EURASIP Journal on Applied Signal Processing 2004:12, 1921–1930

Design of a Low-Power VLSI Macrocell for Nonlinear Adaptive Video Noise Reduction

Sergio Saponara
Department of Information Engineering, University of Pisa, Via Caruso, 56122 Pisa, Italy
Email: sergio.saponara@iet.unipi.it

Luca Fanucci
Institute of Electronics, Information Engineering and Telecommunications, National Research Council, Via Caruso, 56122 Pisa, Italy
Email: luca.fanucci@iet.unipi.it

Pierangelo Terreni
Department of Information Engineering, University of Pisa, Via Caruso, 56122 Pisa, Italy
Email: pierangelo.terreni@iet.unipi.it

Received 26 August 2003; Revised 19 February 2004

A VLSI macrocell for edge-preserving video noise reduction is proposed in the paper. It is based on a nonlinear rational filter enhanced by a noise estimator for blind and dynamic adaptation of the filtering parameters to the input signal statistics. The VLSI filter features a modular architecture allowing the extension of both mask size and filtering directions. Both spatial and spatiotemporal algorithms are supported. Simulation results with monochrome test videos prove its efficiency for many noise distributions with PSNR improvements up to 3.8 dB with respect to a nonadaptive solution. The VLSI macrocell has been realized in a 0.18 μm CMOS technology using a standard-cells library; it allows for real-time processing of main video formats, up to 30 fps (frames per second) 4CIF, with a power consumption in the order of few mW.

Keywords and phrases: nonlinear image processing, video noise reduction, adaptive filters, very large scale integration architectures, low-power design.

1. INTRODUCTION

Noise reduction is a key issue in any video system to improve the visual appearance of the images. Especially in consumer electronics the sources of images such as video recorders, video cameras, satellite decoders, and so on are affected by different kinds of noise [1, 2, 3]. White Gaussian distribution is usually adopted to model the noise in case of digital video broadcasting [3] or CCD/CMOS cameras [1, 2, 3] while impulsive-like noise usually affects images from satellite TV decoders [2, 3]. An impulsive noise model is also used for faulty bits during coding and transmission or for video scanned from damaged films. To remove meaningless noise information, while preserving fine image details, a large variety of nonlinear filtering methods [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] have been proposed in literature since conventional linear filters are known to blur the images. They typically involve weighted averaging masks in case of Gaussian noise or order-statistic filtering in case of impulsive one. In some cases both methods have been combined to better withstand the different noise distributions in various video applications.

For the real-time implementation of these techniques several solutions, based on dedicated applied specific integrated circuits (ASIC) technology or software realization for commercial digital signal processors (DSPs) have been proposed [2, 6, 10, 12, 13]. The above approaches are typically affected by two main drawbacks. Firstly, some of them do not provide a noise estimation unit for a blind and dynamic adaptation to the input signal characteristics. Thus, to achieve better filtering performances, the noise distribution must be known a priori or an external circuit must be used for its estimation. Secondly, the filters are not optimized for low-power consumption which is mandatory for the success of any battery-powered video application such as wireless cameras, 3G mobile phones, and personal digital assistants to name but a few. Conventional DSP implementations of noise smoothing filters (e.g., using a Texas Instruments C80 [6, 13] or a Philips Trimedia1000 [2]) require thousands of mW while the power

budget for system-on-chip (SoC) video communications terminals (implementing a video codec plus pre/postfiltering units) is often bounded to few hundreds of mW [14, 15]. Reducing power consumption is also a key issue in highly integrated SoC to avoid heat removal problems requiring the use of costly packaging and cooling mechanisms. Therefore, the realization of cost-effective SoC video communication terminals requires the integration of a low-power filtering coprocessor (tens of mW) based on a modular architecture with automatic tuning and designed as an intellectual property (IP) macrocell to enable design reuse.

To address the above issues in this paper we present a very large scale integration (VLSI) architecture for edge-preserving noise reduction in different video applications. It is based on a rational filter enhanced by a noise estimator for blind and dynamic adaptation to the input signal characteristics. Implemented in a $0.18\,\mu$m, 1.6 V CMOS technology using a standard-cells library, the circuit minimizes power consumption in the order of few mW while keeping real-time processing for the main video formats. After this introduction, Section 2 describes the nonlinear adaptive algorithm adopted for noise reduction. Section 3 details its mapping into a power-optimized VLSI architecture. Section 4 discusses the characteristics of the achieved CMOS implementation and analyzes the algorithmic performance of the VLSI filter applied to monochrome test videos. Finally, conclusions are drawn in Section 5.

2. NONLINEAR ADAPTIVE NOISE REDUCTION ALGORITHM

2.1. Nonlinear noise reduction filter

The proposed algorithm is based on the class of rational filters that is a powerful tool for edge-preserving smoothing of different types of noise. A rational operator is basically a nonlinear lowpass filter with variable cut-off frequency and is expressed as a ratio of two polynomial functions: a built-in edge sensor (denominator) modulates the coefficients of a linear lowpass filter (numerator) to limit its action in presence of image details. Thus, meaningless noise information can be removed without blurring picture edges. Both spatial (4 filtering directions [5]) and spatio-temporal (5 to 13 filtering directions [6]) processing can be adopted. Equation (1) shows the general expression of a spatio-temporal rational filter working on 3×3 sample masks centred on the pixel to be filtered X_0^t (belonging to the current frame t). Y_0^t is the output filter result, X_i^t and X_j^t are spatially neighbouring input pixels (i.e., belonging to a 3×3 sample mask centred in the current frame t around X_0^t) while X_h^{t-1} and X_p^{t+1} are temporally neighbouring input pixels (i.e., belonging to 3×3 sample masks centred around the position of X_0^t in the previous frame $t-1$ and the following frame $t+1$). Finally, S and T represent the set of indices for the spatial and temporal filtering directions,

