HARDWARE IMPLEMENTATION OF A STEREO CO-PROCESSOR IN A MEDIUM-SCALE FPGA

J. A. Kalomiros[1] and J. Lygouras[2]

[1)] Department of Informatics and Communications, School of Technological Applications, Serres Institute of Education and Technology, Terma Magnisias, 62100 Serres, Greece, ikalom@teiser.gr

[2)] Section of Electronics and Information Systems Technology, Department of Electrical and Computer Engineering, Polytechnic School of Xanthi, Democritus University of Thrace, Greece

Abstarct: We present the design of a hardware co-processor for stereo depth detection, based on a parallel implementation of the Sum of Absolute Differences algorithm. We follow model-based design and create a parametrizable open source VHDL library component appropriate for integration within a System-On-a-Programmable Chip (SOPC). We target a Field Programmable Gate Array (FPGA) board featuring external memory and other peripheral components and implement the control path with a Nios II embedded processor clocked at 100MHz. The hardware co-processor produces dense 8-bit disparity maps of 320x240 pixels at a rate of 25 Mpixels/sec and can expand the disparity range from 32 to 64 pixels with appropriate memory techniques. Essential resources can be as low as 16000 Logic Elements, while by migrating to more complex devices the design can easily grow to support better results.

Keywords: Real-Time Stereo Vision, Hardware design, FPGA, Embedded Processor, Model-based Design

1. INTRODUCTION

Stereo vision is the process of estimating the depth of scene points from their change in position between two or more images. It has potential uses in object recognition, robotics, navigation systems, virtual reality etc. [1]. The heart of stereo vision systems is solving the correspondence problem i.e. finding for each point in one image the matching point in the other image. The resulting displacement of a projected point in one image with respect to the other is called disparity, while the set of all disparities is termed a disparity map [2]. Once correspondence between images is established, a 3-D point in the real scene can be found by triangulation.

Stereo matching is one of the most active research areas in computer vision. Many algorithms have been developed and address various difficulties, like occlusion, lack of texture or noise [3,4]. Most stereo correspondence techniques exploit a binocular geometry constraint referred to as epipolar constraint, which reduces the correspondence problem to a search along respective epipolar lines. By using rectified stereo pairs epipolar lines coincide with image scan-lines [2,3].

Methods to find correspondence are always computationally intensive and when executed by general purpose computers have a limited potential for real-time calculations of dense disparity maps. A few real-time stereo systems based on desktop PCs have appeared recently [5,6], however most real-time implementations make use of special-purpose hardware, like Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs). Custom hardware usually

exploits hardware-friendly algorithms like the Sum of Absolute Differences (SAD) or phase correlation techniques [7,8]. Most real-time systems are inflexible, difficult to design and expensive machines, with virtually no potential to parametrize, re-use or share their resources. In recent years FPGA systems are considered to bridge the flexibility gap between pure software and hardware. They present an advantage over ASICs, in that they are re-programmable, less expensive for prototyping and have a relatively short design cycle. They also perform many times better than pure DSPs since they exploit the inherent parallelism in many vision algorithms [9]. Sometimes designers combine FPGAs with DSPs for superior performance [10].

In this work, we present in detail the design of a FPGA hardware co-processor that produces dense disparity maps from rectified stereo pairs. It is designed as a VHDL library component and can be readily imported in Altera's SOPC-Builder tool as a part of a System-on-a-Programmable-Chip. We follow model-based design and can easily parametrize the function from within the model design before we translate it to VHDL. Pixel streaming is controlled by a Nios II embedded soft processor while the system also includes external memory and a communication channel with other systems. We target a Cyclone II 2C35F672C6 FPGA chip on a DSP development board. The function can be shared with anyone using the same design platform and can target any compatible chip with adequate resources. It can be used with a variety of I/O equipment by using appropriate controllers. Although our implementation is FPGA-specific, the design can also be implemented in ASIC with comparable performance. Designing for ASICs would also address the problem of fixed resources inherent in FPGAs and would be cost-effective in case of high-volume production.

2. SUMMARY OF THE ALGORITHM

In this implementation we use the Sum of Absolute Differences (SAD), which is a classic algorithm for finding stereo correspondence and a variation of the well known Sum of Squared Differences (SSD). Assuming rectified images the best match for a point in one image is found by comparing a square window centered at this point against windows of equal size centered at points on the corresponding scan-line in the other image. The main idea of block matching across epipolar scan-lines is shown in Fig. 1.

The metric used for similarity is the sum of absolute intensity differences across the window:

$$\sum_{u,v} \left| (I_1(u+x, v+y) - I_2(u+x+d, v+y) \right| , \qquad (1)$$

where *(x,y)* is the central pixel in the first image, *(x+d,y)* is a point on the corresponding scan-line in the other image displaced by *d* with respect to its conjugate pair and *u,v* are indices inside the window. The point that minimizes the above measure is selected as the best match. To simplify the matching process the search across a scan-line can be limited within a range of expected disparities instead of shifting the window across the whole scan-line. A good discussion of fast implementations of SAD can be found in references [11,12].

SAD algorithm has its drawbacks, namely it may fail to produce correct disparities at occlusion boundaries and can be inconsistent in the case of non-uniform lighting. Implementing left-right consistency checks or using normalized versions of the algorithm can minimize outlier points due to these effects. Median filters are also commonly used to eliminate outliers caused by occlusion regions and image noise [3,11].

From a hardware point of view the SAD function requires only adders and comparators for which modern FPGAs provide good support. Moreover, correlation algorithms like SAD are characterized by regular structures, abundant parallelism and linear data flow. These features make SAD hardware friendly and a good candidate for reconfigurable hardware.

3. HARDWARE DESIGN

We modeled and simulated the hardware co-processor using DSP-Builder, a tool that combines the Mathworks' MATLAB and Simulink design tools with VHDL design flow. DSP-Builder contains bit- and cycle-accurate Simulink blocks which cover basic digital operations as well as complex functions [13].

The basic design follows two stages. The first stage in Fig. 2 is a scan-line buffer and aims to produce a 3x3 comparison window across both image paths. For this purpose, image lines are shifted into the line-buffer and pixels are stored in delay-lines, as shown in Fig. 2.

Two successive z^{-1} delay blocks produce the two neighboring pixels for each pixel on both scan-lines, while a z^{-320} delay block produces the neighboring pixel in the previous scan-line. This assumes a scan-line of 320 pixels and is a parameter that can change in order to support different image resolutions. Line buffers are implemented as embedded RAM blocks in the FPGA chip.

The next stage, shown as a block in Fig. 3, is a parallel structure that hosts 32 streaming pixels of each of the three input lines of image2 and their neighbors in a 3x3 window. These pixels are compared in parallel with an equal window in image1. In this stage the sum of absolute differences is produced across the 32-pixel parallel structure in one execution cycle. In Fig. 4 four such comparisons are suggested, grouped in a 4-pixel stage, where only one of the three steaming lines per image is shown. The calculations at the first pixel are shown in gate-level detail, while the next 3-pixels calculations are shown in squares. The stage of Fig. 4 is then repeated by cascading eight times to produce 32 values of SAD, as shown in Fig. 5.

The stage of Fig. 3 makes heavy use of chip resources absorbing approximately 6000 Logic Elements for every 16 additional pixels in the range of disparity. It exploits both pixel-level and window-level parallelism: it produces Absolute Differences of all the pixels of reference and comparison windows in parallel, while it makes parallel comparisons of the reference window with all candidate windows.

The sum of absolute differences, produced at the stages of Fig. 4 for every pixel in the disparity range, is input to a next stage, shown as the far right block in Fig. 5, where the minimum absolute difference is calculated. For this purpose, we build a square matrix where every input SAD value is compared with every other input value and sets the output of an AND gate if it is found to be the minimum value. This principle is manifested in Fig. 6, for a total of four pixels. Additional hardware, not shown in Fig. 6, produces a tag-value (0-3) attributed to the winning pixel with the minimum SAD. The minimum input SAD value, out of all four, is also output from this stage.

