Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 962057, 8 pages
doi:10.1155/2010/962057

Research Article

Concurrent Calculations on Reconfigurable Logic Devices Applied to the Analysis of Video Images

Sergio R. Geninatti,[1] José Ignacio Benavides Benítez,[2] Manuel Hernández Calviño,[3] and Nicolás Guil Mata[4]

[1] *Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad de Rosario, Avenue Pellegrini 250, 2000 Rosario, Argentina*
[2] *Escuela Politécnica Superior, Universidad de Córdoba, Menéndez Pidal s/n, 14001 Córdoba, Spain*
[3] *Departamento de Física General, Facultad de Física, Universidad de La Habana, Colina Universitaria El Vedado, 10300 La Habana, Cuba*
[4] *Departamento de Arquitectura de Computadoras, Universidad de Málaga, C. Teatinos, 29071 Málaga, Spain*

Correspondence should be addressed to Sergio R. Geninatti, foco@fceia.unr.edu.ar

Received 25 May 2009; Accepted 11 November 2009

Academic Editor: Elías Todorovich

This paper presents the design and implementation on FPGA devices of an algorithm for computing similarities between neighboring frames in a video sequence using luminance information. By taking advantage of the well-known flexibility of Reconfigurable Logic Devices, we have designed a hardware implementation of the algorithm used in video segmentation and indexing. The experimental results show the tradeoff between concurrent sequential resources and the functional blocks needed to achieve maximum operational speed while achieving minimum silicon area usage. To evaluate system efficiency, we compare the performance of the hardware solution to that of calculations done via software using general-purpose processors with and without an SIMD instruction set.

1. Introduction

The capacity of Reconfigurable Logic for massive concurrent computation makes it wellsuited to implementing complex algorithms. This feature has increased the potential of digital design, as shown by several papers which have proposed implementing mathematical algorithms implemented on FPGA. In almost all cases the topology and the performance obtained depend on the specific application, such that each case must be analyzed individually. References [1–3] show the design of custom image processing systems on FPGA.

This paper presents the advantages of concurrency and parallelism in implementing video temporal segmentation with Reconfigurable Logic Devices, emphasizing the advantages and limitations of FPGA technology and its development tools. We chose to implement a function that has been thoroughly studied in [4] and that measures the similarity between two frames based on their luminance distributions. After breaking the algorithm down into its component parts, we propose specific solutions for each part, while making it clear that obtaining the optimum hardware solution is an iterative process which benefits from all the features of FPGA devices [5].

In Section 2, we briefly present the design philosophy of dedicated hardware, where both Configurable Logic Blocks, (CLBs in XILINX devices), and special-function blocks have been used. In addition, we describe an interface with external memory, an essential feature when dealing with huge volumes of data.

Section 3 offers an introduction to video temporal segmentation through the artificial synthesis of visual human perception and presents the similarity-calculation algorithm on which this paper is based.

Section 4 describes in detail the implementation and optimization of the proposed algorithm, emphasizing the design and evaluation criteria.

Finally, the experimental results section shows the quantitative results obtained by comparing computing time using the proposed hardware solution to that of pure software running on a PC.

2. Design Criteria

The combination of CLB logic with embedded Blocks and data supply are critical aspects of the design to be taken into account.

2.1. CLB Logic versus Embedded Blocks. The main drawback of using a reconfigurable hardware device is the loss of performance imposed by the reconfiguration circuit itself [6]. For this reason, device manufacturers often include embedded Blocks able to perform frequent and specialized tasks, including RAM blocks, multipliers, timers, and communication devices, whose performance is far superior to that of equivalent functions implemented using CLB logic.

Another important issue is the time delay introduced by routing. This factor justifies the inclusion on the FPGA of different-quality routes for different purposes (clocks, near, long, etc.).

The present study will show the many possible alternatives for combining all these elements in a specific design. Those selected in each case will depend on the specific implementation being conducted.

