FPGA Implementation of the N-FINDR Algorithm for Remotely Sensed Hyperspectral Image Analysis

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

***Abstract*—Hyperspectral remote sensing attempts to identify features in the surface of the Earth using sensors that generally provide large amounts of data. The data are usually collected by a satellite or an airborne instrument and sent to a ground station that processes it. The main bottleneck of this approach is the (often reduced) bandwidth connection between the satellite and the station, which drastically limits the information that can be sent and processed in real time. A possible way to overcome this problem is to include onboard computing resources able to preprocess the data, reducing its size by orders of magnitude. Reconfigurable field-programmable gate arrays (FPGAs) are a promising platform that allows hardware/software codesign and the potential to provide powerful onboard computing capability and flexibility at the same time. Since FPGAs can implement custom hardware solutions, they can reach very high performance levels. Moreover, using run-time reconfiguration, the functionality of the FPGA can be updated at run time as many times as needed to perform different computations. Hence, the FPGA can be reused for several applications reducing the number of computing resources needed. One of the most popular and widely used techniques for analyzing hyperspectral data is linear spectral unmixing, which relies on the identification of pure spectral signatures via a so-called endmember extraction algorithm. In this paper, we present the first FPGA design for N-FINDR, a widely used endmember extraction algorithm in the literature. Our system includes a direct memory access module and implements a prefetching technique to hide the latency of the input/output communications. The proposed 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 collected by NASA's Earth Observing-1 Hyperion (a satellite instrument) and the Airborne Visible Infra-Red Imaging Spectrometer over the Cuprite mining district in Nevada and the Jasper Ridge Biological Preserve in California. Experimental results demonstrate that our hardware version of the N-FINDR algorithm can significantly outperform an equivalent software version and is able to provide accurate results in near real time, which makes our reconfigurable system appealing for onboard hyperspectral data processing.**

***Index Terms*—Endmember extraction, field-programmable gate arrays (FPGAs), hyperspectral imaging, N-FINDR, reconfigurable hardware.**

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, 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). Although AVIRIS is a widely used platform, it constitutes only one source of hyperspectral data. For instance, the Hyperion hyperspectral imager aboard NASA's Earth Observing-1 (EO-1) spacecraft was NASA's first hyperspectral imager that became operational on-orbit. It routinely collects images hundreds of kilometers long with 242 spectral bands in the same spectral range as AVIRIS. Table I summarizes spaceborne Earth observation missions with hyperspectral sensors already launched or to be launched in the near future.

As Table I indicates, the proliferation of satellite-based instruments for hyperspectral remote sensing is very notorious, with very high spectral resolution in all cases. With such resolution, the ability to detect and identify individual materials or land-cover classes is greatly enhanced with regard to other techniques available, such as multispectral imaging, which typically just contains tens of images. During the past few years, a great deal of new hyperspectral instruments have been developed for remote sensing applications. In the near future, the use of hyperspectral sensors on spaceborne platforms will produce a nearly continual stream of high-dimensional data, and this expected high data volume will require fast, unsupervised means for storage, transmission, and analysis.

One of the most important challenges in hyperspectral image analysis is computational complexity, which results from the need to process enormous data volumes. With recent

Manuscript received November 30, 2010; revised February 4, 2011; accepted October 1, 2011. Date of publication November 11, 2011; date of current version January 20, 2012. This work was supported by the European Community's Marie Curie Research Training Networks Program under reference MRTN-CT-2006-035927, Hyperspectral Imaging Network (HYPER-I-NET). This work was also supported by the Spanish Ministry of Science and Innovation (HYPERCOMP/EODIX project, reference AYA2008-05965-C04-02).

C. González and D. Mozos are with the Department of Computer Architecture and Automatics, Computer Science Faculty, Complutense University of Madrid, 28040 Madrid, Spain (e-mail: carlosgonzalez@fdi.ucm.es; mozos@fis.ucm.es).

J. Resano is with the Department of Computer Architecture, Polytechnic Center, University of Zaragoza, 50018 Zaragoza, Spain (e-mail: jresano@unizar.es).