This matrix can be expanded by repeating this basic structure and connecting rows and columns to find in parallel the minimum of any number N of input values. The result would be a matrix with NxN comparators. However, in order to avoid the rapid increase in the volume of logic elements required for a massive parallel matrix, we repeat the simple stage of Fig. 6, in order to derive within one clock cycle the minimum SAD at every four pixels. This scheme is implemented with eight stages, for a total of 32 SAD

values. The minimum results of equal number of comparisons are produced, along with the corresponding pixel tag-value. Then, two additional layers with appropriate pipelining are used in order to produce the final minimum output. A final tag value attributed to the winning pixel out of the total of 32 in the disparity range is also derived, and equals the actual disparity value.

In our implementation the two images are inserted in the co-processor as a multiplexed serial pixel stream (8-bit grayscale) from a 16-bit data bus. A total of four main clock cycles are needed in order to obtain one pixel per image and therefore every pixel needs two clock cycles in order to move one step into the processing structure. This results to four main clock cycles for each disparity pixel in the output stage. The input stage is shown in Fig. 7, but it can be customized to meet the requirements of different input schemes. Streaming lines of both images are synchronized by means of proper delays, as shown in Fig. 7. Data transfer and control are described in section 5.

In the above design, frame resolution only affects the depth of the delay lines in the stage of Fig. 2. In order to implement delay lines for a resolution of 320x240 pixels we need approximately 20000 bits of embedded memory out of a total of 480000 available in a 2C35 Cyclone II chip. As a result frame resolution in the above design is not really an issue since pixels are streaming rather than stored. However, the magnitude of the range of disparities supported by the system as well as the magnitude of the comparison window depends on the available resources. In the following section we describe an architecture which is able to double the disparity range from 32 pixels to 64, while keeping the necessary resources low.

4. EXPANDING THE DISPARITY RANGE

Doubling the cascading blocks in Fig. 5 is a direct way to double the disparity range, however this technique is very demanding with respect to chip resources. In this paragraph we describe a more efficient architecture that can double the disparity range, keeping the necessary resources low. In principle, this new design keeps the main structure as in Fig. 3, hosting the same total range of 32 pixels. In this way, the necessary resources remain approximately the same. However, every streaming pixel of image1 is now compared twice with the streaming pixels of image2, hosted in the main SAD structure. Between the two comparisons there is a phase difference of 32 pixels. After the first comparison, the pixel of image1 as well as its disparity in the range from 0 to 31, enter a delay line of 32 pixels and pop-up again when the main structure is filled with the next 32 pixels of the streaming image2. Then a new comparison occurs, while the disparity tag values alternate to represent a range from 32 to 63. The two disparity values are compared with respect to their corresponding SAD values and the minimum SAD wins. Fig. 8 manifests the above principle, for a total range of disparities up to 64 pixels.

Since in the above technique we need two comparisons per pixel, the number of cycles needed to complete all comparisons is doubled. In this sense there is a cost in this procedure, however it can help us to implement large ranges in medium devices. An important aspect of this design is that it can output both disparities, processed with low and large range values, so that one may select the disparity range dynamically and process remote scene points with low range and close points with large range.

5. OVERALL SYSTEM DESIGN AND ARCHITECTURE

The co-processor is embedded in a medium-scale FPGA as part of a System-On-a-Programmable-Chip (SOPC). The system consists of peripheral controllers that interface the reconfigurable chip with external memory and communication devices, the stereo co-processor task-logic, Direct Memory Access (DMA) controllers and a control path implemented with a Nios II software processor [14]. Our target device is a medium-scale Cyclone II 2C35 FPGA chip, placed on a development board featuring external Double Data Rate (DDR2) Synchronous Dynamic Random Access Memory (SDRAM) and Static RAM (SRAM) memory as well as communication channels that can be used to transfer data from a computer or other device. In our test set-up we transfer image data from a desktop computer's file-system to the external DDR2 memory by means of Nios' power to manage and stream data. The overall system architecture is shown in Fig. 9.

The model-based design presented in the previous paragraphs is translated into VHDL code by means of the DSP-Builder conversion engine. In order to interface our task logic with the Nios data and control path we also built hardware fabric implementing the Avalon™ protocol [15]. The design is appropriately packaged into a library component which is readily recognized by the Quartus synthesis software and can be easily integrated within a Nios II system. Our component can be shared as open-source Intellectual Property (IP) and can be parametrized or customized according to specific needs.

Nios processor is programmed with instruction code in order to implement DMA transactions between the system peripheral memory and the Nios data path. It can also be programmed to access a host computer's file-system or communication channels. For this purpose we use the Nios II Integrated Development Environment which is used to develop instruction code for the Nios processor. The output disparity array can be transferred to the host computer as a grayscale image file or it can be projected on a monitor screen by means of an on board VGA controller.

6. RESULTS AND COMPARISON WITH OTHER SYSTEMS

6.1 Output disparity maps

In this section we present hardware output results in the form of dense disparity maps. We use the well known Tsukuba University series of images, one of which is shown in Fig. 10 along with ground truth. Images are translated into 1D arrays and transmitted to the FPGA board over the host/FPGA communication channel. Pixel data are multiplexed one by one so that we maintain the synchronization between the two images. This pre-processing stage is assigned to the embedded Nios II processor. Pixel demultiplexing is performed by the input stage of the stereo co-processor, as shown in Fig. 7. The resulting disparity map is sent back to the computer and is saved as a Bitmap (BMP) image file for evaluation.

We parameterize our module to process images with varying resolutions, with a disparity range 16 or 32 pixels and with a SAD window 3x3 and 5x5. Results are shown in Figs. 11, 12 and 13. Fig. 11 shows a 160x120 pixel disparity map produced with input images of the same resolution. Fig. 12 and 13 exhibit images with resolution 320x240 (8-bit grayscale) processed with windows 3x3 and 5x5 pixels respectively.

All results are characterized by relatively high noise appearing as black and white spots, especially at surfaces with low texture. This result is common with SAD or SSD algorithms, especially with small comparison windows. The wider the window the lower the noise, but at the same time object borders become blurred. A post-processing filter is often adopted in order to correct noise effects [11, 16]. In our experiments

we processed output disparity maps with a median 3x3 filter. In Figs. 12 and 13 the result of post-processing filtering is shown alongside the co-processor output. Although a 5x5 window reduces noise as compared to a 3x3 SAD window, the median filter produces comparable results on both disparity maps. This result is also emphasized by the quality assessment in the following paragraph.

6.2 Performance characteristics

The results presented above are comparable with those produced by a software algorithm working on an equal comparison window and disparity range, but the processing in hardware is more than 2000 times faster. Our comparison is with a Pentium IV at 3GHz running a G language LabVIEW application for SAD, with optimized array processing functions.

Table 1 reports the comparison between software and hardware implementation of our full-SAD algorithm. In order to evaluate the performance in each case we used 1) the RMS (root mean squared) error between the computed disparity maps and the ground truth map and 2) percentage of bad matching pixels, for which the computed disparity differs more than one disparity pixel than the ground truth disparity. Quality metrics are given by eqs. (8) and (9) in [4]. For this performance evaluation the ground-truth image shown in Fig. 10 (b) was used. Only image borders were excluded from the calculation. Both implementations are evaluated using 3x3 and 5x5 comparison windows. The effect of filtering outliers with a post-processing median filter is also measured.

Window size/ Filter	**RMS error in disparity units**		**Bad matches**	
	Software	Hardware	Software	Hardware
3x3/no	0,055	0,052	38%	37%
5x5/no	0,050	0,05	28%	27%
3x3/median	0,042	0,049	28%	26%

Table 1: Comparison between software and hardware implementation of our full-SAD algorithm

Table 2 reports the hardware requirements for the implementation of the stereo co-processor in a Cyclone II 2C35F672 chip. This FPGA has a resource capacity of 33000 Logic Elements and can provide 480000 bits of on-chip RAM memory. Approximately 7000 LE of the reported necessary total recourses are attributed to Nios processor and to embedded peripheral controllers. We note that when we double the disparity range by using the principles of section 4, the required resources do not change substantially (see third line of Table 2). We observe that implementing a 5x5 comparison window in a hardware structure hosting a 32-pixels disparity range requires almost 88% of the total resources and represents the limit of this particular chip. However, the same quality can be achieved by a 3x3 SAD window followed by a median filter, as shown in the evaluation of Table 1.

