Available online at www.sciencedirect.com
ScienceDirect
J. Vis. Commun. Image R. 19 (2008) 1–11
JOURNAL OF VISUAL Communication & IMAGE Representation
www.elsevier.com/locate/jvci

# Hardware implementation of a disparity estimation scheme for real-time compression in 3D imaging applications

D. Chaikalis [a,*], N. Sgouros [a], D. Maroulis [a], P. Papageorgas [b]

[a] National and Kapodistrian University of Athens, Department of Informatics and Telecommunications, Panepistimiopolis, Ilisia, Athens 15784, Greece
[b] Technological Educational Institute of Piraeus, Department of Electronics, Petrou Ralli & Thivon Avenue, Egaleo, Athens 12244, Greece

Received 31 July 2005; accepted 25 September 2007
Available online 12 October 2007

**Abstract**

This paper presents a novel hardware implementation of a disparity estimation scheme targeted to real-time Integral Photography (IP) image and video sequence compression. The software developed for IP image compression achieves high quality ratios over classic methodologies by exploiting the inherent redundancy that is present in IP images. However, there are certain time constraints to the software approach that must be confronted in order to address real-time applications. Our main effort is to achieve real-time performance by implementing in hardware the most time-consuming parts of the compression algorithm. The proposed novel digital architecture features minimized memory read operations and extensive simultaneous processing, while taking into concern the memory and data bandwidth limitations of a single FPGA implementation. Our results demonstrate that the implemented hardware system can successfully process high resolution IP video sequences in real-time, addressing a vast range of applications, from mobile systems to demanding desktop displays.


*Keywords:* 3D Imaging; Integral Photography; Autostereoscopic display; Hardware; FPGA; Real-time; Compression

## 1. Introduction

Over the past few years, the rapid increase in processing power and graphic card acceleration, combined with improvements in high fidelity optical systems, revived the interest for three-dimensional (3D) applications. Many promising technologies have evolved, ranging from polarizing glasses, used at the early stages of 3D cinema, to most sophisticated techniques like shuttering glasses [1] and more recently autostereoscopic display devices [2]. Autostereoscopy is considered to be the "holy grail" of electronic 3D displays [3], mainly because display devices based on this technology provide 3D stereoscopic viewing without the need of additional eyewear, reducing eye fatigue. In addition, most of these displays allow multiple viewers to experience the 3D effect. These display systems are of great use in medical [4], educational and entertainment [5] applications. However, three-dimensional display devices that currently appear in the market, display only a limited number of different views to the observer. Their main disadvantage is the restrained natural perception of a three-dimensional world, where the viewing angle can change in an arbitrary manner.

A special category of autostereoscopic displays that can provide enhanced sense of depth, full colour support and in most cases multidirectional parallax, functions on the principles of Integral Photography (IP), first introduced by Lippman [6] back in 1908. Recent advances in microoptics and display devices manufacturing procedures, allowed the development of a number of different IP display setups that produce high quality 3D IP images. In Fig. 1, a typical IP capturing and display setup is illustrated.

* Corresponding author.
*E-mail addresses:* dhaik@di.uoa.gr (D. Chaikalis), nsg@di.uoa.gr (N. Sgouros), dmarou@di.uoa.gr (D. Maroulis), ppapag@teipir.gr (P. Papageorgas).

1047-3203/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2007.09.003

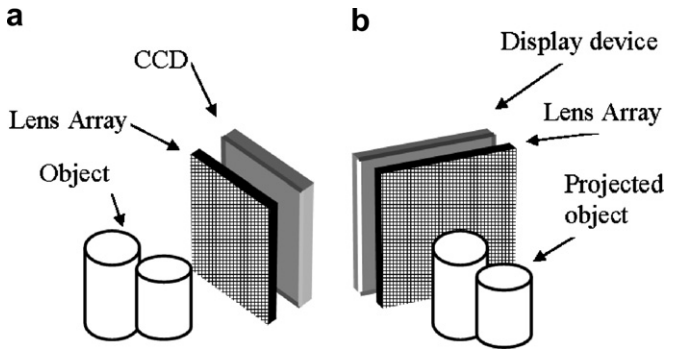


Fig. 1. An IP (a) capturing setup and (b) display setup.

IP images are formed by a number of sub-images that are usually arranged in a square or hexagonal topology, in accordance to the lens array used in the capturing phase. It is evident that neighbouring sub-images exhibit a high degree of correlation, which results in high volumes of redundant information. A portion of an IP image, formed using a lens array with square lenses, is depicted in Fig. 2.

One of the main issues when developing applications based, but not restricted to the IP principle, is the necessity to cope with high resolution images that result in high bandwidth and storage requirements for the capturing and reproduction of 3D objects and scenes. Consequently, a high-efficiency compression scheme of the associated data is crucial. The inherent properties of an IP image gave rise to new techniques for the compression of high correlated data, in order to provide efficient case-oriented algorithms that achieve high compression ratios over traditional methodologies.

To the knowledge of the authors, the most recent compression techniques that are targeted to IP images are mostly based on the utilization of standard lossy compression algorithms like JPEG, based on the two-dimensional Discrete Cosine Transform (2D-DCT) or on higher order DCT techniques [7], with significant results over certain types of IP images. One of the most noticeable drawbacks of the standard JPEG scheme is that it primarily exploits the redundancy present between adjacent pixels inside a confined window that is imposed by the transform used. Such a scheme does not take advantage of the data redundancy that exists in neighbouring windows, and thus creates additional data for image blocks that are highly correlated with previously JPEG encoded blocks. Additionally, the merits of a technique that uses a 3D-DCT transform taking advantage of the data redundancy of adjacent image blocks, are cancelled out by the complexity and the lack of robustness due to proprietary and usually suboptimal quantization tables in high dimensionality spaces.

An alternative methodology for IP image compression is proposed by Sgouros et al. [8], according to which the original MPEG coding scheme is adopted in a certain extent, with major changes in the motion vector estimation unit. Two of the most significant features of this alternative that derive from the properties of the IP image are the a priori knowledge of the search window size and the motion vector directionality. These properties further simplify the architecture, favoring real-time applications, where high compression ratios must be combined with real-time performance of the whole process.

Since the above methodology is targeted not only to imaging but also to video applications, the overall process time becomes a critical factor of the compression technique. Moreover, the time-consuming nature of the specific algorithm imposes the acceleration of the compression task. Aiming to achieve real-time performance, hardware implementation is the dominant way to accelerate the time-consuming parts of the compression algorithm.