2.2. Data Supply. Video processing is a task characterized by a very high demand for data, and when using an FPGA there is only a limited storage capacity available on chip. Thus, an efficient interface is required with external memory capable of feeding and receiving data at the necessary rate. In this case, we used a Xilinx Spartan-3 development board provided with a static external memory bank having 32-bit width and 10-ns access time. This bus width allows us to read and process four bytes at a time. The reading and modification of data storage into the Spartan BlockRAM requires two clock cycles per operation. This memory is synchronous and can operate up to a 200 MHz clock frequency, perfectly matching the 10-ns access time of the external memory.

3. Application: Temporal Video Segmentation

Our aim is the development of a specific architecture for video segmentation. This video analysis technique enables the detection of low-level semantic units in the video, which are called shots. Many applications are based on shot information to carry out higher level analysis, such as video classification, summarization, visual index generation, or sequence comparison [7]. The technique is based on giving a quantitative value to the human perception of similarity between frames. In this theory, it is assumed that the properties of a certain stimulus—in this case an image—can be represented as a vector in a space of characteristics and, as a consequence, the similarity between two images is reduced to the appropriate measurement of a distance using a metric on a psychological space [4].

One of the main features of an image is its luminance histogram, defined as the frequency of occurrence of the pixels' luminance values in each frame. The similarity between two frames is inversely related to the distance between vectors representing its characteristics. In the case of a histogram of luminance, this can be defined through the following normalized equation [4]:

$$f_s(i-1,i) = \frac{\sum_b H_{i-1}[b] \cdot W_i[b]}{\sqrt{\sum_b H_{i-1}[b] \cdot W_{i-1}[b]} \cdot \sqrt{\sum_b H_i[b] \cdot W_i[b]}}, \tag{1}$$

where $H_i[b]$ is the histogram of the bin "b" from frame "i", $\mathrm{W_i}[b]$ is the windowed histogram at level "b" from frame "i", Notice that "$i-1$" and "i" subindexes refer to consecutive frames.

4. Algorithm Segmentation

In order to implement and optimize expression (1), we divide the procedure into the following five sequential stages:

(1) calculation of the histogram,

(2) 1D windowing using a window size of three to calculate $W_i[b]$,

(3) sum of products,

(4) square root,

(5) division.

Multipliers needed for stage 3 were not implemented in VHDL because the Spartan 3 device has dedicated 18-bit multiplier blocks providing much better performance than that attainable using CLB logic.

4.1. Calculating the Histogram. The real bottleneck occurs at the stage responsible for obtaining the histogram, and its interface with external memory is the limiting factor for the overall circuit performance. Therefore, it is advisable to try to reduce the amount of data to be processed. As shown in [8], the use of DC coefficients of compressed frames instead of frame pixels values has no influence on the reliability of the video segmentation technique. Thus, we have reduced the number of data to be read from each video frame by using these DC coefficients. This allows us to achieve a reduction ratio of 64 to 1.

Considering the results obtained in [9], which proved that a circuit for obtaining the histogram using BlockRAMs as accumulators performs better than those using CLB logic, we propose the hardware shown in Figure 1 for the first stage. Note that the dual port feature of the Spartan-3 BlockRAM allows us to extract the histogram's individual values in pairs.

Port B input has been forced to "0" to clear the accumulators in the same clock cycle, leaving the hardware ready for processing the following frame. Several registers were inserted between output adders and at the address input of the BlockRAMs to pipeline the stage, so as to optimize the interface with external memory and double the internal clock frequency.

