FPGA Implementation of Abundance Estimation for Spectral Unmixing of Hyperspectral Data Using the Image Space Reconstruction Algorithm

Carlos González, Javier Resano, *Member, IEEE*, Antonio Plaza, *Senior Member, IEEE*, and Daniel Mozos, *Member, IEEE*

***Abstract*—One of the most popular and widely used techniques for analyzing remotely sensed hyperspectral data is spectral unmixing, which relies on two stages: (i) identification of pure spectral signatures (endmembers) in the data, and (ii) estimation of the abundance of each endmember in each (possibly mixed) pixel. Due to the high dimensionality of the hyperspectral data, spectral unmixing is a very time-consuming task. With recent advances in reconfigurable computing, especially using field programmable gate arrays (FPGAs), hyperspectral image processing algorithms can now be accelerated for on-board exploitation using compact hardware components with small size and cost. Although in previous work several efforts have been directed towards FPGA implementation of endmember extraction algorithms, the abundance estimation step has received comparatively much less attention. In this work, we develop a parallel FPGA-based design of the image space reconstruction algorithm (ISRA), a technique for solving linear inverse problems with positive constraints that has been used to estimate the abundance of each endmember in each pixel of a hyperspectral image. It is an iterative algorithm that guarantees convergence (after a certain number of iterations) and positive values in the results of the abundances (an important consideration in unmixing applications). Our system includes a direct memory access (DMA) module and implements a pre-fetching technique to hide the latency of the input/output communications. The method has been implemented on a Virtex-4 XC4VFX60 FPGA (a model that is similar to radiation-hardened FPGAs certified for space operation) and tested using real hyperspectral data sets collected by the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) over the Cuprite mining district in Nevada and the Jasper Ridge Biological Preserve in California. Experimental results demonstrate that our hardware version can significantly outperform an equivalent software version, thus being able to provide abundance estimation results in near real-time, which makes our reconfigurable system appealing for on-board hyperspectral data processing.**

***Index Terms*—Abundance estimation, field programmable gate arrays (FPGAs), hyperspectral imaging, image space reconstruction algorithm (ISRA), spectral unmixing.**

Manuscript received August 03, 2011; revised September 22, 2011; accepted September 26, 2011. Date of publication November 15, 2011; date of current version February 29, 2012. This work was supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927, Hyperspectral Imaging Network (HYPER-I-NET), and also by the Spanish Ministry of Science and Innovation (AYA2008-05965-C04-02, AYA2009-13300-C03-02, TIN2009-09806).

C. González and D. Mozos are with the Department of Computer Architecture and Automatics, Computer Science Faculty, Complutense University of Madrid, 28040 Madrid, Spain.

J. Resano is with the Department of Computer Architecture, Polytechnic Center, University of Zaragoza, 50018 Zaragoza, Spain.

A. Plaza is with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, 10071 Cáceres, Spain (corresponding author, e-mail: aplaza@unex.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2011.2171673

I. Introduction

HYPERSPECTRAL imaging, also known as imaging spectroscopy, is a technique that has been widely used during recent years in Earth and planetary remote sensing [1]. It generates hundreds of images, corresponding to different wavelength channels, for the same area on the surface of the Earth. The concept of hyperspectral imaging originated at NASA's Jet Propulsion Laboratory in California, which developed instruments such as the Airborne Imaging Spectrometer (AIS), then called AVIRIS (for Airborne Visible Infra-Red Imaging Spectrometer [2]). This system is now able to cover the wavelength region from 400 to 2500 nanometers using 224 spectral channels, at nominal spectral resolution of 10 nanometers. As a result, each pixel (considered as a vector) collected by a hyperspectral instrument can be seen as a *spectral signature* or 'fingerprint' of the underlying materials within the pixel (see Fig. 1).

The number and variety of processing tasks in hyperspectral remote sensing is enormous [3]. However, one of the main problems in hyperspectral data exploitation is the presence of mixed pixels [4], which arise when the spatial resolution of the sensor is not enough to separate spectrally distinct materials [5]. Spectral unmixing is one of the most popular techniques to analyze hyperspectral data [6]. It comprises two stages: (i) identification of pure spectral signatures (endmembers) in the data [7], and (ii) estimation of the abundance of each endmember in each (possibly mixed) pixel [8].

A standard technique for spectral mixture analysis is *linear* spectral unmixing [7], [8], which assumes that the collected spectra at the spectrometer can be expressed in the form of a linear combination of endmembers weighted by their corresponding abundances (see Fig. 2). It should be noted that the linear mixture model assumes minimal secondary reflections and/or multiple scattering effects in the data collection procedure, and hence the measured spectra can be expressed as a linear combination of the spectral signatures of materials present in the mixed pixel [see Fig. 3(a)]. Although the linear model has practical advantages such as ease of implementation and flexibility in different applications [5], *nonlinear* spectral unmixing may best characterize the resultant mixed spectra for certain endmember distributions, such as those in which the endmember components are randomly distributed throughout the field of view of the instrument [4], [9]. In those cases, the

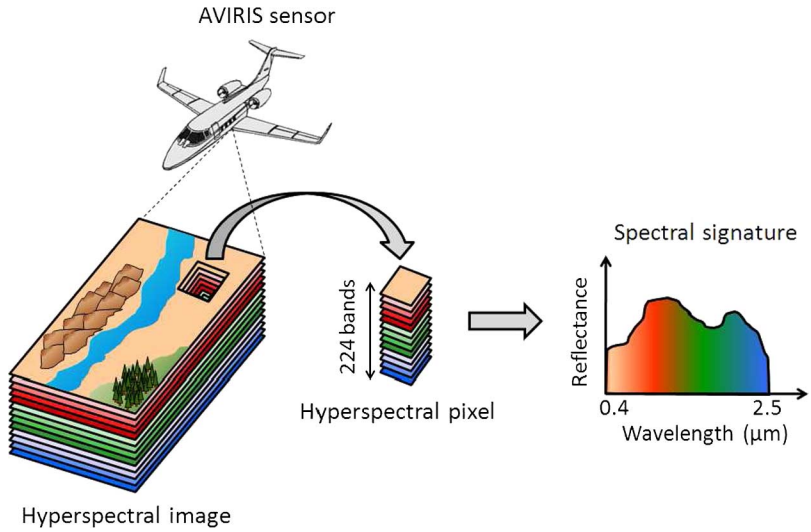

Fig. 1. The concept of hyperspectral imaging.

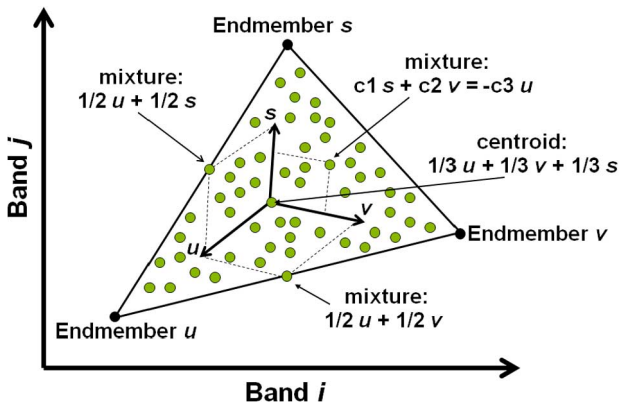

Fig. 2. Graphical interpretation of the linear mixture model.

mixed spectra collected at the imaging instrument is better described by assuming that part of the source radiation is multiply scattered before being collected at the sensor [see Fig. 3(b)]. Despite the fact that the nonlinear model may be more precise in certain circumstances, its proper application requires *a priori* information about the geometry and physical properties of the observed objects, making it difficult to address in situations where there is no such information. In the following, we focus on the linear model for simplicity.

Due to the high dimensionality of the hyperspectral data, spectral unmixing is a very time-consuming task [10]. With recent advances in reconfigurable computing, especially using field programmable gate arrays (FPGAs) [11]–[13], hyperspectral image processing algorithms can now be accelerated for on-board exploitation using compact hardware components with small size and cost. As the hardware design can be carried out at the register-transfer level (RTL), the emerging electronic design automation (EDA) tools allow design engineers to perform hardware/software co-design by transforming implementations developed in high-level programming languages into low level RTL designs in such a way that the design cycle can be greatly simplified. Therefore, as the hardware advances, the real-time implementation of the software and the subsequent on-board processing have become feasible goals in the context of remotely sensed hyperspectral imaging [14], [15]. FPGAs are now fully reconfigurable [16], [17], a technological feature that, in our application context, allows a control station on Earth to adaptively select a data processing algorithm (out of a pool of available algorithms implemented on the FPGA) to be applied on-board. The ever-growing computational demands of remote sensing applications can fully benefit from compact, reconfigurable hardware components and take advantage of the small size and relatively low cost of these units [18]–[20]. Although in previous work several efforts have been directed towards FPGA implementation of endmember extraction algorithms [21]–[26], the abundance estimation step has received comparatively much less attention in this kind of hardware platform.