Hardware implementations can exploit pipelined and massively parallel processing, accelerating time-critical and high complexity tasks, especially for real-time applications. Such tasks include motion estimation used in various real-time compression schemes.

Although Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs) have

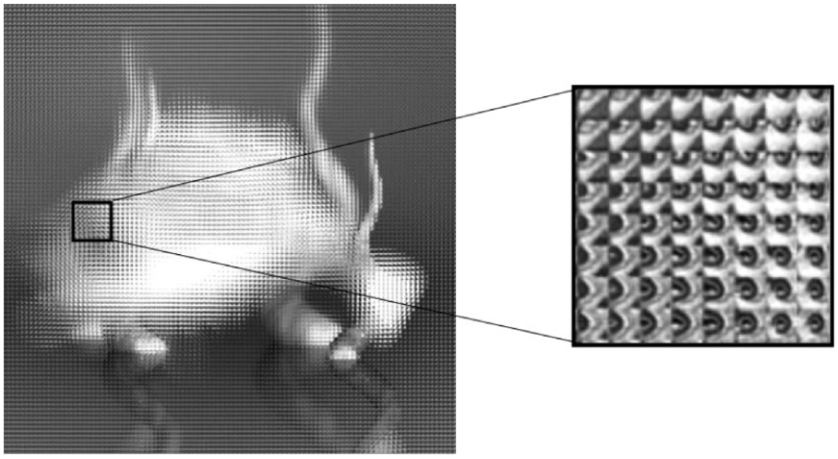

Fig. 2. The inherent redundancy in an IP image.

both been used for hardware implementation, the use of FPGAs has dominated over ASICs in the research and development phase over the last few years [9]. The primary benefit of an FPGA design is a combination of increased flexibility, sufficient performance, faster design times and scalability over ASIC designs [10].

Many different architectures based on the MPEG compression scheme have been proposed for the implementation of video compression modules taking advantage of specific target devices. Recent research interest can be grouped to three main domains:

- ASIC implementations of standard MPEG techniques [11].
- FPGA implementations of parts or simplified MPEG components [9,12,13].
- Novel architectures for parts of the MPEG compression scheme, such as the Sum of Absolute Differences (SAD), the DCT and inverse DCT transforms and the Motion Estimation component [14–16].

A hardware implementation of a system for real-time stereoscopic videoconferencing with viewpoint adaptation is described by Ohm et al. [17]. In this system intermediate views at arbitrary positions are synthesized from the views of a stereoscopic camera system. Disparity Estimation (presented in Section 2) is used for retrieving depth information from two view images in order to synthesize intermediate views by interpolation.

The aforementioned recent works present an interesting approach for image and video compression. However, none of the previous methods has been properly adapted in order to take advantage of IP image or video sequence characteristics.

In this paper, an area-efficient hardware implementation of a real-time motion estimation scheme specially modified for IP images is presented, targeted to real-time 3D capturing and display applications. The main idea is the exploitation of the inherent redundancy that is present in IP images, while considering the memory and data bandwidth limitations of such an implementation, creating a memory-efficient and bandwidth-limited hardware system. Preliminary results presented by Chaikalis et al. [18] revealed that the implementation is deemed suitable for real-time IP image compression.

The rest of the paper is organized in four sections. In Section 2 we describe the algorithm on which the proposed compression architecture is based, and introduce some essential aspects of motion estimation. In Section 3 we illustrate the proposed hardware design, the results of which are apposed in Section 4. Finally, the conclusions and prospects of this study are summarized in the last section.

## 2. Algorithm description

The realized algorithm derives from the basic MPEG scheme for video compression, based on the fact that the elemental images can be treated as a spatial sequence of subsequent frames having a predetermined motion pattern.

A typical MPEG encoder consists of standard quantization maps, encoding procedures, motion vector prediction and coding. Sgouros et al. [8] have proposed a technique based on the above encoder, where certain modifications are made to the motion estimation and vector coding units. These modifications derive from the unique properties of the IP image as described earlier in Section 1.

In detail, the motion estimation module is replaced by a disparity estimation and coding unit where a full pixel disparity estimation strategy is used over an area that is decided to contain the most probable matching parts. This area confinement is easily calculated, considering the geometric relation exhibited by neighboring areas of an IP image. The advantages of this technique derive from the use of standard 2D-DCT and the relatively low-complexity disparity compensation modules, along with the standard quantization and coding techniques.

The proposed method achieves high quality ratios over classic methodologies like JPEG. The Peak Signal-to-Noise Ratio (PSNR) objective metric was used to assess the results of the algorithm, over baseline JPEG, showing that quality gain is over 2 dBs for a wide range of bit rates [8]. Fig. 3 depicts the coding gain for a typical IP image as the one presented in Fig. 2.

The reduced complexity of the compression method allows easy hardware prototyping and scalability of the algorithm. Additionally, the aforementioned properties of an IP image favor the development of highly parallel and thus time-efficient hardware implementations.

### 2.1. Motion estimation

One of the critical tasks of a compression scheme based on an MPEG-like architecture is motion estimation. Motion estimation has been introduced in an attempt to trace the motion of objects within a video scene, i.e. iden-

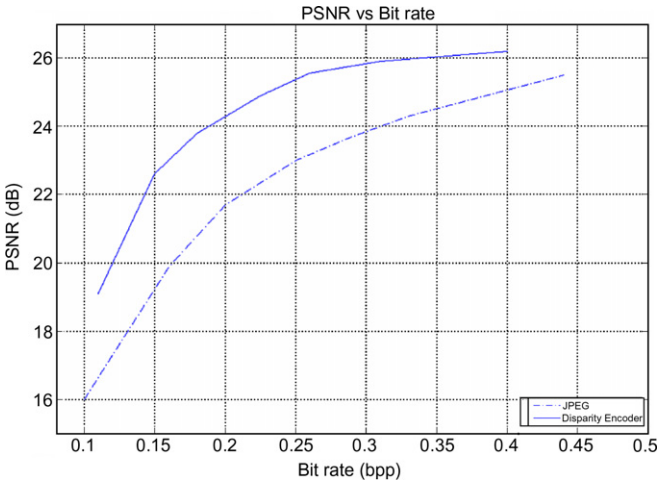


Fig. 3. PSNR values of the proposed disparity encoder compared to baseline JPEG for several bit rates.

tifying the best match between pixels in the current frame and pixels in the reference frame. To this end, a search area within the reference frame must be traversed in order to find the best match. After that, the intensity differences between the pixels must be coded along with the difference in coordinates between the matched locations in the current and reference frame (motion vector). In block-based motion estimation, a search is performed in the reference frame to locate the best match for a block of pixels (named macroblock in MPEG) in the current frame.