Image resolution	SAD window	Disparity range (pixels)	Logic Elements (LE)	Overhead needed for Nios +peripherals (LE)	Total Memory bits
80x60	3x3	16	3300	7000	176000
160x120	3x3	16	8000	7000	186000
160x120	3x3	16x2 (double comparison)	9600	7000	202700
320x240	3x3	32	15000	7000	196000
320x240	5x5	32	22000	7000	216832

Table 2: Logic Elements and Memory bits for different parameters of the co-processor

Migrating to more complex devices, like Stratix II, can give us an overhead to build a much more massive stereo co-processor, based on the same premises as above.

As already mentioned, one pixel disparity calculation needs four main clock cycles in our design. With a main on-board clock of 100MHz we process a frame with resolution 320x240 pixels in 3.1 ms. This is the equivalent of 325 frames per second. Equivalently, the system processes 25 Mpixels per second.

By applying the PDS metric (Points times Disparity per second) we measure a PDS value of $25x10^6x32=800x10^6$ disparities per second, which is above most reported hardware systems [17]. In a fully parallel system this metric is factored by the main clock frequency and the maximum sustained disparity range. The high disparity throughput can be of particular importance for a number of future applications, like fast part inspection in industrial automation, obstacle detection for autonomous navigation, deformation and vibration measurements, mapping of the environment from moving vehicles etc [18]. Fast 3D analysis is now possible using high-speed digital cameras able to transmit hundreds or thousands of frames per second [19].

However, true processing rate is also defined by other latencies, like Direct Memory Access (DMAs) and other data transfer procedures and depends on how the co-processor is interfaced with peripheral circuitry. It is also submitted to the limits of conventional NTSC or PAL video cameras and frame-grabbers. We measure that a practical stereo system as in Fig. 9, can render about 12-14 disparity frames per second on a video monitor, when it receives a double-frame video sequence from a host computer through high-speed USB2.0 channel. Details on how a Nios II system can interface with a USB2.0 controller can be found in [20].

6.3 Comparison with other FPGA implementations

In this paragraph we present the main idea behind other FPGA-based stereo implementations and compare them with our system.

Hariyama et al. [21] present a coarse to fine iterative SAD approach, using adaptive window size in order to reduce ambiguity at smaller detail. They exploit a more sophisticated algorithm than simple SAD, which however requires complex data scheduling and is demanding with respect to hardware resources. They result in a complicated and massive architecture that can only be accommodated with a bit-serial pipeline that reduces parallelism. They make heavy use of on-chip memory modules to load reference and search image regions and they can extend to practical full images only with a multichip design. Their FPGA

implementation utilizes 42570 Logic Elements of an Altera APEX20KE chip in order to process a 64x64 image.

Lee et al. [22] implement versions of SAD algorithms in VHDL and synthesize them to determine resource requirements and performance. Their main design principle is similar to our own presented in this article and their estimation of the required slices of a Virtex II Xilinx device are closely equivalent to the Logic Elements we need from our Altera Cyclone II chip. They do not make use of an embedded processor in their design and they report estimated pixel clock rate up to 10 MHz. They find that a rectangular window can save chip resources without degrading the quality of the resulting disparity map.

Darabiha et al. [17] describe the FPGA implementation of a stereo depth measurement algorithm based on Local Weighted Phase-Correlation. They use a custom board featuring four Xilinx Virtex2000E FPGAs, external memory and communication channels, as well as a FPGA controller chip. Their design approach is specific to their system, which is capable of processing disparity maps at video-rate.

Miyajima et al. [23] report on a compact stereo system based on a FPGA prototype board and external memory. Their system can support disparity maps at 20 frames per second and includes a camera calibration circuit and left-right consistency check.

Other FPGA implementations also exist (e.g. [24, 25]). They all show that a reconfigurable hardware platform is well suited for the design of a real-time stereo system for depth detection. Our system compares favorably with other implementations by virtue of its highly parallel, highly scalable design and its modest demand of chip resources. However its strongest virtue is its SOPC-ready concept of a parametrizable megafunction, which is able to interface with a commodity controller like Nios II. Our Avalon-interfaced co-processor can make the design of the overall stereo system very flexible.

Table 3 reports on the comparison between our implementation and other previous FPGA-based stereo systems.

Author	**Technology used**	**Function of implementation**	**Algorithm**	**Area utilization**	**Image size/ max disparity**	**Performance**
Present work	FPGA-based board +Nios II controller +SOPC-ready	Co-processor megafunction	SAD	22000 LE of a Cyclone II FPGA	320x240/ 64pixels	25 Mpixels/s or 800x10^6 PDS
S. Lee et al. [22]	VHDL Synthesis	Simulation	SAD	10000 Virtex II slices	320x240/ 64 pixels	10 Mpixels/s
M. Hariyama et al. [21]	FPGA APEX20KE	Co-processor	SAD	42570 LE	64x64/ -	21 Mpixels/s
Y. Miyajima et al. [23]	FPGA-based prototype board	Real-Time System	SAD	7100 slices of a XC2V6000 Xilinx FPGA	640x480/ 80 pixels	20 fps
A. Darahiba et al. [17]	Custom FPGA board	Real-Time System	Local Weighted Plase-Correlation	4 Virtex 2000E Xilinx FPGAs	256x360/ 20 pixels	60x10^6 PDS or 30 fps
J. Diaz et al. [25]	FPGA-based prototype board	Real-Time System	Phase-based stereo	9200 slices of a Virtex II FPGA	640x480/ 9 pixels	65 Mpixels/s or 585x10^6 PDS

Table 3: Comparison with other FPGA-based stereo implementations

7. CONCLUSIONS

We presented the design of a SOPC-ready hardware stereo co-processor, based on a full SAD algorithm implemented in a medium-scale FPGA. The design preserves window- and pixel-level parallelism and is fully scalable in the sense that it can grow to accommodate larger disparities by simply cascading the main blocks. The design is Avalon-interfaced and can be added as a VHDL megafunction in a Nios-controlled system-on-a-programmable-chip. It can be integrated with other VHDL blocks to form a fast vision-based depth-extracting application. By doubling the number of comparisons per pixel we can expand the disparity range while keeping resources low, which is desirable when a low-cost medium-scale FPGA chip is used.

The presented design can be enhanced by adding a hardware post-processing stage handling outlier points, like an appropriate median-filtering technique. Migrating the design to Stratix FPGAs can give adequate headroom to grow the design up to any desired window size or disparity range.

ACKNOWLEDGMENT

Author J. Kalomiros wishes to acknowledge financial support provided by the Research Committee of the Serres Institute of Education and Technology.

REFERENCES

[1] Davies, E.R.: "Machine Vision: Theory, Algorithms, Practicalities", (Elsevier, 2004, 3nd Edition).

[2] Klette, R., Schluns, K., and Koschan, A.: "Computer Vision – Three dimensional Data from Images" (Singapore: Springer 1998).

[3] Brown, M.Z., Burschka, D., and Hager, G.: "Advances in Computational Stereo", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25, (8), pp. 993-1008.

[4] Scharstein, D., and Szeliski, R.: "A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms", International Journal of Computer Vision, 2002, 47, (1-3), pp. 7-42.

[5] Hirschmuller, H.: "Improvements in Real-Time Correlation-Based Stereo Vision", Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01), Dec. 2001, p. 141-148.

[6] Forstmann, S., Kanou, Y., Ohya, J., Thuering, S., Schmitt, A.: "Real-Time Stereo by Using Dynamic Programming", Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'04), June 2004, Vol. 3, Washington D.C., USA, p. 29-36.

[7] Kanade, T., Yohsida, A., Oda, K., Kano, H., and Tanaka, M.: "A Stereo Machine for Video-rate Dense Depth Mapping and Its New Applications", Proceedings of the 15th Computer Vision and Pattern Recognition Conference (CVPR), June 18-20 1996, San Francisco, USA, pp. 196-202.

[8] Masrani, D. K., and MacLean, W. J.: "A Real-Time Large Disparity Range Stereo-System using FPGAs", Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS 2006), Jan. 4-7 2006, N.Y., USA, p. 13-20.