In this work, we develop a parallel FPGA-based design of the image space reconstruction algorithm (ISRA) [27], a technique for solving linear inverse problems with positive constraints that has been used in hyperspectral imaging applications to estimate the proportion of each endmember in each pixel of a hyperspectral image [28], [29]. ISRA is one of numerous existing algorithms for abundance estimation in hyperspectral image data, including expectation-maximization maximum likelihood (EMML) [30], fully constrained least-squares unmixing (FCLSU) [8], and non-negative constrained least-squares unmixing (NNLSU) [31]. All these algorithms are computationally intensive and place a heavy burden on computing systems. The main distinguishing features of ISRA are its iterative unmixing nature [32], and the fact that it guarantees convergence (after a certain number of iterations) and positive values in the results of the abundances. This is an important consideration in unmixing applications, as the derivation of negative abundances (which is possible if an unconstrained model for abundance estimation is applied [5]) is not physically meaningful. Our main reasons for selecting the ISRA algorithm for hardware implementation over other abundance estimation algorithms are: (i) the fact that this algorithm provides physically meaningful abundance estimations, (ii) its iterative nature, which allows controlling the quality of the obtained solutions depending on the number of iterations executed, and (iii) the higher computational complexity of this algorithm as compared to other abundance estimation solutions, which demands fast implementations in order to be

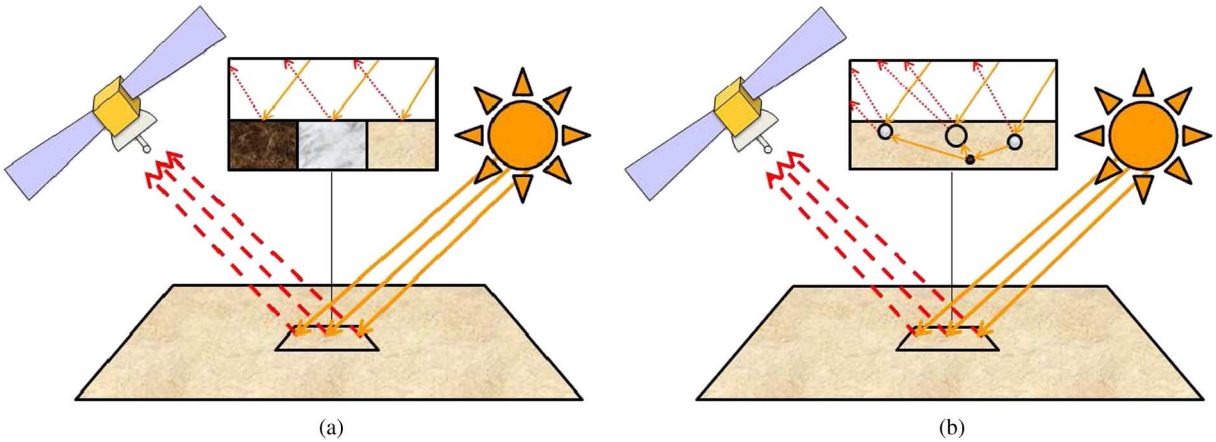

Fig. 3. Linear versus nonlinear mixture models: (a) single scattering versus (b) multiple scattering.

fully operational. Our parallel design to accelerate the ISRA algorithm has been implemented on a Virtex-4 XC4VFX60 FPGA (a model that is similar to radiation-hardened FPGAs certified for space operation) and tested using real hyperspectral data sets collected by AVIRIS.

The remainder of the paper is organized as follows. Section II formulates the spectral unmixing problem in mathematical terms and describes the ISRA algorithm. Section III describes its parallel implementation on a Xilinx Virtex-4 XC4VFX60 FPGA. Section IV provides an experimental assessment of both abundance estimation accuracy and parallel performance of the proposed FPGA-based implementation using well-known hyperspectral data sets collected by AVIRIS over two different sites: the Cuprite mining district in Nevada, and the Jasper Ridge Biological Preserve in California. Finally, Section V concludes with some remarks and hints at plausible future research lines.

II. Spectral Unmixing and the Image Space Reconstruction Algorithm (ISRA)

Let us assume that a hyperspectral scene with n bands is denoted by $\mathbf{I}$, in which a (possibly mixed) pixel of the scene is represented by an n-dimensional vector $\mathbf{x} = [x_1, x_2, \cdots, x_n] \in \Re^n$, where $\Re$ denotes the set of real numbers in which the pixel's spectral response x_i at sensor wavelengths $i = 1, \ldots, n$ is included. Under the linear mixture model assumption [6], each pixel vector can be approximated using the following expression:

$$\mathbf{x} \approx \sum_{j=1}^{p} \mathbf{e}_j \cdot \Phi_j + \mathbf{w} = \mathbf{E} \cdot \Phi + \mathbf{w} \tag{1}$$

where $\mathbf{e}_j$ denotes the spectral response of an endmember, Φ_j is a scalar value designating the fractional abundance of the endmember $\mathbf{e}_j$, p is the total number of endmembers, and $\mathbf{w}$ is a noise vector. It should be noted that $\mathbf{E} = \{\mathbf{e}_j\}_{j=1}^{p}$ can be seen as a $n \times p$ matrix. The solution of the linear spectral mixture problem described in (1) relies on the correct determination of a set of p endmembers and their correspondent abundance fractions $\Phi = \{\Phi_j\}_{j=1}^{p}$ at each pixel $\mathbf{x}$. A p-dimensional unconstrained abundance estimate in $\mathbf{x}$ can be simply obtained (in least squares sense) by the following expression [5]:

$$\hat{\Phi}^{\mathrm{UC}} = \left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T\mathbf{x}. \tag{2}$$

The main advantages of the unconstrained abundance estimation approach in (2) are: (i) the simplicity of its implementation, and (ii) its fast execution. However, under this unconstrained model the derivation of negative abundances is possible if the model endmembers are not pure or if they are affected by variability caused by spatial or temporal variations [7]. To address this issue, two physical constrains can be imposed into the model described in (1), these are the abundance non-negativity constraint (ANC), i.e., $\Phi_j \geq 0$, and the abundance sum-to-one constraint (ASC), i.e., $\sum_{j=1}^{p} \Phi_j = 1$ [8]. Imposing the ASC constraint results in the following optimization problem:

$$\min_{\Phi \in \Delta} \left\{ (\mathbf{x} - \Phi \cdot \mathbf{E})^T (\mathbf{x} - \Phi_j \cdot \mathbf{E}) \right\}, \quad \text{subject to} : \Delta = \left\{ \Phi \sum_{j=1}^{p} \Phi_j = 1 \right\}. \tag{3}$$

Similarly, imposing the ANC constraint results in the following optimization problem:

$$\min_{\Phi \in \Delta} \left\{ (\mathbf{x} - \Phi \cdot \mathbf{E})^T (\mathbf{x} - \Phi \cdot \mathbf{E}) \right\}, \quad \text{subject to} : \Delta = \{\Phi | \Phi_j \geq 0 \text{ for all } j\}. \tag{4}$$

As indicated in [8], a fully constrained (i.e. ASC-constrained and ANC-constrained) estimate can be obtained in least-squares sense by solving the optimization problems in (3) and (4) simultaneously. While partially constrained solutions imposing only the ANC have found success in the literature [31], the ASC is however, prone to strong criticisms because, in a real image, there is a strong signature variability [33] that, at the very least, introduces positive scaling factors varying from pixel to pixel in the signatures present in the mixtures. As a result, the signatures are defined up to a scale factor, and thus, the the ASC should be replaced with a generalized ASC of the form $\sum_{j=1}^{p} \xi_j \cdot \Phi_j = 1$, in which the weights ξ_j denote the pixel-dependent scale factors. What we conclude is that the non-negativity of the endmembers automatically imposes a generalized ASC. For this

reason, in this work we focus on solutions that do not explicitly impose the ASC constraint but only the ANC constraint.

A non-negative constrained least-squares (NCLS) algorithm can be used to obtain a solution to the ANC-constrained problem described in (4) in iterative fashion [31]. A successful approach for this purpose in different applications [34] has been ISRA [27], a multiplicative algorithm for solving NCLS problems. The algorithm is based on the following iterative expression:

$$\hat{\Phi}^{k+1} = \hat{\Phi}^{k} \left(\frac{\mathbf{E}^T \cdot \mathbf{x}}{\mathbf{E}^T \mathbf{E} \cdot \hat{\Phi}^{k}} \right) \tag{5}$$

where the endmember abundances at pixel $\mathbf{x}$ are iteratively estimated, so that the abundances at the $k+1$-th iteration, $\hat{\Phi}^{k+1}$, depend on the abundances estimated at the k-th iteration, $\hat{\Phi}^{k}$. The procedure starts with an unconstrained abundance estimation $\hat{\Phi}^{\mathrm{UC}}$ which is progressively refined in a given number of iterations. For illustrative purposes, Algorithm 1 shows the ISRA pseudocode for unmixing one hyperspectral pixel vector $\mathbf{x}$ using a set of $\mathbf{E}$ endmembers. For simplicity, in the pseudocode $\mathbf{x}$ is treated as an n-dimensional vector, and $\mathbf{E}$ is treated as a $n \times p$-dimensional matrix. The estimated abundance vector $\hat{\Phi}$ is a p-dimensional vector, and variable $iters$ denotes the number of iterations per pixel in the abundance estimation process. The pseudocode is subdivided into the numerator and denominator calculations in (5). When these terms are obtained, they are divided and multiplied by the previous abundance vector. It is important to emphasize that the calculations of the fractional abundances for each pixel are independent, so they can be calculated simultaneously without data dependencies, thus increasing the possibility of parallelization.