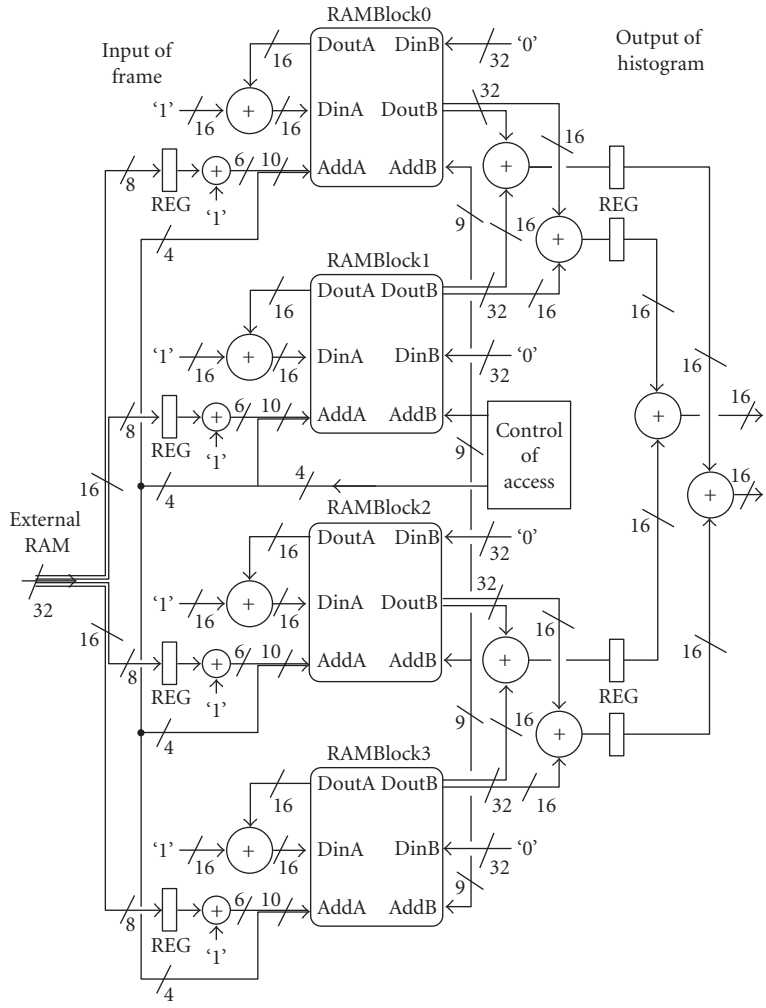

Figure 1: Circuit for calculating the histogram.

4.2. Calculating the Windowed Terms $W_i[b]$. To reduce the influence of slight variations in image luminance, each term of the histogram has been replaced with the sum of its adjacent neighbors. Figure 2 shows how the E_i windowed values are generated, as well as the convenience of using concurrent calculations due to the fact that the majority of H_i values are used in more than one E_i calculation. Excluding the two boundary values, each H_i value contributes to E_{i-1}, E_i, and E_{i+1}.

Figure 3 shows how the algorithm symmetry allows the same sum term to be used twice in calculating the windowed value. This stage was designed to calculate six windowed values concurrently, starting from eight histogram values at the input. The concurrent supply of these eight values is achieved by using two-storage BlockRAMs. As shown below, these two memories are also used for obtaining the sum terms needed for the correlation. These two blocks have four 32-bit-width output ports each, and thus are capable of supplying the eight necessary histogram values simultaneously.

Figure 4 shows the first version of this stage. The folded buses that appear at Port B inputs of both blocks are used to repeat some histogram values in particular positions of the memory bank, thus solving the boundary problem mentioned in the above calculation. The output summing stages were also pipelined to keep the maximum clock frequency of the whole stage close to 200 MHz, while the three output buses enable the concurrent extraction of six windowed values in each clock cycle.

4.3. Coupling Considerations. To prevent the delay introduced by the next calculation from piling up, BlockRAMs 4 and 5 retain data belonging to a frame while the following one is being processed in a different memory page. The BlockRAM dual port feature allows for decoupling and behaves like another pipeline stage. The only difference in this case is that the retention unit is the whole frame and not a single clock cycle.

Bearing in mind that 64 is the number of established histogram levels, each one represented by a 16-bit integer, the proposed structure can process frames containing up to 65,536 pixels. Taking as a reference a frame of only 1600 pixels, Table 1 shows the calculation time expressed as the number of elapsed clock cycles.

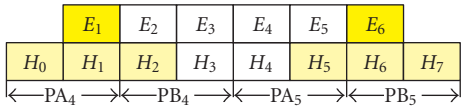

Figure 2: Concurrent windowing definition.