A. Plaza is with the Department of Technology of Computers and Communications, Polytechnic School of Cáceres, University of Extremadura, 10071 Cáceres, Spain (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/TGRS.2011.2171693

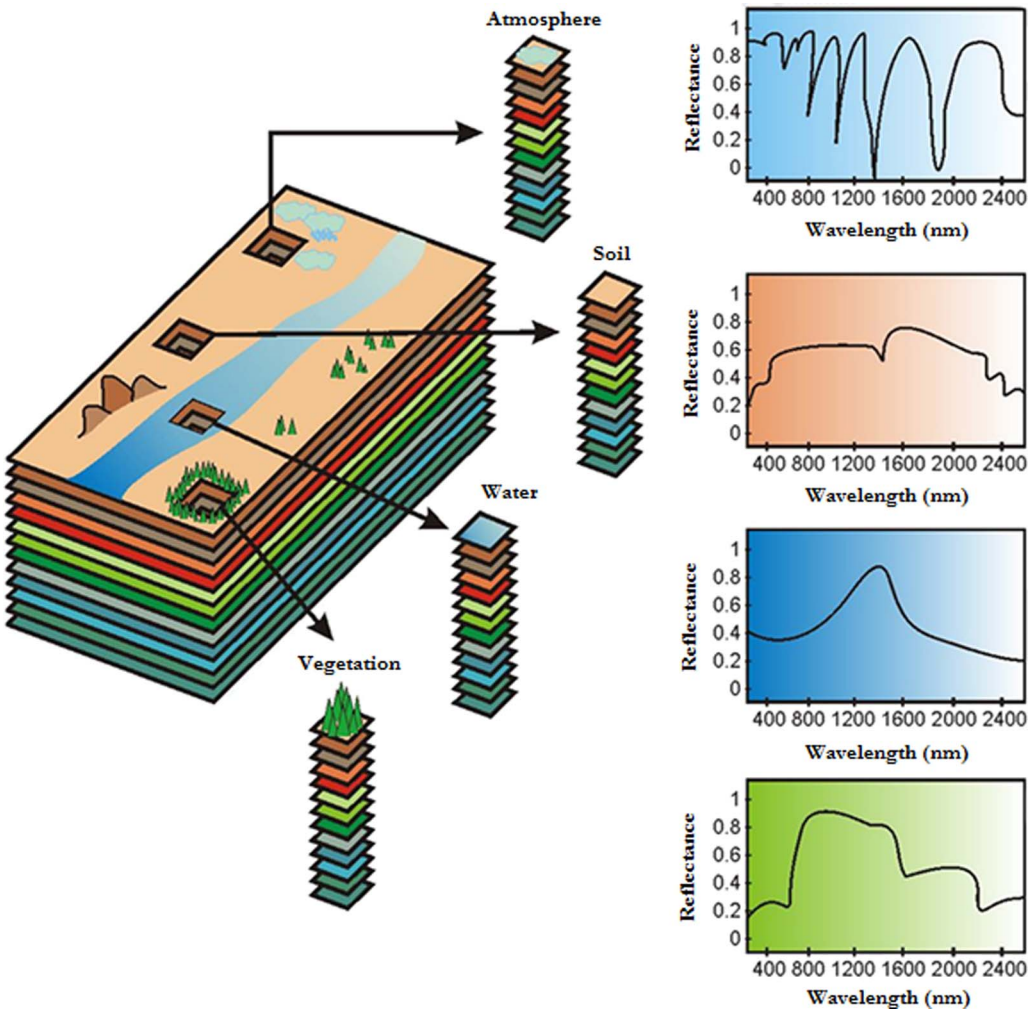

Fig. 1. Concept of hyperspectral imaging (inspired by the AVIRIS concept illustration available online at: http://aviris.jpl.nasa.gov/html/aviris.concept.html).

TABLE I
OVERVIEW OF SOME PRESENT AND FUTURE SPACEBORNE REMOTE SENSING MISSIONS INCLUDING HYPERSPECTRAL IMAGING INSTRUMENTS. EO-1 HYPERION (http://eo1.gsfc.nasa.gov). PRISMA (http://www.asi.it/en/flash_en/observing/prisma). ENMAP (http://www.enmap.org). HYSPIRI (http://hyspiri.jpl.nasa.gov)

	EO-1 Hyperion	**Prisma**	**EnMAP**	**HyspIRI**
Country of origin	USA	Italy	Germany	USA
Spatial Resolution	30 meters	5-30 meters	30 meters	60 meters
Revisit Time	16 days	3/7 days	4 days	18 days
Spectral Range	400-2500 nanometers	400-2500 nanometers	420-2450 nanometers	380-2500 nanometers
Spectral Resolution	10 nanometers	10 nanometers	6.5-10 nanometers	10 nanometers
Swath width	7.7 kilometers	30 kilometers	30 kilometers	120 kilometers
Earth coverage	Partial	Full	Full	Full
Launch	2000	2010	2012	2018
Lifetime	10 years	$\approx$ 6 years	$\approx$ 6 years	$\approx$ 6 years

advances in reconfigurable computing, particularly using field-programmable gate arrays (FPGAs) [3]–[5], hyperspectral image processing algorithms can now be accelerated using high-performance FPGAs. As hardware design can be carried out at the register-transfer level (RTL), the emerging electronic design automation tools allow design engineers to perform hardware/software codesign by transforming implementations developed in high level 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 onboard processing become feasible goals [6], [7]. FPGAs are now fully reconfigurable [8], [9], 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 onboard. The ever-growing computational demands of hyperspectral imaging applications can fully benefit from compact, reconfigurable hardware components and take advantage of the small size and relatively low cost of these units.