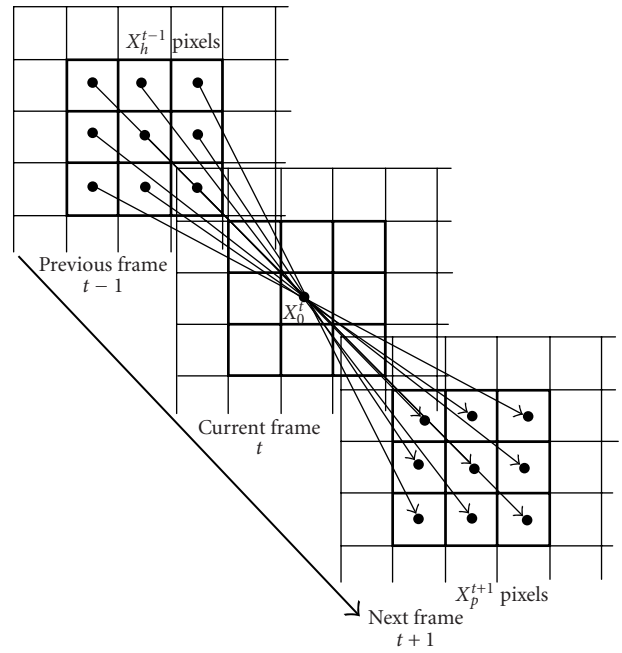

Figure 1: Pixels and set of directions considered for the temporal processing on 3×3 sample masks.

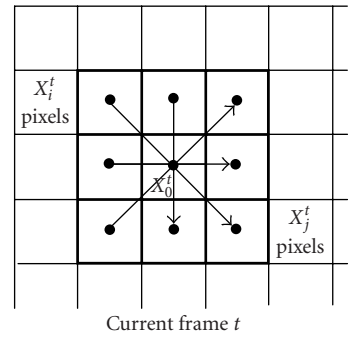

Figure 2: Pixels and set of directions considered for the spatial processing on a 3×3 sample mask.

respectively,

$$Y_0^t = X_0^t + \sum_{i,j \in S} \frac{\overbrace{b_i X_i^t + a_0 X_0^t + b_j X_j^t}^{\text{Linear filter}}}{1 + K_S \underbrace{\left(X_i^t - X_j^t \right)^2}_{\text{Edge sensor}}} + \sum_{h,p \in T} \frac{\overbrace{b_h X_h^{t-1} + a_0 X_0^t + b_p X_p^{t+1}}^{\text{Linear filter}}}{1 + K_T \underbrace{\left(X_h^{t-1} - X_p^{t+1} \right)^2}_{\text{Edge sensor}}}. \tag{1}$$

With reference to the filter expression proposed in (1), Figures 1 and 2 show pixels and sets of directions considered for the temporal (Figure 1, up to 9 filtering directions)

and spatial (Figure 2, up to 4 filtering directions) processing. To optimize filtering performance the coefficients in (1) (b_i, b_j, b_h, b_p, a_0, K_S, K_T) have to be properly selected according to the noise distribution: Gaussian, contaminated-Gaussian, [16] or impulsive noise.

The rational algorithm has been selected since, as proved in [2, 4, 5, 6], it outperforms a large variety of linear and nonlinear (e.g., sigma filter, center-weighted median filter, L-filters) methods [7, 8, 9, 10, 11] in terms of computational complexity versus noise reduction trade-off. Moreover, it is characterized by a regular computational flow (i.e., the ratio of two polynomial functions) that simplifies hardware implementation through VLSI array processing. To better understand the VLSI mapping of the algorithm the expression of the filter in (1) can be rewritten as reported in (2) (case of 3×3 sample masks). Equation (2) shows that the backbone of the nonlinear algorithm is a set of 3-tap (Z-tap for generic $Z \times Z$ sample masks) linear filters (LF): one LF for each spatial or temporal filtering direction. The LF outputs are weighted by nonlinear terms ($\beta_{i,j}$ and $\beta_{h,p}$) produced according to the difference of proper control pixels, spatial (X_i^t and X_j^t) and temporal (X_h^{t-1} and X_p^{t+1}) neighbours of the pixel to be processed X_0^t.

$$Y_0^t = X_0^t + \sum_{i,j\in S} \overbrace{\beta_{i,j}}^{\text{Nonlinear weight}} \overbrace{[\, b_i X_i^t + a_0 X_0^t + b_j X_j^t \,]}^{\text{Linear filter}} + \sum_{h,p\in T} \overbrace{\beta_{h,p}}^{\text{Nonlinear weight}} \overbrace{[\, b_h X_h^{t-1} + a_0 X_0^t + b_p X_p^{t+1} \,]}^{\text{Linear filter}}, \tag{2}$$

where $\beta_{i,j} = 1/(1+K_S(X_i^t - X_j^t)^2)$ and $\beta_{h,p} = 1/(1+K_T(X_h^{t-1} - X_p^{t+1})^2)$.

2.2. Filter extension

The rational algorithm features a modular structure allowing the extension of the filtering directions and/or the number of LF taps by the iterative use of a single filtering macrocell or by the cascade of several macrocells.

As an example of the extension of processing directions, the spatio-temporal algorithm with 13 directions in (2), factorized as reported in (3a) and (3b), can be implemented in two iterative steps by a single filtering macrocell supporting in parallel up to 9 filtering directions.

$$Y = X_0^t + \sum_{i,j\in S} \beta_{i,j}\left[b_i X_i^t + a_0 X_0^t + b_j X_j^t\right], \tag{3a}$$

$$Y_0^t = Y + \sum_{h,p\in T} \beta_{h,p}\left[b_h X_h^{t-1} + a_0 X_0^t + b_p X_p^{t+1}\right]. \tag{3b}$$

According to the above factorization the algorithm in (2) can be also implemented by cascading two macrocells supporting in parallel up to 4 (the first filter) and 9 (the second filter) processing directions: in this case the first filter implements the spatial part of the algorithm (denoted as in (3a)) while the second filter implements concurrently the temporal part (denoted as in (3b)).

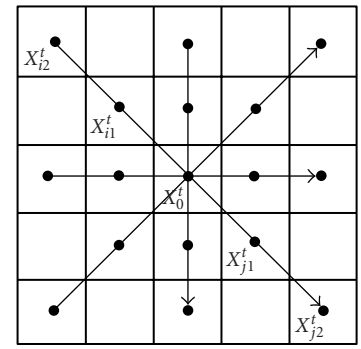

Figure 3: Pixels and set of directions considered for the spatial processing on 5×5 sample masks.

