Low Bit Rate Image Compression Core for Onboard Space Applications

Pasquale Corsonello, *Member, IEEE*, Stefania Perri, *Member, IEEE*, Giovanni Staino, Marco Lanuzza, and Giuseppe Cocorullo, *Member, IEEE*

***Abstract*—This paper presents low-cost, purpose optimized discrete wavelet transform-based image compressors for future spacecrafts and microsatellites. The hardware solution proposed here exploits a modified set partitioning in hierarchical trees algorithm and ensures that appropriate reconstructed image qualities can be achieved also for compression ratios over 100:1.**

Several implementations are presented varying the parallelism level and the tile size. Obtained results demonstrate that, using a parallel implementation operating on a 64 × 64 size tile, a maximum data rate of about 18 Mpixels/s can be sustained. In this case, only 4500 slices and 24 BlockRAMs of a XILINX Virtex II device are required.

***Index Terms*—Field programmable gate arrays (FPGAs), image coding, set partitioning in hierarchical trees, wavelet transform.**

I. Introduction

FUTURE spacecrafts and microsatellites will have the ability to capture an increasingly larger number of images by means of multispectral, high-resolution cameras [1], [2]. Captured images are transmitted to Earth through communication subsystems whose bandwidth often represents an unavoidable bottleneck for the transmission of such information. Furthermore, considering the orbit on which the satellites flight, they will have a few tens of minutes per day of contact time with the receiver ground station. Due to this, efficient onboard image compression subsystems and appropriately sized memory buffers become ever more necessary. Finally, the designer has to accurately take into account the power and mass budget that can be allocated in such an image processing payload.

It is well known that for the above applications software solutions are not adequate. In fact, their limited speed performances do not allow the desired frame-rate to be achieved. For this reason, on purpose hardware systems are most often the preferred solution. Main characteristics of such image compression hardware are: 1) flexibility in image resolution; 2) possibility of implementation on reconfigurable platform [i.e., field programmable gate arrays (FPGAs)] for easy upgrade; 3) limited usage of memory resources; 4) very high compression capability (i.e., over 100:1). The latter characteristic makes traditional compression techniques inadequate, unless severe degradation of the image is tolerable.

Manuscript received June 11, 2004; revised February 3, 2005. This paper was recommended by Associate Editor A. Tabatabai.

The authors are with the Department of Electronics, Computer Science and Systems, University of Calabria, Arcavacata di Rende-87036-Rende (CS), Italy (e-mail: p.corsonello@unical.it; perri@deis.unical.it; staino@deis.unical.it; lanuzza@deis.unical.it; cocorullo@deis.unical.it).

Digital Object Identifier 10.1109/TCSVT.2005.856925

In the past decade, the discrete cosine transform (DCT) has been the most popular transform for image compression because it provides good performance and can be hardware implemented at a reasonable cost [3]–[5]. However, as is well known, at high compression ratios image quality is very poor.

In the last few years, discrete wavelet transform (DWT) based image compression has emerged as a more efficient technique [6]. Nowadays, the JPEG2000 standard [7] is the best-known example of DWT-based intra-frame compression algorithm and synthesizable cores were recently made available [8]–[10]. Most of them can be implemented in FPGAs but, they require a large amount of resources and often do not reach the target performance either in terms of frame rate or in terms of image quality at high compression ratio.

In this context the lack of a design solution optimized for low-power high-performance FPGAs applications can be noted. The above reasons motivate our efforts in the design of a low-cost DWT-based architecture capable of reaching high throughput and high visual quality at low bit rate. To this aim three DWT-based coding algorithms have been analyzed as possible candidates: the embedded zero tree (EZT) [11], the embedded block coding with optimized truncation (EBCOT) [12] and the set partitioning in hierarchical trees (SPIHT) [13]. All these encoding algorithms produce embedded bit streams with a further benefit to progressive transmission. Both the EZT and EBCOT algorithms need a binary arithmetic encoding, which is a complex subsystem requiring a nonnegligible amount of logic and memory. On the contrary, the SPIHT algorithm does not actually need the arithmetic encoding subsystem. Thus it is expected to offer cheaper hardware solutions. Moreover, as demonstrated in several studies [13], [14] the SPIHT algorithm surpasses the EZT in terms of image quality especially at low bit rate.

For all the above reasons, the VLSI compressor core presented in this paper is based on the SPIHT algorithm. As detailed in the following, the compressor here proposed exploits an inherent capability to obtain the superior visual quality offered by the SPIHT algorithm to consistently reduce the hardware cost and consequently the power dissipation.

The whole system has been implemented on an XILINX VirtexII XC2V1000-4 FPGA device. Compression ratio ranges from 4:1 to 160:1 and throughputs up to 18 Mpixels/s are achieved in the worst case compression condition (i.e., 4:1 compression).

The paper is organized as follows. In Section II, the algorithms implemented for the wavelet transformation and the encoding process are described. Section III details the compressor hardware implementation, with major modules, the

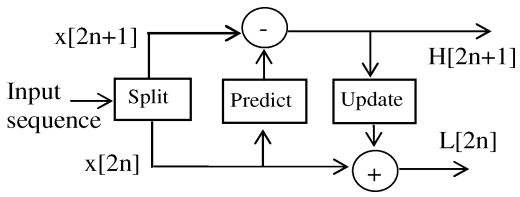

Fig. 1. Lifting scheme.

circuit used for the DWT and the encoder, being presented in order. Section IV describes the implementation in FPGAs and some experimental results are shown. Finally, conclusions are given in Section V.

II. Compression Algorithm

The hardware here presented is based on 5/3 integer filtering [15], [16] for the wavelet transformation and on a purpose modified version of the no list SPIHT (NLS) algorithm [17] for the subsequent encoding process. Most of the modifications introduced to the original algorithms have allowed a more efficient hardware implementation to be carried out.

A. Discrete Wavelet Transform (DWT)

Two conventional approaches are largely used to perform the DWT. The first one, known as the filterbank method [18], is based on convolution, whereas the second approach is lifting based filtering [19]. The latter offers several advantages: it assures that faster implementations can be obtained with fewer resources than the filterbank method; it can operate using just the memory resources necessary to store the original signal; the inverse transform is easily obtained; it supports perfectly reversible integer-to-integer wavelet transformation. All the above-mentioned advantages suggest that lifting based filtering represents a good approach for hardware implementations of DWT. For this reason it has been chosen for the compressor core here presented.

As shown in Fig. 1, lifting consists of *split*, *prediction*, and *updating* steps. Typically, during the split phase, the N-sample input sequence x is partitioned into even and odd components. That is, the terms $x[2n]$ and $x[2n+1]$ are formed (with $n = 0, \ldots, (\mathrm{N}/2) - 2$ and n even). In this way, a new representation of the original sequence is obtained. The prediction phase is equivalent to a high-pass filtering applied to x from which detail coefficients $H[2n+1]$ are generated. On the contrary, the update phase is equivalent to a low-pass filtering applied to x and a subsampling from which the approximation coefficients $L[2n]$ are generated.

When two-dimensional (2-D) signals, such as an $N \times N$ image, have to be wavelet transformed, the DWT can be performed by first applying the one-dimensional (1-D) filtering to the rows of the image (in the following this phase will be named the horizontal step), and then repeating the same filtering on its columns (in the following this phase will be named the vertical step). The horizontal step generates two $N \times N/2$ subbands L and H, which represent coarse and detail information, respectively, of the original image in the horizontal direction. On the contrary, the vertical step produces four $(N/2) \times (N/2)$ subbands: LL, LH, HL, and HH. When j levels of 2-D DWT are executed, the decomposition is then recursively repeated on the approximation subband LL, whereas detail subbands LH, HL, and HH are not processed anymore.

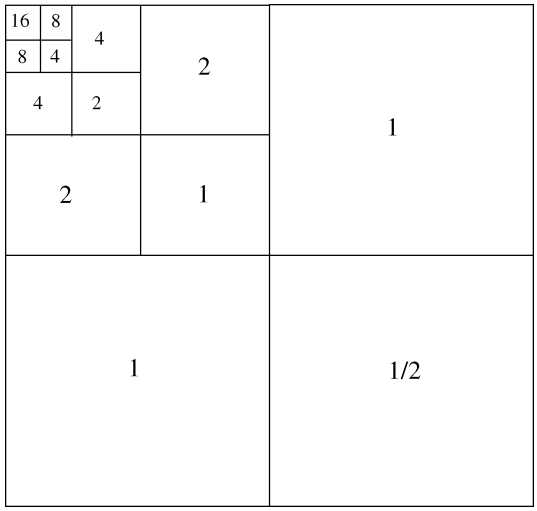

Fig. 2. Normalization factors for 4-level 2-D DWT.