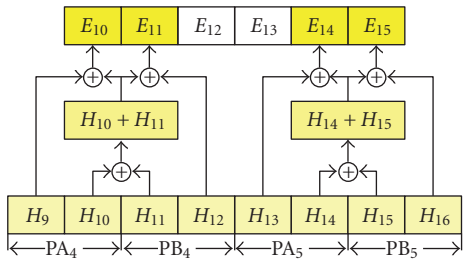

Figure 3: Grouping the windowed sum terms.

Table 1: Number of clock cycles per operation.

Histogram calculation	800
Intermediate storage	32
Windowing	11

At the same time that storing and windowing are being performed on the previous histogram, the input stage builds the following histogram. After the first frame is complete, the remaining calculations are done in parallel within the 800 clock cycles taken by the first stage.

4.4. Calculating the Correlation and the Sum. To implement this stage, we have modified the circuit depicted in Figure 4 to calculate two correlation sum terms simultaneously. We have placed six multipliers next to the windowing stage followed by a matrix of a three-level adder, responsible for obtaining the correlation. The whole block has been pipelined using six registered stages, keeping the clock frequency close to 140 MHz.

Two of the partial sums needed to calculate the similarity coefficient as defined by (1) are stored in the output. Partial terms of the denominator will be reused in the following frame calculation; hence, they are stored to avoid repeating the calculation (see Figure 5). Note that both summing terms use the windowed term of the same frame $W_i[b]$, but while one is multiplied by the current histogram $H_i[b]$, the other is multiplied by the previous one $H_{i-1}[b]$. The term $W_i[b]$ is retained during two clock cycles in the registers (shadowed in Figure 5) placed at the input of the multipliers, ensuring that in each clock cycle the correct histogram corresponds to its associated sum term.

Starting from the histograms stored in BlockRAM belonging to two frames, the stage only takes 28 clock cycles to complete two windowed sums. The key to such high performance lies in the concurrency of operations, the organization of the data, and the pipeline structure. It is also easy to interface this block with its neighbors because the input and output buses are only 32- and 64-bits wide, respectively. In contrast, the internal bus-width parallelism reaches 192 bits in the multiplier layer.

4.5. Square Root Calculation. The purpose of this stage is to obtain the square root of the denominator in (1). This calculation is performed by using shift and sum operations according to the algorithm described in [10]. The circuit depicted in Figure 6 shows the hardware implementation of this operation, starting from a slight modification at the output of the circuit shown in Figure 5.

It takes 16 clock cycles to complete processing a 32-bit-long input data. The sum term $S[h(n).w(n)]$ is available at the previous correlation block one clock cycle prior to completing the sum term $S[h(n-1)w(n)]$, a fact that can be useful in anticipating the beginning of the square root calculation. It should also be emphasized that some terms in the numerator of (1) belonging to histogram "n" will be used in the same calculation of the next frame. For this reason, a register has been provided to retain it.

4.6. Product and Division. Following the sequential stage responsible for calculating the square root, a multiplying block provides the product of the denominator in (1). In this way, numerator and denominator are ready and available at the output of the circuit shown in Figure 6 to make the final division and obtain the normalized similarity coefficient between two frames.

The division between two unsigned integers employing a shift and subtract algorithm is done by a circuit similar to the one shown in Figure 6, as described in [10]. The full process takes 32 clock cycles for the 32-bit solution required.

4.7. Control and Timing. The stages referred to above require a set of control and synchronization signals to properly manage data flow. Therefore, a control circuit is required to generate the appropriate timing signals. Figure 7 shows the diagram of the Finite State Machine responsible for controlling the whole system. This was implemented in VHDL description language and placed and mapped with the rest of the module at the top level of the hierarchy tree.

The design of this finite state machine is crucial, because the generated signals have to drive all the calculating modules, such that the fan-out and route length of these lines determine the maximum system clock frequency.