As an example of the extension of LF taps, the rational algorithm in (4) (5-tap LFs and 4 spatial directions reported in Figure 3), factorized as reported in (5a) and (5b), can be implemented by the iterative use of a single macrocell supporting 3-tap LFs and 4 processing directions. According to the factorization in (5a) and (5b), the algorithm in (4) can be also implemented by the cascade of two macrocells (each supporting only 3-tap LFs) working concurrently.

$$Y_0^t = X_0^t + \sum_{i,j\in S} \beta_{i,j}\left[b_{i2}X_{i2}^t + b_{i1}X_{1i}^t + a_0 X_0^t + b_{j1}X_{j1}^t + b_{j2}X_{j2}^t\right], \tag{4}$$

$$Y = X_0^t + \sum_{i,j\in S} \beta_{i,j}\left[b_{i1}X_{i1}^t + a_0 X_0^t + b_{j1}X_{j1}^t\right], \tag{5a}$$

$$Y_0^t = Y + \sum_{i,j\in S} \beta_{i,j}\left[b_{i2}X_{i2}^t + b_{j2}X_{j2}^t\right]. \tag{5b}$$

Therefore, given a generic filtering macrocell supporting the elaboration of Z-tap LFs and D processing directions, the cascade of F macrocells or the iterative use for F steps of a single macrocell allow the implementation of rational algorithms characterized by: (i) Z-tap LFs and $F \times D$ filtering directions; (ii) $F \times Z$-tap LFs and D filtering directions.

2.3. Data flow organization

The expression of the algorithm reported in (2) refers to the processing of one output pixel Y_0^t using a filtering mask centred on the input pixel X_0^t. The processing of a whole frame is obtained by applying the expression in (2) to all pixels in the frame, scanned in raster (row-by-column) mode. This way, as it emerges from Figure 4, the filtering masks centred on successive input pixels X_n^t and X_{n+1}^t are overlapping; 6 pixels out of 9 for 3×3 masks. A local buffer (i.e., a copy of the overlapping data) can be inserted to exploit data reuse and reduce by a factor of 3 the frequency of data transfers between the background frame memories and the filter. As widely proved in literature [17, 18, 19], the insertion at algorithmic level of such data copies enables, at architectural level, the reduction of power consumption during I/O data exchange with

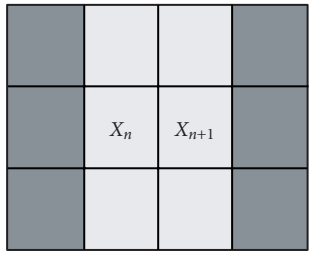

Figure 4: Overlapped processing masks of successive input pixels.

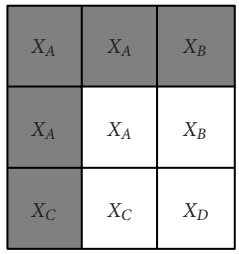

Figure 5: Filtering mask when elaborating the pixel in the top-left corner of a frame.

external frame memories (which entails the major contribution to power consumption in multimedia systems). The proposed approach implements a horizontal window overlapping.[1]

Due to the usage of 3×3 filtering masks centred on the pixel to be processed a problem arises when handling the image border: the values of the pixels beyond the border are missing. To overcome this problem we envisage a replication of the image border. As an example, Figure 5 shows the 3×3 sample mask adopted to elaborate the pixel in the top-left corner of a frame (X_A within the white box): the missing pixels within the shaded boxes are obtained as a copy of the neighbouring pixels located within the image (white boxes).

2.4. Adaptive filter scheme

Simulation results incorporating synthetic and real-world noise show that the optimal set of filtering parameters depends on the type of noise. Therefore a noise estimation step has been inserted to adaptively select the best set of coefficients for the filter reported in (2). The noise estimator receives as input the difference between the original noisy image and the image processed using the nonlinear filtering algorithm. Starting from these samples, some noise parameters such as variance (σ^2), mean value (μ), and 4th central moment of data (μ_4) are firstly evaluated; then, noise statistics are compared to proper thresholds to discriminate the different distributions and select the optimal set of filter coefficients. To reduce the computational and data transfer workload of the noise estimator the estimation technique is based on the assumption that the type of noise uniformly applies over every individual image of the sequence. Accordingly, only a part of every image needs to be analyzed. A 32×32 pixel subimage, obtained by subsampling (drop of every second sample) a 64×64 area positioned on the centre of the frame, was selected by computer simulations as the best trade-off between computational complexity and estimation accuracy. The algorithmic performance of the proposed filter is analyzed in Section 4.2 by comparing results achieved when the filter coefficients are either adapted using noise estimation, or kept fixed during processing time, whereas finite precision arithmetic conditions related to VLSI architecture implementation are taken into account in both cases.

3. VLSI ARCHITECTURE

3.1. Architecture overview

The nonlinear adaptive noise reduction algorithm described in Section 2 is implemented by a VLSI filtering macrocell whose global architecture is sketched in Figure 6. This architecture is made up of the following building blocks:

(i) a programmable nonlinear filtering core implementing the algorithm in (2);
(ii) a unit for noise estimation and filter tuning;
(iii) memory resources for data flow management, including also the local input buffer that implements the horizontal window overlapping described in Section 2.3;
(iv) a control unit[2] that provides all relevant control signals (dashed lines in Figure 6).

The *programmable nonlinear filter* block in Figure 6 is implemented by the circuit sketched in Figure 7. The core of this circuit is an array of 3-tap programmable LFs (LF in Figure 7 and (2)) processing the input samples of 3×3 masks centred on the pixel to be filtered X_0^t. Each LF is realized using a carry-save multiplier (see Section 3.3 for further details) cascaded by an accumulator unit. The relevant outputs are weighted by nonlinear terms produced according to the absolute difference (AD in Figure 7) of suitably chosen pixels ("control pixels" in Figure 7 indicated as X_i^t, X_j^t and X_h^{t-1}, X_p^{t+1} in Figures 1 and 2 and (2)). The results of all filtering directions are provided to the adder tree that produces the final output value. The adder tree unit is followed by a programmable output stage that allows the extension of both mask size and filtering directions. The circuit of Figure 7 processes concurrently up to 5 filtering directions supporting both spatial (4 directions [5]) and spatio-temporal (5 directions [6]) rational algorithms. Following the approach described in Section 2.2, the extension of both mask size and processing directions can be obtained by a cascade of filtering circuits or by the iterative

[1] As proved in [17, 18] other possibilities of data flow organization (and design of a memory hierarchy between background frame memories and data paths of the video processor) exist enabling also vertical window overlapping (not supported in the current filter implementation).