Among the existing filters for the 2-D DWT, the circuit presented in this paper exploits the well known 5/3 integer filter described by (1). However, more complex or combined filters can be easily used inside the architecture [20]

$$H[2n+1] = \frac{1}{\sqrt{2}} \cdot \left(-\frac{x[2n]}{2} + x[2n+1] - \frac{x[2n+2]}{2} \right)$$
$$L[2n] = \sqrt{2} \cdot \left(-\frac{x[2n-2]}{8} + \frac{x[2n-1]}{4} + \frac{3 \cdot x[2n]}{4} + \frac{x[2n+1]}{4} - \frac{x[2n+2]}{8} \right) \quad (1)$$

It is important to underline that in a hardware implementation the factors $1/\sqrt{2}$ and $\sqrt{2}$ are not considered during the filtering. They can be applied as normalization factors after all wavelet transforms have taken place. It can be easily verified that in this way the final wavelet coefficients have to be multiplied by normalization factors that are powers of 2. As an example, Fig. 2 shows the normalization factors to use when four levels of DWT are performed. Thus, extracting the factors $1/\sqrt{2}$ and $\sqrt{2}$ from (1) and applying the lifting approach, the following is obtained:

$$H[2n+1] = x[2n+1] - \left\lfloor \frac{x[2n] + x[2n+2]}{2} \right\rfloor$$
$$L[2n] = x[2n] + \left\lfloor \frac{H[2n+1] + H[2n-1] + 2}{4} \right\rfloor. \quad (2)$$

As is well known the hardware implementations of DWT-based compression algorithms can require a large amount of memory resources, which increase with the image size. In order to reduce hardware resources, the tiling approach is typically used. It consists in partitioning the original $N \times N$ image into N^2/h^2 independent $h \times h$ portions (called tiles). In this way each tile can be elaborated independently of each other and efficient low-cost circuits can be realized. Moreover, the tiling approach allows different compression ratios to be used on different portions of the original image, thus making the region

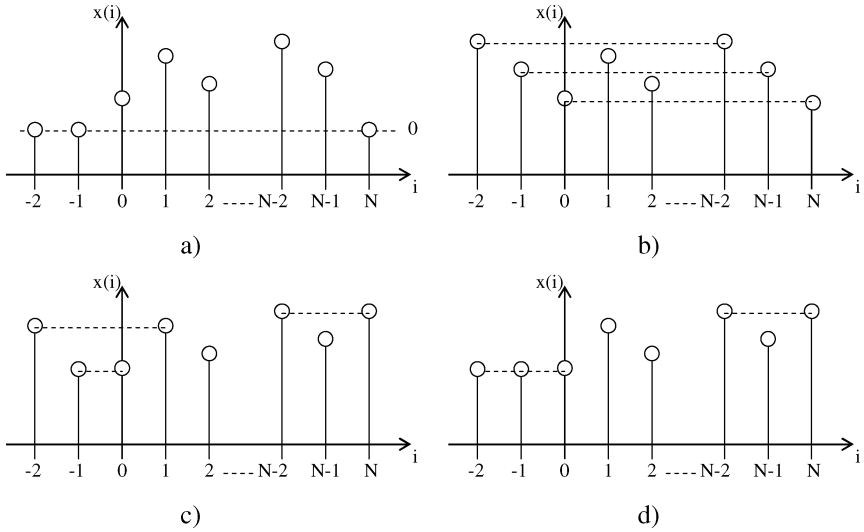

Fig. 3. Border extension techniques. (a) Padding with zeros. b) Periodic extension. c) Symmetric extension. (d) Method used here.

of interest (ROI) processing effective [21]. On the other hand, tiling an image and processing tiles separately introduce tiling artifacts on the reconstructed image. This is due to the fact that, at the boundary of the generic row/column, data out of the currently examined tile are needed in (2). In fact, when $2n = 0$, to generate $L[0]$ the term $H[-1]$ has to be computed. In order to do this, pixels $x(-1)$ and $x(-2)$ are required. In a similar way, when $2n = N - 2$, the pixel $x(N)$ is needed to compute $H[N - 1]$. Several choices can be exploited to set these pixels. The most popular are: padding with zeros; the periodic extension; the symmetric extension. In the compressor core here presented a mixed approach has been chosen for simplifying the hardware module needed for the boundary treatment. Fig. 3 shows the conventional choices and that used here [Fig. 3(d)]. Fortunately, efficient techniques can be used at the decoder side to significantly reduce the effect of these artifacts [22].

Hardware resources requirement could be further reduced discarding the subbands LH_1, HL_1, and HH_1 resulting from the first level of transformation [1]. Several purpose performed tests demonstrated that in this way a significant reduction is obtained in terms of memory resources and computational time. As discussed in the following Sections, for low compression factors (i.e., less than 30), the proposed approach leads to image quality lower than JPEG2000 compression algorithm. However, as demonstrated in Section IV, the visual quality of reconstructed images is not significantly compromised, since just details of the original images are neglected. On the contrary, for compression factors higher than 30, that is the target of this work, the implemented algorithm allows better reconstructed image qualities to be obtained.

B. Encoding Algorithm

In the pyramidal decomposition produced by the 2-D DWT a spatial self-similarity between subbands exists [13]. Consequently, as shown in Fig. 4, the wavelet coefficients $w(i, j)$, with $i = 0, \ldots, (N/2) - 1$, and $j = 0, \ldots, (N/2) - 1$, are organized in hierarchical trees, also called *spatial orientation tree*. Each node of the trees has either no offspring (e.g. the nodes

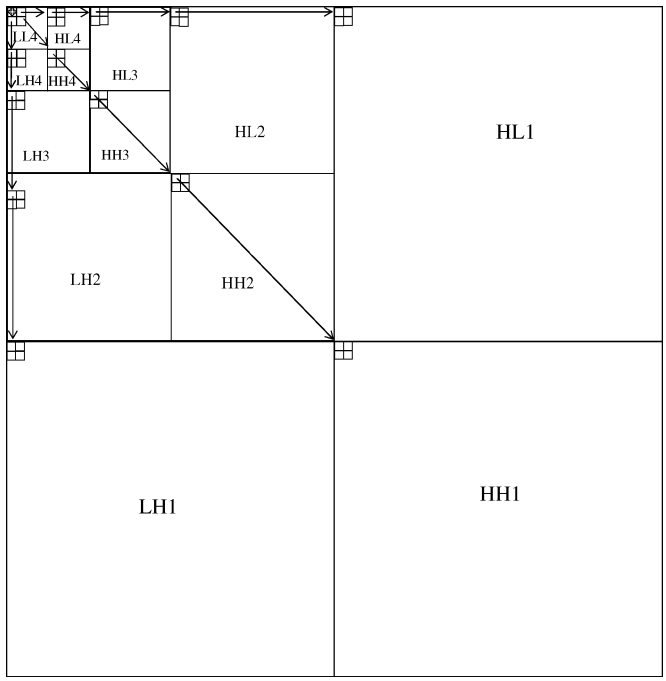

Fig. 4. Spatial orientation tree structure.

belonging to the subband LH_1) or four offsprings that form a group of 2×2 adjacent coefficients belonging to the same spatial orientation in the next finer level of the pyramid. The arrows in Fig. 4 are oriented from the parent node to its four offsprings. The wavelet coefficients in the subband LL_4 are the tree roots. In this structure, partitioning subsets are formed and information about them is encoded instead of information about the isolated coefficient. More explicitly, information to encode is obtained by comparing the magnitude of coefficients to a threshold value th. The latter is initially set to $2^{\lfloor \log_2[\max\{w(i,j)\}] \rfloor}$ and it is then iteratively divided by two. Magnitude tests flag which coefficients and subsets are significant or insignificant with respect to the current threshold value. The generic coefficient $w(i, j)$ is significant if it is greater than or equal to th, whereas a subset is significant if it contains at least one significant coefficient.

```
Initialization Phase
for n=0:1:N_DC-1
        mark[n]=MIP;                          /* mark each DC Band coeff. as insignificant
for n=N_DC:4:4N_DC-1
        mark[n]=MD;                           /* initial spatial orientation hierarchical trees
      push(n);                                /*define push(n)
                                                      mark[4n]=MN2; mark[16n]=MN3;Ö ..

Refinement Pass
1) n=0; while n<N^2                           /* start the Refinement scanning
2)      if mark[n]= MSP;                      /* significant coeff.
3)           output ( bit=val[n] AND th);     /* send the significance
4)           n=n+1;                           /* go to the next coeff.
5)      else
6)           n= n + skip(mark[n] );           /* skip the current set and go to the next coeff.
Insignificant Pixel Pass
7) n=0; while n< N^2                          /* start the Insignificant Pixel scanning
8)      if mark[n]= MIP;                      /* insignificant coeff.
9)           output(bit=val[n] AND th);       /* send the coeff. significance
10)          if bit=1                         /* if significant
11)            output( bit=sign[n]);          /* send the coeff. sign
12)            mark[n]= MSP;                  /* mark the coeff. as significant
13)          n=n+1;                           /* go to the next coeff.
14)         else
15)          n=n+skip (mark[n] );             /* skip the current set and go to the next coeff.
Insignificant Set Pass
16) n=0; while n< N^2                         /* start the Insignificant Set scanning
17)     if mark[n]= MD;                       /* set of descendants
18)          output(bit=dmax [ n/4] AND th);  /* send the set significance
19)          if bit=1                         /* if significant
20)             mark[4n]= MG;                 /*  locate the granddescendants set
21)             for j=n to n+3                /* for all its 4 children
22)                output(bit=val[j] AND th); /* send the coeff. significance
23)                if bit=1;                  /* if significant
24)                  output( bit=sign[j]);    /* send the coeff. sign
25)                  mark[j]= MSP;            /* mark the coeff. as significant
26)                else
27)                  mark[j]=MIP;             /* mark the coeff. as insignificant
28)             n=n+4;                        /* go to the next coeff.
29)          else
30)               n=n+4;                      /* go to the next descendants set
31)       elseif mark[n]= MG;                 /* set of granddescendants
32)          output(bit= gmax [n/16]AND th);  /* send the set significance
33)          if bit=1                         /* if significant
34)             j=n;    mark[j]= MD;  push(j);/* move current index n to an index j and use
35)             j=j+4; mark[j]= MD;  push(j); /* the new index to split the current set into
36)             j=j+4; mark[j]= MD;  push(j); /* four new sub-sets
37)             j=j+4; mark[j]= MD;  push(j);
38)          else
39)             n=n+16 ;                     /* go to the next granddescendants set
40)      else
41)        n=n+isskip (mark[n] );             /* skip the current set and go to the next coeff.
Threeshold Update
```

Fig. 5. Encoding algorithm.