Algorithm 1 Pseudocode of ISRA algorithm for unmixing one hyperspectral pixel vector $\mathbf{x}$ using a set of $\mathbf{E}$ endmembers

```
// For a certain number of iterations
for (k = 0; k < iters; k++) {
    // For all endmembers
    for (j = 0; j < p; j++) {
        // For all bands
        for (i = 0; i < n; i++) {
            // Calculate the numerator of (5)
            numerator = numerator + E[i][j] * x[i];
            // Calculate the denominator of (5) using Φ̂
            from previous iteration (in the first iteration,
            Φ̂ = Φ̂^UC)
            for (s = 0; s < p; s++) {
                dot+ = E[i][s] * Φ̂[s];
            end for
            denominator+ = dot * E[i][j];
            dot = 0;
        } end for
        // Calculate the new Φ̂
        Φ̂[j] = Φ̂[j] * (numerator/denominator));
        numerator = 0;
        denominator = 0;
    } end for
} end for
```

In order to achieve the highest degree of parallelization and hardware reuse as possible, we have rewritten Algorithm 1 in order to make sure that each processing unit consumes the least possible resources. For this purpose, in Algorithm 2 the numerator is obtained by first calculating the dot-product between the pixel and the endmember, and then the result is multiplied by the previous abundance fraction. Subsequently, in order to calculate the denominator we use the same structure to obtain partial sums that will be accumulated until we obtain the final result. Finally, we divide the numerator by the denominator at each iteration. In the following section we develop an FPGA implementation of Algorithm 2.

Algorithm 2 Modified version of Algorithm 1 in order to achieve higher degree of parallelization

```
// For a certain number of iterations
for (k = 0; k <= iters; k++) {
    // For all endmembers
    for (j = 0; j < p; j++) {
        // For all bands
        for (i = 0; i < n; i++) {
            numerator = numerator + E[i][j] * x[i];
        } end for
        numerator = numerator * Φ̂[j];
        for (i = 0; i < n; i++) {
            for (s = 0; s < p; s++) {
                dot+ = E[i][s] * Φ̂[s];
            } end for
            denominator+ = dot * E[i][j];
            dot = 0;
        } end for
        // Calculate the new Φ̂
        Φ̂[j] = (numerator/denominator);
        numerator = 0;
        denominator = 0;
    } end for
} end for
```

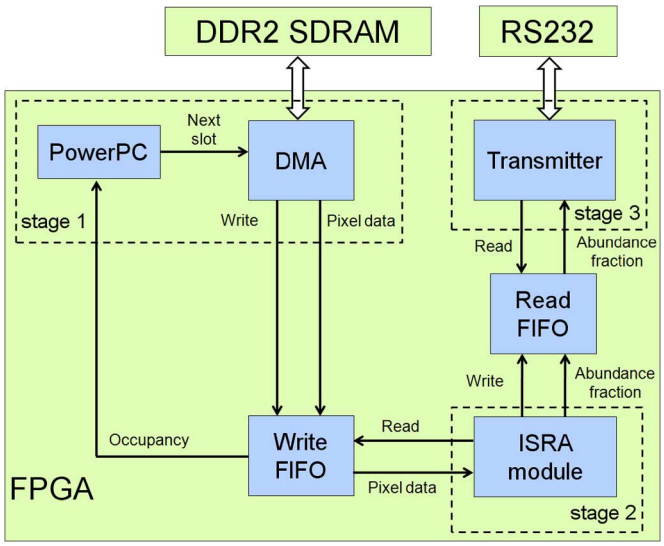

Fig. 4. Hardware architecture to implement the complete system.

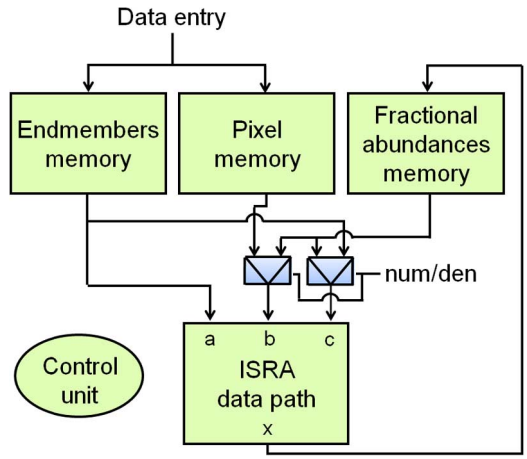

Fig. 5. Hardware architecture to implement the ISRA basic unit.

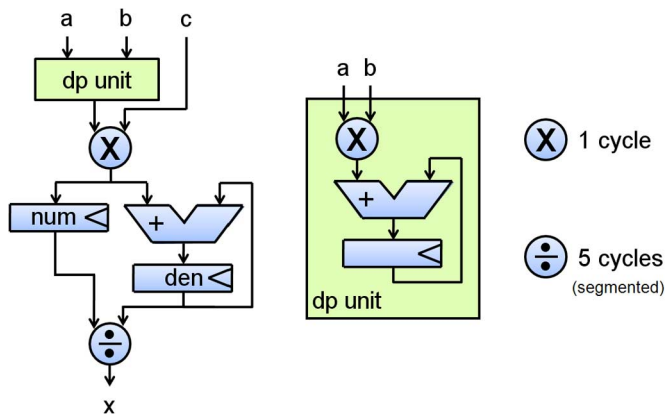

Fig. 6. Hardware architecture to implement the ISRA data path.

III. FPGA Implementation of ISRA

Fig. 4 shows the architecture of the hardware used to implement the ISRA algorithm, along with the input/output (I/O) communications. For data input, we use a double data rate (DDR2)-type synchronous dynamic random access memory (SDRAM) and a direct memory access (DMA) module (controlled by a PowerPC) with a first-in first-out (FIFO) unit to store pixel data. The ISRA module is used to implement our parallel version of the ISRA. Finally, a transmitter is used to send the fractional abundances via a RS232 port. The general structure of the hardware used to implement the ISRA can be seen as a pipelined architecture. We can distinguish three stages which are communicated using the FIFO structure: the first stage provides the necessary data for the system (endmembers and image data), the second stage carries out the abundance estimation process for each pixel, and finally the third stage sends such fractional abundances via a RS232 port. In our implementation these three steps work in parallel. In other words, while our design is sending some results, it is also computing the following results and reading the next data needed.

Fig. 5 shows the hardware architecture used to implement the ISRA basic unit. Three different memories are used to store the endmembers, the current pixel, and the fractional abundances for the current pixel, respectively. The ISRA data path represents the hardware architecture used to perform the calculations. The control unit carries out the ISRA execution: it reads the appropriate memory locations for each of the memories and updates the fractional abundances. In addition, we use a combinational circuit based on multiplexers to select the appropriate input data. Once calculated, the system writes the estimated abundances to the read FIFO.

Fig. 6 describes the architecture of the data path used to implement the ISRA basic unit following the design criteria discussed in the previous section. The dot-product unit is used for calculating both the numerator and denominator in (5), allowing a proper reuse of hardware resources. To perform the update of a fractional abundance value, it proceeds as follows: during n cycles (where n is the number of bands) it computes the dot-product between the current pixel and the endmember corresponding to the proportion of abundance that is being updated. In the next clock cycle, the result of the dot-product is multiplied by the previous abundance fraction and the result is stored in a register called `num`, thus concluding the calculation of the numerator. To calculate the denominator, the aforementioned procedure is repeated p times (where p is the number of endmembers) with the appropriate input data, while partial results are accumulated using an adder and the register `den`. The calculation of the denominator requires therefore $p \times (n+1)$ clock cycles. The process finalizes with the division between the numerator and the denominator in 5 clock cycles.

Since the fractional abundances can be calculated simultaneously for different pixels, the calculation of the ISRA algorithm can be parallelized as illustrated in Fig. 7. As we can see, the ISRA basic unit is replicated u times and we add two referees, one to read from the write FIFO and to send data to the corresponding basic unit or units, and the other to write the abundance fractions to the read FIFO. The number of times that we can replicate the ISRA basic unit is determined by the amount of available hardware resources and fixes the speedup of the parallel implementation. Since the FIFOs are now shared for all the ISRA basic units, a mechanism that guarantees that only one unit is using the FIFOs at each time must be incorporated. The

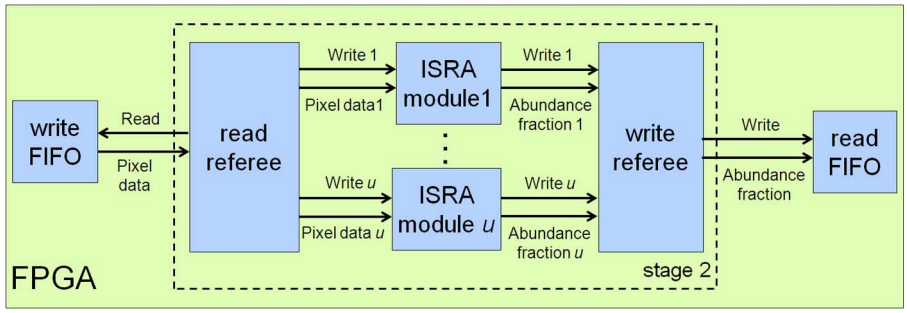

Fig. 7. Hardware architecture to implement the parallel ISRA module.