Two key issues are associated with motion estimation in general, namely the size of the search area and the metric used for determining the "best match". For both issues, many methods have been proposed [14] in order to reduce the number of computations.

The most commonly used metric is the Sum of Absolute Differences (SAD), which adds up the absolute differences between corresponding elements in the macroblocks. The SAD value calculation process is time-consuming due to the complex nature of the absolute value calculation procedure and the subsequent multitude of additions. For an $8 \times 8$ block, the SAD is calculated as follows:

$$\mathrm{SAD}(U,V) = \sum_{x=0}^{7} \sum_{y=0}^{7} |U(x,y) - V(x,y)|$$

where

$x, y = 0, 1, 2 \ldots 7$ are the spatial coordinates in the pixel domain and

$U(x,y)$, $V(x,y)$ represent arbitrary $8 \times 8$ blocks in adjacent frames. The actual coordinates of these blocks in the frames are chosen by the search algorithm used.

A second issue is the type of transform to be used for the coding of the reference and the residual frames. The Discrete Cosine Transform (DCT) that is used in a variety of multimedia compression systems (JPEG, MPEG, H261), is the dominant candidate, thanks to its relatively low complexity. Despite its design simplicity, even a typical $8 \times 8$ 2D-DCT has a significant computational cost, since it requires execution of 4096 multiplications per window [19].

In multimedia compression applications, it is widely acceptable that, the choice of an $8 \times 8$ window size for the 2D-DCT is a good trade-off between the transform complexity and the achieved spatial decorrelation [16]. The expression of an $8 \times 8$ 2D-DCT is given by the following equation:

$$F_{m,n} = \sum_{i=0}^{7} \sum_{j=0}^{7} f_{i,j} \cos\left(\frac{(2i+1)m\pi}{16}\right) \cos\left(\frac{(2j+1)n\pi}{16}\right)$$

where

$m, n, i, j = 0, 1, 2 \ldots 7.$

### 2.2. Compression method

The compression method is applied on an IP image in order to exploit its correlation properties, using each sub-image as a spatial frame and assuming that is highly correlated with its neighbouring sub-images. In this manner, all time dependent quantities present in MPEG are transformed to equivalent spatial ones.

One of the most valued features of an MPEG algorithm are motion vectors that determine the displacement of the best match macroblock in the reference frame with respect to the macroblock in the predicted frame. These vector quantities are substituted by disparity vectors that reflect the coordinates offset of the best match between two sub-images which act as a reference and predicted frames. Moreover, disparity vectors are expected to have constraints regarding the maximum and minimum offset in coordinates between two sub-images, due to the knowledge of the characteristics of the elemental lenses that comprise a lens array.

In this work, the size of the search area is defined by the properties of the IP images as already discussed. In detail, horizontally subsequent sub-images are perfectly aligned and this constraints the search window to an $8 \times 32$ pixel area. The absence of vertical disparity between horizontally adjacent lenses is due to the manufacturing specifications of the lens array [8]. Hence, a unidirectional exhaustive block search method is performed ensuring an optimal block match. The time performance of this method is close to the ones used in the general case by MPEG due to the unidirectionality of the search and the search area selection, which is based on the IP image properties, reducing the overall computational cost. It must be noted though that the exhaustive block search method is selected, since the proposed technique targets to high quality compressed images. This differentiates the technique from the temporal MPEG algorithm aiming to achieving low bit rates for the encoded sequence.

Fig. 4 is an illustration of an IP image segmentation in spatial blocks along with the search method followed. An intra sub-image (I-type) of size $32 \times 32$ is encoded in a JPEG like manner by the use of the two-dimensional DCT transform, followed by quantization and entropy coding to optimize performance. Specifically, the I-type sub-image is segmented to $8 \times 8$ pixel blocks and a 2D-DCT is applied on each block using the quantization table as described in the JPEG standard [20]. The coefficients are zigzag scanned and finally coded with a combination of run-length and Huffman procedures.

Based on the reconstructed I-type sub-image, the disparity vector matrices are created for the two neighbouring P-type sub-images, which are located to the left and to the right of the I-type sub-image. This group of three sub-images form what will hereafter be referred to as a P–I–P sub-image triplet. Due to the unidirectional search method, the disparity vector degenerates to a scalar, which denotes the horizontal disparity offset of each P block in its corresponding I-type sub-image row, and has a value from 0 to 24. After the disparity vector matrices' creation, an estimation $P_{\mathrm{est}}$ of the two P-type sub-images of $32 \times 32$ pixel size is formed by using the reconstructed I-type

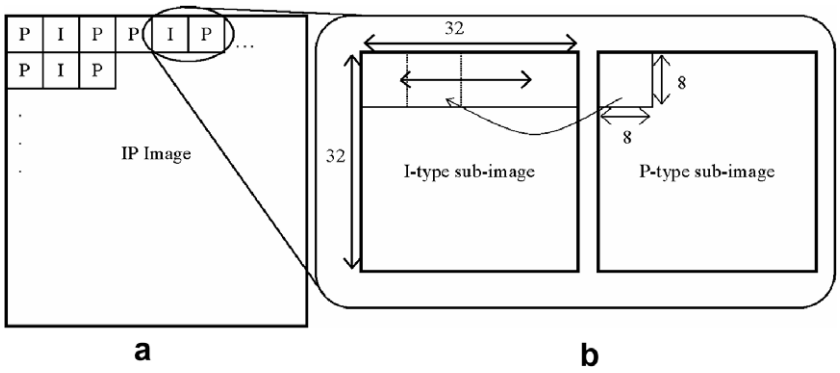


Fig. 4. (a) The I-type and P-type sub-images in an IP image, (b) block search area outline.

sub-image and the proper disparity vectors for each P-type sub-image. Finally, the $P - P_{est}$ difference between the estimated and the actual P-type sub-images is calculated and compressed.