Because of the intensive use of pipeline architecture, the system exhibits a finite latency time from which a new value is output each 800 clock cycles (5.7 μs at 110 MHz clock frequency). Data flow has been organized in such a way that the first stage calculates the histograms on alternate RAM pages, taking 800 clock cycles to process each 1600-pixel frame. The rest of the calculation, which takes 109 clock cycles to complete, starts each time the processing of a frame is over. Except for the first stage (see Figure 1), the rest of the stages remain idle 86% of the time.

Clearly, the bottleneck is the width of the bus that connects the external memory with the above-described system. Widening this bus from 32 to 128 bits would make it possible to process 16 pixels at a time, reducing the time needed to

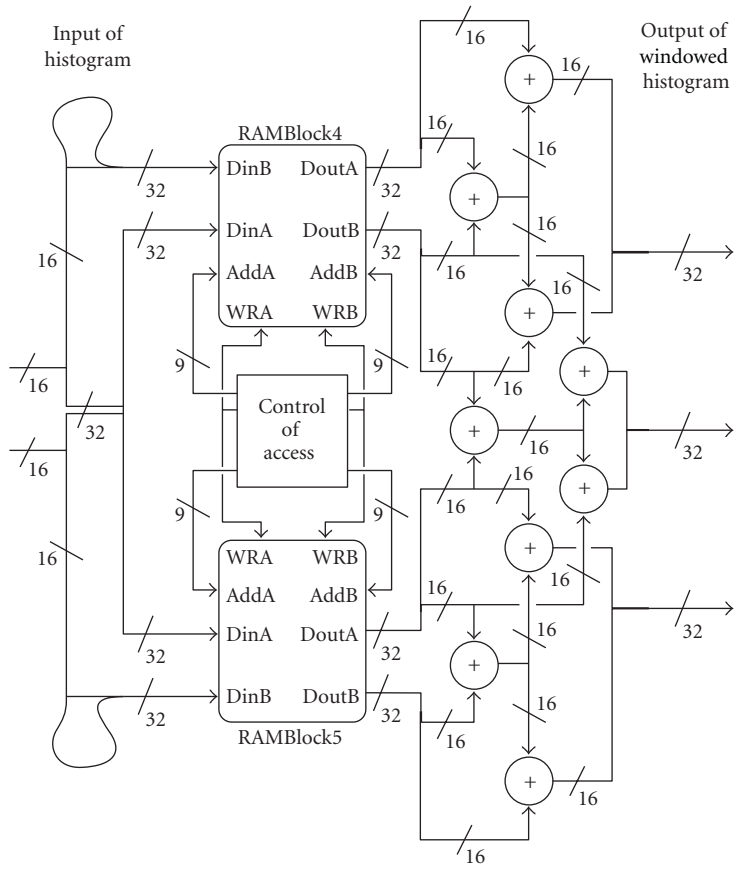

Figure 4: Circuit for calculating the windowed terms.