TABLE I
SUMMARY OF RESOURCES USED FOR THE FPGA IMPLEMENTATION OF THE ISRA DATA PATH IN THE VIRTEX-4 XC4VFX60

Component analyzed	Number of slice flip flops	Number of 4 input LUTs	Number of slices
`dp` unit	132	891	492
`num` or `den` register	32	0	0
Multiplier	33	167	88
Adder	0	538	301
Divisor	219	775	441
ISRA data path (with divisor)	458	2427	1367
ISRA data path (without divisor)	239	1652	929

most straightforward solution for this problem is to include a referee module that checks which units need to use the FIFOs and acknowledges the right to use one of the units following some criteria. At this point, it is important to clarify that introducing a referee for a large number of modules would not determine the critical path of the system. This is because for all pixels the number of iterations is the same, and hence the behavior of the referees are quite predictable since the readings from the write FIFO and the writings to the read FIFO will always be in order.

To reduce the amount of hardware resources needed to implement the ISRA module, we have analyzed the amount of hardware resources required for the ISRA basic unit in a specific hardware architecture (Xilinx Virtex-4 XC4VFX60) that will be described later on. Table I shows the results of this analysis. As we can see in Table I, the divisor required approximately one third of the available resources of the data path and yet is the component used the least time. Therefore, it is a very good candidate to improve resources consumption. To address this issue, we propose to have a single divisor shared by all ISRA basic units and meet the request of each ISRA basic unit using a referee. However, since the number of clock cycles elapsed between two divisions for the same ISRA basic unit, $c = 5 + (n + 1) + p \times (n + 1)$, is high compared with the number of read clock cycles n for the pixels (also, the time between the divisions for two consecutive basic units), if the number of ISRA basic units that can operate in parallel, u, satisfies: $u < LCM(c, n)/n$, we can ensure that the referee is not necessary because there will not be any collisions. If we reach or exceed this amount, the use of a referee is then essential. Fig. 8 shows a timing diagram with ISRA basic units in parallel that illustrates the collision phenomena.

To conclude this section, we provide a step-by-step description of how the proposed architecture performs the abundance estimation with a set of p endmembers for each pixel $\mathbf{x}$ from a hyperspectral image:

- Firstly, the PowerPC sends an order to the DMA to write the set of p endmembers, $\mathbf{E} = \{\mathbf{e}_j\}_{j=1}^{p}$, in the write FIFO.
- Later, the read referee reads these endmembers and sends them to all ISRA basic units where are stored in the endmembers memory.
- After the PowerPC has written the set of endmembers, it sends an order to the DMA to start copying a piece of the image from the DDR2 SDRAM to the write FIFO. The main bottleneck in this kind of system is frequently the data input, which is addressed in our implementation by the incorporation of a DMA that eliminates most I/O overheads. Moreover, the PowerPC monitors the write FIFO and sends a new order to the DMA every time that it detects that the write FIFO is half-empty. This time, the DMA will bring a piece of the image that occupies half of the write FIFO total capacity.
- When the first pixel data are written in the write FIFO, the read referee sends them to the first ISRA basic unit and it starts working. While storing the pixel data, we initialize the values of the fractional abundances to $1/p$, with the idea that it is a good starting point for this kind of iterative algorithm. Once the pixel is stored, the fractional abundances are estimated using the proposed ISRA procedure explained above, and finally the vector of estimated

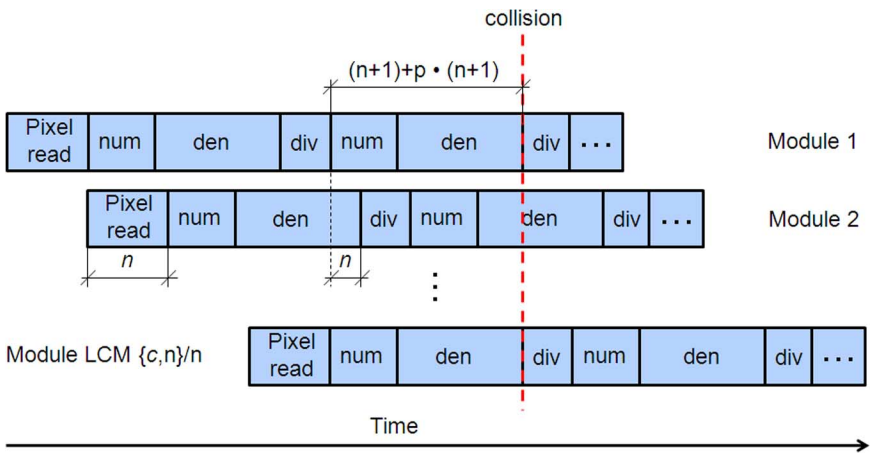

Fig. 8. Timing diagram with ISRA basic units in parallel.

abundances $\hat{\Phi}$ is written in the read FIFO through the write referee.

- Finally, the transmitter extracts the fractional abundance vector $\hat{\Phi}$ obtained for pixel $\mathbf{x}$ from the read FIFO and sends them via a RS232 port.
- As the next pixel data are written to the write FIFO, the read referee sends them to the appropiate ISRA basic unit when it finishes the execution with a previous pixel. Thus, the different ISRA basic units work in parallel. It should be noted that, except initially, there will not be any collisions in data transmissions to the different ISRA basic units and we will never have a conflict in the writing of the abundance values, mainly because in our implementation the number of endmembers p is always inferior to the number of bands n.

IV. Experimental Results

In this section we illustrate the unmixing accuracy and parallel performance of the proposed FPGA implementation. The section is organized as follows. In Section IV-A we describe the FPGA board used in our experiments. Section IV-B describes the hyperspectral data sets that will be used for demonstration purposes. Section IV-C evaluates the fractional estimation accuracy of the considered implementation. Finally, Section IV-D shows the resources used by our hardware implementation and further performs a comparison (in terms of computational performance) between our proposed FPGA design with an equivalent optimized software version.

A. FPGA Architecture

The hardware architecture described in Section III has been implemented using the VHDL language.[1] Further, we have used the Xilinx ISE environment[2] and the Embedded Development Kit (EDK) environment[3] to specify the complete system. The full system has been implemented on a ML410 board (see Fig. 9), a low-cost reconfigurable board with a single Virtex-4 XC4VFX60 FPGA component, a DDR2 SDRAM DIMM slot which holds up to 2GBytes, a RS232 port, and some additional components not used in our implementation. We use a Xilinx Virtex-4 XC4VFX60 FPGA because is based on the same architecture as other FPGAs [35] that have been certified by several international agencies for space operation. This FPGA is very close to the space-grade Virtex-4QV XQR4VFX60 FPGA so we could easily implement our design on it. We have followed a balance optimization goal in the mapping process of the design. We use the IP Core Generator to generate the floating point multiplier, adder and divider units with single precision (32 bits) following the IEEE 754 standard. These units limit the clock cycle, so as future work we are planning to implement our own floating point units.

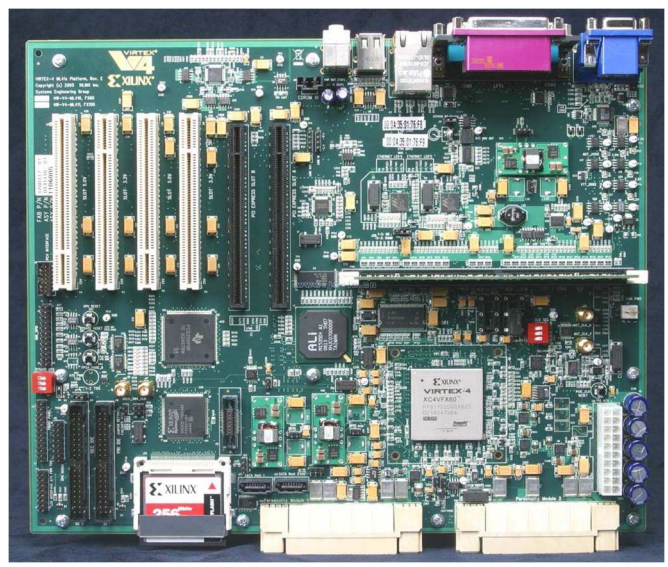

Fig. 9. Xilinx ML410 board with a Virtex-4 XC4VFX60 FPGA component.

B. Hyperspectral Image Data Sets

Several different hyperspectral data sets have been used in our experiments:

• The first one is the well-known AVIRIS Cuprite scene [see Fig. 10(a)], collected in the summer of 1997 and available online in reflectance units after atmospheric correction.[4] The portion

[1]http://www.vhdl.org.

[2]http://www.xilinx.com/support/download/index.htm.

[3]http://www.xilinx.com/ise/embedded/edk_pstudio.htm.

[4]http://aviris.jpl.nasa.gov.