In this paper, we develop an FPGA-based hardware version of a popular algorithm for hyperspectral data analysis. It belongs to the category of spectral unmixing [10]–[12], which has been an alluring exploitation goal since the earliest days of hyperspectral image and signal processing. No matter what the spatial resolution is, the spectral signatures collected in natural environments are invariably a mixture of the signatures of the various materials found within the spatial extent of the ground instantaneous field view of the imaging instrument. For instance, the pixel vector labeled as "vegetation" in Fig. 1 may actually comprise a mixture of vegetation and soil, or different types of soil and vegetation canopies. In this case, several spectrally pure signatures (called *endmembers* in hyperspectral imaging terminology) are combined into the same (mixed) pixel. Winter's N-FINDR algorithm [13] approaches the automatic identification of such endmembers using volume-related computations associated to the calculation of determinants. In this paper, we use this popular algorithm in the endmember extraction and unmixing communities as a case study to illustrate the advantages and potential challenges of applying software/hardware codesign principles for porting hyperspectral imaging algorithms to specialized modules (intended for onboard operation) in both airborne and spaceborne platforms, with emphasis on the technological advances needed to fully achieve such incorporation.

The remainder of the paper is organized as follows. Section II discusses the role of reconfigurable hardware in remote sensing missions. Section III describes the spectral unmixing problem and the original N-FINDR algorithm. Section IV describes its implementation on a Xilinx Virtex-4 XC4VFX60 FPGA. Section V provides an experimental assessment of both endmember extraction accuracy and processing performance of the proposed FPGA-based algorithm, using well-known hyperspectral data sets collected by NASA's AVIRIS and EO-1 Hyperion instruments over two different sites: the Cuprite mining district in Nevada, and the Jasper Ridge Biological Preserve in California. Finally, Section VI concludes with some remarks and hints at plausible future research lines.

II. Role of Reconfigurable Hardware in Remote Sensing Missions

Future remote sensing missions such as those described in Table I will require significant technological advances to achieve high data processing rates. For instance, recent internal studies at NASA's Jet Propulsion Laboratory estimate that a volume of 1–5 TB of raw data (uncompressed) per day can be expected from future imaging instruments such as HyspIRI. As a result, efficient implementations of onboard processing algorithms able to perform data reduction will be required to drastically reduce data volumes within the downlink capabilities of the satellite and existing ground stations. A trend in the design of hardware modules for remote sensing missions has been using hardware devices with small size and cost, but with flexibility and high computational power [22], [23]. Onboard processing, as a solution, allows for a good reutilization of expensive hardware resources. Furthermore, it allows making autonomous decisions onboard that can potentially reduce the delay between the image capture, analysis, and the related action. Implementations of onboard processing algorithms to perform data reduction can dramatically reduce data transmission rates (see Fig. 2). However, many available systems are characterized by their high power consumption, cost, and requirement of additional interface boards. An interesting alternative is to include recently developed hybrid FPGAs, such as the Xilinx Virtex-4FX60 and Virtex-5. These FPGAs not only include a larger hardware area to implement custom accelerators, but also embedded processors and memory resources. This option offers versatility in running diverse software applications on embedded processors, while taking advantage of reconfigurable hardware resources, all on the same chip package. These tightly coupled hardware/software codesigned systems combine the flexibility of traditional microprocessors with the power and performance of custom hardware implementations, leading to new architectures for remote sensing missions. There are at least three companies that have introduced 8- and 32-bit embedded system microcontrollers which have a portion of an integrated circuit that contains reconfigurable logic [24]–[26]. The advantage of this setup is that the communication delays between the embedded processor and the associated reconfigurable logic are reduced through the incorporation of internal buses. It also has the advantage that power consumption and external input/output (I/O) wiring is significantly reduced.