As it is well known, several algorithms exist in which the above described approach is exploited to lead with efficient embedded encoding process [13], [17], [23]. Among them the NLS algorithm [17] has been referenced for the compressor core here proposed, because it is expected offering hardware implementations cheaper than other algorithms. As detailed in the following, several modifications have been purposely introduced in the original NLS algorithm to make its hardware implementation more approachable.

The pseudo-code of the encoding algorithm implemented for the elaboration of each discrete wavelet transformed tile is shown in Fig. 5. It is worth noting that, in order to make magnitude tests simpler, the sign-magnitude binary representation is assumed for the wavelet coefficients that are also supposed to be stored into a 1-D array in accordance with the Morton ordering [24]. As in the conventional NLS algorithm [17], for the generic coefficient two sets of descendants are defined: D and G. D is the set of all descendants of the referred coefficient, whereas G is the set of all descendants, the offsprings excluded. Each set consists of a proper number of generations depending on the executed number of transformation levels. For example, child and grandchild of a coefficient belong to the second generation and to the third generation, respectively, of the related D set. To efficiently encode the wavelet coefficients and their sets bit plane by bit plane (each bit plane is equivalent to a threshold value), state markers are defined. For the wavelet coefficients

which are insignificant or untested for the current bit plane the marker *MIP* is used. On the contrary, the marker *MSP* is assigned to significant coefficients. Three further markers are also used: *MD*, *MG*, and *MNy*. The former marks the first child into the generic D set; the second marker identifies the first grandchild into the generic G set; and finally *MNy* marks the first descendant at the yth generation of an insignificant set. In the above definitions, the term "*first*" means "*the first encountered during the scanning*." In other words, those referenced as "*first*" are the child, the grandchild or the descendant with the lowest index.

The markers *MNy* keep track of which wavelet coefficients are members of insignificant sets. Therefore, they are particularly useful for reducing the number of memory accesses. In fact, if a coefficient marked in this way is encountered during the scanning, all the $2^{2\cdot y}$ coefficients belonging to the $(y+1)$th generation of the related set can be skipped without requiring their examination. As an example, the presence of a $MN2$ coefficient implies that all the 16 coefficients belonging to the third generation of the related set can be skipped.

As detailed in Fig. 5, to properly define the initial markers values, an initialization phase is performed. During this phase, the wavelet coefficients in the LL_4 subband are marked as *MIP*, whereas the first child and first descendant at the yth generation of the generic set are marked as *MD* and *MNy*, respectively.

In order to start the encoding phase, the initial threshold and information about the maximum magnitude of coefficients belonging to the sets having the ith wavelet coefficient as the root must be pre-computed. In Fig. 5, this information is named $dmax[i]$ and $gmax[i]$ to indicate that they are related to the D and G sets, respectively, defined for the ith coefficient. These data can be stored into two separate one-dimensional arrays of size $N^2/4$ and $N^2/16$, respectively [17].

The implemented algorithm consists of three main steps: the *Refinement Pass (RP)*, the *Insignificant Pixel Pass (IPP)* and the *Insignificant Set Pass (ISP)*. It is worth pointing out that, the execution of the *Refinement Pass* as the first step allows the memory requirement to be reduced with respect to the case in which it is the final step [23]. In fact, in this way additional markers needed in the original NLS to indicate that a pixel that has newly become significant are eliminated.

In the *IPP*, only wavelet coefficients, which were insignificant in the previous iteration, are tested and their significance is outputted. If one coefficient of these becomes significant, it is marked as *MSP* and its sign is also outputted. On the contrary, a proper number of coefficients are skipped in accordance to the rules summarized in the first column of Table I. It is important to note that for testing the significance of the generic coefficient with respect to the current threshold, a simple bitwise logical *AND* between the coefficient magnitude (i.e., *val*) and the current threshold th is needed. In fact, for the wavelet coefficients the sign-magnitude representation is used.

During the *ISP*, for both D and G sets the significance with respect to the current threshold is checked. When a given set contains at least one significant coefficient, also the subsets included in it are tested for significance. Moreover, when significant coefficients are identified, their sign is outputted and they are marked as *MSP*. On the contrary, a proper number of coefficients can be skipped as ruled by the second column in Table I,

TABLE I
SKIPPING RULES

mark[n]	*skip*	*isskip*
MIP	1	4
MSP	1	4
MD	4	4
MG	16	16
MN2	16	16
MN3	64	64

which indicates how many coefficients can be skipped during the *ISP* depending on the current marker value. It is worth underlining that the above-described *ISP* is much simpler than the original NLS. In fact, the absence of markers needed to identify newly significant coefficients consistently simplifies the control logic. Obviously, it is expected that this also provides a simplification in a VLSI implementation.

After the threshold update, the *RP* progressively refines the wavelet coefficients previously marked as significant. The steps described above are then iterated until either all bit planes are scanned (i.e., a lossless compression is performed) or the desired bitrate is reached (i.e., a lossy compression is performed).

Comparing the bitstream produced by the original NLS algorithm to that generated by the above described algorithm, it can be observed that encoded information appear in a different order. This is due to the execution of the *RP* as the first step. This modification does not change the bitstream length, but it can have effects on the quality of the reconstructed images. In fact, it can be expected that, when truncating bitstreams at arbitrary rates, the information reordering can cause larger variations in the decoded pixel values. In order to examine these effects, the peak signal-to-noise ratio (PSNR) has been measured applying both algorithms on several known benchmark 128×128 images. In Figs. 6–8, the obtained results are summarized. It can be seen that, when the desired bitrate causes the whole encoding of all inspected bit planes, the compared algorithms lead to identical PSNR. On the contrary, when the desired bitrate causes a partial encoding of a bit plane (for example truncation happens at the end of the refinement pass), differences in the reconstructed image qualities are observed. Even though in the above tests the modified algorithm allows higher image qualities to be obtained with respect to the conventional one, this result does not guarantee that one method always performs better than the other.

III. HARDWARE IMPLEMENTATION

The top-level architecture ideated for the compressor core here proposed is illustrated in Fig. 9. The module used to perform the 2-D DWT receives two 8-bit image pixels at a time through the input lines *Datain_P1* and *Datain_P2*. The wavelet transformation starts when the signal *newtile* is asserted to flag that a new valid tile is ready for elaboration.

The resulting wavelet coefficients are stored in a dual-port memory. At the end of the 2-D DWT the signal *end_dwt* goes high, thus activating the encoder. The latter outputs the addresses *Addout_P1* and *Addout_P2* to read the wavelet

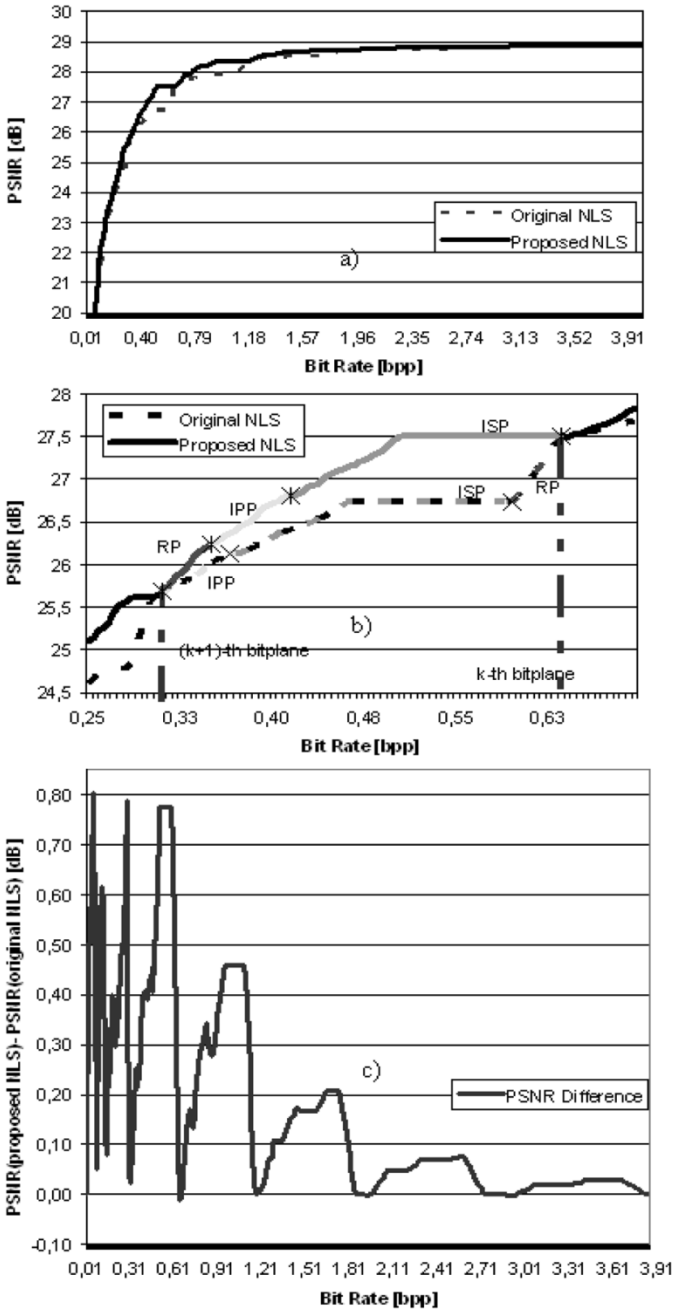

Fig. 6. Results obtained referring to the image *Lena*. (a) PSNR. (b) Zoom. (c) PSNR difference.