[2] The control logic managing the communication between the architecture and the external frame memories is not included. In its current implementation the proposed VLSI macrocell acts as a slave when integrated in a SoC video device.

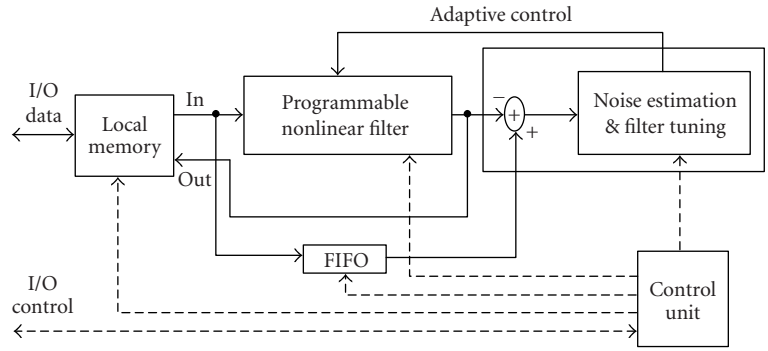

Figure 6: Block diagram of the VLSI architecture.

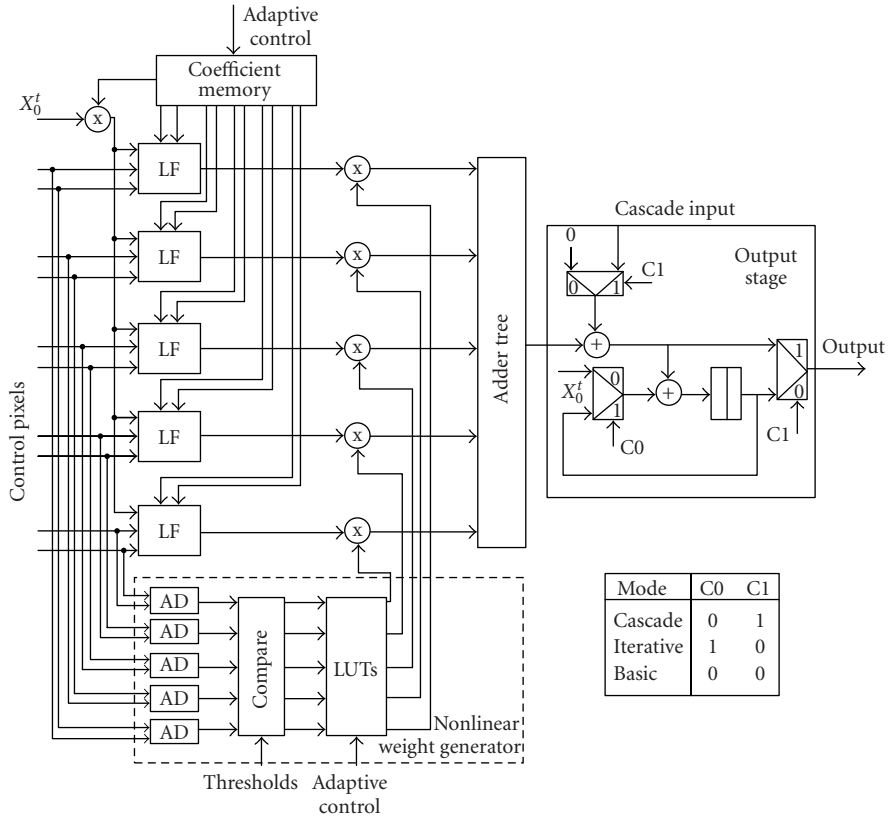

Figure 7: Core of the programmable nonlinear filter implemented in VLSI.

use of a single unit. The preferred mode can be selected by the user by proper programming of the output stage according to the modes and parameters settings specified in the bottom-right corner of Figure 7. In the first "cascade" mode, by exploiting architectural parallelism, real-time processing can be achieved with reduced clock speed and reduced supply voltage thus minimizing power consumption. In the second "iterative" mode the adoption of a single filtering unit allows for silicon area minimization. For instance, starting from the circuit in Figure 7 the spatio-temporal rational algorithm with 13 filtering directions in [6] can be implemented using 3 processing iterations of a single unit, by selecting the "basic" mode configuration of the output stage for the first iteration, and the iterative mode for the remaining iterations. Alternatively, it is also possible to cascade 3 units, by selecting the basic mode for the first stage, and the cascade mode for the remaining stages of the cascaded structure.

The unit for *noise estimation and filter tuning* in Figure 6 implements the adaptive control scheme proposed in Section 2.4. With reference to a 32×32 pixel subimage positioned on the centre of the frame (see Section 2.4), the noise estimator receives as input the difference between the

pixels of the original noisy image and the pixels of the filtered image. A first in first out (FIFO) buffer (see Figure 6) is used to ensure data synchronization at the input of the noise estimator. Starting from these samples, some noise parameters such as variance (σ^2), mean value (μ), and 4th central moment of data (μ_4) are firstly evaluated; then, noise statistics are compared to proper thresholds to discriminate the different distributions and select the optimal set of filter coefficients. A clock-gating strategy is applied to reduce power consumption: if the target is only to discriminate short-tailed from long-tailed distributions, then the knowledge of σ^2 and μ is enough and the circuitry for μ_4 computation is powered down. Otherwise, when the video application requires a more detailed tuning of the filter, μ_4 is evaluated and compared to σ^4 to discriminate among Gaussian ($\mu_4 \sim 3\sigma^4$), contaminated Gaussian ($\mu_4 > 3\sigma^4$) and impulsive noise ($\mu_4 \gg 3\sigma^4$). The latter approach is similar to the one presented in [2].

In the proposed adaptive scheme the estimation of the noise statistics for the current frame t occurs concurrently to the filtering of the same frame t. When processing a video sequence, several strategies can be devised to select the optimal set of filter coefficients for each frame. The VLSI architecture described in this paper supports two different strategies. According to the first strategy, adopted for tests described in Section 4.2, each current frame t of the video sequence is filtered once and the selection of the filter coefficients is based on the noise statistics estimated when processing the previous frame $t-1$. According to the second strategy, each frame t of the video sequence is filtered at least twice. During the first filter application, the filter coefficients are selected according to the noise statistics estimated while processing the previous frame $t-1$. During the second filter application, the filter coefficients assigned to frame t are selected according to the noise statistics estimated for the same frame t during the first filter application. The first strategy leads to costs, expressed in terms of computational complexity and data loading, that are twice lower (B times lower in case of B successive filter applications) than for the second strategy. If the type of noise remains the same over all frames of the video sequence, the quality of the noise reduction is similar for both strategies. However, as it will be further detailed in Section 4.2, when the noise type changes over frames of the video sequence, the first strategy is less effective regarding noise reduction.