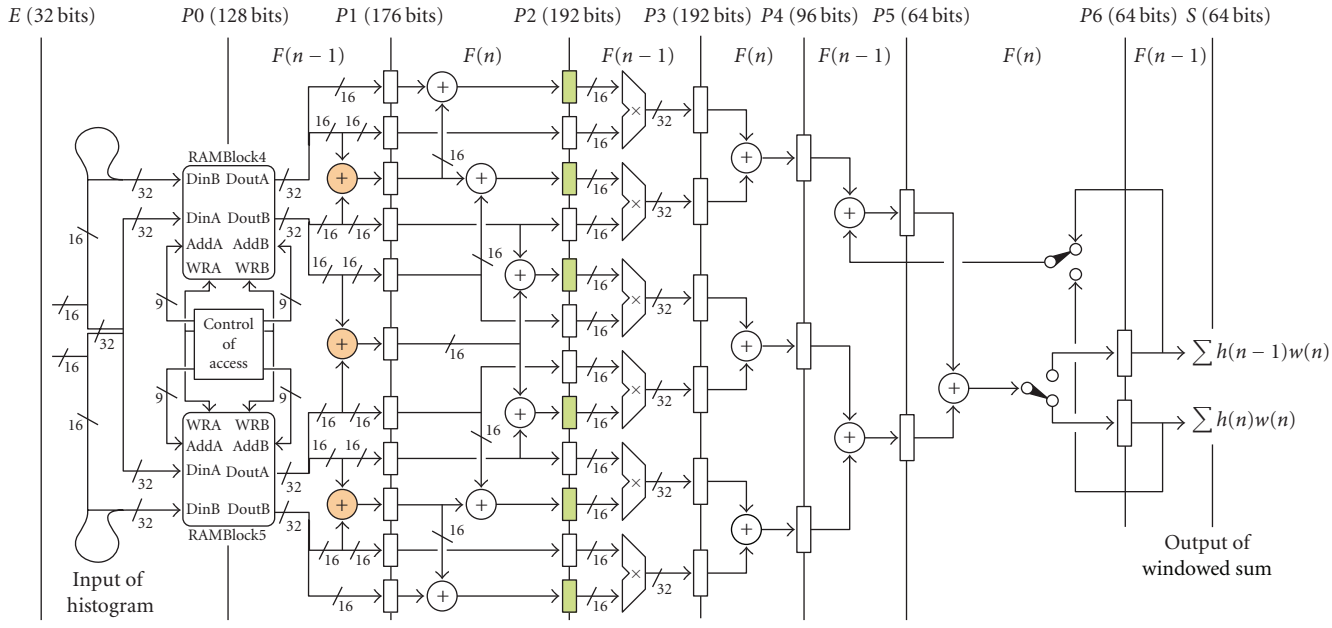

Figure 5: Circuit for calculating the windowed correlation (includes windowed calculation pipelined).

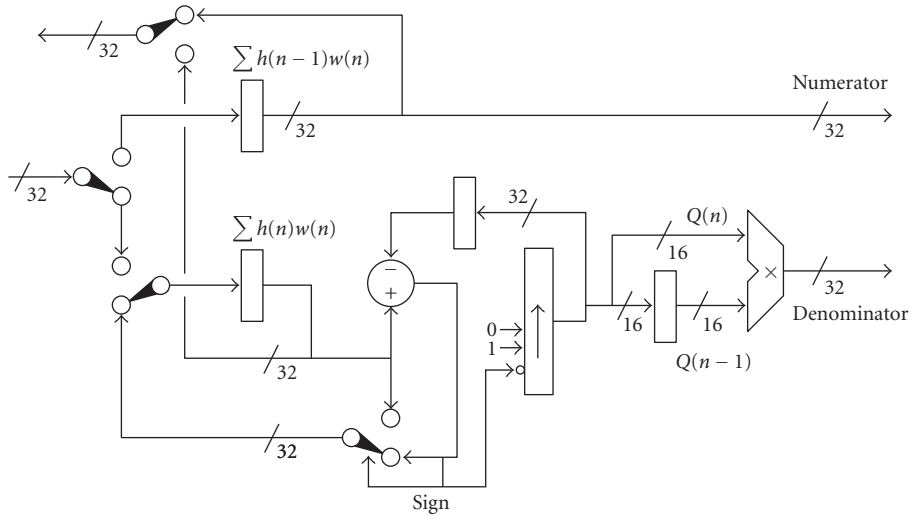

Figure 6: Circuit for obtaining the square root and terms of division.