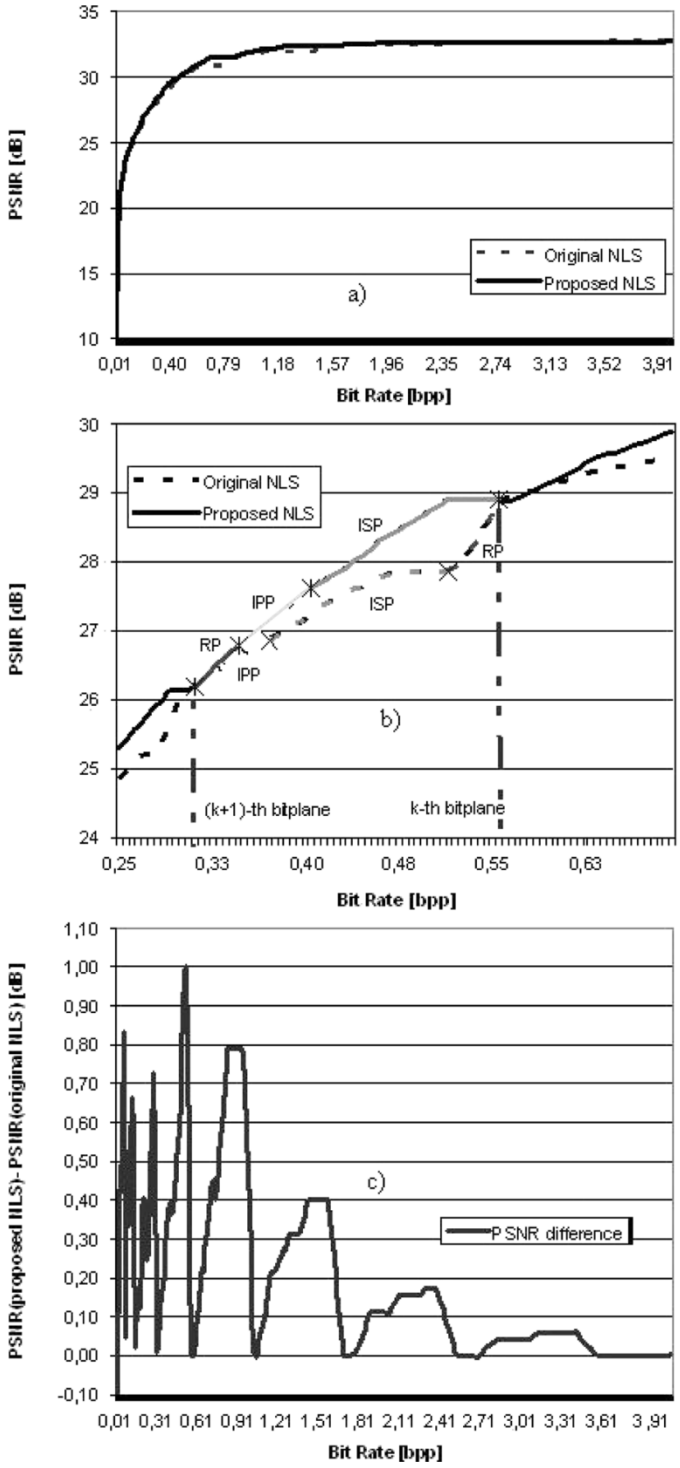

Fig. 7. Results obtained referring to the image *Barbara*. (a) PSNR. (b) Zoom. (c) PSNR difference.

coefficients that are transferred to the encoder through the data lines *Dataout_P1* and *Dataout_P2*.

The bitstream produced by the encoder is outputted one byte at a time through the data bus $byte[7:0]$. The signal *new_byte* is also generated to flag the validity of a new byte of the bitstream. The encoding process ends (i.e., the signal *end_spiht* is asserted) when the desired number of bytes, specified through the input bus *desired_byte*, and the number of produced bytes, specified by the bus *byte_count*, are equal.

For realizing the whole system, a purpose VHDL code has been written and then synthesized and optimized for the target FPGA platform. The used design flow and software tools are detailed in Fig. 10. As explained in Section IV, the Microblaze microprocessor and the Xilinx EDK software tool have been utilized just for testing purposes.

The two main modules of the realized compressor core are detailed in the following subsections.

A. Two-Dimensional DWT Circuit

The circuit purposely designed for the 2-D wavelet transformation is structured as depicted in Fig. 11. It can be seen that

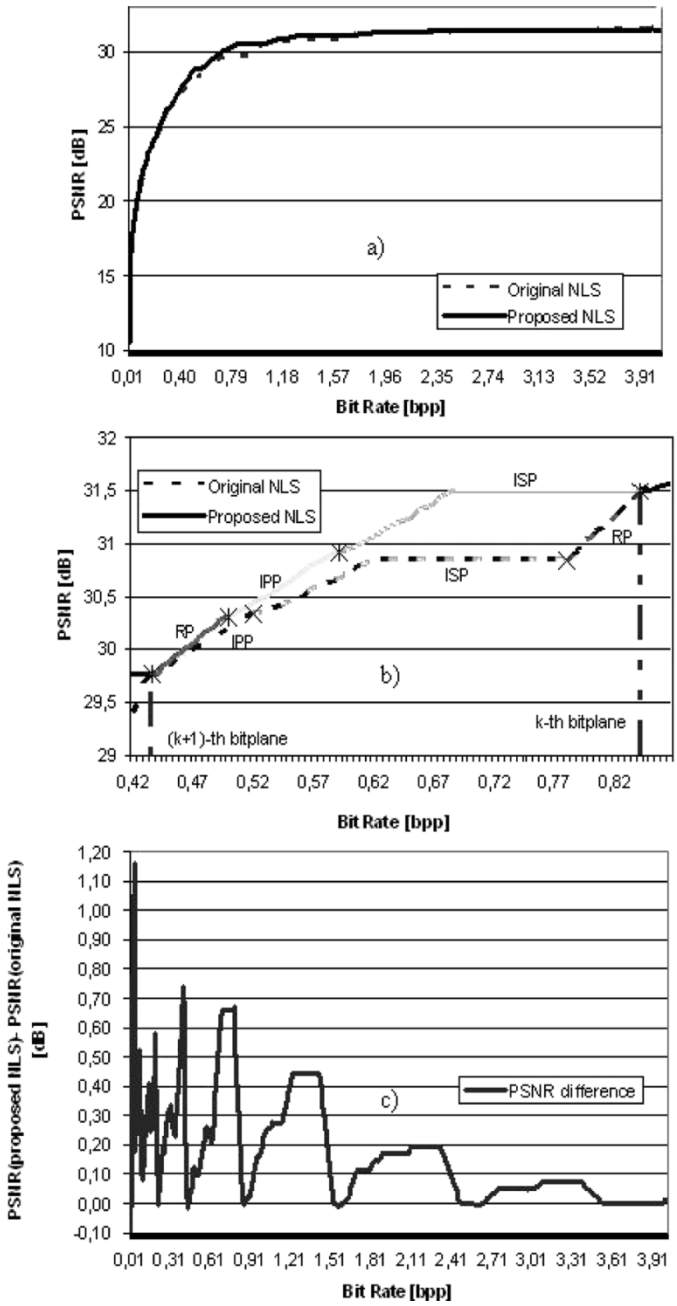

Fig. 8. Results obtained referring to the image *Gold Hill*. (a) PSNR. (b) Zoom. (c) PSNR difference.

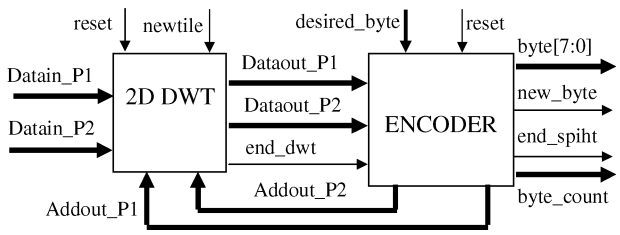

Fig. 9. Proposed compressor.

the circuit consists of two wavelet processors, Prow and Pcol, that perform the horizontal and the vertical steps, respectively; two dual-port memory banks, MEMa and MEMb, necessary to store the results coming from the processors; a control unit CU

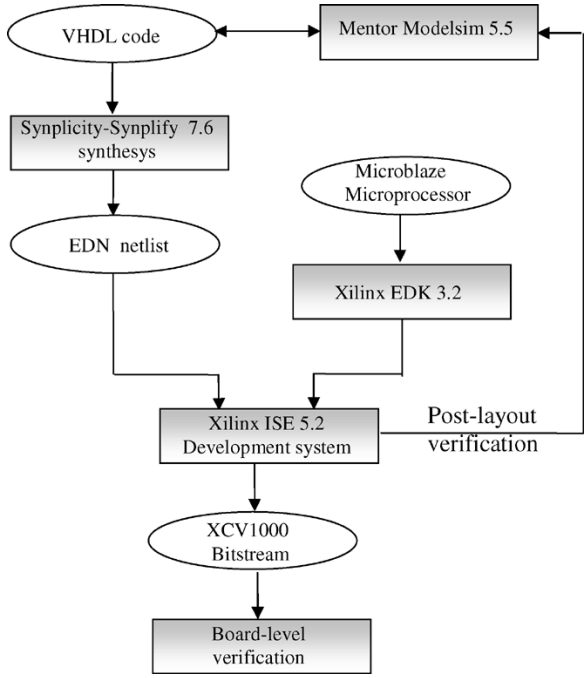

Fig. 10. Design flow.

that orchestrates the elaboration; and a module used for conversion and normalization, which, as better explained in the following, are needed to properly adjust the wavelet coefficients for the subsequent encoding process.