[9] MacLean, W.J.: "An Evaluation of the Suitability of FPGAs for Embedded Vision Systems", Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), June 2005, Vol. 3, San Diego, CA, USA, p. 131-137.

[10] Batlle, J., Marti, J., Ridao, P., and Amat, J.: "A new FPGA/DSP-Based Parallel Architecture for Real-Time Image Processing", Real-Time Imaging, 2002, 8, pp. 345-356.

[11] Muhlmann, K., Maier, D., Hesser, J., and Manner, R.: "Calculating Dense Disparity maps from Color Stereo Images, an Efficient Implementation", International Journal of Computer Vision, 2002, 47, (1-3), pp. 79-88.

[12] Sunyoto, H., van der Mark, W., Gavrila, D.: "A Comparative study of Fast Dense Stereo Vision Algorithms", Proceedings of the 2004 IEEE Intelligent Vehicles Symposium, June 14-17, 2004, Parma, Italy, p. 319-324.

[13] Altera Technical paper MNL-DSPBLDR-7.2: "DSP-Builder v. 7.2 Reference Manual", Altera Corporation, Oct. 2007.

[14] Altera Technical paper NII5V1-7.2: "Nios II Processor Reference Handbook", Altera Corporation, Oct. 2007.

[15] Altera Technical paper, MNL-AVABUSREF-3.1: "Avalon Interface Specification", 2005.

[16] Kotoulas, L., Gasteratos, A., Sirakoulis, G., Georgoulas, Ch., and Andreadis, I.: "Enhancement of Fast Aquired Disparity Maps Using a 1-D Cellular Automaton Filter", Proceedings of the Fifth IASTED International Conference on Visualization, Imaging and Image Processing, Sept. 7-9 2005, Benidorm, Spain, p. 355-359.

[17] Darabiha, A., Rose, J., and MacLean, W. J.: "Video-Rate Depth Measurement on Programmable Hardware", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03), June 18-20 2003, Vol. 1, Madison, Wisconsin, USA, p. 203-210.

[18] Woodfill, J.I., Gordon, G., and Buck, R.: "Tyzx DeepSea High Speed Stereo Vision System", Computer Vision and Pattern Recognition Workshop, 27-02 June 2004, Washington D.C., USA, p. 41.

[19] NAC Image Technology web page: http://nacinc.com, accessed January 2008.

[20] Kalomiros, J., and Lygouras, J., "Design and Evaluation of a Hardware/Software Architecture for Fast Image Processing", Microprocessors and Microsystems, 2008, doi: 10.1016/j.micpro.2007.09.001.

[21] Hariyama, M., Kobayashi, Y., Sasaki, H., and Kameyama, M.: "FPGA Implementation of a Stereo Matching Processor Based on Window-Parallel-and-Pixel-Parallel Architecture", IEICE Trans. Fundamentals, Dec. 2005, E88-A, (12), pp. 3516-3522.

[22] Lee, S., Yi, J., and Kim, J.: "Real-Time Stereo Vision on a Reconfigurable System", Proceedings of the 5th International Workshop on Systems, Architectures, Modeling, and Simulation, Samos, Greece, July 18-20, 2005 (T.D. Hamalainen et al. (Eds): LNCS Vol. 3553, Springer-Verlag, Berlin-Heidelberg, 2005), pp. 299-307.

[23] Miyajima Y., and Maruyama, T.: "A Real-Time Stereo Vision System with FPGA", Field Programmable Logic and Applications, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, Vol. 2778, 2003, pp. 448-457.

[24] Kang, B., Woo, K., Hong, Ch., Hong, D., Yang, H.: "Design of three-dimensional real-time system using stereo images", Current Applied Physics, 2004, 4, pp. 31-36.

[25] Diaz, J., Ros, E., Mota, S., Ortigoza, E. M., and del Pino, B.: “High Performance Stereo Computation Architecture”, Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL’05), 24-26 Aug. 2005, Tampere, Finland, pp. 463-468.

FIG. 1

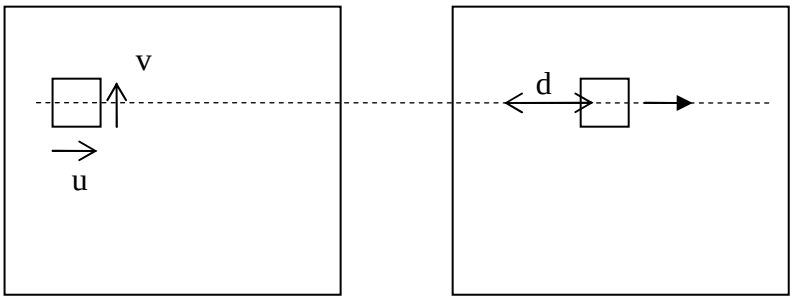


Fig. 1 A square window centered in one pixel of the left image is compared with all equal windows across the same epipolar scan-line on the right image. The blocks match when the metric of eq. (1) is minimized.