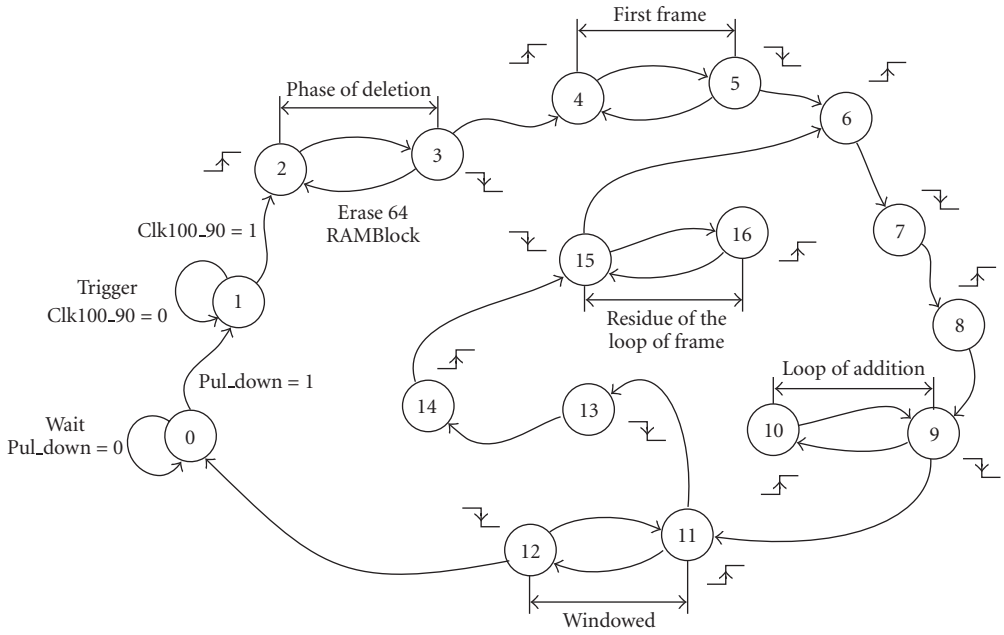

Figure 7: State diagram of the control machine.

complete the calculation to 200 clock cycles. Even with this addition, the usage factor of the remaining stages is never higher than 55% of the total time. However, the inactivity of these stages has no influence on the total processing time as they work in parallel with the histogram stage.

5. Experimental Results

The entire design was simulated and implemented using the software package ISE 8.2i from Xilinx and tested on a development board "SPARTAN-3 Starter Board" from Digilent provided with an XC3S1000 FT256 speed grade 4 chip, 1 Mbyte of SRAM (256 Kb × 32), and 10 ns access time.

The SOFT-HARD comparison uses four compressed videos from an MPEG-7 content set belonging to different genres (drama, sport, and news), all of which have an average length of 25,000 frames. The algorithm was implemented using fixed point arithmetic. Overflow has been avoided by selecting suitable resource sizes for each stage. The main considerations are discussed in the following.

The bus width at the multiplication output is twice as wide as the incoming factors and only half as wide for the square root case. The size of the accumulators for the sums of products enables storing the quantity $N \times \mathrm{MAX}$, where MAX is the maximum value attained by these factors and N is the number of terms to be added.

Table 2: Summary of resources usage.

Selected device: 3s1000ft256-4			
Resource	Used	Available	Ratio
Slices	952	7680	12%
Slice Flip Flops	1170	15360	7%
4 input LUTs	1531	15360	9%
Bonded IOBs	80	173	46%
BRAMs	7	24	29%
MULT18X18s	7	24	29%
GCLKs	3	8	37%
DCMs	1	4	25%

Source: ISE 8.2i provided by XILINX.

Table 3: Summary of number of clock cycles taken per operation.

Histogram calculation	800
Intermediate storage	32
Windowed correlation (2 sums)	28
Square root	16
Product	1
Division	32

The Xilinx ISE synthesis tool provided the report shown in Table 2 concerning the resources occupied and available at the chip level. In regard to these data, it should be made clear that in applications like the one described in this paper—where there is no direct link between the number of BlockRAMs and multipliers used—the report is inaccurate, since the use of a multiplier block is assumed to disable the use of its associated BlockRAM and vice versa, due to the peculiarities of the Xilinx FPGA. Thus, the usage factor must be calculated considering them as available pairs, in this case 14 multiplier-BlockRAM pairs out of 24 available (58% busy), which is almost twice the value (29%) reported by the Xilinx synthesis tool.

The interface with the external memory was debugged using an Agilent 64622D oscilloscope.

Close examination of the summary of the number of clock cycles taken by each individual stage shown in Table 3 suggests that it might be desirable to reduce the parallelism of nearly idle stages, which are very resource hungry. One example is the circuit depicted in Figure 5, which employs such scarce and valuable resources such as multipliers.