Each processor consists of a pipelined 16-bit data-path structured as depicted in Fig. 12. It can be noted that to implement (2), just additions, multiplexing and right shifting operations are needed in the data-path.

Observing Fig. 11, it can be seen that the splitting phase of the lifting based filtering is simply performed by simultaneously accessing two input data: *DATAIN_P1* and *DATAIN_P2*. The wavelet coefficients computed by the processor Prow during the horizontal step are stored into the memory MEMb in the locations indicated by the addresses *ADDINT_Mb_P1* and *ADDINT_Mb_P2*. MEMb is then accessed through two separate ports by the processor Pcol for the execution of the vertical step. The resulting wavelet coefficients are stored in the memory MEMa, which provides the input data for all the subsequent iterations of the 2-D DWT. On the basis of the currently executed iteration, the control unit sets the signal *Mux_input* that indicates if the processor Prow has to receive external data (i.e., the first level is executed) or it must operate on data stored in MEMa (i.e., the LL subband resulting from the previous level has to be elaborated). The final wavelet coefficients can be transferred from MEMa to the *DATAOUT_P1* and *DATAOUT_P2* output lines by sending the external address *ADDOUT_P1* and *ADDOUT_P2*. It is worth underlining that the processors used to perform the 2-D DWT operates in 2's complement fashion. Both the binary notation and the normalization of the wavelet coefficients resulting from the 2-D DWT have been properly managed.

In particular, normalization is implemented as discussed in Section II and taking into account that the subbands LH_1, HL_1

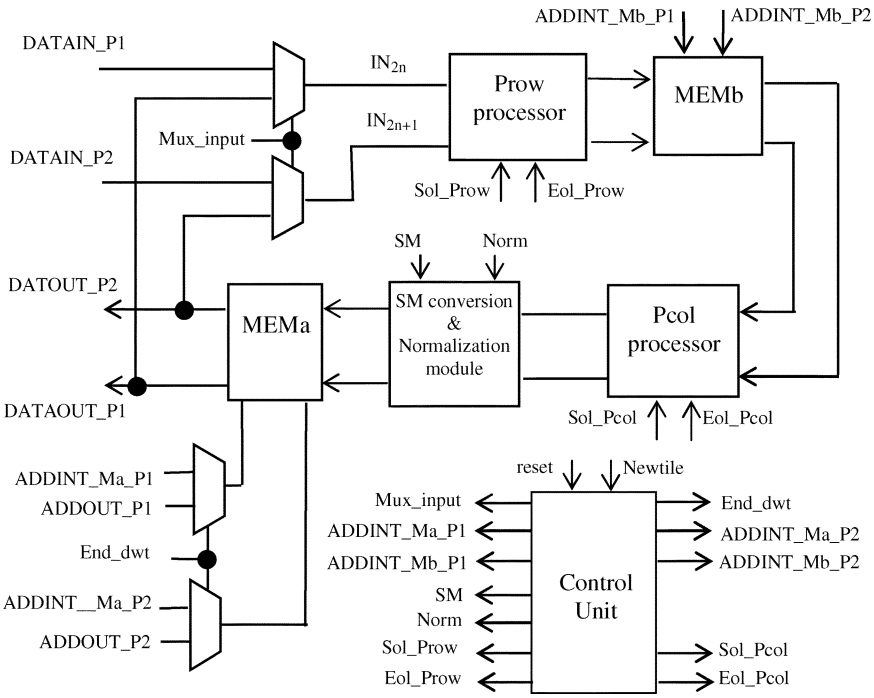

Fig. 11. Two-dimensional DWT circuit.

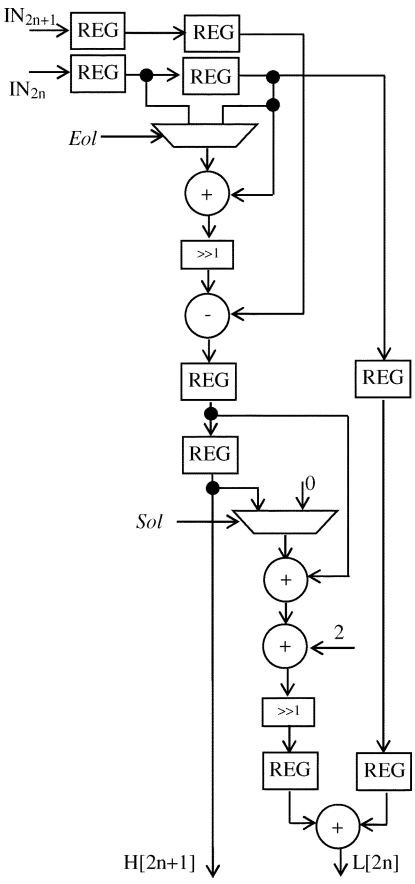

Fig. 12. Data-path in processors for 2-D DWT.

and HH_1 are discarded. Moreover, considering that the subsequent encoding step operates on wavelet coefficients in sign-module notation, conversion from 2's complement to sign-magnitude representation is also executed. Conversion and normalization operations are performed by the SM conversion&normalization module at the end of each vertical step just on the wavelet coefficients in the current subbands LH, HL, and HH.

The control signal *SM* indicates if the current coefficients belong to these subbands. This event is easily recognized by monitoring the signals *ADDINT_Ma_P1* and *ADDINT_Ma_P2* that are the addresses of memory locations of MEMa in which the current coefficients will be stored. These addresses depend on which subband the current coefficients belong to. Also the signal *Norm* is easily determined monitoring the above addresses. This signal represents the number of bit positions by which the current coefficients have to be left shifted for normalization.

The dual-port memories MEMa and MEMb store the wavelet coefficients in a row-based style. Furthermore, since the detail subbands LH_1, HL_1, HH_1 are discarded, for realizing MEMa and MEMb $N \times (N/2)$ and $(N/2) \times (N/2)$ 16-bit locations are used, respectively.

Memory accesses are controlled by the Control Unit that also manages the interface with external circuits, the boundary treatment and the coefficients conversion/normalization. The control unit is organized as depicted in Fig. 13 and runs as follows. A high-to-low transition of the signal *newtile* flags that the elaboration of a new tile can initiate. During the computation, the Top_FSM module communicates with the memory management unit (MMU) to set the signals *Currlevel* and *Dir*. They indicate which level of DWT is executed and in what direction (horizontal or vertical), respectively. This information is needed to establish which memory has to be accessed and to properly generate memory addresses. To this purpose the MMU is divided into two portions for separately controlling the accesses to the dual-port memories MEMa and MEMb. Each portion uses two programmable counters, GEN_Mb_P1 and GEN_Mb_P2 (GEN_Ma_P1 and GEN_Ma_P2), which generate the proper memory addresses, and a local finite state machine (FSM). The

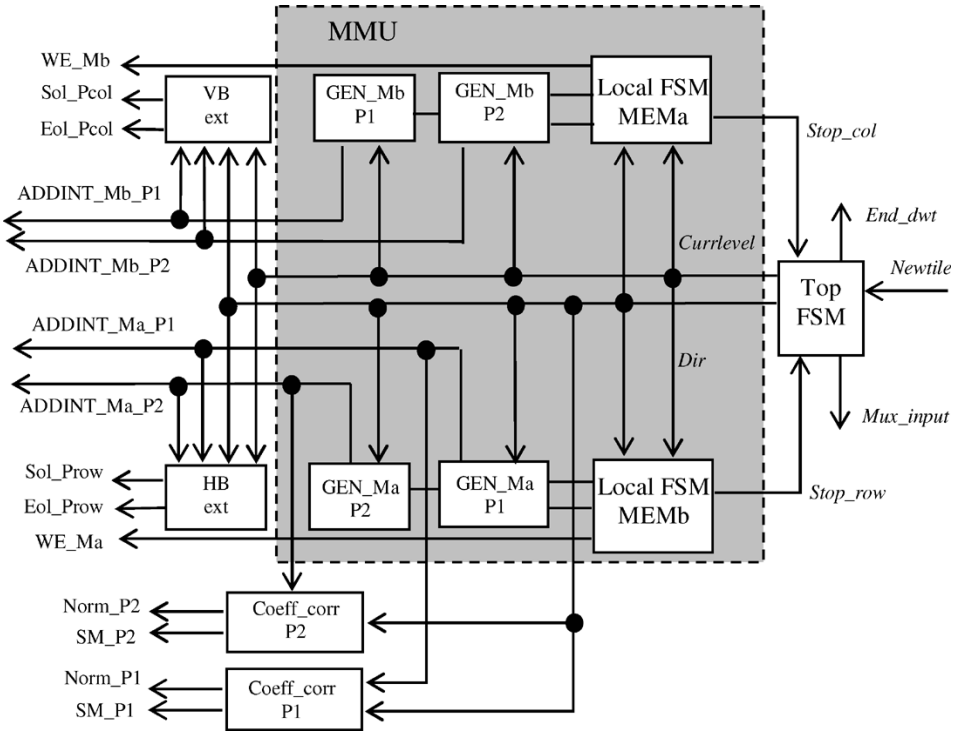

Fig. 13. Control unit.