FIG. 2

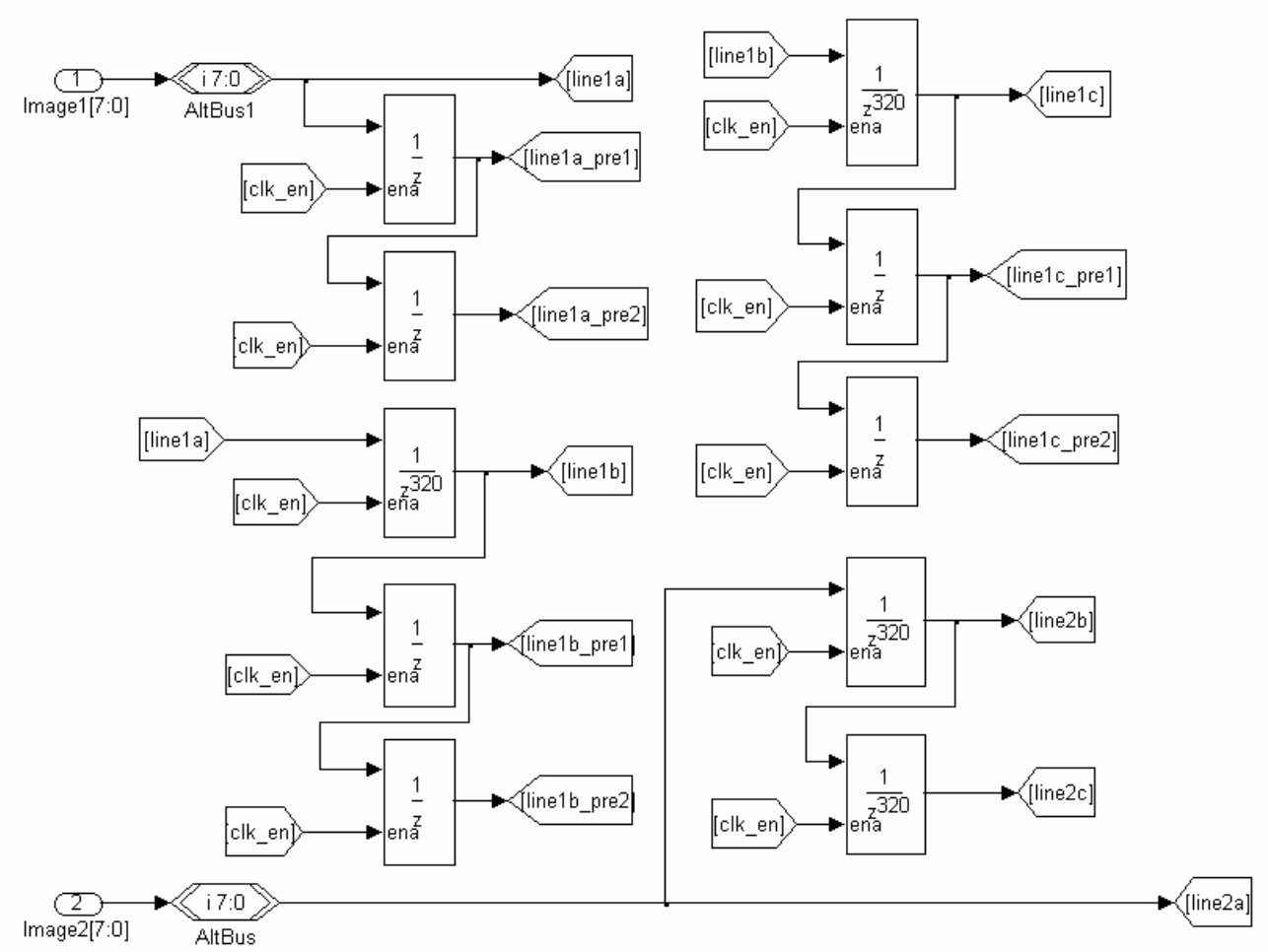


Fig. 2 Line-buffer for the production of a 3x3 comparison window

FIG. 3

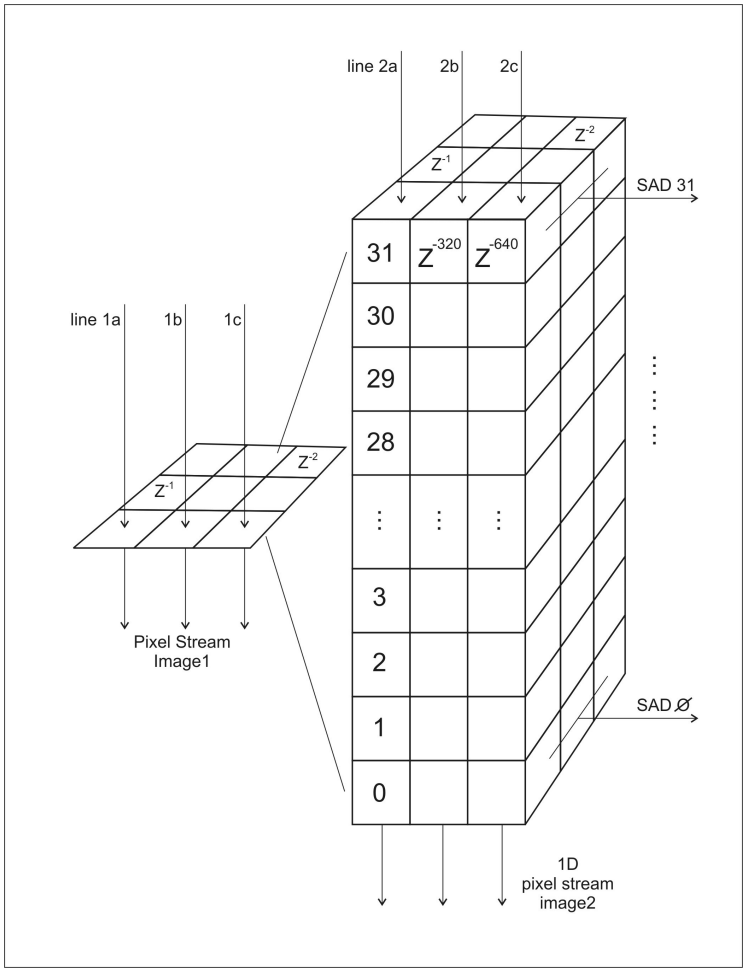


Fig. 3 The main parallel structure that computes SADs across a disparity range of 32 pixels in parallel.

FIG. 4

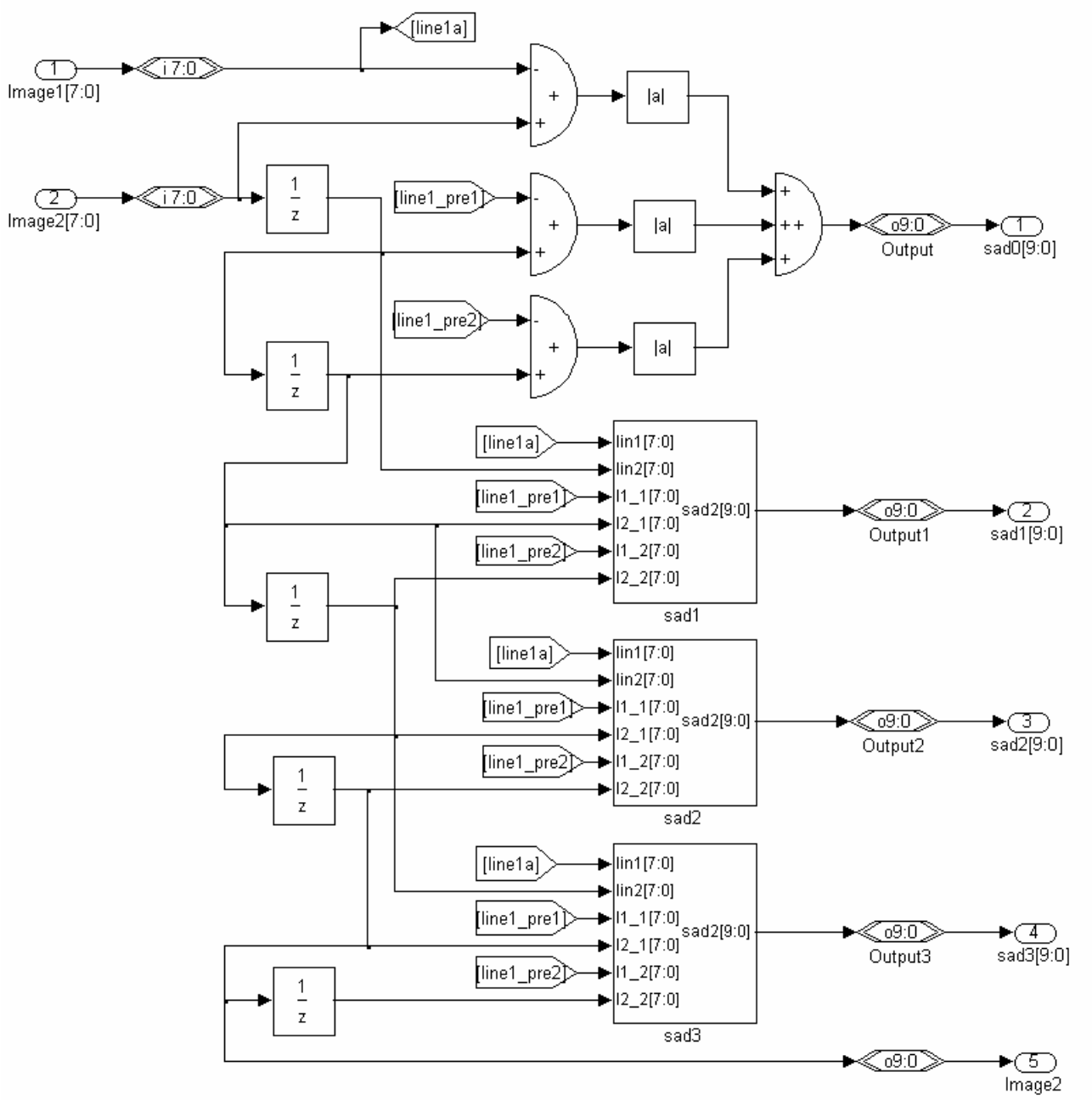


Fig. 4 A 4-pixel stage for the calculation of the sum of absolute differences. The first square is shown in gate-level detail. The stage can grow by cascading, as shown in Fig. 5, to form a 32-pixel disparity structure. Only one streaming line per image (line a) is shown.

