the higher tested resolution of 2,048 × 1,576 pixels. In addition, compared to the software approach, the implemented encoder has demonstrated an acceleration factor lying in the range of 300–370 for the processing time. Figure 12, which illustrates the timing performances for the software and hardware approaches, reveals the high acceleration achieved when using the proposed architecture.

5 Conclusions

The use of 3D stereoscopic and autostereoscopic systems leads to a more natural perception of objects in a scene. This can benefit applications where user experience heavily relies on the 3D representation quality of the objects. Typical applications include everyday communication needs, ranging from entertainment and standard broadcasting environments to more specific and demanding applications such as specially designed simulators for medical or other uses. As the need for high resolution of the 3D representation grows, numerous challenges in design and implementation of robust real-time performing systems emerge. The amount of data contained in the different views of a 3D system leads to excessive bandwidth and storage requirements. These problems can be alleviated using efficient compression schemes that properly exploit the particular characteristics of the 3D image data.

In the present paper, an integrated FPGA-based InIm and Integral video encoder has been proposed that targets real-time performance in high-quality InIm applications. This digital system is based on an efficient compression algorithm [26] which exploits the high volumes of redundant information enclosed in this type of images. The encoder uses the notion of disparity estimation between neighboring EIs and specific image features such as prior knowledge on the directionality of the disparity vectors to reduce the computational load introduced by the motion estimation modules in MPEG-2. In addition, exhaustive search methods that improve the quality of the encoded InIms can be used without greatly increasing the complexity of the algorithm. The proposed single-FPGA encoder's features are focused on reduced memory operations and extensive pipelining where applicable in order to efficiently address a wide variety of real-time InIm and Integral video applications.

The obtained results demonstrate that the proposed hardware implementation can successfully process in real-time InIms and integral video streams suitable for typical 3D broadcast applications, such as mobile and network communications, as well as for desktop applications with

demanding resolutions. The acceleration rate achieved with this implementation, compared to the current software solution, ranges from 300 to 370 times for several InIm resolutions. This verifies that the proposed system can be used as an acceleration component for robust real-time 3D InIm and integral video compression applications.

Acknowledgments This work was realized under the framework 8.3 of the Reinforcement Programme of Human Research Manpower ("PENED 2003"-03ED656), cofunded 25% by the General Secretariat for Research and Technology, Greece, 75% by the European Social Fund and by the private sector.