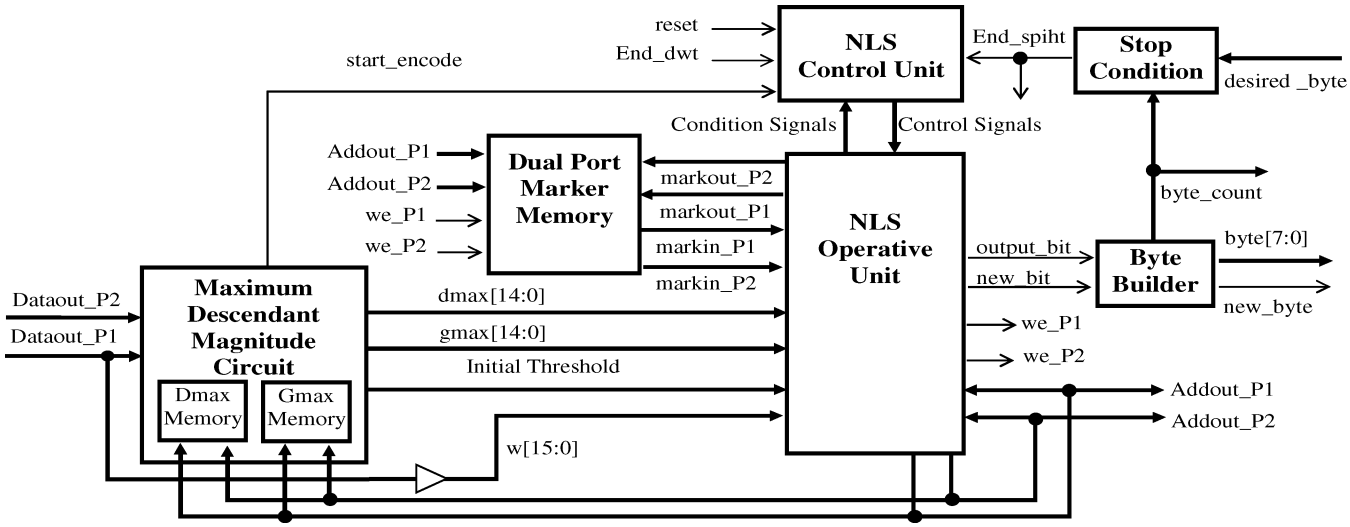

Fig. 14. Encoding circuit.

latter is needed to disable and reset the counters when required. On the basis of the memory addresses and the signal *Currlevel*, the modules HB_ext and VB_ext are activated, during the horizontal and the vertical step, respectively. These modules generate the control signals *Sol_Prow*, *Eol_Prow*, *Sol_Pcol* and *Eol_Pcol* that are used for establishing when boundary treatment is needed. In particular, the control signals *Sol_Prow* and *Eol_Prow* flag which pixel starts and ends the generic row, respectively. On the contrary, the control signals *Sol_Pcol* and *Eol_Pcol* flag which pixel starts and ends the generic column, respectively. The modules Coeff_corr_P1 and Coeff_corr_P2 depicted in Fig. 13 activate the conversion and the normalization of the wavelet coefficients belonging to the subbands LH, HL, and HH resulting from the current level of DWT. When all wavelet transforms have taken place, the signal *End_dwt* goes high to flag that the wavelet coefficients stored in the memory MEMa can be read for the subsequent encoding step.

B. The Encoder

Fig. 14 shows the top-level architecture purposely designed for hardware implementing the NLS algorithm described in Section II. The *maximum descendant magnitude circuit* (MDMC) reads the wavelet coefficients from MEMa in a depth-first search order and computes the initial threshold and the information about the maximum magnitude of coefficients

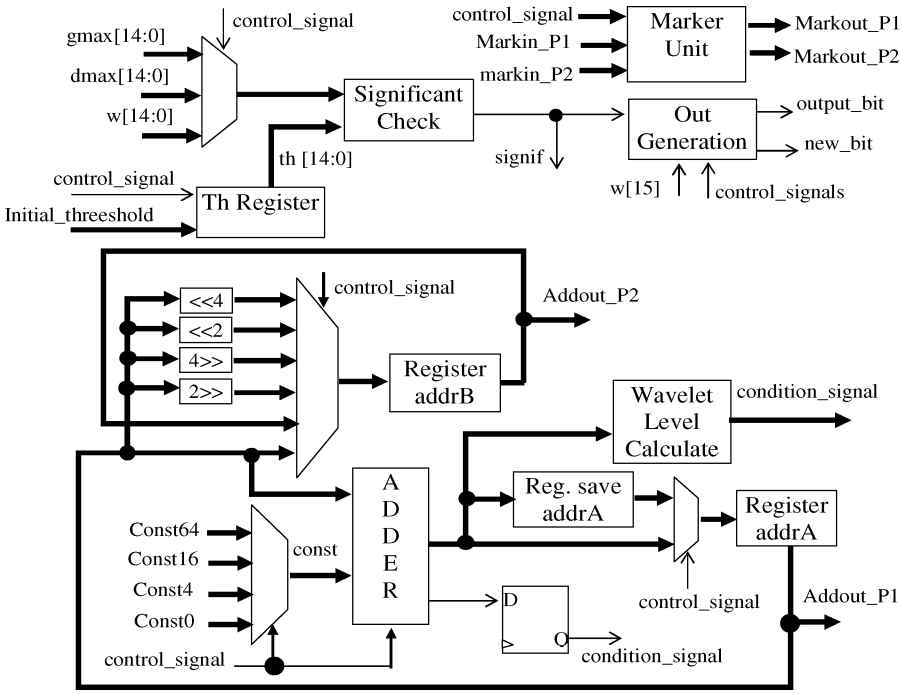

Fig. 15. NLS operative unit.

in each set. The latter are computed by a simple combinatorial circuit that implements the following:

$$g\max[n] = \mathrm{OR}\,(d\max[4n], d\max[4n+1], d\max[4n+2], d\max[4n+3]) \quad (3)$$

$$d\max[n] = \mathrm{OR}\,(val[4n], val[4n+1], val[4n+2], val[4n+3], g\max[n]) \quad (4)$$

with $gmax[n] = 0$ for $n \geq N^2/16$ and $0 \leq n \leq N^2/4$.

Obtained results are stored into the dual-port shared memories $Dmax$ and $Gmax$. When this phase ends, the signal *start_encode* is sent to the *NLS Control Unit*. The *NLS Operative Unit* receives the 16-bit wavelet coefficients in sign-magnitude representation, the 15-bit maximum magnitude data $dmax$ and $gmax$ and the 15-bit threshold as input. The bitstream obtained by applying the algorithm of Fig. 5 is outputted bit by bit through the *output_bit* line. The signal *new_bit* flags the production of a new valid compressed bit. The *Byte Builder Circuit* then rearranges the bitstream as a stream composed by a proper number of bytes (*byte_count*). The latter is then compared to the desired number of output bytes and the stop condition is determined. When the desired number of output bytes have been generated or all the thresholds have been examined the signal *end_spiht* is asserted. The desired number of output bytes $(dnoB)$ is easily obtained from the desired bitrate (dbr) applying (5), in which $N \times N$ 8-bit pixels original images are assumed

$$dnoB = \frac{N \times N}{8} \times dbr \quad (5)$$

The *NLS Operative Unit* is structured as illustrated in Fig. 15. When the *IPP* of the implemented algorithm is executed, the wavelet coefficients are tested for significance. During the *ISP*, either the D type sets or the G type ones are checked. A proper control signal coming from the *NLS Control Unit* establishes what input the circuit has to operate on. In any case, the *Significant Check Circuit* performs tests for significance using a simple two-level *AND-OR* combinational circuit which outputs the signal *signif*. This signal is then transferred as condition signal to the *NLS Control Unit* and as a new output bit to the *Output Generation Circuit*. The latter also receives the signal $w[15]$ (i.e., the sign bit of the currently checked coefficient) and appropriate control signals needed to establish whether the sign bit of the checked coefficient has to be also outputted (this occurs only when the coefficient is significant). The right-shift *Th Register* is initialized with the initial threshold produced by the MDMC. Thereafter, just before each *Refinement Pass* it is updated through a right shift. Based on the control signals coming from the *NLS Control Unit*, the memory addresses *Addout_P1* and *Addout_P2* are generated. They indicate the next coefficients and maximum magnitude information to be accessed for the encoding process. The next value of *Addout_P1*, used for accessing the memory MEMa and the marker memory, is obtained by adding to its current value the constant value const determined by applying the skipping rules shown in Table I. The *Save AddrA Register* is needed to save the current value of *Addout_P1* when required during the *Insignificant Set Pass* (line 34 of the pseudo-code in Fig. 5). The next value of *Addout_P2*, used for accessing $Gmax$ and $Dmax$ memories, is obtained by simple logical shift operations performed on the current value of *Addout_P2*. It is worth underlining that on the basis of *Addout_P1*, appropriate condition signals are generated to establish at which pyramid level the current wavelet coefficients belong and when the current encoding iteration ends (i.e., when all the tests for significance needed for the current threshold have taken place). This information is needed to prepare the encoding circuit for the next iteration (i.e., to update the threshold or to flag the stop condition). As is clear from the above descriptions, in the proposed circuits a strongly controlled process is performed. Thus, during the entire elaboration the *NLS Control Unit* plays

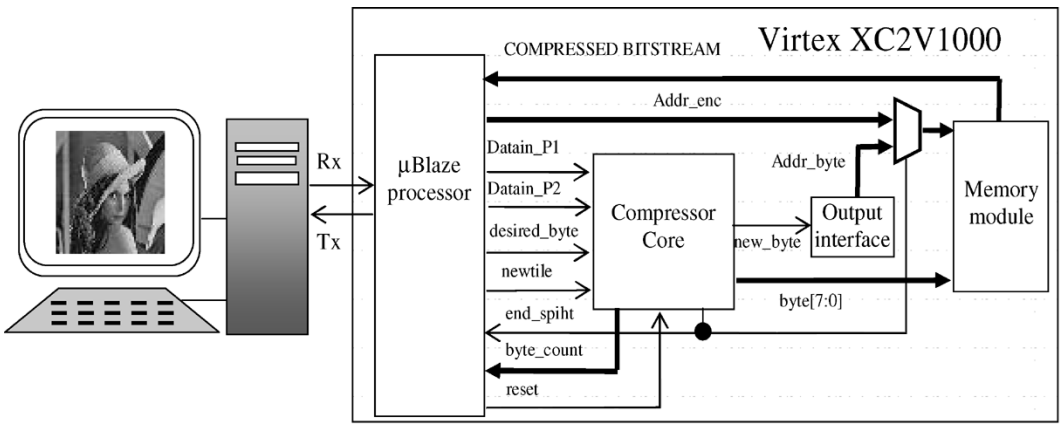

Fig. 16. Testing system.

a crucial role. It has implemented as a one-hot state machine and performs the following functions.