FIG. 5

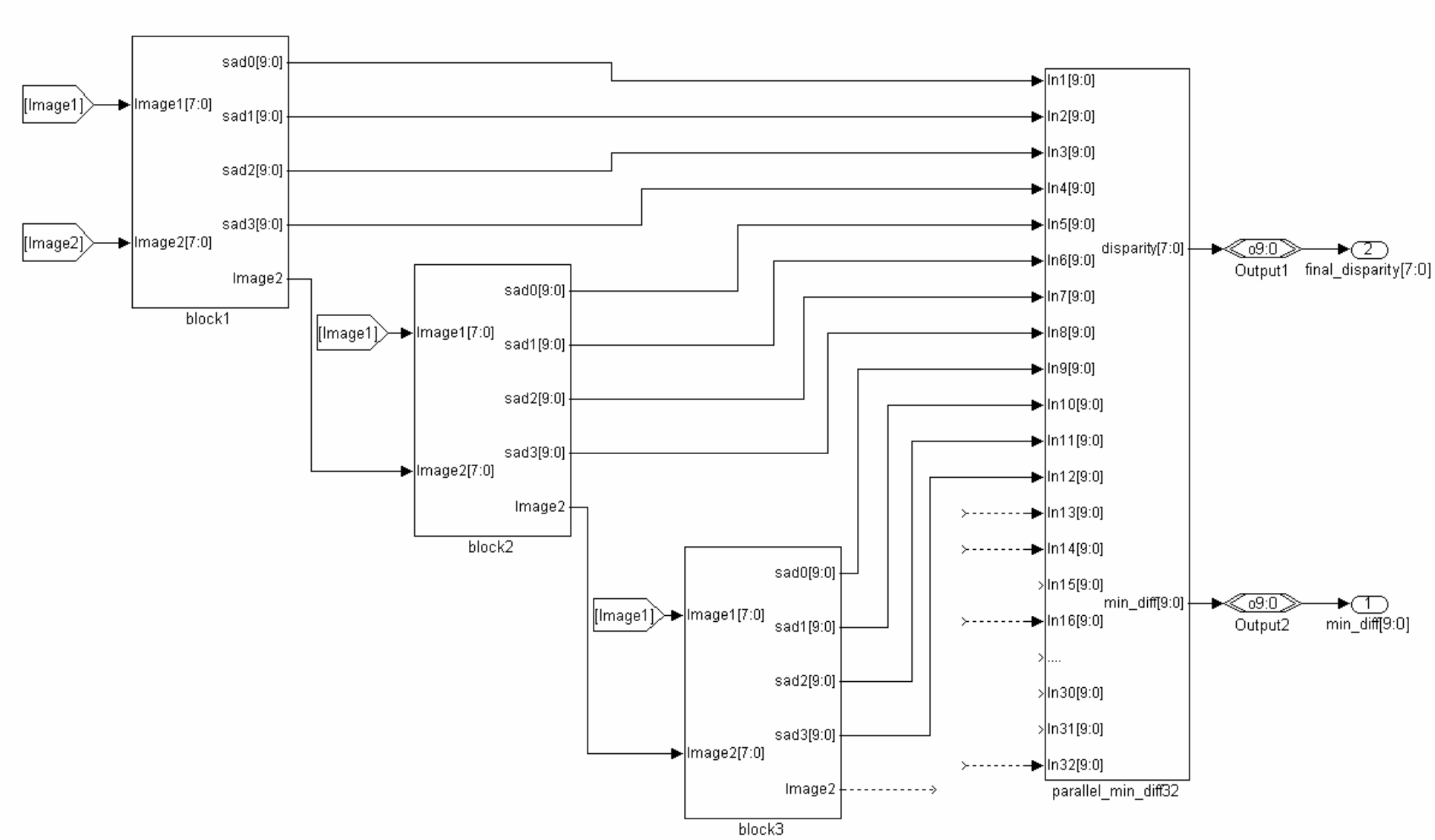


Fig. 5 Cascading 4-pixel blocks form the main parallel structure. Only three of eight blocks are shown. The last block on the right derives in parallel the minimum SAD.

FIG. 6

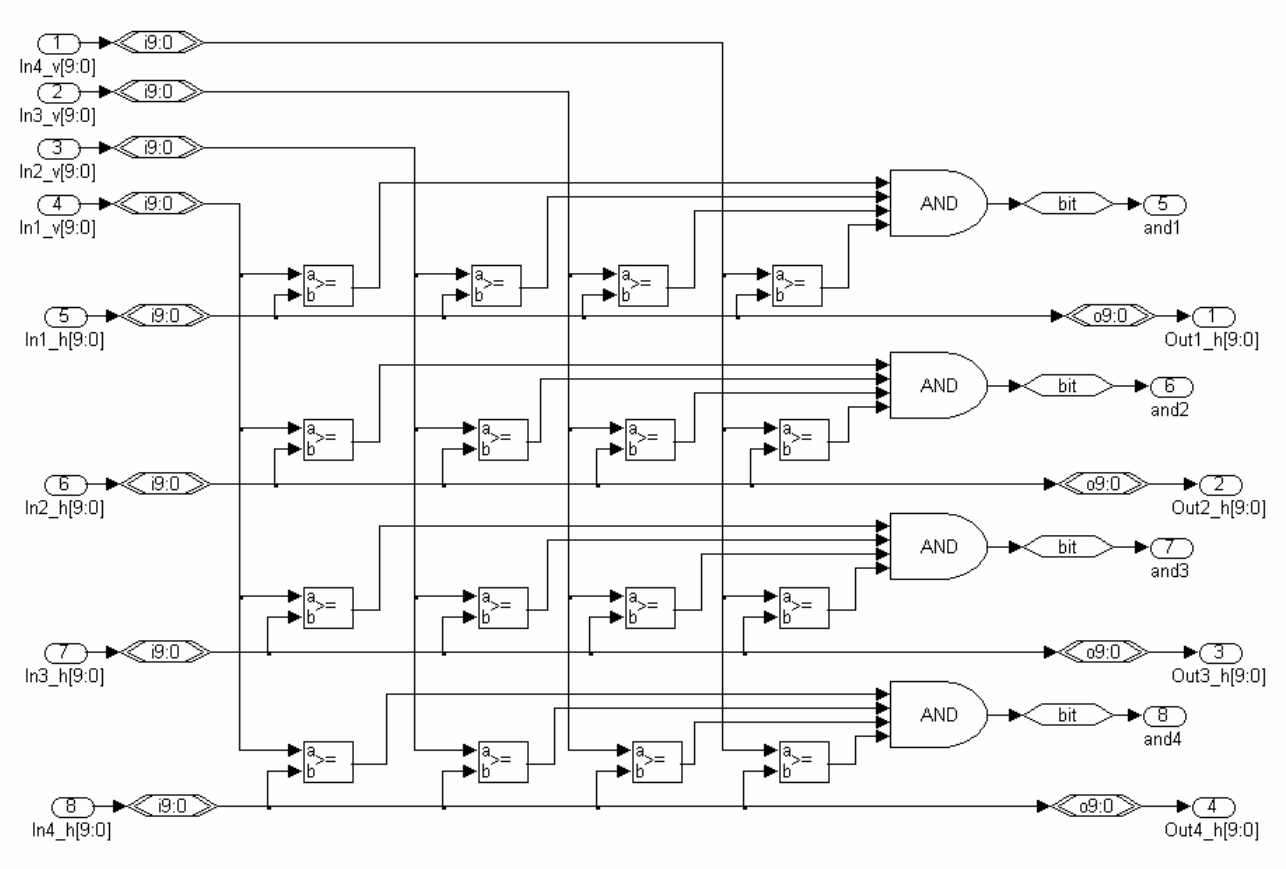


Fig. 6 Parallel derivation of the minimum among four values. The matrix can expand by connecting rows and columns.

FIG. 7

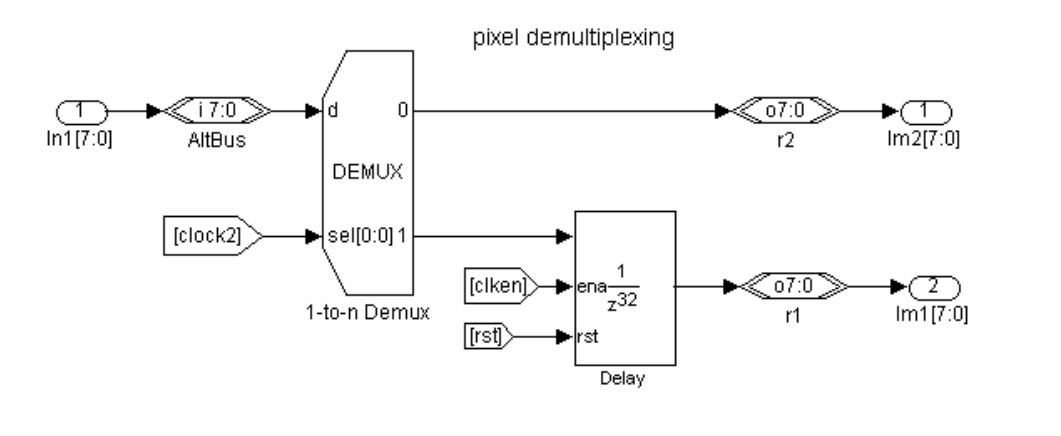


Fig. 7 Pixel de-multiplexing at the input stage. Image1 pixels are delayed in a 32-pixel-deep delay-line, while pixels of image2 are stored in the host structure of Fig. 3.