3.2. Nonlinear weight generation

To avoid the use of power-consuming divider and square operators, the generation of the nonlinear weights ($\beta_{i,j}$ and $\beta_{h,p}$ in (2)) is based on a lookup table (LUT) approach as in [2, 6]. In this work, for each filtering direction 6 possible LUTs are defined, each optimized for a different noise distribution (Gaussian, contaminated-Gaussian [16], or impulsive noise) and for both spatial and spatio-temporal processing. As an example, Figure 8 represents the shape of the generic nonlinear term $\beta_{i,j}$ in (2) versus the absolute difference of the control pixels X_i^t and X_j^t considering 8-bit input pixels and $K_S = 10^{-3}$. Figure 9 shows its representation when using quantization levels stored in an LUT. In Figure 9 the term $\beta_{i,j}$ is approximated by a staircase-shaped function[3] obtained after subdivision of the horizontal axis into 8 nonuniformly distributed intervals, whereas the vertical axis is homogeneously quantized into 64 levels. The higher the number of horizontal intervals and vertical quantization levels, the higher the approximation accuracy of the curve in Figure 8 and the circuit complexity (number of comparators in Figure 7 and size of the LUT). More precisely, each LUT contains P words of L bits, thus resulting in a global size of $P \times L$ bits, where P indicates the number of horizontal intervals considered in Figure 9, whereas the number of vertical quantization levels is specified by 2^L (with $P = 8$ and $L = 6$ in Figure 9).

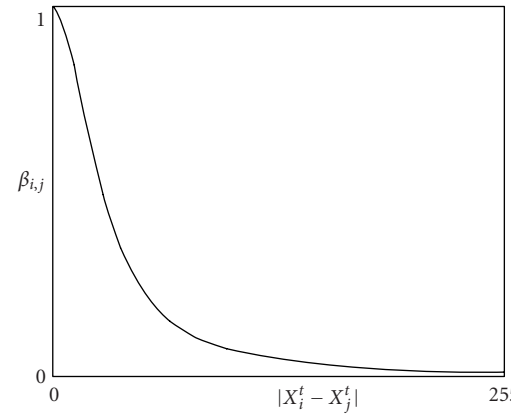

Figure 8: Shape of the generic nonlinear term $\beta_{i,j}$ in (2), with $K_S = 10^{-3}$.

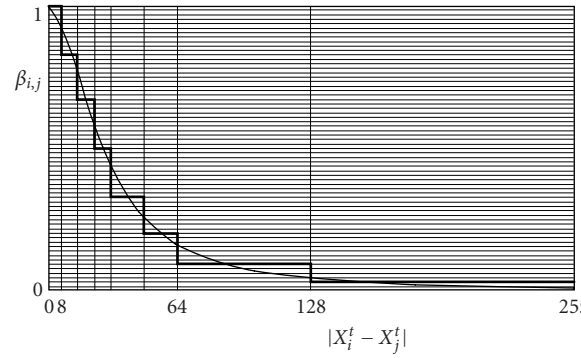

Figure 9: Approximation of the generic nonlinear term using the LUT approach.

[3] In the reported example $\beta_{i,j}$ is equal to: 1 when $|X_i^t - X_j^t|$ ranges from 0 to 7, 53/64 when $|X_i^t - X_j^t|$ ranges from 8 to 15, 43/64 when $|X_i^t - X_j^t|$ ranges from 15 to 23, 1/2 when $|X_i^t - X_j^t|$ ranges from 24 to 31, 21/64 when $|X_i^t - X_j^t|$ ranges from 32 to 47, 13/64 when $|X_i^t - X_j^t|$ ranges from 48 to 63, 6/64 when $|X_i^t - X_j^t|$ ranges from 64 to 127, and 2/64 when $|X_i^t - X_j^t|$ ranges from 128 to 255.

The approach proposed in Figure 9 exploits a nonuniform distribution of the horizontal intervals to increase the processing accuracy while limiting the LUT size. As widely proved in the field of video coding, the input pictures of typical video applications are characterized by high level of spatial correlation. The absolute differences of neighbouring input pixels (such as X_i^t and X_j^t in (2)) are concentrated in the first half, 0–127, of the whole range of possible values 0–255. Tests with pictures extracted from different video sequences (e.g., Akiyo, Foreman, Basketball, and Coastguard) prove that only a low percentage of the absolute differences $|X_i^t - X_j^t|$ is in the range 128–255. As a consequence, concentrating the horizontal intervals around low values of the range 0–255 leads to a more efficient estimation of the $\beta_{i,j}$ term compared to a uniform distribution. For instance, computer simulations demonstrate that the noise reduction performances of the whole filter when adopting the 8 horizontal intervals in Figure 9, delimited by the values $(0, 8, 16, 24, 32, 48, 64, 128)$, or adopting 32 horizontal intervals with a fixed step, delimited by the values $(0, 8, 16, 24, 32, 40, \ldots, 248)$, are roughly the same. The former case with nonuniform distribution entails a LUT size 4 times lower than the latter case with a uniform distribution. The above approach, described for $\beta_{i,j}$, is also applicable to the nonlinear term $\beta_{h,p}$, in which case the nonuniform distribution of the intervals exploits the temporal data correlation typically occurring in video sequences.