In order to calculate the disparity vectors, the P-type sub-image is segmented to sixteen $8 \times 8$ blocks (Fig. 4), so 4 rows and 4 columns of blocks are created. The SAD metric is used for determining the best match of each $8 \times 8$ block of each P-type sub-image into the I-type sub-image. For each $8 \times 8$ block, the search area in the I-type sub-image is defined starting by the same top-left coordinates as the ones of the P block and advancing until the right end of the line of blocks in the sub-image. The exhaustive search method indicates that the search advances one pixel at a time, comparing in this manner the leftmost P block with 25 blocks in the I-type sub-image, traversing the row from its leftmost edge up to its rightmost edge - horizontal coordinates from 0 up to 24. The next P block is compared with 17 blocks—coordinates from 8 up to 24—and the third P block with 9 blocks—coordinates from 16 up to 24. In total, $25 + 17 + 9 = 51$ comparisons are needed for each row of blocks in the P-type sub-image, and for each comparison one SAD value is calculated, resulting to 51 SAD values for each row of blocks in the P-type sub-image.

The aforementioned calculations and the block matching search are implemented into hardware, in order to achieve real-time performance. The block diagram of the above compression scheme is presented in Fig. 5. The shaded components represent the part that is implemented in hardware, which will be referred to as Disparity Vectors Matrices' Generator (DVMG) henceforth.

## 3. Hardware design

The DVMG architecture contains the initial compression parts of the algorithm, and the creation of the disparity vectors. In order to create a robust digital system, certain facts are taken into account, concerning the implementation platform and the hardware requirements. Given that one of our objectives is a timing comparison to the software realization, the FPGA fitted on the Celoxica RC1000-PP PCI board was tested against the software version of the algorithm. Moreover, the additional hardware on the development board is exploited to a maximum degree in favour of the digital architecture residing in the FPGA device.

The proposed hardware is designed using Very High Scale Integrated Circuits (VHSIC) Hardware Description Language (VHDL). The hardware modules are implemented in FPGA using a Celoxica RC1000-PP PCI board. The FPGA on the development board is a Xilinx Virtex XCV-2000E chip, with an equivalent area to 2 million logic gates. The board includes 8 MB memory modules organized in four different banks, each with separate address and data buses. In this way, every memory bank can be accessed independently in the same clock cycle. The FPGA implements a dedicated 640 kbits (80 Kbytes) dual-ported memory, which can be used in any width desired [21].

In total, four FPGA memory modules are realized in the FPGA dedicated memory. Three FPGA memory modules are used for storing the sub-images and one for storing the results. In order to maximize memory utilization and optimize the speed of the read operations, a cell size equal to 64 bits was chosen, that can store the intensity values of 8 pixels. For each memory module 11 bits address space is required. Each memory module used for sub-image storage can store up to 16 sub-images with size $32 \times 32$ bytes. These three FPGA modules occupy 48 Kbytes out of the 80 Kbytes of the dedicated memory available on the device and store 16 P–I–P sub-image triplets in each processing phase. The rest of the dedicated memory is allocated to the results' memory module where the calculated disparity vectors are stored.

Each disparity vector is expressed using 5 bits, but is allocated an 8-bit value corresponding to the width of the memory cell of the FPGA dedicated memory. For the 16 P–I–P sub-image triplets that are stored in the FPGA sub-image memory modules, 32 disparity vector matrices are created (see Fig. 4). Each matrix contains $4 \times 3$ disparity vector values, since no disparity estimation is performed for the rightmost P block of each row, which is independently coded. In total, $4 \times 3 \times 32 = 384$ bytes of results' memory are needed to store the disparity vector matrices for 16 P–I–P triplets.

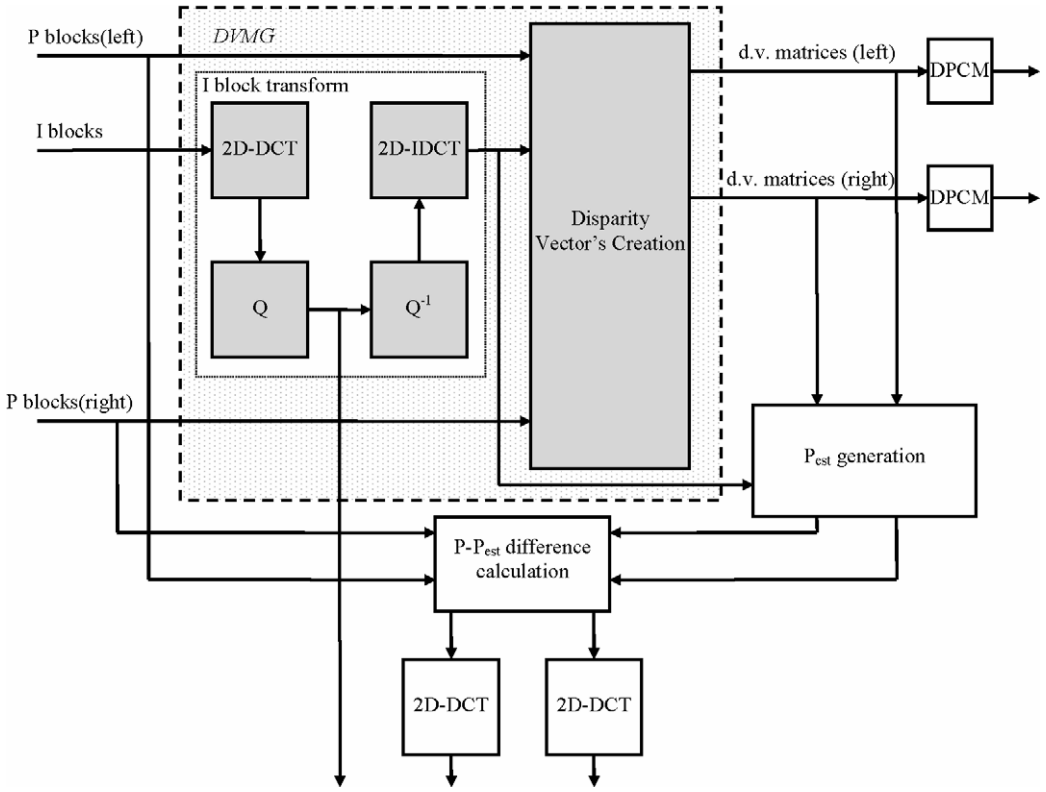


Fig. 5. Block diagram of the IP image compression scheme as detailed in [8]. The Disparity Vector Matrices' Generation (DVMG) represents the part that is implemented in hardware.

In our implementation, each sub-image type is downloaded from the host to a separate board memory bank. In total, 6 Mbytes of image data are transferred to the board memory. In this way, the simultaneous access of the board memory banks by the DVMG allows pixels of three of the sub-images forming the P–I–P sub-image triplets to be transferred to the FPGA memories in parallel.