1) During the initialization phase, it establishes the initial marker values.
2) It ensures that memories are accessed in accordance with the Morton Scan order.
3) It controls the results obtained from the checks for significance and, based on marker values, establishes the next operation to execute.
4) It controls the threshold update.

IV. Results

Three implementations of the proposed architecture with different tile sizes (i.e., 64 × 64, 128 × 128, and 256 × 256) have been evaluated and compared to previous proposals. Furthermore, two different parallel implementations have been carried out. The first one consists of four encoders each elaborating 64 × 64 tiles, whereas the second parallel implementation uses two compression cores each operating on 128 × 128 tiles. All the proposed implementations use four levels of wavelet transformation and they exhibit a 100-MHz running frequency.

The number of transformation levels can be easily varied without hardware modifications.

A prototype testing system has been realized to perform hardware verification. The encoders have been synthesized on a XILINX Virtex XC2V1000-4 FPGA device, using the XILINX Integrated Software Environment (ISE) 5.2i, and tested through a Memec VirtexII V2MB1000-Rev3A board.[1] The architecture of the testing system is illustrated in Fig. 16. The system communicates with a host PC by means of a RS232 serial link. The host PC executes a Visual Basic [25] software routine to send tiles to the board through the serial RS232 interface, and to retrieve the compressed tiles. As shown in Fig. 16, the FPGA chip also contains a MicroBlaze soft processor[2] and an auxiliary memory module for storing the bit stream generated by the compressor. The soft processor is used as a top-level controller unit that interfaces the compressor core with the RS232 link. The host PC sends to the MicroBlaze information about the desired bitrate (i.e., the desired number of output bytes) and

[1][Online] Available: www.insight.na.memec.com/Memec/iplanet/link1.

[2][Online] Available: www.xilinx.com/ipcenter/processor_central.

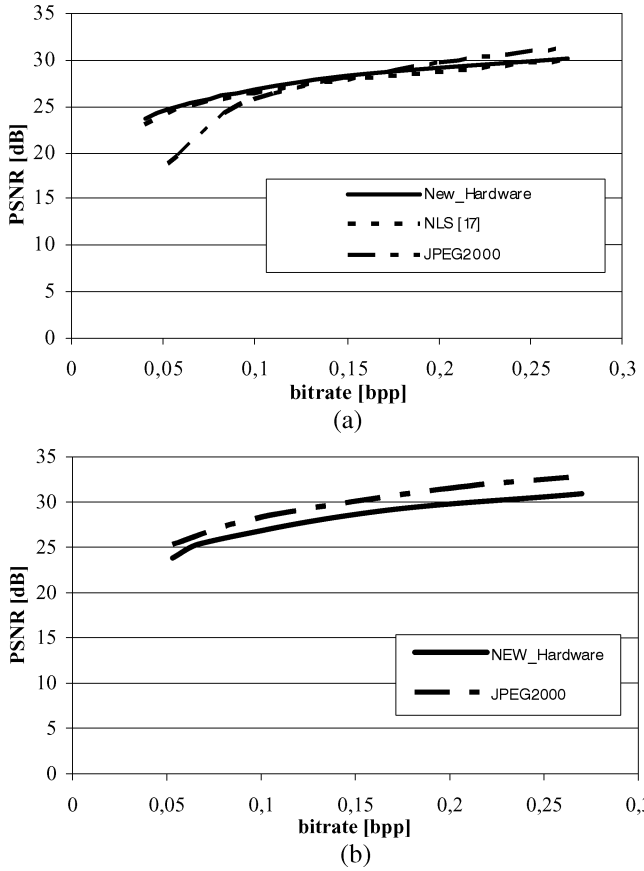

Fig. 17. Results obtained for the 512 × 512 Lena with (a) 64 × 64 tiles and (b) 128 × 128 tiles.

the number of tiles to compress. Each tile received through the serial Rx line is sent to the compressor core with the required control signals (*newtile* and *desired_byte*). The compressor core elaborates the current tile and then stores the compressed bitstream into the memory module. Finally, the MicroBlaze reads the resulting bitstream from the memory module and transmits it to the host PC through the serial line Tx. It is worth underlining that in order to manage read and write operations on the memory module the *Output interface* block is used.

When all compressed tiles have been transferred to the host PC, a software routine purpose implemented for the inverse

TABLE II
COMPARISON RESULTS

Type	*Tile Size*	*Slices*	*Bram or Ram size*	*Mult.*	*#Trans. or gates*	*Mpixels/sec.*	*Frequency*	*Technology*	*Algorithm*
Proposed	64x64	1035	6	-	-	4.6^	100MHz	VIRTEX II -4	SPIHT
Proposed	4×(64x64)	4500	24	-	-	**18.4^**	100MHz	VIRTEX II -4	SPIHT
Proposed	128x128	**1237**	**15**	**-**	**-**	7.2^	100MHz	VIRTEX II -4	SPIHT
Proposed	2×(128x128)	2362	30	-	-	14.4^	100MHz	VIRTEX II -4	SPIHT
Proposed	256x256	1350	53	-	-	12^	100MHz	VIRTEX II -4	SPIHT
[9]	128x128	14036	64	-	-	16 *	40.4 MHz	VIRTEX II**	JPEG2000
[8]	256x256	12935	31	9	-	10 *	106MHz	VIRTEX II -6	JPEG2000
[10]	128x128	6153	45	2	-	10.3 *	101MHz	VIRTEX II -6	JPEG2000
[31]	-	5984	10	19	-	N.A.	84 MHz	VIRTEX II -6	JPEG
[5]	-	-	23Kb	-	411k Trans	N.A.	27MHz	0.6um ASIC	JPEG
[29]	-	-	11.5Kb	-	50K gates	N.A.	30MHz	0.35um ASIC	JPEG

^ Worst case 2bpp * Declared typical condition **Speed grade not available

Fig. 18. Reconstructed images: compression factor 160:1, 64 × 64 tile, 4-levels 5-3 DWT.

DWT and the decoding process is executed to reconstruct and display the image. In order to reduce the tiling effect expected on the reconstructed images, the filter based approach proposed in [22] has been exploited.

Fig. 19. Reconstructed images: compression factor 4:1, 64 × 64 tile, 4-levels 5-3 DWT.

Many research papers claim to discuss hardware implementations of image encoding circuits. Although interesting, most of them just preliminarily describe works in the VLSI field [26] and only a few examples of hardware-verified encoders can be found in literature [5], [27], [28]. For this reason, comparison has been extended to commercial IP Cores whose behavior is described in the company datasheets [8]–[10].

In Table II characteristics of the encoders here proposed are summarized. Features of commercial encoders available for JPEG and JPEG2000 standards and of ASIC implementations available in literature for JPEG are also reported [5], [27], [29]. The implementation presented in [28] is not included in the comparison Table due to its very high resources and power requirements (it involves three different XILINX Virtex chips) even if the nine less significant bit planes of the wavelet transformed image are discarded.

As is evident from Table II, the highest throughput is obtained using the parallel implementation consisting of four cores each operating on 64 × 64 tiles. As already stated in the previous Sections, the proposed compression system has been optimized for compression ratios higher than 30:1. Performed tests demonstrated that it allows acceptable quality of decompressed images to be obtained for compression ratios up to 160:1. The main innovation is that such performances are obtained also using a small tile size (64 × 64) that implies the consistent reduction in memory usage required by the target application. Simulations of the JPEG2000 algorithm have been performed through the JPEG2000 VM 7.1 software version. For fair comparison, also for the JPEG2000 simulations, the 5/3 filter and four levels of wavelet transformation have been used. Obtained results demonstrated that using 64 × 64 tiles and the same filter for artifacts removal, decompressed images obtained by JPEG2000 compression algorithm are degraded for very low bit rates [see Fig. 17(a) and Fig. 18]. In Fig. 17, the proposed hardware solution is compared to the original NLS algorithm [17] and to the software JPEG2000 compression in terms of PSNR. The Lena 512 × 512 benchmark image has been referenced for this goal. Fig. 17(a) and Fig. 17(b) show the obtained PSNR for the cases in which 64 × 64 tiles and 128 × 128 tiles are used, respectively. It can be noted that, in the range of interest, the proposed hardware achieves an image quality at least similar to that attainable by JPEG2000 software.

In Fig. 18, some examples of reconstructed images for compression ratio 160:1 are reported. The JPEG2000 reconstructed images have been obtained with and without R/D optimization.