3.3. Multiplier optimization

The optimization of the multiplier is a main issue for the cost-effective design of VLSI filters since it usually represents the bottleneck in terms of circuit complexity and processing speed. This is in particular the case of parallel architectures, similar to the one described in Figure 7, in which the same multiplier unit is instantiated several times. Many architectures have been proposed in literature [20] to implement multiplications with different trade-offs between circuit complexity and processing speed depending on the operand types (range of possible values, number of bits for each value). For the proposed video filter we have designed and compared two different multipliers: (i) carry-save multiplier (based on a cascade of carry-save adders); (ii) ROM-based multiplier (all the results of the multiplication between the input samples and the set of filter coefficients are precalculated and stored in a ROM). Figures 10 and 11 compare their performance in terms of circuit complexity and processing time (and its inverse, the maximum processing frequency) for different bit sizes N and M of the operands. Reported values have been extracted from gate-level synthesis results in a 0.18 μm CMOS technology using a standard-cells library. The filter proposed in Figure 7 involves two types of multiplications: (i) input pixels multiplied by the coefficients of the LF; (ii) LF outputs multiplied by the nonlinear coefficients β.

For multimedia systems the incoming video frames are represented with a resolution from 4 to 12 bits/pixel [21], the typical value being 8 bits/pixel. Accordingly, the number of bits N assigned to the frame pixels, and the number of bits

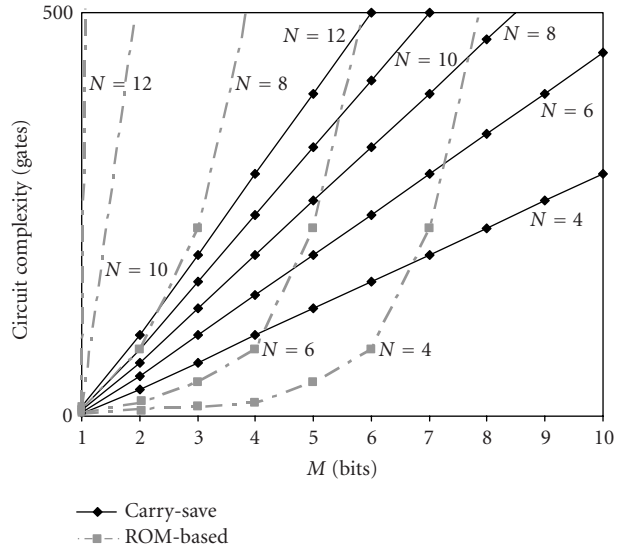

Figure 10: Circuit complexity for carry-save and ROM-based $N \times M$ bits multipliers.

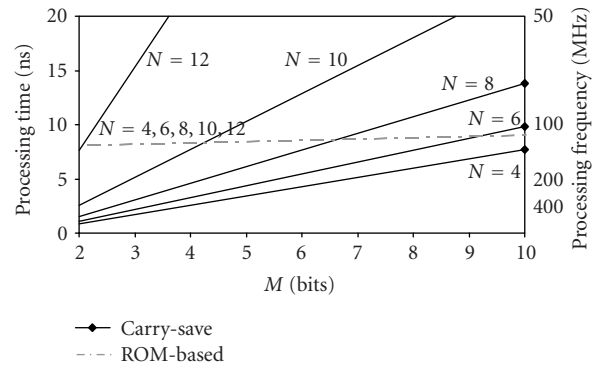

Figure 11: Processing time for carry-save and ROM-based $N \times M$ bits multipliers.

M considered for the filter coefficients, are specified within ranges 4 to 12 and 1 to 10, respectively, in order to evaluate the complexity and processing time in Figures 10 and 11. For $M = 1$ the $N \times 1$ bits multiplication is merely realized using an N-bit AND gate.

In the ROM-based approach the circuit complexity increases exponentially with the size of the operands (see Figure 10) while the processing time, for the considered operand range and CMOS technology, is roughly estimated as being constant (see Figure 11), and limited by the read access time of the ROM. On the contrary the carry-save multiplier is characterized by a linear increase of both circuit complexity (Figure 10) and processing time (Figure 11) versus the size of the operands. The ROM-based approach is suitable for low-cost, low-quality video applications since the small number of used bits can be exploited to reduce circuit complexity with respect to the carry-save technique ($N \times M$ bits multiplications up to 6×4 or 4×6 bits) achieving

Table 1: Resulting PSNR (dB) for different statistical noise sources.

Video signal	Type of statistical noise affecting the video signal		
	Gaussian	Contaminated Gaussian	Impulsive
Original noisy video signal	25.22	14.58	18.11
Output video signal filtered by the proposed algorithm	28.61	23.29	26.39
Output video signal filtered by the algorithm [6]	28.61	23.36	26.01

a computational throughput of roughly 120 MHz. For higher operand sizes the carry-save operator minimizes circuit complexity. The processing speed is enough for the target applications of the proposed filter (compare the results in Figure 11 to the clock frequency requirements reported in Section 4.1 for the noise smoothing of main video formats). Therefore the carry-save approach has been selected to implement the VLSI filter architecture.

4. PERFORMANCE OF THE ACHIEVED CMOS IMPLEMENTATION

4.1. CMOS implementation results

The whole VLSI filter architecture is conceived as a parametric IP macrocell according to a design reuse policy. All parameters of the very high speed integrated circuits hardware description language (VHDL) description such as word lengths of internal and I/O data paths, LUT size, number of LF taps, and comparators can be modified before logic synthesis by the IP user integrator to achieve the desired balance between circuit complexity and filtering performances.

The VLSI macrocell has been synthesized, within the Synopsys^{TM} CAD environment, in a 0.18 μm, 6 metal level CMOS technology using a standard-cells library. The following set has been considered for the VHDL parameters: 5 filtering directions, 3-tap programmable LFs, 8-bit I/O pixels, and 30 different LUTs (6 for each filtering direction) of 48 bits ($P = 8$ words of $L = 6$ bits).

The circuit complexity amounts to roughly 20 Kgates (6 Kgates for noise estimation, 13 Kgates for the nonlinear filter, and 1 Kgate for control logic) plus 180 bytes of ROM and about 10 Kbits of a low-power SRAM. It is noticed that 40% of the filter core complexity is due to the carry-save multipliers: with reference to Figure 7, 8×6 bits multipliers for the LF array and 12×6 bits multipliers for the nonlinear weights β. The computational throughput is up to 0.5 samples/cycle. The circuit clock frequency, required for the real-time processing of 30 fps QCIF (176×144 pixels), CIF (352×288 pixels), and 4CIF (704×576 pixels) video formats, amounts to 2.28, 9.12, and 36.48 MHz, respectively. The average power consumption (extracted from gate-level simulations of different test video sequences using Synopsys Power Compiler tool) is 1.1 mW for 30 fps QCIF, 2.3 mW for 30 fps CIF, and 7.1 mW for 30 fps 4CIF. The power consumption results refer to the gate-level synthesis of the whole architecture described in Figure 6 (not including the contribution of the clock tree and the I/O pad capacitances) in the 0.18 μm CMOS technology considering a supply voltage of 1.6 V and a temperature of 85 C°. The above results are very interesting when compared to known implementations of nonlinear noise reduction filters [2, 6, 10, 12, 13]. The low complexity of the proposed macrocell is also suitable for realization into field programmable gate array (FPGA) technology. The VLSI macrocell has been synthesized, without any optimization for the specific device, on a Xilinx XCV1000 (0.22 μm, 5 metal levels) occupying 15% of the available FPGA hardware resources.