The memory layout of the FPGA and board memory modules along with their sizes and interconnections are illustrated in Fig. 6. The memory interface is designed in the FPGA and rearranges the image data while transferring them from the board memory modules to the FPGA ones.

In the remainder of this section, the DVMG's architecture and operation phase are presented in detail.

### 3.1. System overview

Right after the FPGA device is programmed with the DVMG, the host downloads a part of an IP image up to 6 MB in size to the board memory using DMA transfer mode of operation. The DMA transfer is performed in a rate of approximately 90 MB/s; for the appropriate data to be downloaded to the board memory, about 66.7 ms are required. Upon completion, the DVMG is signalled through a host-to-FPGA signal in order to start processing the data.

The first task of the DVMG is to transfer 16 P–I–P sub-image triplets to the FPGA memory. From there on, a series of operations are executed in order to calculate and write to the results' memory the final data, which are the disparity vector matrices for the 16 sub-image triplets. Each processing phase ends when all the data in the available FPGA memory are processed. The next processing phase initiates by transferring the next 16 P–I–P sub-image triplets from the board memory to the FPGA memory, and the disparity vector matrices are created in the same fashion. The operation phase of the DVMG ends when all the image data residing in the board memory have been processed and the results are transferred from the results' memory module to the board memory.

When the DVMG completes its operation phase, the desired results are residing in the board memory. The host is notified and uploads the final data, and the FPGA system returns to its initial state, waiting for a new start signal from the host.

The DVMG is comprised of the following sub-systems: the two-dimensional Discrete Cosine Transform (2D-DCT) Unit, the Quantizer and the Inverse Quantizer, the 2D Inverse-DCT (2D-IDCT) Unit, the Disparity Vectors' Creation Unit (DVC Unit or DVCU), an Address Generation Unit, and the Control Unit along with the necessary FPGA memory modules. The first four units (see Fig. 5) mentioned above are grouped to the I block transform module, that is responsible for compressing and decompressing the I-type sub-images. Fig. 6 presents a block diagram of the hardware system.

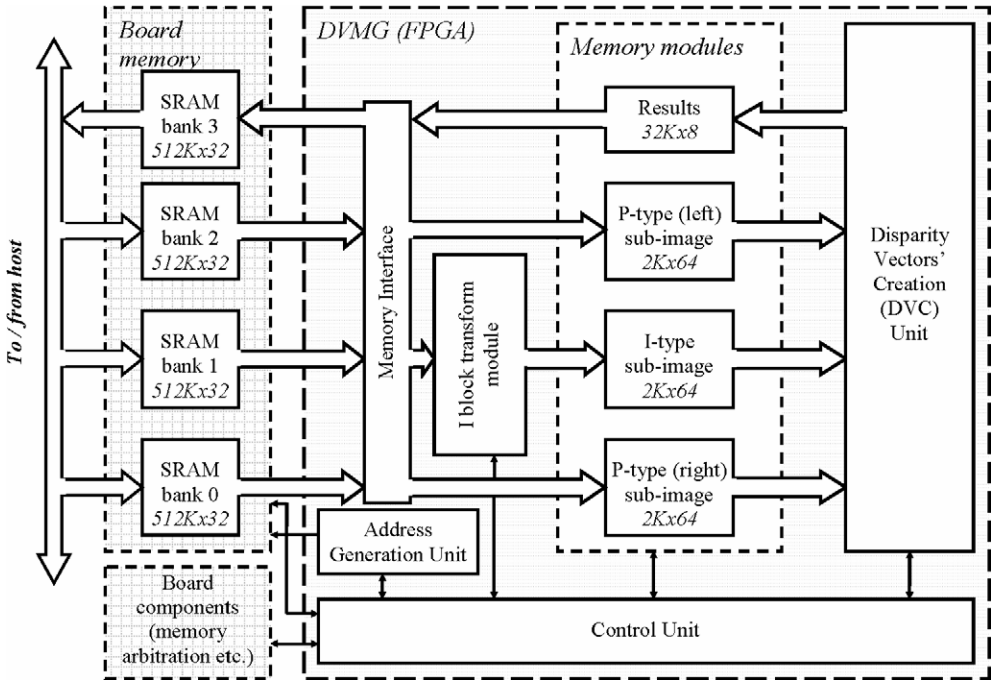


Fig. 6. An overview of the DVMG.

### 3.2. Datapath

The FPGA memory used to store the sub-images is organised into three different banks. The P-type sub-images are written in two separate P-type sub-image memories, one for the P-type sub-images preceding an I sub-image and one for the P-type sub-images following an I-type sub-image, while the I-type sub-images are stored in a separate bank.

As soon as the image data are read from the board memory, the I-type sub-images are passed through an $8 \times 8$ 2D-DCT transform, and then quantized according to the standard JPEG luminance table [20]. Finally, they are inverse-quantized and fed in to an $8 \times 8$ inverse 2D-DCT transform. At the end of this processes, the reconstructed I-type sub-images are stored in the I-type sub-image FPGA memory.

The Disparity Vector's Creation Unit is responsible for generating the disparity vector matrices for the P-type sub-images by calculating the minimum SAD value for each $8 \times 8$ block (see Fig. 4). The SAD value calculation is performed by comparing I-type and P-type sub-image pixel values, according to the procedure explained in this section.

For the creation of the disparity vectors, the Sum of Absolute Differences (SAD) metric is implemented (see Section 2.1). The SAD calculations could be performed in a completely parallel manner, increasing the overall speed of the system. Such an approach though poses the problem of high bandwidth and area demands, which requires a multitude of FPGAs [15].

In the proposed method, instead of implementing one Absolute Difference (AD) unit for each pair of pixels in the compared blocks (e.g. 64 AD units for an $8 \times 8$ block comparison), we implement only 8 of these units for each $8 \times 8$ block comparison, in order to avoid the distribution of the implementation to more than one FPGAs. By successive repetition, each SAD unit produces a SAD value for an $8 \times 8$ block in 8 clock cycles, by processing in parallel 8 pixel pairs from one P-type and I-type sub-image block in every clock cycle. The hardware design of the implemented AD calculation and the adder tree are based on a method proposed in [15]. Fig. 7 presents the SAD unit for 8 pixel pairs, which is designed in VHDL and implemented in the system.