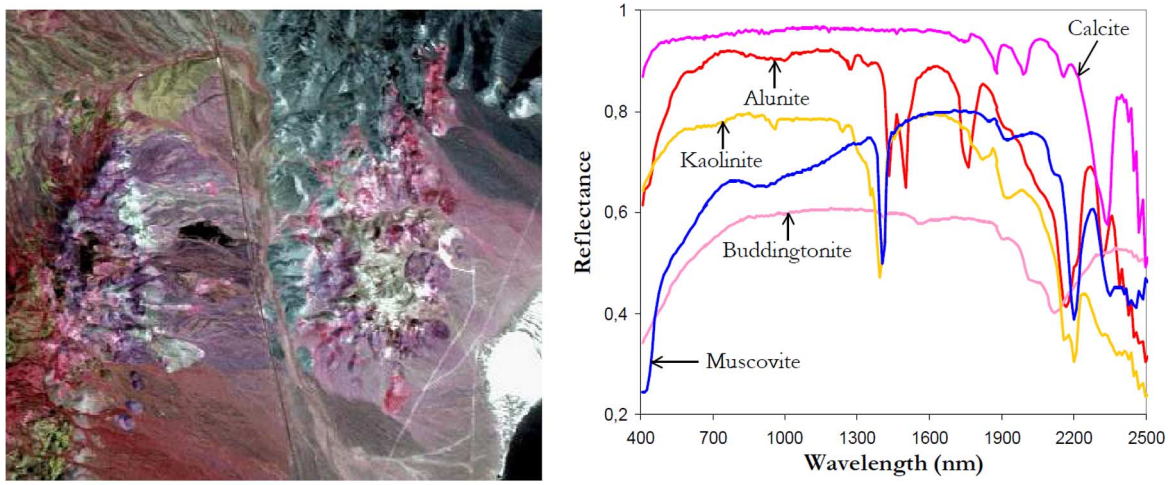

Fig. 10. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada. (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

used in experiments corresponds to a 350 × 350-pixel subset of the sector labeled as f970619t01p02_r02_sc03.a.rfl in the online data which comprises 224 spectral bands in the range from 400 to 2500 nanometers, and a total size of around 50 Megabytes. Bands 1–3, 105–115, and 150–170 were removed prior to the analysis due to water absorption and low SNR in those bands. The site is well understood mineralogically, and has several exposed minerals of interest including *alunite*, *buddingtonite*, *calcite*, *kaolinite* and *muscovite*. Reference ground signatures of the above minerals [see Fig. 10(b)], available in the form of a U.S. Geological Survey library (USGS)[5] will be used to estimate the fractional abundances in this work.

• Second, we have used a set of two AVIRIS images taken over the Jasper Ridge Biological Preserve in California. The datasets are available in both radiance (uncorrected) and reflectance (atmospherically corrected) units. Each of the data sets, acquired on April 1998, consist of 512 × 614 pixels and 224 spectral bands (for a total size of around 140 Megabytes each). Water absorption and low SNR bands were removed prior to the analysis. In a previous study of surface materials over this area, image endmembers were derived from the scenes above based on extensive ground knowledge [36]. Fig. 11 plots spectral signatures in radiance and reflectance units associated to the main constituent materials at Jasper Ridge. These signatures, corresponding to materials such as *soil*, *evergreen forest*, *dry grass*, *chaparral vegetation*, and *shade*, were obtained from the image scene by using a hybrid method combining visual inspection and prior information about the scene. The location of these materials is also identified in Fig. 11. Ground knowledge was used to identify homogeneous vegetation, shadow and soil areas in the scene. Inside those areas, representative pixels were selected as ground-truth spectra by comparing them to a spectral library of field data, used to represent landscape components at Jasper Ridge. In this process, we ensured that library spectra matched the phenology at the time of the image, and that there was little mis-calibration between field spectra and image spectra.

[5]http://speclab.cr.usgs.gov/spectral-lib.html.

C. Analysis of Abundance Estimation Accuracy

Before analyzing the parallel properties of the proposed implementation, we first conducted an experiment-based cross examination of the fractional abundances estimated by our proposed ISRA implementation using different hyperspectral scenes. First, we emphasize that our hardware version gives exactly the same results as our software implementation. Both implementations have been validated after testing them with a wide set of synthetic data with known results, in order to guarantee that both versions offer exactly the same results for the considered scenes. In this work, we assume that the spectral endmembers are known in advance. These endmembers comprise a set of USGS spectral library signatures (convolved to the image spectral bands) in the case of the AVIRIS Cuprite scene (available in reflectance units), and a set of image pixels labeled as spectrally pure in the case of the AVIRIS Jasper Ridge scene (available both in radiance and reflectance units). Based on these endmembers, we estimated the fractional abundances for each pixel in the considered scenes. The metric used for quantitative comparison in our experiments is the root mean square error (RMSE) between the original and the reconstructed hyperspectral scenes [7], which can be defined as follows. Let us assume that $\mathbf{I}^{(O)}$ is the original hyperspectral scene, and that $\mathbf{I}^{(R)}$ is a reconstructed version of $\mathbf{I}^{(O)}$, obtained using (5) with a set of endmembers and their corresponding estimated fractional abundances. Let us also assume that the pixel vector at spatial coordinates (i, j) in the original hyperspectral scene is given by $\mathbf{x}^{(O)}(i,j) = [x_1^{(O)}(i,j), x_2^{(O)}(i,j), \cdots, x_n^{(O)}(i,j)]$, while the corresponding pixel vector at the same spatial coordinates in the reconstructed hyperspectral scene is given by $\mathbf{x}^{(R)}(i,j) = [x_1^{(R)}(i,j), x_2^{(R)}(i,j), \cdots, x_n^{(R)}(i,j)]$. With the

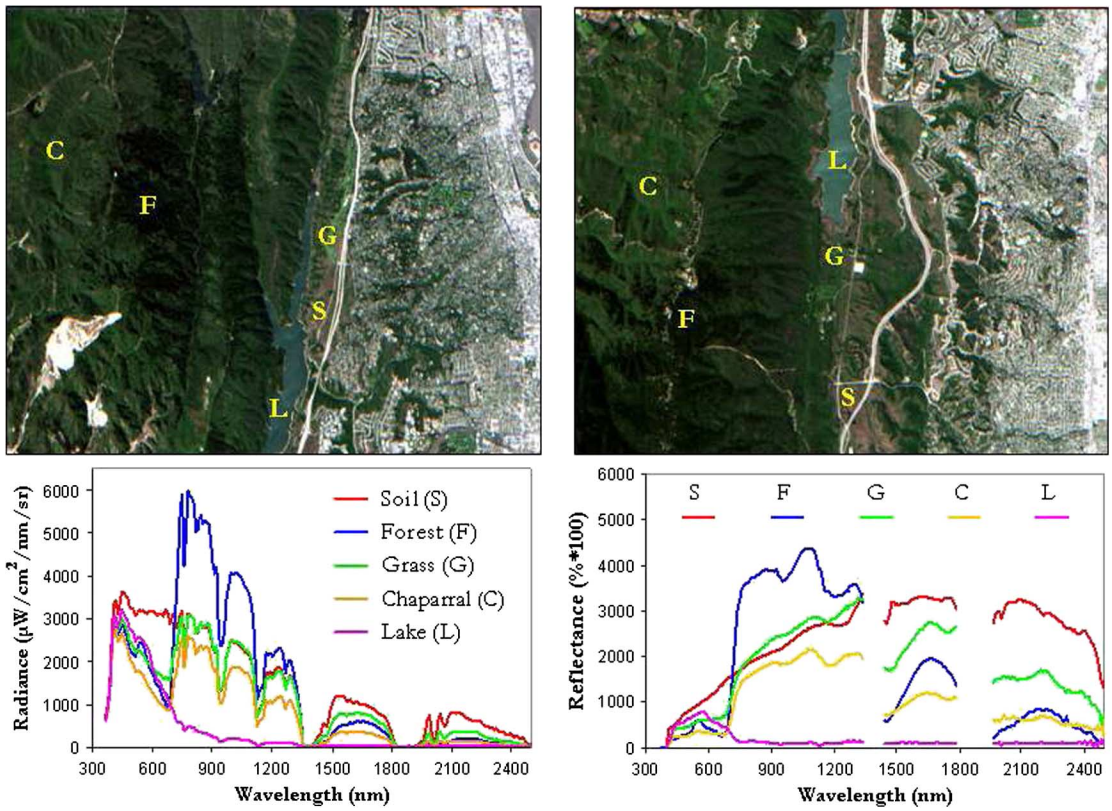

Fig. 11. AVIRIS hyperspectral images collected over Jasper Ridge Biological Preserve in radiance (left) and reflectance (right) units, along with the spectral signatures and spatial location of representative endmembers in the two considered scenes.

above notation in mind, the RMSE between the original and the reconstructed hyperspectral scenes is calculated as follows:

$$\text{RMSE}\left(\mathbf{I}^{(O)}, \mathbf{I}^{(R)}\right) = \frac{1}{s \times l} \sum_{i=1}^{s} \sum_{j=1}^{l} \left(\frac{1}{n} \sum_{k=1}^{n} \left[x_k^{(O)}(i,j) - x_k^{(R)}(i,j) \right] \right)^{\frac{1}{2}} \quad (6)$$

where s denotes the number of samples and l is the numbers of lines (hence, $s \times l$ denotes the total number of pixels). As a result, low RMSE scores mean high similarity between the compared images.

Figs. 12, 13, and 14, respectively, represent the per-pixel and overall RMSE (in the parentheses) between the original and the reconstructed hyperspectral scene for the AVIRIS Cuprite and AVIRIS Jasper Ridge (reflectance and radiance) scenes. It should be noted that the only input parameter used for ISRA (besides the set of endmembers and the input hyperspectral scene) is the number of iterations $iters$ in Algorithms 1 and 2. In our experiments, this parameter was varied between 10 and 600 to illustrate the convergence of the algorithm (we experimentally observed that, beyond 600 iterations, there were no variations in the obtained results). In the figures, warmer colors indicate greater error. From the results in Figs. 12, 13 and 14 we can conclude that, after 100-150 iterations the RMSE decreases very slowly, so that the observed improvements are quite small. It is noteworthy that the error in the reconstruction of the AVIRIS Cuprite scene is greater than the error for the Jasper Ridge AVIRIS scene. This is because in the first case the signatures used as endmembers are extracted from the USGS library collected on the field. Despite an atmospheric correction algorithm was applied to derive the AVIRIS Cuprite data in reflectance units, the ground signatures are free of atmospheric interferers and hence it is expected that higher reconstruction errors are observed in this case. On the other hand, in the AVIRIS Jasper Ridge images (both radiance and reflectance) we used pure spectral signatures directly derived from the hyperspectral images. As a result, the RMSE reconstruction errors are smaller in both cases.