To clearly show the limit of the proposed approach we now discuss the counterparts of the adopted design choices. As described in Section II and as demonstrated in Fig. 17, the subbands elimination leads to image quality higher than or similar to that obtained by the JPEG 2000 software model for compression factors higher than 30. Outside the range of interest for our space applications, the penalty due to the subbands elimination is more evident. The lack of details corresponding to the neglected wavelet coefficients makes the proposed approach unsuitable for particular applications, such as the storage of medical images. Fig. 19 illustrates decompressed images at 2 bpp.[3]

Their visual qualities are still fair, even though evident PSNR's degradations are observable. The presence of many details in HH_1, HL_1, and LH_1 subbands of the Barbara wavelet transformed picture leads to the worst PSNR difference.

Data reported in Table II, also show that the proposed encoder using 128×128 tile size offers the best resource-performance tradeoff. In fact, when compared to the cheaper known core [10], it requires 79% less slices and 67% less memory and reaches a throughput rate of 7.2 Mpixels/s (i.e., only 30% less than the referred commercial core). Obviously, higher performances are reached by the parallel architecture in which two cores are used each elaborating 128×128 tiles. As shown in Table II, faster compressions are performed still saving a large amount of resources and power with respect to the existing solutions.

V. Conclusion

The DWT-based image compressors here presented offer appropriate image qualities also at very high compression ratios (over 100:1). Tests performed on a prototype FPGA-based testing system have proven that the proposed implementations are very efficient with respect to existing compressors at both commercial and research levels.

The proposed architecture is easily scalable so it can be adapted to several tile sizes. Moreover, parallel implementations can be efficiently realized to obtain higher speed with a reasonable increase in hardware resources requirement. For example, using four parallel cores each operating on 64×64 tiles, rates up to 18.4 Mpixels/s are reached while occupying just 4500 slices and 24 BRAM.

Acknowledgment

The authors would to thank the Associate Editor and the anonymous reviewers for their thorough revision and precious suggestions.

References

[1] N. Ismailoglu, O. Benderli, I. Korkmaz, M. Durna, T. Kolcak, and Y. C. Tekmen, "A real time image processing subsystem: GEZGIN," presented at the *16th Annual/USU Conf. Small Satellites*, Logan, UT, 2002.

[2] R. Manduchi, S. Dolinar, F. Pollara, and A. Macache, "Onboard science processing and buffer management for intelligent deep space communications," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, 2000, pp. 329–339.

[3] M. Kovac and N. Ranganathan, "JAGUAR: a fully pipelined VLSI architecture for JPEG image compression standard," *Proc. IEEE*, vol. 83, no. 2, pp. 247–258, Feb. 1995.

[4] W. Namgoong and T. H. Meng, "A low-power encoder for pyramid vector quantization of subband coefficients," *J. VLSI Signal Process.*, vol. 16, no. 1, pp. 9–23, May 1997.

[5] S. H. Sun and S. J. Lee, "A JPEG chip for image compression and decompression," *J. VLSI Signal Process.*, vol. 35, no. 1, pp. 43–60, Aug. 2003.

[6] M. Weeks and M. Bayoumi, "Discrete wavelet transform: architectures, design and performance issues," *J. VLSI Signal Process.*, vol. 35, no. 2, pp. 179–186, Sep. 2003.

[7] A. Skodras, C. Christopulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[8] Cast, Inc.. (2002, Oct.) Datasheet JPEG2k_E: JPEG 2000 Encoder Core. [Online]. Available: www.cast-inc.com

[9] Amphion Semiconductor Ltd. (2003, Jan.) Datasheet CS6510: JPEG 2000 Encoder. [Online]. Available: www.amphion.com

[10] Barco-Silex. (2004, Feb.) Datasheet BA112JPEG2000E: JPEG 2000 Encoder. [Online]. Available: www.barco-silex.com

[11] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[12] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[13] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[14] H. J. So, Y. J. Jung, J. S. Koh, and N. C. Kim, "Performance analysis and comparison of nonembedded wavelet coders," *IECE Trans. Inf. Syst.*, vol. E86-D, no. 6, pp. 1103–1109, Jun. 2003.

[15] D. Le Gall and A. Tabatabai, "Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, New York, 1988, pp. 761–765.

[16] P. Y. Chen, "VLSI implementation of lifting discrete wavelet transform using the 5/3 filter," *IEICE Trans. Inf. Syst.*, vol. E85-D, no. 12, pp. 1893–1897, Dec. 2002.

[17] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing ICASSP 2000*, vol. 6, Jun. 5–9, 2000, pp. 2047–2050.

[18] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge, 1996.

[19] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, 1996.

[20] G. Dillen, B. Georis, J. D. Legat, and O. Cantineau, "Combined line-based architecture for the 5-3 and 9-7 wavelet transform of JPEG2000," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 9, pp. 944–950, Sep. 2003.

[21] C. Christopoulos, J. Askelöf, and M. Larsson, "Efficient methods for encoding regions of Interest in the upcoming JPEG 2000 still image coding standard," *IEEE Signal Process. Lett.*, vol. 7, no. 9, pp. 247–249, Sep. 2000.

[22] J. Liang, C. Tu, and T. D. Tran, "Optimal pre- and post-processing for JPEG2000 tiling artifact removal," presented at the *37th Annu. Conf. Information Systems and Science (CISS 2003)*, Baltimore, MD, March 12–14, 2003.

[23] J. Ritter, G. Fey, and P. Molitor, "SPIHT implemented in a XC4000 device," in *Proc. 45th Midwest Symp. Circuits and Systems*, vol. 1, Aug. 2002, pp. 239–242.

[24] V. R. Algazi and R. R. Estes Jr, "Analysis based coding of image transform and subband coefficients," in *Proc. SPIE Visual Communications Image Processing Conf.*, San Jose, CA, 1995, pp. 11–21.

[25] *Microsoft Visual Basic 6.0 Programmer's Guide*, Microsoft Press, Redmond, WA, 1998.

[26] K. Andra, C. Chakrabarti, and T. Acharya, "A high-performance JPEG2000 architectured," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 3, pp. 209–218, Mar. 2003.

[3]It should be noted that, for the proposed compressor, 2 bpp represents the worst-case compression condition. Most often, due to subbands elimination, bit rate higher than 2 bpp cannot be reached.

[27] J. K. Hunter, J. V. McCanny, A. Simpson, Y. Hu, and J. G. Doherty, "JPEG encoder system-on-a-chip demonstrator," in *Proc. 33rd Asilomar Conf.*, 1999, pp. 762–766.

[28] T. W. Fry and S. Hauck, "Hyperspectral image compression on reconfigurable platforms," in *Proc. 10th Annu. IEEE Symp. Field-Programmable Custom Computing Machines*, Apr. 22–24, 2002, pp. 251–260.

[29] Cast, Inc. (2002, Oct.) Datasheet JPEG_Fast_C: High Performance JPEG Codec Core. [Online]. Available: www.cast-inc.com

Pasquale Corsonello (M'97) was born in 1964. He received the Master's degree in electronics engineering from the University of Naples, Naples, Italy, in 1988.

He joined the Institute of Research on Parallel Computer System (IRSIP), National Council of Research, Naples, Italy, working on the design and modeling of electronic transducer for high-precision measurement. In 1992, he joined the Department of Electronics, Computer Sciences and Systems of the University of Calabria, Rende, Italy, where he was involved in application specific integrated circuit design. From 1997 to 2004, he was with the University of Reggio Calabria, Reggio Calabria, Italy. Currently he is an Associate Professor of Electronics at the Department of Electronics Computer Science and System of the University of Calabria. His main research interests are in arithmetic circuit, reconfigureble computing and very large-scale integration design.

Stefania Perri (M'01) was born in 1971. She received the Master's degree in computer science engineering and the Ph.D. degree in electronics engineering from the University of Calabria, Rende, Italy, in 1996 and 2000, respectively.

In 1996, she joined the Department of Electronics, Computer Sciences and Systems of the University of Calabria, where she is currently an Assistant Professor of Electronics. Her current research interests include arithmetic circuit, asynchronous circuits, reconfigurable computing and very large-scale integration design.

Giovanni Staino was born in Cosenza, Italy, in 1970. He received the M.S. degree in computer science engineering and the Ph.D. degree in electronics engineering from the University of Calabria, Cosenza, Italy, in 1997 and 2003, respectively.

Since 2001, he has been with the Department of Electronics, Computer Sciences and Systems of the University of Calabria, where he is currently Research Assistant. His research interests include VLSI, signal processing, and CAD tools.

Marco Lanuzza was born in Reggio Calabria, Italy, in 1974. He received the M.S. degree in electronic engineering from the University "Mediterranea" of Reggio Calabria, Italy, 2000. He is currently working toward the Ph.D. degree at the Department of Electronics, Informatics and System, University of Calabria, Cosenza, Italy.

From 2001 to 2002, he was with Marconi Mobile, Genova, Italy, as Hardware Designer of Telecommunication Systems. His research interests include VLSI, low-power and high-speed arithmetic circuits, image compression, and signal processing.

Giuseppe Cocorullo (M'90) was born in 1952. He received the Dr. Eng. degree in electronics from the University of Naples, Naples, Italy, in 1978.

From 1983 to 1992, he was with National Council of Research, IRECE Institute, Naples, where he was in charge of the microelectronic department. Since 1992, he has been an Associate Professor of Electronics at the University of Calabria, Italy. Since 2000, he has been Full Professor of Electronics at the University of Calabria. His main research interest are in the fields of silicon optoelectronics and application specific IC design.