4.2. Algorithmic performance analysis

To assess the algorithmic performance of the VLSI macrocell (architecture with finite precision arithmetic using the VHDL parameter configuration described in Section 4.1) several tests have been carried out using monochrome (greyscale 8 bits/pixel) input sequences. For test sequences originally furnished in color, the greyscale images have been obtained from the luma components discarding the chroma samples. Each frame of a video sequence is filtered once and the selection of its processing coefficients is based on the noise statistics estimated when elaborating the previous frame.

Table 1 compares, for several input noise statistics, the performance of the proposed IP macrocell to a software implementation of the 3D nonlinear filter presented in [6]. The noise type is the same for all the frames of the input video sequence. Reported values refer to the average peak signal-to-noise ratio (PSNR), evaluated on whole frames, obtained for the spatio-temporal processing of the Basketball test sequence. The results of Table 1 show that the proposed VLSI macrocell achieves the same good filtering performances of the algorithm in [6]. Similar results have been obtained for other test video sequences (e.g., Akiyo, Coastguard, Foreman, and Miss America).

Figure 12 shows the improvement of the adaptive filter scheme versus the fixed one in case of videos with rapidly changing noise statistics. The input video is the Basketball sequence corrupted with additive Gaussian (frames 1 to 5), impulsive (frames 6 to 10), and contaminated-Gaussian (frames 11 to 15) noise distributions. More precisely, this figure shows the difference between the PSNR of the sequence filtered using the VLSI macrocell with adaptive filter tuning, response curve labelled "Adaptive" in Figure 12, and the PSNR of the sequence filtered using the same macrocell without noise estimation and adaptive tuning, response curve labelled "Fixed" in Figure 12. Both adaptive

and fixed cases implement the spatial algorithm (4 directions depicted in Figure 2) under the same finite precision arithmetic conditions. The three curves in Figure 12 refer to different settings of the fixed filter scheme optimized for Gaussian (triangles), impulsive (squares), and contaminated-Gaussian noise (circles). The adaptive scheme is initialized with a set of parameters tailored for the Gaussian statistics.

In frames 1 to 5 of Figure 12, when the noise type is Gaussian, the performance of the adaptive filter versus the fixed one is roughly independent from the temporal succession of the frames since the adaptive scheme is initialized with a set of parameters tailored for the Gaussian distribution (the PSNR improvement is equal to its maximum value of 2 dB versus fixed-contaminated Gaussian, 1 dB versus fixed impulsive, and 0 dB versus fixed Gaussian). When in frame 6 the noise type changes from Gaussian to impulsive, the adaptive filter selects the processing coefficients according to the noise type estimated in frame 5, corresponding to Gaussian type. Therefore the performance of the adaptive filter is the same as for the fixed-Gaussian noise, as confirmed by the PSNR improvement of 0 dB observed for frame 6, and it is slightly worse than the performance achieved for fixed-impulsive noise, as indicated by the negative PSNR improvement obtained for frame 6. The noise type affecting the input video frames remains the same over frames 6 to 10, hence the noise estimation is gradually refined to let the set of filter coefficients stabilize around optimal values, as observed for the frames 8 to 10. The PSNR improvement versus fixed schemes is then maximum, amounting to 0 dB versus fixed-impulsive noise, 1 dB versus fixed-contaminated Gaussian noise, and 2 dB versus fixed-Gaussian noise, respectively. Similar results are obtained when the noise type changes from impulsive to contaminated-Gaussian noise over frames 11 to 15, with a maximum PSNR improvement of 0 dB versus fixed-contaminated Gaussian noise, 1 dB versus fixed-impulsive noise, and 3.8 dB versus fixed-Gaussian noise, respectively. Globally considered for the given example, it is shown that even in case of a video sequence with rapidly changing noise statistics, the proposed adaptive noise reduction algorithm applied using the first strategy defined in Section 3.1 leads to an improvement of the PSNR up to 3.8 dB with respect to a nonadaptive solution. It is worth noting that the different video noise distributions reported in Table 1 and Figure 12 are generated according to the mathematical models proposed in [16].

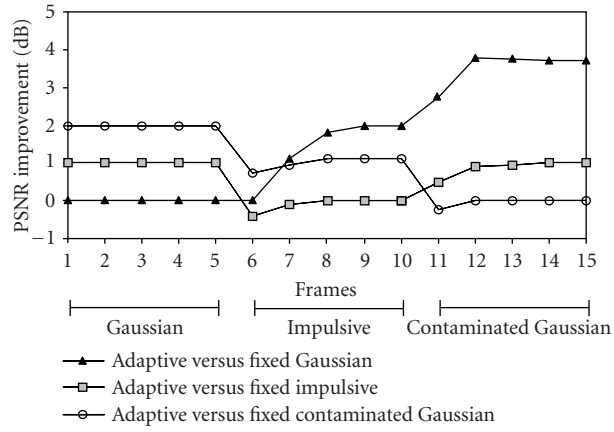

Figure 12: PSNR improvement of the adaptive filter scheme versus the fixed one; Basketball test sequence corrupted with different noise distributions.

5. CONCLUSIONS

A VLSI filter architecture for video noise reduction is presented in the paper. Based on a nonlinear rational operator the filter is enhanced by a noise estimator for blind and dynamic adaptation to the input signal characteristics. The architecture is conceived as an IP macrocell to enable design reuse and features a modular structure allowing the extension of mask size and filtering directions. Both spatial and spatio-temporal algorithms are supported. Realized in 0.18 μm CMOS technology using a standard-cells library, the VLSI filter allows for real-time processing of main video formats, up to 30 fps 4CIF, with a power consumption in the order of few mW. Simulation results with monochrome test videos prove its efficiency for many noise distributions with PSNR improvements up to 3.8 dB with respect to a nonadaptive solution. This way it represents an optimal solution for edge-preserving noise reduction in low-power and/or high-throughput SoC video devices. Current research activities aim at extending the application of the proposed macrocell to handle color images in 4:2:0 YUV format, and at considering its integration with an MPEG-4 compliant video encoder.