The DVMG effectively reduces the I-type sub-image pixel reads for calculating the minimum SAD value for a line of block in the P-type sub-image. The sub-images are only transferred once from the FPGA memory to the DVC Unit (Fig. 6) in order to create the disparity vectors for the P blocks. This translates to 92% less clock cycles needed for performing the SAD calculations compared to the simplest approach of reading each pixel as many times as it participates in the calculations.

As explained in Section 2.2, 51 SAD values are computed for each line of blocks in the P-type sub-image. Therefore, the straightforward way to calculate these values in parallel would be to use 51 SAD Units [18]. Such an implementation though would not fit into our target FPGA device, so we experimented with several other schemes in order to reduce the number of units. It was concluded that the use of 25 SAD Units offers the best compromise between area coverage and architecture complexity. Specifically, if less than 25 SAD Units were implemented, we would need a very complex control mech-

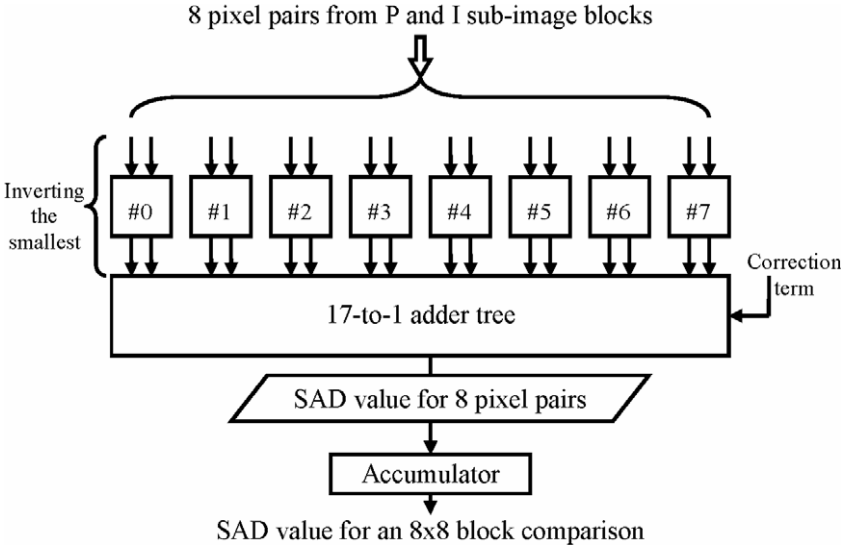


Fig. 7. The SAD unit for 8 pixel pairs, used to calculate the SAD value for an $8 \times 8$ block.

anism and connection scheme in order to distribute the image data to the processing units. By using 25 SAD Units we only need to insert an idle clock cycle after the SAD Unit has been active for 8 clock cycles. During this idle cycle, the data is cleared from its accumulators and registers.

A simplified timing chart of an operation phase of the SAD Units inside the DVC for a $8 \times 32$ pixel area is presented in Fig. 8. For the specific clock cycles, the $8 \times 32$ pixel areas of a P-type and I-type sub-image are read and 51 SAD values are calculated, as explained in Section 2.2. In each clock cycle, one 8-pixel column of the P-type and I-type sub-image is retrieved from the respective FPGA memory and forwarded to the DVC Unit. The P, I addresses indicate pixel columns of the P and I $8 \times 32$ area respectively that are available in the input stage of the DVC in a specific clock cycle. Two delay lines are created inside the DVC, in order to feed all the SAD Units with the appropriate data, and are denoted in the chart using (addr-1) and (addr-2). The I pixel columns are forwarded to the SAD Units only on the cycle that they are read from the FPGA memory, while the P pixel columns are rippled through the Units using a structure of delay registers in order to be available throughout the 32 clock cycles.

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. The Comparison module also generates the offset value to pair with the smaller SAD value. The offset value represents the I block to which the minimum SAD value corresponds. This offset value constitutes the disparity vector that will be an element to the block's disparity matrix.

As long as the 32 disparity vector matrices for the 16 P–I–P sub-image triplets that reside in the FPGA memory are stored in the results' memory, the next P–I–P triplets are transferred from the board memory to the FPGA memory and 32 more disparity vector matrices are generated with the processes described in the above sections. When all image data is processed and the matrices are complete, they are transferred from the FPGA results' memory to the board memory, so that the host can access them. The DVMG then notifies the host that the results for the image are ready, and returns to its initial state, waiting for a host signal.

## 4. Results

As described in the previous sections, the DVMG has the ability to calculate the disparity vector matrices for an IP image by processing 16 P–I–P sub-image triplets on each processing phase. These matrices are temporarily stored to FPGA memory and, upon processing of the whole image, they are transferred to the board memory, where they are accessible by the host computer.

A simplified timing chart of the main units comprising the DVMG, during a processing phase of 16 P–I–P triplets is illustrated in Fig. 9. The chart also illustrates the time measured in clock cycles, starting from 0, on which the most important transitions during the processing phase take place. The total time required for calculating the disparity vector matrices for 16 P–I–P sub-image triplets adds up to 20 869 clock cycles. The DVMG can be clocked with a maximum frequency of 11.29 MHz occupying 87% of the Virtex XCV-2000E area. The overall time for processing an entire image depends on the image size. As an example, for an image with a dimension of $1024 \times 768$ pixels, 31.4 ms are required. In other words, 31 such images are processed per second by the implemented hardware system. Fig. 10 depicts the number of IP images that the hardware system can process per second, for images with several dimensions.

The operating clock rate of the DVMG is defined by the maximum operation frequency of its slowest compo-