FIG. 8

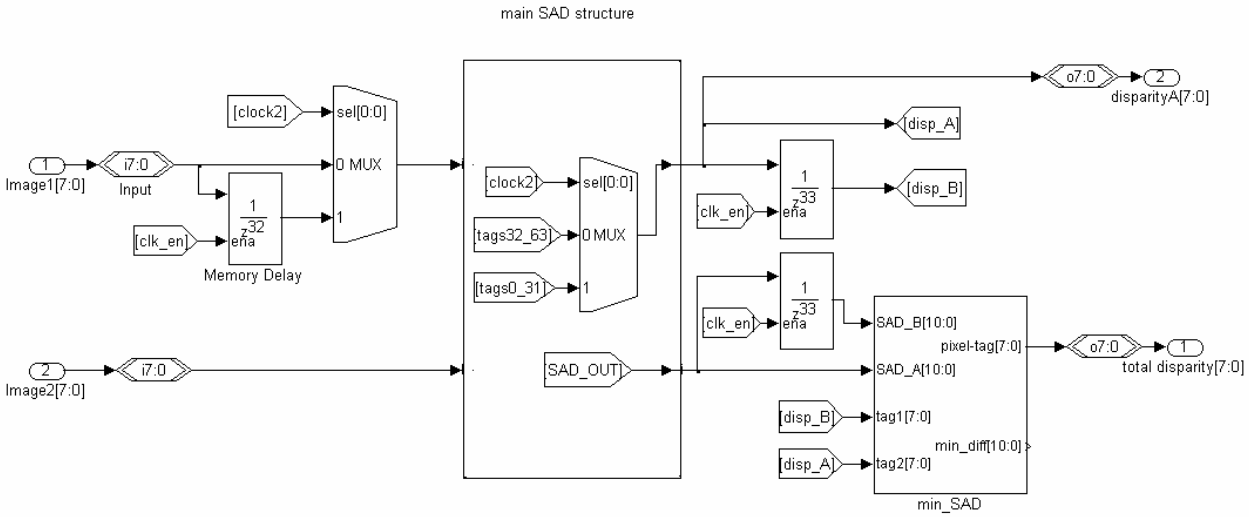


Fig. 8 Double comparison principle with the help of delay lines for expanding the disparity range

FIG. 9

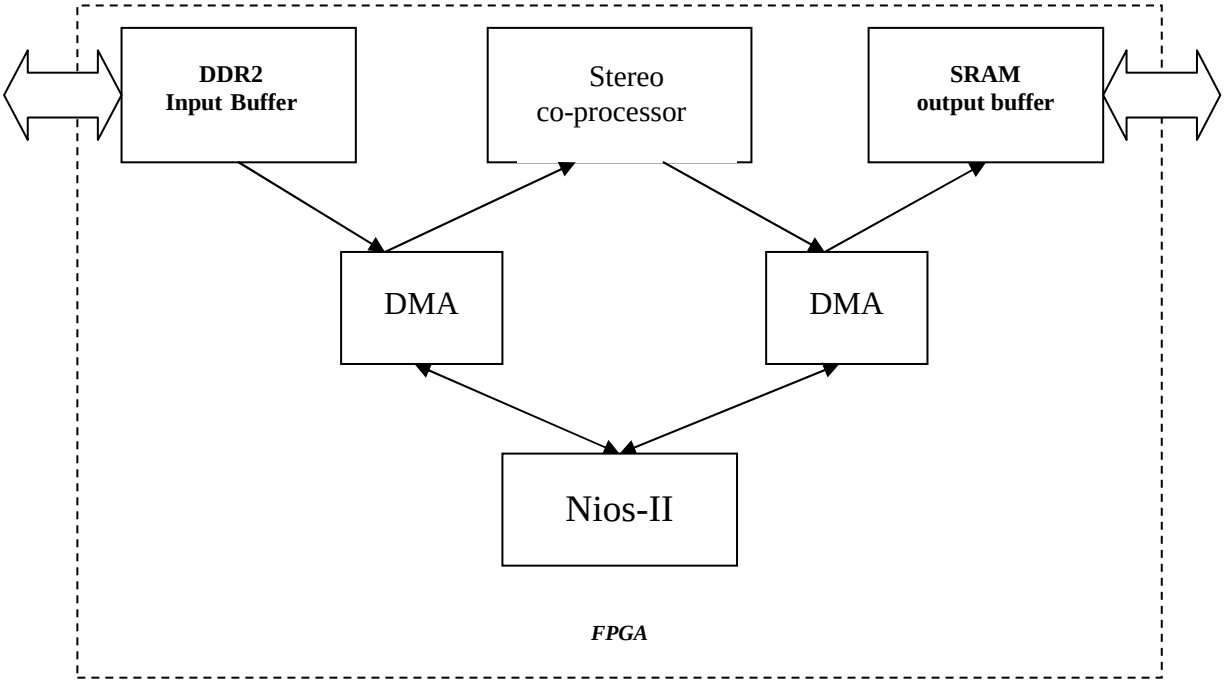


Fig. 9 Overall architecture of the FPGA system-on-a-programmable chip

FIG. 10

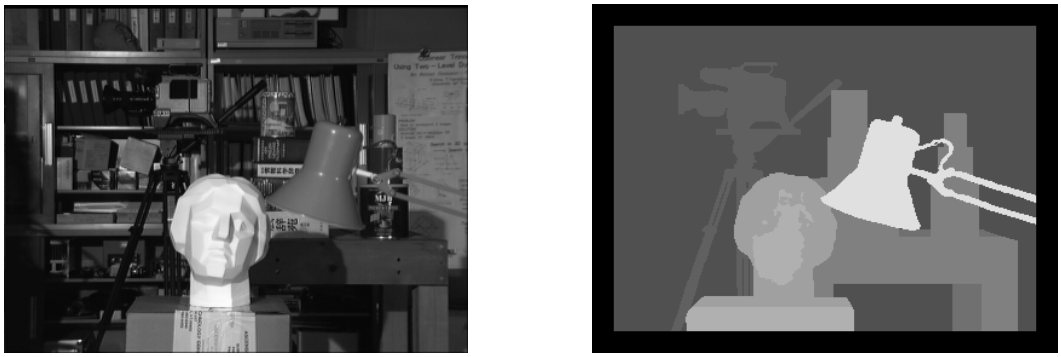
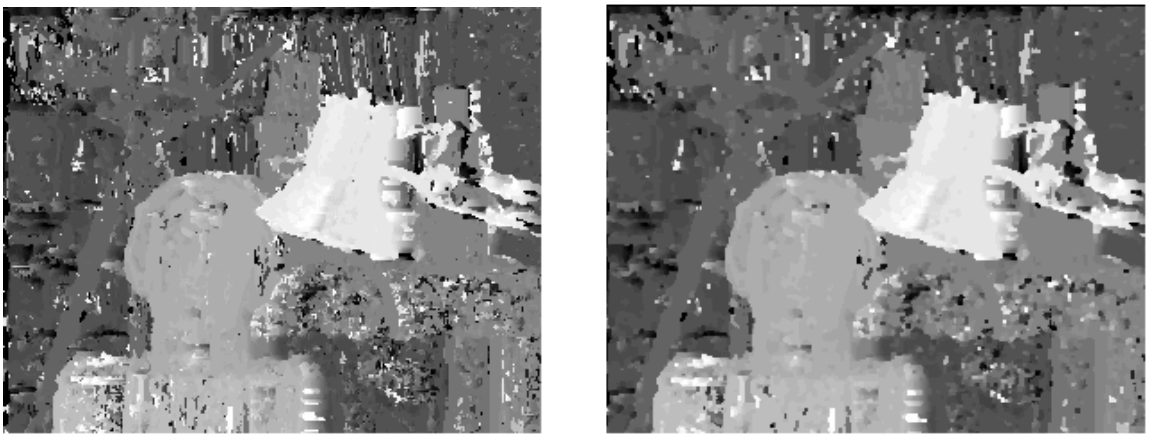


Fig. 10 Reference image from the so-called Tsukuba University series (left) and ground truth (right)

FIG. 11

Fig. 11 Hardware result on a 160x120 resolution image processed with disparity range 16 pixels and a 3x3 SAD window.