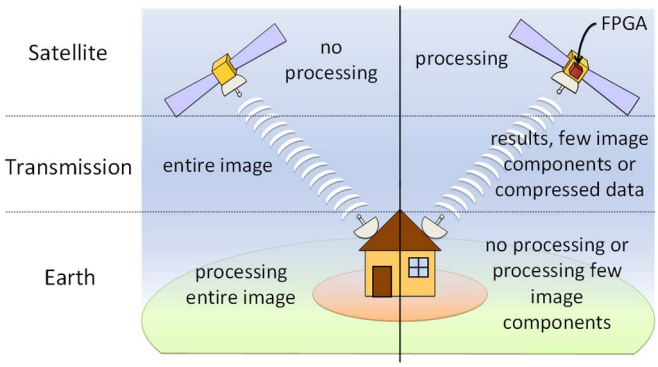

Fig. 2. Potential advantages of using reconfigurable hardware in remote sensing data processing.

It should be noted that current remote sensing systems extensively utilize embedded microcontrollers and dedicated hardware peripherals. There is much flexibility present in those systems, but this flexibility is restricted to the software portion of the design while the hardware portion remains fixed from the time of its fabrication. Unfortunately, embedded microcontrollers are sequential in nature and spend much time simply moving data around but not performing useful computations. Reconfigurable hardware allows the degree of flexibility needed to avoid this bottleneck. With reconfigurable hardware, it is possible to apply much of the flexibility that was formally restricted to software developments only, to highly parallel

hardware resources. The idea is that FPGAs can be reconfigured on the fly. This approach is called *temporal partitioning* [27], [28] or *run-time reconfiguration* [29]. Basically, the FPGA (or a region of the FPGA) executes a series of tasks one after another by reconfiguring itself between tasks [30]. The reconfiguration process updates the functionality implemented in the FPGA, and a new task can then be executed. This time-multiplexing approach allows for the reduction of hardware components onboard since one single reconfigurable module can substitute several hardware peripherals carrying out different functions during different phases of the mission.

The flexibility provided by reconfigurable hardware can also be used to modify the functionality of the satellite instrument during the flight or to automatically recover from malfunction. Moreover, the hardware design cycle for FPGAs is much shorter than the one for custom-integrated circuits, mainly because the design can be tested on the target platform since the first steps of the design process, thus avoiding a complex chip fabrication process. Another important outcome of reconfigurable computing is the possibility to achieve real-time processing. This is because it can provide a mechanism for deterministic execution. Many of the hardware modules associated with modern embedded microcontrollers (such as those associated with interrupt processing and cache memory operations) are highly nondeterministic in nature. This nondeterminism compromises the attempts to guarantee that all time constraints will always be met. The incorporation of reconfigurable hardware gives the remote sensing system designer additional flexibility which could be used to assign many time-critical applications to their own hardware, making these functions more independent of such nondeterministic operations.

Moreover, satellite-based remote sensing instruments can only include chips that have been certified for space operation. This is because space-based systems must operate in an environment in which radiation effects have an adverse impact on integrated circuit operation [31]. Ionizing radiation can cause soft errors in the static cells used to hold the configuration data. This will affect the circuit functionality and ultimately result in system failure. This requires special FPGAs that provide on-chip reconfiguration error detection and/or correction circuitry. High-speed, radiation-hardened FPGA chips with million gate densities have recently emerged to support the high throughput requirements for remote sensing applications. In fact, radiation-hardened FPGAs are in great demand for military and space applications. For instance, industrial partners such as Actel Corporation[1] or Xilinx[2] have been producing radiation-tolerant antifuse FPGAs for several years, intended for high-reliability space-flight systems. Actel FPGAs have been onboard more than 100 launches, and Xilinx FPGAs have been used in more than 50 missions [31]. In this paper, we use a Xilinx Virtex-4 XC4VFX60 FPGA as a baseline architecture because it is similar to other FPGAs that have been certified by several international agencies for remote sensing applications [32]. They are based on the same architecture so porting our design to those platforms is a straightforward process.

[1]http://www.actel.com
[2]http://www.xilinx.com

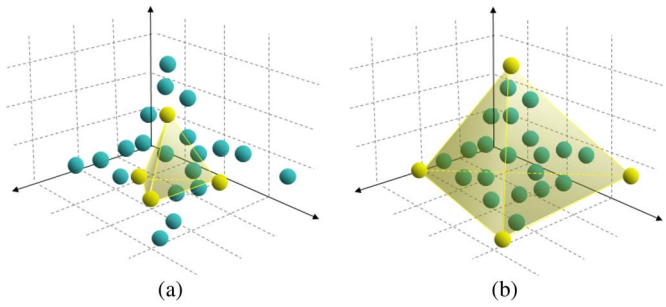

Fig. 3. Graphical interpretation of the N-FINDR algorithm in a 3-D space. (a) N-FINDR initialized randomly ($p = 4$). (b) Final volume estimation by N-FINDR.