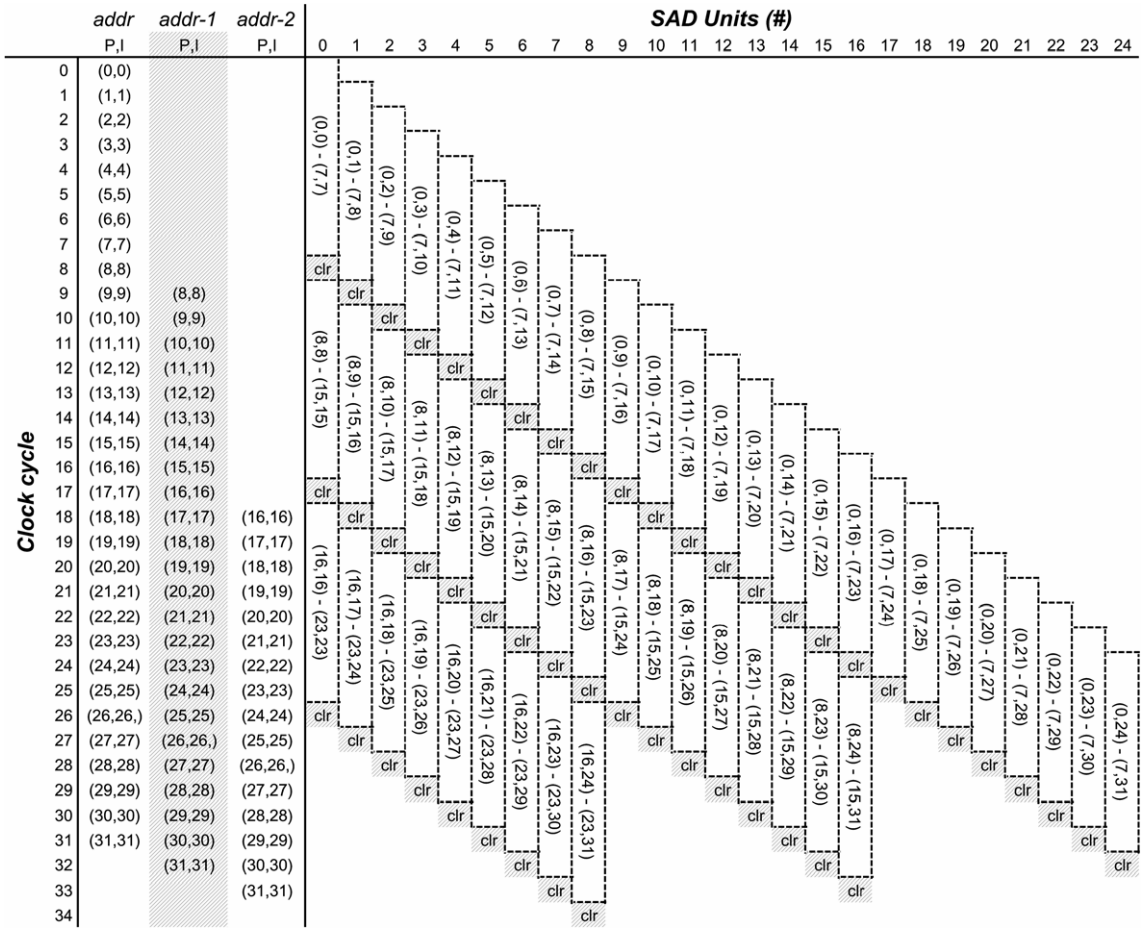


Fig. 8. A simplified timing chart of an operation stage of the SAD Units.

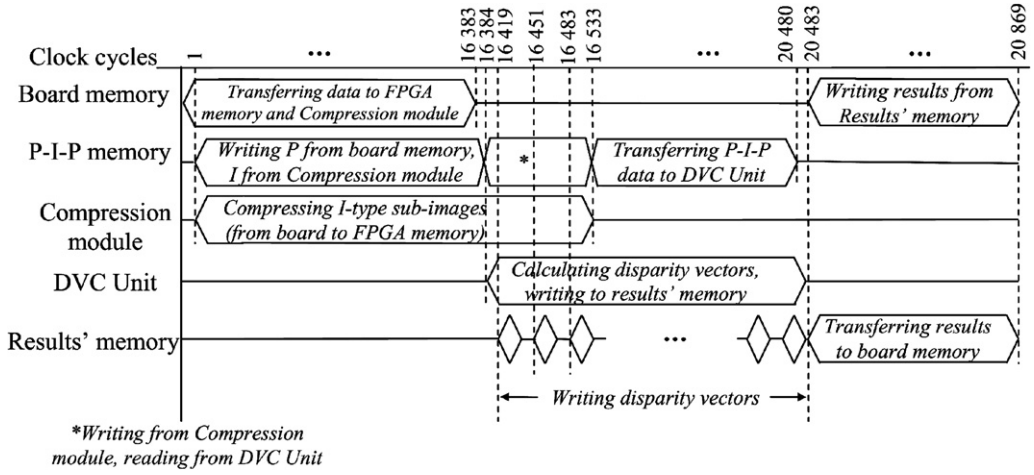


Fig. 9. Timing chart of the main units comprising the DVMG. The clock cycle count corresponds to a processing phase of 16 P–I–P triplets.

nent. Being this the case, the low clock rate of the system is due to the combinatorial logic of the adder trees in the SAD units. If extensive pipelining were to be used for the current architecture, area coverage would exceed the available FPGA device. Using larger FPGA devices would improve the performance of the hardware system, as discussed in the last section. Nevertheless, as revealed in Fig. 10, the DVMG is able to process IP images of significant size in real time, targeting a resolution range from moderate resolutions, normally used in mobile systems, to high resolutions for demanding 3D desktop applications. Therefore, the implemented hardware system is deemed suitable for real-time processing of video sequences.

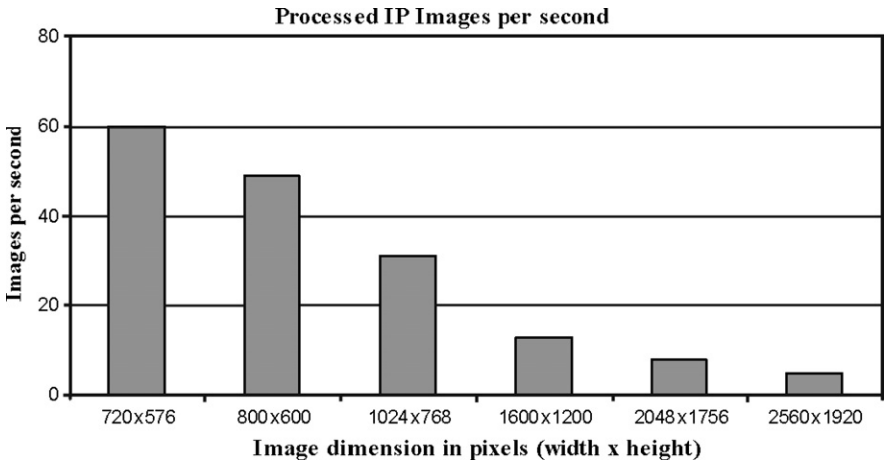


Fig. 10. The number of IP images that the DVMG can process in every second.