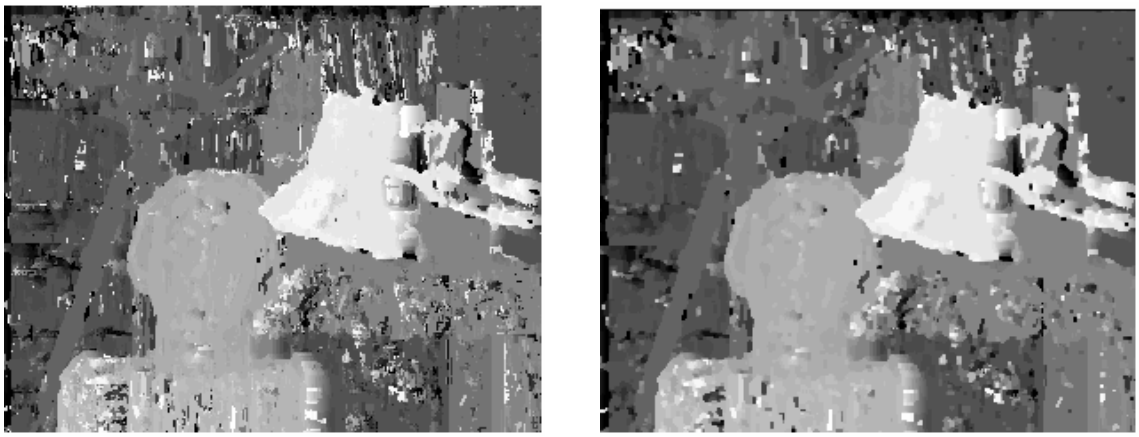
FIG. 12



a b

Fig. 12 (a) 320x240 pixels disparity map after processing with disparity range 32 pixels and a SAD window 3x3. (b) Median filtering of the disparity map, with a 3x3 mask.

FIG. 13



a b

Fig. 13 (a) 320x240 pixels disparity map, after processing with disparity range 32 pixels and a SAD window 5x5. (b) Median filtering of the disparity map, with a 3x3 mask.

Table 1: Comparison between software and hardware implementation of our full-SAD algorithm

Window size/ Filter	RMS error in disparity units		Bad matches	
	Software	Hardware	Software	Hardware
3x3/no	0,055	0,052	38%	37%
5x5/no	0,050	0,05	28%	27%
3x3/median	0,042	0,049	28%	26%

Table 2: Logic Elements and Memory bits for different parameters of the co-processor

Image resolution	SAD window	Disparity range (pixels)	Logic Elements	Overhead needed for Nios +peripherals (LE)	Total Memory bits
80x60	3x3	16	3300	7000	176000
160x120	3x3	16	8000	7000	186000
160x120	3x3	16x2 (double comparison)	9600	7000	202700
320x240	3x3	32	15000	7000	196000
320x240	5x5	32	22000	7000	216832

Table 3: Comparison with other FPGA-based stereo implementations

Author	Technology used	Function of implementation	Algorithm	Area utilization	Image size/ max disparity	Performance
Present work	FPGA-based board +Nios II controller +SOPC-ready	Co-processor megafunction	SAD	22000 LE of a Cyclone II FPGA	320x240/ 64pixels	25 Mpixels/s or 800x10^6 PDS
S. Lee et al. [22]	VHDL Synthesis	Simulation	SAD	10000 Virtex II slices	320x240/ 64 pixels	10Mpixels/s
M. Hariyama et al. [21]	FPGA APEX20KE	Co-processor	SAD	42570 LE	64x64/ -	21 Mpixels/s
Y. Miyajima et al. [23]	FPGA-based prototype board	Real-Time System	SAD	7100 slices of a XC2V6000 Xilinx FPGA	640x480/ 80 pixels	20 fps
A. Darahiba et al. [17]	Custom FPGA board	Real-Time System	Local Weighted Plase-Correlation	4 Virtex 2000E Xilinx FPGAs	256x360/ 20 pixels	60x10^6 PDS or 30 fps
J. Diaz et al. [25]	FPGA-based prototype board	Real-Time System	Phase-based stereo	9200 slices of a Virtex II FPGA	640x480/ 9 pixels	65 Mpixels/s or 585x10^6 PDS

FIGURE CAPTIONS

Fig. 1 A square window centered in one pixel of the left image is compared with all equal windows across the same epipolar scan-line on the right image. The blocks match when the metric of eq. (1) is minimized.

Fig. 2 Line-buffer for the production of a 3x3 comparison window

Fig. 3 The main parallel structure that computes SADs across a disparity range of 32 pixels in parallel.

Fig. 4 A 4-pixel stage for the calculation of the sum of absolute differences. The first square is shown in gate-level detail. The stage can grow by cascading, as shown in Fig. 5, to form a 32-pixel disparity structure. Only one streaming line per image (line a) is shown.

Fig. 5 Cascading 4-pixel blocks form the main parallel structure. Only three of eight blocks are shown. The last block on the right derives in parallel the minimum SAD.

Fig. 6 Parallel derivation of the minimum among four values. The matrix can expand by connecting rows and columns.

Fig. 7 Pixel de-multiplexing at the input stage. Image1 pixels are delayed in a 32-pixel-deep delay-line, while pixels of image2 are stored in the host structure of Fig. 3.

Fig. 8 Double comparison principle with the help of delay lines for expanding the disparity range

Fig. 9 Overall architecture of the FPGA system-on-a-programmable chip

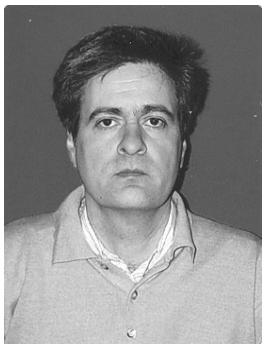
Fig. 10 Reference image from the so-called Tsukuba University series (left) and ground truth (right)

Fig. 11 Hardware result on a 160x120 resolution image processed with disparity range 16 pixels and a 3x3 SAD window.

Fig. 12 (a) 320x240 pixels disparity map after processing with disparity range 32 pixels and a SAD window 3x3. (b) Median filtering of the disparity map, with a 3x3 mask.

Fig. 13 (a) 320x240 pixels disparity map, after processing with disparity range 32 pixels and a SAD window 5x5. (b) Median filtering of the disparity map, with a 3x3 mask.

VITAE



John A. Kalomiros received the degree of Physics at the Aristotle University of Thessaloniki in 1984 and a Masters degree in Electronics in 1987. His doctoral thesis is on characterization of materials for micro-electronic devices. His recent research interests include microcontrollers and digital systems design with applications in Robotics. He is also working on systems for digital measurements and instrumentation. He teaches electronics and related subjects at the Serres Institute of Education and Technology, in Greece.

John N. Lygouras was born in Kozani, Greece in May 1955. He received the Diploma degree and the Ph.D. in Electrical Engineering from the Democritus University of Thrace, Greece in 1982 and 1990, respectively, both with honors. From 1982 he was a research assistant and since 2000 he is an Associate Professor at the Democritus University of Thrace, Department of Electrical and Computer Engineering. In 1997 he spent six months at the University of Liverpool, Department of Electrical Engineering and Electronics as a Honorary Senior Research Fellow. His research interests are in the field of robotic manipulators trajectory planning and execution. His interests also include the research on analog and digital electronic systems implementation and position control of underwater remotely operated vehicles.