References

1. Agostini, L.V., Silva, I.S., Bampi, S.: Pipelined fast 2D DCT architecture for JPEG image compression. In: Proceedings of the 14th Symposium on Integrated Circuits and Systems Design. Pirenopolis, Brazil, pp. 226–231 (2001)
2. Celoxica, R.C.: 1000-PP development board: hardware reference, http://www.celoxica.com
3. Chaikalis, D., Sgouros, N., Maroulis, D., Papageorgas, P.: Hardware implementation of a disparity estimation scheme for real-time compression in 3d imaging applications. J. Vis. Commun. Image Represent. **19**(1), 1–11 (2008)
4. Chen, G.B., Lu, X.N., Wang, X.G., Liu, J.L.: A complexity-scalable software-based MPEG-2 video encoder. J. Zhejiang Univ. Sci. **5**(5), 572–578 (2004)
5. Cheng, S.C., Hang, H.M.: A comparison of block-matching algorithms mapped to systolic-array implementation. IEEE Trans. Circuits Syst. for Video Technol. (CSVT) **7**(5), 741–757 (1997)
6. Kim, S.H., Kim, N.K., Ahn, S.C., Kim, H.G.: Object oriented face detection using range and color information. In: Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition, pp. 76–81 (1998)
7. Kuhn, P.: Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation. Kluwer, Dordrecht (1999)
8. Lee, S., Yi, J., Kim, J.: Real-time stereo vision on a reconfigurable system. In: LNCS Embedded Computer Systems: Architectures, Modeling, and Simulation, pp. 299–307 (2005)
9. Maroulis, D., Sgouros, N., Chaikalis, D.: FPGA-based architecture for real-time ip video and image compression. In: IEEE International Symposium on Circuits and Systems (ISCAS), May 21–24, Island of Kos, Greece (2006)
10. Martina, M., Molino, A., Vacca, F.: Reconfigurable and low power 2D-DCT IP for ubiquitous multimedia streaming. IEEE Int. Conf. Multimedia Expo (ICME) **2**, 177–180 (2002)
11. Moshnyaga, G.V., Tamaru, K.: A memory efficient array architecture for real-time motion estimation. In: 11th International Parallel Processing Symposium (IPPS), Geneva, Switzerland, pp. 28–32 (1997)
12. Naemura, T., Yoshida, T., Harashima, H.: 3-D computer graphics based on integral photography. Opt. Express **8**(2), 255–262 (2001)
13. O'Connor, N., Muresan, V., Kinane, A., Larkin, D., Marlow, S., Murphy, N.: Hardware acceleration architectures for MPEG-based mobile video platforms: a brief overview. In: Proceedings of the WIAMIS 2003—4th Workshop on Image Analysis for Multimedia Interactive Service, 9–11 April, London, UK (2003)
14. Ohm, J.-R., Grüneberg, K., Hendriks, E., Izquierdo, M.E., Kalivas, D., Karl, M., Papadimatos, D., Redert, A.: A Realtime hardware system for stereoscopic videoconferencing with viewpoint adaptation. Signal Process Image Commun. **14**, 147–171 (1998). doi:10.1016/S0923-5965(98)00034-4
15. Olsson, R., Sjöström, M.: A Depth dependent quality metric for evaluation of coded integral imaging based 3D-images. In: Proceedings of the 3DTV-Conference 2007, pp. 1–4 (2007)
16. Olsson, R., Sjöström, M., Xu, Y.: Evaluation of combined pre-processing and H.264-compression schemes for 3D integral images. In: Proceedings of the SPIE Electronic Imaging—VCIP, vol. 6508, 65082C (2007)
17. Park, J.-H., Jung, S., Choi, H., Kim, Y., Lee, B.: Depth extraction by use of a rectangular lens array and one-dimensional elemental image modification. Appl. Opt. **43**(25), 4882–4895 (2004). doi: 10.1364/AO.43.004882
18. Pennebaker, B.W., Mitchell, L.J.: JPEG Image Compression Standard. Van Nostrand Reinhold, New York (1993)
19. Ramachandran, S., Srinivasan, S.: FPGA implementation of a Novel, fast motion estimation algorithm for real-time video compression. In: 9th International Symposium on FPGAs, Monterey, Canada, pp. 213–219 (2001)
20. Rao, K.R., Hwang, J.J.: Techniques and standards for image, video and audio coding. Prentice-Hall PTR, New Jersey (1996)
21. Rathnam, S., Slavenburg, G.: An architectural overview of the programmable multimedia processor, TM-1. In: Proceedings of the COMPCON, pp. 319–326 (1996)
22. Reddy, V.S.K., Sengupta, S., Iatha, Y.M.: A high-level pipelined FPGA based DCT for video coding applications. In: Proceedings of the TENCON 2003 vol 2, pp. 561–565 (2003)
23. Redert, P.A.: Acquisition and presentation of 3D video in the Panorama WP2 Hardware Chain. Rapport aan, E.G, Brussels (1997)
24. Roma, N., Dias, T., Sousa, L.: Customisable core-based architectures for real-time motion estimation on FPGAs. In: Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL), Lisboa - Portugal, 1–3 September, pp. 745–754 (2003)
25. Sexton, I., Surman, P.: Stereoscopic and autostereoscopic display systems. IEEE Signal Proc. Mag. **16**(3), 85–99 (1999)
26. Sgouros, N., Andreou, A., Sangriotis, M., Papageorgas, P., Maroulis, D., Theofanous, N.: Compression of IP images for autostereoscopic 3D imaging applications. In: 3rd International Symposium on Image and Signal Processing and Analysis (ISPA), Rome, Italy, September 18–20 (2003)
27. Sgouros, N., Kontaxakis, I., Sangriotis, M.: Effect of different traversal schemes in integral image coding. To appear in. Appl. Opt. **47**(19), D28–D37 (2008). doi:10.1364/AO.47.000D28
28. Shah, D., Dodgson, N.A.: Issues in multi-view autostereoscopic image compression. In: Proceedings of the SPIE 4297, Symposium on Stereoscopic Displays and Applications XII, pp. 307–316, (2001)
29. Sorbier, F., Nozick, V., Biri, V.: GPU rendering for autostereoscopic displays. In: Proceedings of the 4th International Symposium on 3D Data Processing, Visualization and Transmission (3DPTV) (2008)
30. Stern, A., Javidi, B.: Integral image compression methods. In: Proceedings of the SPIE vol. 6311, pp. 631104-1–631104-11 (2006)
31. Torres-Huitzil, C., Arias-Estrada, M.: Real-time image processing with a compact FPGA-based systolic architecture. In: Elsevier Journal Real-Time Imaging. vol. 10, pp. 177–187 (2004)
32. Wong, S., Stougie, B., Cotofana, S.: Alternatives in FPGA-based SAD Implementations. In: IEEE International Conference on Field Programmable Technology (FPT), Hong Kong, pp. 449–452 (2002)
33. Wong, S., Vassiliadis, S., Cotofana, S.: A sum of absolute differences implementation in FPGA Hardware. In: 28th Euromicro Conference, Dortmund, Germany, pp. 183–186 (2002)