D. Parallel Performance Evaluation

In this subsection we conduct an experimental evaluation of the computational performance of our proposed FPGA implementation. For illustrative purposes, Table II shows the resources used for our hardware implementation of the proposed ISRA design for different values of the ISRA basic units that can operate in parallel, u. The FPGA design was implemented on the Xilinx Virtex-4 XC4VFX60 FPGA of the ML410 board. This FPGA has a total of 25280 slices, 50560 slice flip flops and 50560 four input look-up tables (LUTs) available. In addition, the FPGA includes some heterogeneous resources such as two PowerPCs, 128 DSP48Es, and distributed block RAMs. In our implementation, we took advantage of these resources to optimize the design. One PowerPC monitors the communications and the block RAMs are used to implement the FIFO, so the vast majority of the slices are used for the implementation of the ISRA together with the DSP48Es multipliers.

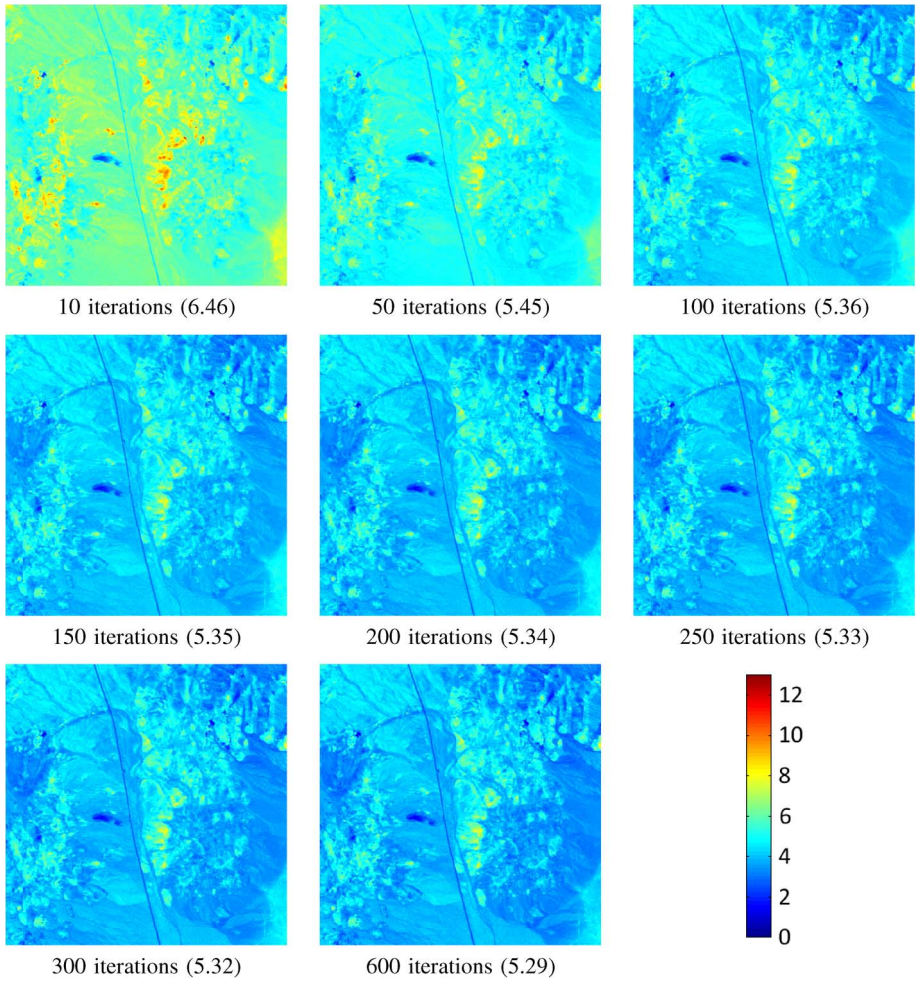

Fig. 12. Per-pixel and overall (in the parentheses) RMSE reconstruction errors for different number of iterations after reconstructing the AVIRIS Cuprite scene with the USGS reference signatures.

TABLE II
SUMMARY OF RESOURCE UTILIZATION FOR THE FPGA-BASED IMPLEMENTATION OF ISRA FOR DIFFERENT NUMBERS OF MODULES IN PARALLEL ON A VIRTEX-4 XC4VFX60 FPGA

Component	Number of modules (u)	Number of DSP48Es	Number of slice flip flops	Number of 4 input LUTs	Number of slices	Percentage of total	Maximum operation frequency (MHz)
Parallel ISRA module	10	100	3368	23357	12719	50.31	44.7
	11	110	3683	25654	13956	55.2	44.5
	12	120	3990	27920	15186	60.07	44.4
	13	128	4383	30394	16547	65.45	44.3
	14	128	4758	34300	18660	73.81	44.1
	15	128	5147	38112	20717	81.95	43.9
	16	128	5532	42047	22773	90.08	43.8
RS232 Transmitter	-	0	69	128	71	0.28	208
DMA Controller	-	0	170	531	367	1.45	102

As shown by Table II, our design can be scaled up to $u = 16$ ISRA basic units in parallel. An interesting behavior of the proposed parallel ISRA module is that it can be scaled without significant increase in the critical path delay (the clock frequency remains almost constant). For the maximum number of basic units used in our experiments ($u = 16$), the percentage of total hardware utilization is 90.08% thus reaching almost full hardware occupancy. However, other values of u tested in our experiments, e.g. $u = 10$, still leave room in the FPGA for additional algorithms. In any case, this value could be significantly increased in more recent FPGA boards. For example, in a Virtex-4 FPGA XQR4VLX200 (89,088 slices) certified for space, we could have 3.5 times more ISRA units in parallel simply synthesizing the second stage of the hardware architec-

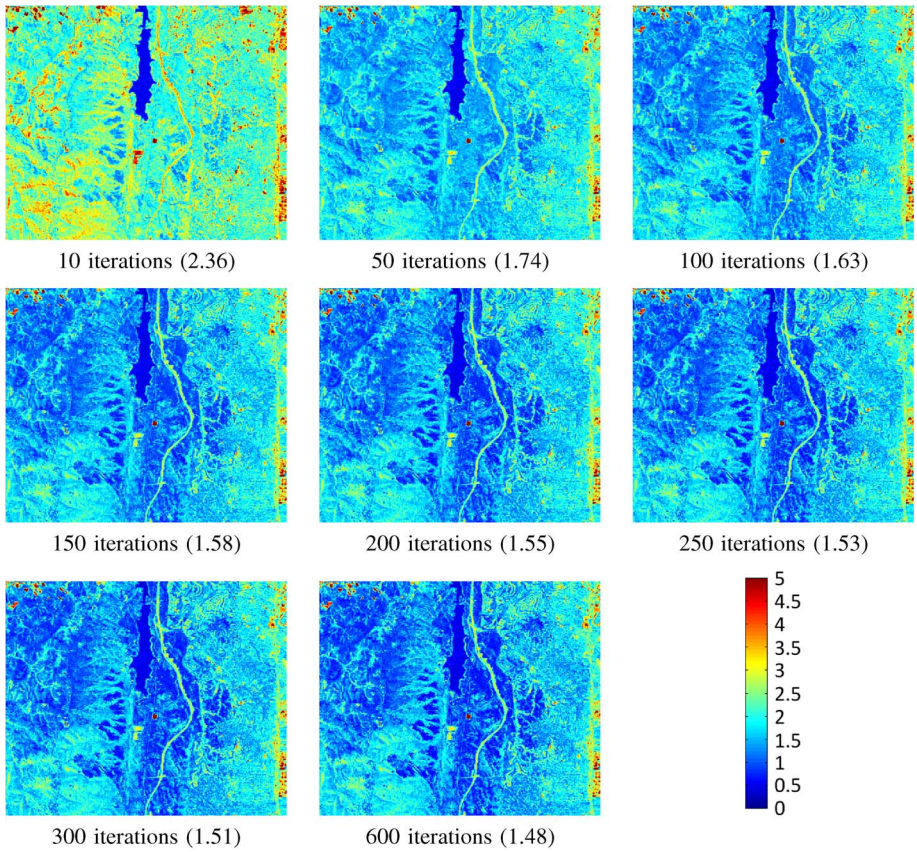

Fig. 13. Per-pixel and overall (in the parentheses) RMSE reconstruction errors for different number of iterations after reconstructing the AVIRIS Jasper Ridge scene in reflectance units with the image pixels labeled as spectrally pure.

ture for the new number of units in parallel. In this way, we could achieve a speedup of 3.5 without any modifications in our proposed design. In the case of an airborne platform without the need for space-certified hardware, we could use a Virtex-6 XQ6VLX550T (550000 logics cells) which has nearly 10 times more logic cells than the FPGA used in our experiments.