ACKNOWLEDGMENTS

This work was partially supported by the Italian National Research Council in the framework of the 5% *Microelectronics* project. Discussions with Professor Ramponi, University of Trieste, within the same project are gratefully acknowledged. We would like to thank the anonymous reviewers for useful comments and suggestions.

REFERENCES

[1] G. E. Healey and R. Kondepudy, "Radiometric CCD camera calibration and noise estimation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 267–276, 1994.

[2] L. Tenze, S. Carrato, C. Alessandretti, and S. Olivieri, "Design and real-time implementation of a low cost noise reduction video system," in *Proc. COST 254-Workshop on Intelligent Communication Technologies and Applications*, pp. 36–40, Neuchatel, Switzerland, May 1999.

[3] A. Amer and H. Schröder, "A new video noise reduction algorithm using spatial subbands," in *Proc. IEEE Conference on Electronics, Circuits and Systems*, vol. 1, pp. 45–48, Rodos, Greece, October 1996.

[4] S. Mitra and G. Sicuranza, *Nonlinear Image Processing*, Academic Press, San Diego, Calif, USA, 2000.

[5] G. Ramponi, "The rational filter for image smoothing," *IEEE Signal Processing Letters*, vol. 3, no. 3, pp. 63–65, 1996.

[6] F. Cocchia, S. Carrato, and G. Ramponi, "Design and real-time implementation of a 3-D rational filter for edge preserving smoothing," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 4, pp. 1291–1300, 1997.

[7] S.-J. Ko and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Trans. Circuits and Systems*, vol. 38, no. 9, pp. 984–993, 1991.

[8] J.-S. Lee, "Digital image smoothing and the sigma filter," *Computer Vision, Graphics and Image Processing*, vol. 21, no. 3, pp. 255–269, 1983.

[9] I. Pitas and A. N. Venetsanopoulos, "Application of adaptive order statistic filters in digital image/image sequence filtering," in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 327–330, Chicago, Ill, USA, May 1993.

[10] G. de Haan, T. Kwaaitaal-Spassova, M. Larragy, O. Ojo, and R. Schutten, "Television noise reduction IC," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 143–154, 1998.

[11] G. Arce, "Multistage order statistic filters for image sequence processing," *IEEE Trans. Signal Processing*, vol. 39, no. 5, pp. 1146–1163, 1991.

[12] G. Bernacchia and S. Marsi, "A VLSI implementation of a reconfigurable rational filter," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 1076–1085, 1998.

[13] Y. Loh, L. Chew, and U. Chan, "Multiprocessor denoising of weak video signals in strong noise," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, pp. 3124–3127, Orlando, Fla, USA, May 2002.

[14] M. Takahashi, T. Nishikawa, M. Hamada, et al., "A 60-MHz 240-mW MPEG-4 videophone LSI with 16-Mb embedded DRAM," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1713–1721, 2000.

[15] A. Chimienti, L. Fanucci, R. Locatelli, and S. Saponara, "VLSI architecture for a low-power video codec system ," *Microelectronics Journal*, vol. 33, no. 5-6, pp. 417–427, 2002.

[16] M. Gabbouj and I. Tabu, *TUT Noisy Image Database v.1.0*, Tampere University of Technology, Tampere, Finland, 1994.

[17] F. Catthoor, S. Wuytack, E. Greef, F. Balasa, L. Nachtergaele, and A. Vandecappelle, *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design*, Kluwer Academic Publishers, Boston, Mass, USA, 1998.

[18] F. Catthoor, K. Danckaert, C. Kulkarni, et al., *Data Access and Storage Management for Embedded Programmable Processors*, Kluwer Academic Publishers, Boston, Mass, USA, 2002.

[19] T. Meng, B. Gordon, E. Tsern, and A. Hung, "Portable video-on-demand in wireless communication," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 659–680, 1995.

[20] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, Reading, Mass, USA, 1985.

[21] ISO/IEC 14496-2 (MPEG-4), "Generic coding of audio visual objects," 1998.

Sergio Saponara received his M.S. degree in electronics and his Ph.D. degree in information engineering, both from the University of Pisa, Italy, in 1999 and 2003, respectively. In 2001, he collaborated with Consorzio Pisa Ricerche on a MEDEA+ project related to the low-power design of an xDSL modem. In 2002, he was with multimedia image compression systems (MICS) group at Interuniversity Microelectronics Centre (IMEC), Leuven, Belgium, under a Marie Curie research scheme working on the complexity analysis of advanced video coding standards. Currently, he is a Researcher at Pisa University, working on algorithms and VLSI architecture design for multimedia and low-power CMOS design methodologies.

Luca Fanucci was born in Montecatini Terme, Italy, in 1965. He received the Doctor Engineer (with the highest honors) and the Ph.D. degrees, both in electronic engineering, from the University of Pisa, Pisa, Italy, in 1992 and 1996, respectively. From 1992 to 1996, he was with the European Space Agency's Research and Technology Center, Noordwijk, the Netherlands, where he was involved in several activities in the field of VLSI for digital communications. He is currently a Research Scientist at the Italian National Research Council in Pisa. Since 2000, he has been Professor of microelectronics at the University of Pisa, Italy. His main interests are in the areas of system-on-chip design, low-power systems, VLSI architectures for real-time image and signal processing, and applications of VLSI technology to digital and RF communication systems.

Pierangelo Terreni is Full Professor of electronics at the Engineering Faculty of the University of Pisa. He is involved in research activities in VLSI design for many years. In particular, he worked on the design of real-time high-performance systems for digital signal processing. In cooperation with other colleagues, he participated in identifying, realizing, and testing a design methodology based on systolic arrays. For the past years he has been involved in the design of high-performance low-power digital systems. Professor Terreni is National Coordinator of a research project cosponsored by Ministry of University and Research (MURST) and he is Manager of a section of the national research project on microelectronics and VLSI architectures of the National Research Council.