34. Yeom, S., Stern, A., Javidi, B.: Compression of 3D color integral images. Opt. Express **12**, 1632–1642 (2004). doi:10.1364/OPEX.12.001632
35. Zandonai, D., Carro, L., Bampi, S., Suzin, A.A.: An architecture for MPEG motion estimation. In: VII Workshop Iberchip (IWS), Montevideo, vol. 1, pp. 90–95 (2001)

Author Biographies

Dionisis P. Chaikalis received his B.Sc. in Informatics and Telecommunications from the National and Kapodistrian University of Athens. His main research interests lie in the fields of digital system design, parallel hardware implementations and real-time 3D image and video processing. Currently he is a Ph.D. candidate in the Department of Informatics and Telecommunications at the National and Kapodistrian University of Athens.

Nicholas P. Sgouros received his B.Sc. in Physics and subsequently his M.Sc. and Ph.D. in Informatics and Telecommunications from the National and Kapodistrian University of Athens. His research interests are in the areas of 3D digital image and video processing, with specific focus on 3D image and video analysis, compression and real-time 3D imaging systems. Currently he is a Postdoctoral Research Fellow in the Department of Informatics and Telecommunications at the National and Kapodistrian University of Athens.

Dimitris E. Maroulis received the B.Sc. degree in physics, M.Sc. degrees in radioelectricity and cybernetics and Ph.D. degree in informatics from the University of Athens, Athens, Greece, in 1973, 1977, 1980 and 1990, respectively. Currently, he is an Associate Professor in the Department of Informatics and Telecommunications, University of Athens. He is a coauthor of more than 90 articles in scientific journals and conference proceedings on data acquisition systems, real-time systems, image analysis and biomedical applications. His research interests include data acquisition, real-time image/signal processing systems and bioinformatics.

Manolis S. Sangriotis received B.Sc. and Ph.D degrees from Physics Department of Athens University in Greece. In 1981, he was with the Department of Physics in Athens University. Since 1990 he has been with the Department of Informatics and Telecommunications in Athens, Greece, where he is currently an Associate Professor. His research interests include Image Analysis and Image Coding.