Another important aspect in our hardware implementation is the issue of communications, which are often the main bottleneck of a parallel system. We have paid special attention to this issue in our design. To reduce the I/O overheads, we have included a DMA and we have applied a pre-fetching approach in order to hide the communication latency. Basically, while the ISRA modules are processing a set of data, the DMA is fetching the following data set, and storing it in the write FIFO. Having in mind the proposed optimization concerning the use of available resources, it is important to find a balance between the number of DMA operations and the capacity of the destination FIFO. In other words, we need to fit enough information in the FIFO so that the ISRA modules are always busy. In addition, the greater the FIFO capacity the fewer DMA operations will be required. We have evaluated several FIFO sizes and identified that, for 1024 positions or more, there are no penalties due to reading of the input data. To demonstrate the advantages of using a DMA, we have developed another version in which the image data are read from memory and written to the write FIFO by the PowerPC instead of the DMA. In this version, the execution time was increased more than an order of magnitude so we can conclude that the resources used for the DMA (367 slices) are well spent.

TABLE III
EXECUTION TIMES (MINUTES) MEASURED FOR THE FPGA HARDWARE IMPLEMENTATION (HW) AND FOR AN EQUIVALENT SOFTWARE (SW) VERSION OF ISRA USING THE CONSIDERED HYPERSPECTRAL IMAGES AND DIFFERENT NUMBERS OF ITERATIONS

Number of iterations	AVIRIS Cuprite		AVIRIS Jasper Ridge	
	HW	SW	HW	SW
10	0.15	1.61	0.41	5.043
50	0.79	8.05	2.05	25.34
100	1.59	16.11	4.10	50.45
150	2.39	24.17	6.15	75.89
200	3.19	32.22	8.20	101.05
250	3.99	40.28	10.25	126.66
300	4.79	48.34	12.30	151.59
600	9.59	96.68	24.61	302.80

Table III reports the processing times achieved by the considered FPGA implementation with regards to an equivalent software version, developed in the C programming language (without math library optimization) and executed on a PC with a single AMD Athlon 2.6 GHz processor and 512 Mb of RAM. Although the 512 MB memory system may not be enough for many applications, it is enough for this case: the biggest image demands 135 MB and the abundance fractions 6 MB. Hence there is still enough space for the intermediate calculations and OS memory system. This version has been extensively

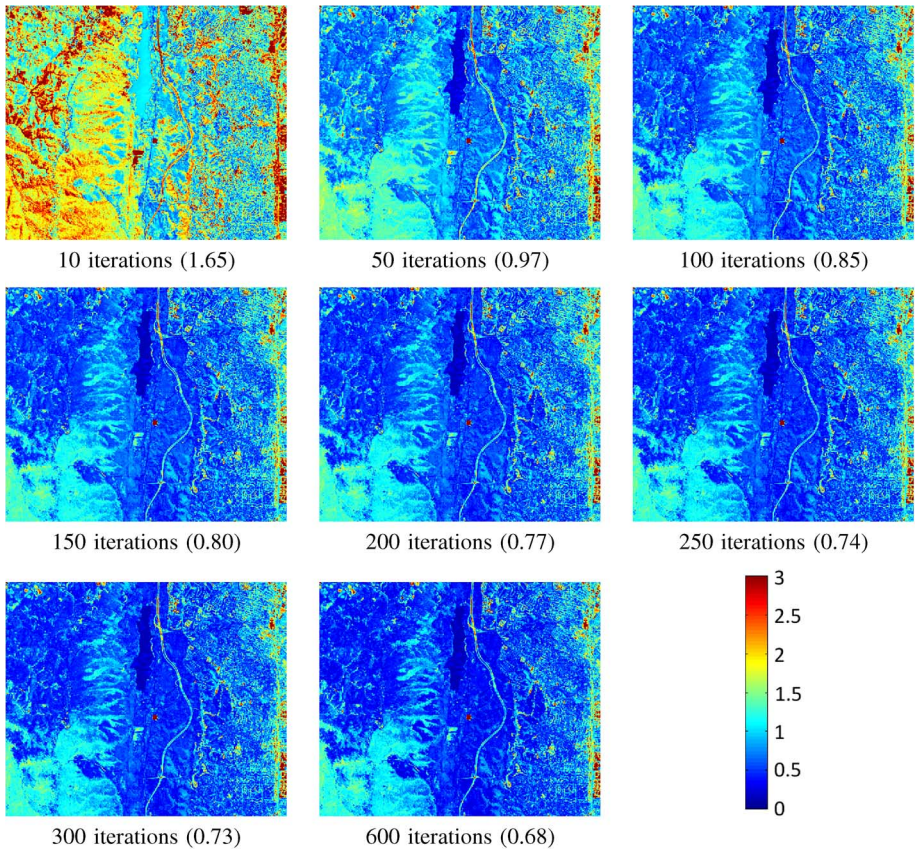

Fig. 14. Per-pixel and overall (in the parentheses) RMSE reconstruction errors for different number of iterations after reconstructing the AVIRIS Jasper Ridge scene in radiance units with the image pixels labeled as spectrally pure.

fine-tuned using autovectorization capabilities and optimization flags. Since the processing times for the AVIRIS Jasper Ridge data in radiance and reflectance units are exactly the same, we only report one of them in Table III. In all cases, the number of ISRA modules used in parallel for the FPGA version is $u = 16$. In our experiments, the processing times achieved in the FPGA board show a speedup of around $10\times$ when processing the AVIRIS Cuprite scene and a speedup over $12\times$ when it comes to the two Jasper Ridge AVIRIS scenes. It is also interesting to compare the time needed to process the two different images. For the hardware implementation, processing the second image consumes 2.6 times more execution time than processing the first one. This is considered a good result, as the second image is 2.6 times bigger than the first one. Hence, the execution time scales linearly with the size of the image. Regarding the software implementation, we experimentally observed that the execution time increases 3.1 times, which is a slightly worst result.

At this point, we should emphasize that the proposed FPGA implementation is not only interesting due to its good performance. In fact, very good parallel performance results have also been reported in the literature for the proposed approach implemented in commodity graphics processing units (GPUs) [37]. We are aware that the use of these systems may provide better performance than our FPGA implementation. However, the main contribution of our work is to demonstrate that a small device (certified for space applications) provides good performance with reasonable power/energy features, and that this device can operationally perform computations on board. This reference also includes a general comparison in terms of abundance estimation accuracy and computational cost concerning different hyperspectral abundance estimation algorithms.

Finally, we emphasize that other FPGA implementations of ISRA are available in the literature [28], [29]. However, the authors indicate that their full FPGA design and implementation (tested on a Virtex II Pro FPGA) required more execution time than the serial implementation presented in [34], [38]. This was due to data transmission and I/O bottlenecks. Since our approach optimizes I/O significantly, a comparison with regards to the approach in [28], [29] was not deemed suitable in terms of execution time. On the other hand, we synthesized our proposed implementation on the same Virtex II Pro FPGA in order to draw a comparison in terms of area occupation with regards to available approaches. While the implementation in [28], [29] requires 93% of the available resources for two units in parallel, our proposed implementation can scale up to 10 units in parallel and requires 90% of the available resources.

V. Conclusions and Future Research Lines

In this paper, we have described a parallel implementation of the image space reconstruction algorithm (ISRA), a popular approach to estimate positive fractional abundances in spectral unmixing of remotely sensed hyperspectral data. Our experimental results, conducted on a Virtex-4 XC4VFX60 FPGA (a platform with the same architecture and similar area than radi-

ation-hardened FPGAs that have been certified by international remote sensing agencies and are commonly used in both airborne and spaceborne Earth Observation missions) demonstrate that our hardware implementation can significantly outperform (in terms of computation time) an equivalent software version, and is also able to provide accurate abundance estimation results in reconfigurable hardware with compact size, thus making our system appealing for on-board and near real-time hyperspectral image processing. An important advantage of our design, which carefully optimizes input/output (I/O) communications, is its scalability, which makes it easy to be ported to new (possibly certified) FPGA developments as they become available. As future work, we plan to modify the algorithm to make the number of iterations per pixel adaptive, in the sense that the estimation does not need to be a fixed and constant value for all the pixels that compose the hyperspectral image. Different pixels may require different numbers of iterations, and we could iterate until the observed reconstruction error is below a given threshold. Although the proposed design is portable, in the future we will focus on improving the proposed implementation to make better use of hardware resources and decrease processing times.

ACKNOWLEDGMENT

The authors gratefully thank Dr. Robert O. Green at NASA/JPL for providing the AVIRIS data sets used in our experiments. The authors also would like to take this opportunity to gratefully thank the three anonymous reviewers for providing outstanding comments which helped us improve the technical quality and presentation of this manuscript.

REFERENCES

[1] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for Earth remote sensing," *Science*, vol. 22, pp. 1147–1153, 1985.

[2] R. O. Green *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, pp. 227–248, 1988.

[3] A. Plaza, J. A. Benediktsson, J. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, J. A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. 110–122, 2009.

[4] J. Plaza, A. Plaza, R. Perez, and P. Martinez, "On the use of small training sets for neural network-based characterization of mixed pixels in remotely sensed hyperspectral images," *Pattern Recognit.*, vol. 42, p. 30323045, 2009.

[5] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York: Springer, 2007.

[6] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.*, vol. 19, pp. 44–57, 2002.

[7] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, pp. 650–663, 2004.

[8] D. Heinz and C.-I. Chang, "Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, pp. 529–545, 2001.