To evaluate circuit performance, we compared a software implementation presented in Table 4 (on a PC Pentium IV at 2 Ghz) for the calculation of windowed values and sum of products to the hardware implementation on FPGA (Spartan 3 at 120 MHz) developed in this work. This comparison is very illuminating because the FPGA is a reconfigurable device that is impaired by the overhead of configuration circuits that slow down clock rate and increase the silicon area compared to a custom VLSI device. Despite this handicap, the calculation speed obtained with the FPGA exceeds that obtained using SIMD instructions (SSE multimedia extension) by a factor of 1.75 : 1 as regards nominal delay, and by a factor of 25 : 1 as regards clock cycles. We attribute this improvement to the optimization of data flow and to concurrence.

Table 4: Hard-Soft solution comparison.

Implementation	ms	cycles
PC (sequential programming)	2500	5000
PC (sequential optimized)	750	1500
PC (SSE with Intrincsics)	350	700
FPGA (SPARTAN 3-VHDL)	200	28

6. Conclusions

In this paper we have presented an FPGA implementation to calculate similarities between two frames. Our design works with the DC coefficients of compressed video frames to compute the frame histogram. Similarity is calculated by applying a windowed cross-correlation to frame histograms. Our implementation has efficiently solved the problems arising from the management of constant data flow from video frames. Thus, we have designed a windowed sum of products stage which completes the calculation of 64 sum terms in only 28 clock cycles.

Although our system is a hardware implementation of a video segmentation technique, extending these results to other data processing applications is possible, simply because any calculations involving data correlation always takes the form of a sum of products. Its success in such applications will depend strongly on both the ability to maintain constant data flow through the system and how the numerical resolution of each stage is chosen.

Acknowledgments

This work was supported in part by the Ministry of Education and Science (CICYT) of Spain under Contract TIN2006-01078 and Junta de Andalucia under Contract TIC-02800.

References

[1] F. Bensaali, A. Amira, and A. Bouridane, "Accelerating matrix product on reconfigurable hardware for image processing applications," *IEE Proceedings: Circuits, Devices and Systems*, vol. 152, no. 3, pp. 236–246, 2005.

[2] N. İsmailoğlu, O. Benderli, S. Yeşil, R. Sever, B. Okcan, and R. Öktem, "GEZGİN-2: an advanced image processing subsystem for earth-observing small satellites," in *Proceedings of the 2nd International Conference on Recent Advances in Space Technologies (RAST '05)*, pp. 605–610, IEEE JNL, Istanbul, Turkey, June 2005.

[3] R. Porter, K. McCabe, and N. Bergmann, "An applications approach to evolvable hardware," in *Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardware*, pp. 170–174, IEEE JNL, Pasadena, Calif, USA, 1999.

[4] E. Saez Peña, *Segmentación automática de video*, Tesis Doctoral, 2006.

[5] H. Quinn, L. A. S. King, M. Leeser, and W. Meleis, "Runtime assignment of reconfigurable hardware components for image

processing pipelines," in *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '03)*, pp. 173–182, April 2003.

[6] T. Sugimura, S. Jeoung Chill, H. Kurino, and M. Koyanagi, "Parallel image processing field programmable gate array for real time image processing system," in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT '03)*, pp. 372–374, 2003.

[7] E. Sáez, J. I. Benavides, and N. Guil Mata, "Reliable real time scene change detection in MPEG compressed video," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '04)*, vol. 1, pp. 567–570, Taipei, Taiwan, June 2004.

[8] F. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*, Chapman & Hall, Boca Raton, Fla, USA, 1996.

[9] M. Calviño, J. I. Benavides, S. Geninatti, F. J. Hormigo, and J. Villalva, "Hardware accelerators for the microblaze software embedded processor," in *Proceedings of the 6th Southern Programmable Logic Conference (SPL '07)*, 2007.

[10] J. P. Deschamps, G. J. A. Bioul, and G. D. Sutter, *Synthesis of Arithmetic Circuits FPGA, ASIC, and Embedded Systems*, John Wiley & Sons, New York, NY, USA, 2006.