The timing results are compared to a software approach, created in C programming language, that produces the same disparity vector matrices for an IP image. The host PC used for the realisation of the FPGA and software approaches was a Pentium 4 2.8 GHz unit, equipped with 512 MB of RAM. Measuring the time needed for the software to complete the generation of the disparity vectors for several sizes of IP images, a speedup of 2 orders of magnitude was measured in favour of the hardware realisation. Specifically, several seconds are required for the software to create the matrices even for small image sizes, while the FPGA produces the final data approximately 100 times faster. Table 1 illustrates the advantage of FPGA processing when compared to the software approach.

## 5. Discussion—Future work

The continuously growing interest for three-dimensional applications has revived the research for methods and techniques for 3D image capturing and reproduction. Today's means of processing and display need to confront the high bandwidth and storage requirements derived from high resolution 3D images. In this paper, a hardware implementation of a disparity estimation scheme is presented, targeted to IP image coding.

Area optimization was the most critical task of the DVMG, given that multiple FPGAs should be used if a completely parallel processing scheme is adopted. Targeting to a single Virtex-E FPGA with an area capacity equivalent to 2 million logic gates, we succeeded in implementing a novel, area-efficient, real-time disparity estimation hardware system. The latter has the ability to process video-sized images in a rate greater than 30 frames per second, being 100 times faster compared to the current software realisation. Hence, it is evident that the proposed hardware system can constitute a robust accelerating component in a hybrid IP image compression system.

Using an FPGA device with larger available area could lead to improvements in the overall performance with the hardware implementation of additional time consuming modules of the compression algorithm. Moreover, in larger FPGAs, a more efficient pipelining scheme would be realised, leading to higher clock rates and additional performance gain.

Table 1
Timing comparison between FPGA and software approach for disparity vector matrices' generation for an IP image

| Data processor | FPGA (ms) | Software (ms) |
|---|---|---|
| *Image dimension in pixels* | | |
| $800 \times 600$ | 20.33 | 2800 |
| $1024 \times 768$ | 31.42 | 4100 |
| $1600 \times 1200$ | 75.79 | 8700 |

## Acknowledgments

This work was realized under the framework of the Reinforcement Programme of Human Research Manpower ("PENED 2003"—03ED656), cofunded by the General Secretariat for Research and Technology, Greece, and the European Social Fund.

## References

[1] Developers Handbook, Stereographics 1997, available from:www.stereographics.com.
[2] M. Halle, Autostereoscopic displays and computer graphics, computer graphics, ACM SIGGRAPH 31 (2) (1997) 58–62.
[3] Janusz Konrad, Visual communications of tomorrow: natural, efficient and flexible, IEEE Communications Magazine 39 (1) (2001) 126–133.
[4] H. Liao, S. Nakajima, et al., Intra-operative real-time 3D information display system based on integral videography, MICCAI' 01, LNCS 2208 (2001) 392–400.
[5] P. Harman, Home based 3D entertainment—an overview, in: Proc. ICIP(1), 2000, pp. 1–4.
[6] G. Lippman, La Photographie Integrale, Comptes Rendus de L Academie Des Sciences 146 (1908) 446–451.
[7] R. Zaharia, A. Aggoun, M. McCormick, Adaptive 3D-DCT compression algorithm for continuous parallax 3D integral imaging, Elsevier, Signal Processing: Image Communication 17 (2002) 231–242.

[8] N. Sgouros, A. Andreou, M. Sangriotis, P. Papageorgas, D. Maroulis, N. Theofanous, Compression of IP images for autostereoscopic 3D imaging applications, in: Third International Symposium on Image and Signal Processing and Analysis (ISPA03), Rome, Italy, September 18–20, 2003.

[9] S. Ramachandran, S. Srinivasan, FPGA Implementation of a novel, fast motion estimation algorithm for real-time video compression, in: Ninth International Symposium on FPGAs, Monterey, Canada, 2/2001.

[10] S. Rathnam, G. Slavenburg, An architectural overview of the programmable multimedia processor, TM-1, in: Proc. COMPCON'96, 1996, pp. 319–326.

[11] V.G. Moshnyaga, K. Tamaru, A memory efficient array architecture for real-time motion estimation, in: Eleventh International Parallel Processing Symposium (IPPS'97), Geneva, Switzerland, April 01–05, 1997, pp. 28–32.

[12] N. Roma, T. Dias, L. Sousa, Customisable core-based architectures for real-time motion estimation on FPGAs, in: Proc. 13th International Conference on Field Programmable Logic and Applications—FPL'03, Lisboa—Portugal, 1–3 September, 2003, pp.745–754.

[13] D. Zandonai, L. Carro, S. Bampi, A.A. Suzin, An architecture for MPEG motion estimation, VII Workshop Iberchip IWS'2001, 2001, Montevideo, 1. pp. 90–95.

[14] S. Wong, S. Vassiliadis, S. Cotofana, A sum of absolute differences implementation in FPGA hardware, in: 28th Euromicro Conference (EUROMICRO'02), Dortmund, Germany, September 04–06, 2002, pp. 183–186.

[15] S. Wong, B. Stougie, S. Cotofana, Alternatives in FPGA-based SAD implementations, in: IEEE I.C. on Field Programmable Technology 2002 (FPT'02), Hong Kong, December 2002.

[16] M. Martina, A. Molino, F. Vacca, Reconfigurable and low power 2D-DCT IP for ubiquitous multimedia streaming, in: IEEE International Conference on Multimedia and Expo (ICME 2002) 2, August 26–29, 2002, pp. 177–180.

[17] J.-R. Ohm, K. Gróneberg, E. Hendriks, E. Izquierdo M., D. Kalivas, M. Karl, D. Papadimatos, A. Redert, A realtime hardware system for stereoscopic videoconferencing with viewpoint adaptation, Image Communication, special issue on 3D technology, January 1998.

[18] D. Chaikalis, D. Maroulis, N. Sgouros, P. Papageorgas, N. Theofanous, An area-efficient FPGA implementation of a disparity estimation scheme for real-time compression of IP images, in: International Conference on Signals and Electronic Systems (ICSES), Poznan, Poland, 2004, pp. 309–312.

[19] R. Woods, A. Cassidy, J. Gray, VLSI architectures for field programmable gate arrays: a case study, IEEE Symposium on FPGAs for Custom Computing Machines, 1996.

[20] W.B. Pennebaker, J.L. Mitchell, JPEG Image Compression standard, VNR, NY, 1993.

[21] Celoxica RC1000-PP development board: Hardware Reference, www.celoxica.com.