[9] K. J. Guilfoyle, M. L. Althouse, and C.-I. Chang, "A quantitative and comparative analysis of linear and nonlinear spectral mixture models using radial basis function neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, pp. 2314–2318, 2001.

[10] A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*. Boca Raton, FL: CRC Press, 2007.

[11] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs," in *Proc. Int. Conf. Field Programmable Logic and Applications*, 2006, vol. 1, pp. 1–6.

[12] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, vol. 34, pp. 171–210, 2002.

[13] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: A survey," *J. VLSI Signal Process. Syst.*, vol. 28, pp. 7–27, 2001.

[14] D. A. Buell, T. A. El-Ghazawi, K. Gaj, and V. V. Kindratenko, "Guest Editors' introduction: High-performance reconfigurable computing," *IEEE Computer*, vol. 40, pp. 23–27, 2007.

[15] T. A. El-Ghazawi, E. El-Araby, M. Huang, K. Gaj, V. V. Kindratenko, and D. A. Buell, "The promise of high-performance reconfigurable computing," *IEEE Computer*, vol. 41, pp. 69–76, 2008.

[16] A. DeHon and J. Wawrzynek, "Reconfigurable computing: What, why, and implications for design automation," in *Proc. IEEE/ACM Design Automation Conf.*, 2009, vol. 1, pp. 610–615.

[17] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. San Mateo, CA: Morgan Kaufmann, 2007.

[18] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, Sep. 2011.

[19] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 528–544, Sep. 2011.

[20] D. Kohlert and F. Schreier, "Line-by-line computation of atmospheric infrared spectra with field programmable gate arrays," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 701–709, Sep. 2011.

[21] D. D. Lavenier, J. P. Theiler, J. J. Szymanski, M. Gokhale, and J. R. Frigo, "FPGA implementation of the pixel purity index algorithm," in *Proc. SPIE*, 2000, vol. 4212, p. 3041.

[22] J. Theiler, D. D. Lavenier, N. R. Harvey, S. J. Perkins, and J. J. Szymanski, "Using blocks of skewers for faster computation of pixel purity index," in *Proc. SPIE*, 2000, vol. 4132, p. 6171.

[23] C. Gonzalez, J. Resano, D. Mozos, A. Plaza, and D. Valencia, "FPGA omplementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis," *EURASIP J. Adv. Signal Process.*, vol. 2010, 2010, Article ID 969806, 13 pp.

[24] M. Hsueh and C.-I. Chang, "Field programmable gate arrays (FPGA) for pixel purity index using blocks of skewers for endmember extraction in hyperspectral imagery," *Int. J. High Performance Computing Applications*, vol. 22, no. 4, p. 408423, 2008.

[25] A. Plaza and C.-I. Chang, "Clusters versus FPGA for parallel processing of hyperspectral imagery," *Int. J. High Performance Computing Applications*, vol. 22, pp. 366–385, 2008.

[26] A. Plaza, J. Plaza, A. Paz, and S. Sanchez, "Parallel hyperspectral image and signal processing," *IEEE Signal Process. Mag.*, vol. 28, pp. 119–126, 2011.

[27] M. E. Daube-Witherspoon and G. Muehllehner, "An iterative image space reconstruction algorithm suitable for volume ECT," *IEEE Trans. Med. Imag.*, vol. 5, pp. 61–66, 1986.

[28] J. Morales, N. Medero, N. G. Santiago, and J. Sosa, "Hardware implementation of image space reconstruction algorithm using FPGAs," in *Proc. IEEE Int. Midwest Symp. Circuits and Systems*, 2006, vol. 1, pp. 433–436.

[29] J. Morales, N. G. Santiago, and A. Morales, "An FPGA implementation of image space reconstruction algorithm for hyperspectral imaging analysis," in *Proc. SPIE*, 2007, vol. 6565, pp. 1–8.

[30] J. M. P. Nascimento and J. M. Bioucas-Dias, "Does independent component analysis play a role in unmixing hyperspectral data?," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, pp. 175–187, 2005.

[31] C.-I. Chang and D. Heinz, "Constrained subpixel target detection for remotely sensed imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 38, p. 11441159, 2000.

[32] M. Velez-Reyes, A. Puetz, M. P. Hoke, R. B. Lockwood, and S. Rosario, "Iterative algorithms for unmixing of hyperspectral imagery," in *Proc. SPIE*, 2003, vol. 5093, p. 418429.

[33] C. A. Bateson, G. P. Asner, and C. Wessman, "Endmember bundles: A new approach to incorporating endmember variability into spectral mixture analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 38, p. 10831094, 2000.

[34] M. Velez-Reyes, A. Puetz, M. P. Hoke, R. B. Lockwood, and S. Rosario, "Iterative algorithms for unmixing of hyperspectral imagery," in *Proc. SPIE*, 2003, vol. 5093, pp. 418–429.
[35] Xilinx. [Online]. Available: http://www.xilinx.com/publications/prod_mktg/virtex5qv-product-table.pdf
[36] M. Garcia and S. L. Ustin, "Detection of interannual vegetation responses to climatic variability using AVIRIS data in a coastal savanna in California," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, pp. 1480–1490, 2001.
[37] S. Sanchez, A. Paz, G. Martin, and A. Plaza, "Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units," *Concurrency and Computation: Practice and Experience*, vol. 23, pp. 1538–1557, 2011.
[38] S. Rosario, "Iterative algorithms for abundance estimation on unmixing of hyperspectral imagery," Master thesis, Univ. Puerto Rico, Mayaguez, 2004.

Carlos González received the M.S. and Ph. D. degrees in computer engineering from the Complutense University of Madrid, Madrid, Spain, in 2008 and 2011, respectively.

He is currently a Teaching Assistant in the Department of Computer Architecture and Automation of the Universidad Complutense Madrid. As a research member of GHADIR group, he is mainly focuses on applying run-time reconfiguration in aerospace applications. His research interests include remotely sensed hyperspectral imaging, signal and image processing, and efficient implementation of large-scale scientific problems on reconfigurable hardware. He is also interested in the acceleration of artificial intelligence algorithms applied to games.

Dr. González won the Design Competition of the IEEE International Conference on Field Programmable Technology in 2009 (FPT'09) and in 2010 (FPT'10). He received the Best Paper Award of an Engineer under 35 years old in the International Conference on Space Technology in 2011.

Javier Resano (M'09) received the Bachelor degree in physics in 1997, the Master degree in computer science in 1999, and the Ph.D. degree in 2005 from the Universidad Complutense of Madrid, Spain.

Currently he is an Associate Professor in the Computer Engineering Department of the Universidad de Zaragoza, Spain, and he is a member of the GHADIR research group, from Universidad Complutense, and the GAZ research group, from Universidad de Zaragoza. He has collaborated with the Digital Design Technology Group of IMEC Laboratory since 2002. His research has been focused in hardware/software codesign, task scheduling techniques, dynamically reconfigurable hardware and FPGA design.

Antonio Plaza (SM'07) received the M.S. and Ph.D. degrees in computer engineering from the University of Extremadura, Caceres, Spain.

He was a Visiting Researcher with the Remote Sensing Signal and Image Processing Laboratory, University of Maryland Baltimore County, Baltimore, with the Applied Information Sciences Branch, Goddard Space Flight Center, Greenbelt, MD, and with the AVIRIS Data Facility, Jet Propulsion Laboratory, Pasadena, CA. He is currently an Associate Professor with the Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain, where he is the Head of the Hyperspectral Computing Laboratory (HyperComp). He was the Coordinator of the Hyperspectral Imaging Network (Hyper-I-Net), a European project designed to build an interdisciplinary research community focused on hyperspectral imaging activities. He has been a Proposal Reviewer with the European Commission, the European Space Agency, and the Spanish Government. He is the author or coauthor of around 300 publications on remotely sensed hyperspectral imaging, including more than 50 Journal Citation Report papers, 20 book chapters, and over 200 conference proceeding papers. His research interests include remotely sensed hyperspectral imaging, pattern recognition, signal and image processing, and efficient implementation of large-scale scientific problems on parallel and distributed computer architectures.

Dr. Plaza has coedited a book on high-performance computing in remote sensing and guest edited seven special issues on remotely sensed hyperspectral imaging for different journals, including the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (for which he serves as Associate Editor on hyperspectral image analysis and signal processing since 2007), the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, the International Journal of High Performance Computing Applications, and the Journal of Real-Time Image Processing. He has served as a reviewer for more than 280 manuscripts submitted to more than 50 different journals, including more than 140 manuscripts reviewed for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. He has served as a Chair for the IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing in 2011. He has also been serving as a Chair for the SPIE Conference on Satellite Data Compression, Communications, and Processing since 2009, and for the SPIE Remote Sensing Europe Conference on High Performance Computing in Remote Sensing since 2011. Dr. Plaza is a recipient of the recognition of Best Reviewers of the IEEE Geoscience and Remote Sensing Letters in 2009 and a recipient of the recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING in 2010. He is currently serving as Director of Education activities for the IEEE Geoscience and Remote Sensing Society.

Daniel Mozos (M'06) received the B.S. degree in physics and the Ph.D. degree in computer science from the Complutense University of Madrid, Spain.

He is a permanent Professor in the Department of Computer Architecture and Automatics of the Complutense University of Madrid, where he leads the GHADIR research group on dynamically reconfigurable architectures. His research interests include design automation, computer architecture, and reconfigurable computing.