III. Spectral Unmixing and the N-FINDR Algorithm

To define the spectral unmixing problem in mathematical terms, let us assume that a remotely sensed hyperspectral scene with n bands is denoted by $\mathbf{I}$, in which the pixel at the discrete spatial coordinates (i, j) of the scene is represented by a vector $\mathbf{X}(i, j) = [x_1(i, j), x_2(i, j), \ldots, x_n(i, j)] \in \Re^n$, where $\Re$ denotes the set of real numbers in which the pixel's spectral response $x_k(i, j)$ at sensor channels $k = 1, \ldots, n$ is included. Under the linear mixture model assumption [14], [18], each pixel vector in the original scene can be modeled using the following expression:

$$\mathbf{X}(i,j) = \sum_{z=1}^{p} \Phi_z(i,j) \cdot \mathbf{E}_z + \mathbf{n}(i,j) \tag{1}$$

where $\mathbf{E}_z$ denotes the spectral response of endmember z, $\Phi_z(i, j)$ is a scalar value designating the fractional abundance of the endmember z at the pixel $\mathbf{X}(i, j)$, p is the total number of endmembers, and $\mathbf{n}(i, j)$ is a noise vector. The solution of the linear spectral mixture problem described in (1) relies on the correct determination of a set $\{\mathbf{E}_z\}_{z=1}^{p}$ of endmembers and their correspondent abundance fractions $\{\Phi_z(i, j)\}_{z=1}^{p}$ at each pixel $\mathbf{X}(i, j)$. The derivation and validation of the correct suite of endmembers have remained a challenging goal for the past years (not only in terms of adequate spectral signature extraction [19], but also in terms of computational complexity [20]).

Over the last decade, several algorithms have been developed for automatic or semiautomatic extraction of spectral endmembers [19]. Winter's N-FINDR [13] has been one of the most succesfully applied techniques for automatically determining endmembers in hyperspectral image data. This algorithm looks for a set of pixels with the largest possible volume by "inflating" a simplex inside the data. The procedure begins with a random initial selection of pixels [see Fig. 3(a)]. Every pixel in the image must be evaluated to refine the estimate of endmembers, looking for the set of pixels that maximizes the volume of the simplex defined by selected endmembers. The corresponding volume is calculated for every pixel in each endmember position by replacing that endmember and finding the resulting volume. If the replacement results in an increase of volume, the pixel replaces the endmember. This procedure is repeated until there are no more endmember replacements [see Fig. 3(b)].

The mathematical definition of the volume of a simplex formed by a set of endmember candidates is proportional to the determinant of the set augmented by a row of ones. The determinant is only defined in the case where the number of features is $p-1$, being p the number of desired endmembers [21]. Since in hyperspectral data typically the number of spectral bands is much larger than the number of endmembers, i.e., $n \gg p$, a transformation that reduces the dimensionality of the input data is required. While the endmember determination step of N-FINDR in the commercial version distributed by Pacific Spectral Technology[3] has been optimized for high speed processing, the computational performance of the algorithm depends on the accuracy of the initial random selection of endmembers and, most importantly, on the dimensions of the hyperspectral scene and the number of endmembers to be found, p.

In the following, we provide a detailed step-by-step algorithmic description of the original N-FINDR algorithm developed by Winter [13]. It is interesting to notice that the algorithm below represents our own effort to delineate the steps implemented by N-FINDR using available references in the literature [13], [33]. However, it is also worth noting that the N-FINDR algorithm has never been fully disclosed. As a result, this description was developed based on the limited published results available and our own interpretation. Nevertheless, the algorithm below has been verified using the N-FINDR software, provided by the authors, where we have experimentally tested that the software produces essentially the same results as the code below, provided that initial endmembers are generated randomly. The original N-FINDR algorithm can be summarized by the following steps.

1) *Feature reduction*. Apply a dimensionality reduction transformation such as the minimum noise fraction [34] or the principal component analysis [35] to reduce the dimensionality of the data from n to $p-1$, where p is an input parameter to the algorithm (number of endmembers to be extracted).
2) *Initialization*. Let $\{\mathbf{E}_1^{(0)}, \mathbf{E}_2^{(0)}, \ldots, \mathbf{E}_p^{(0)}\}$ be a set of endmembers randomly extracted from the input data.
3) *Volume calculation*. At iteration $k \geq 0$, calculate the volume defined by the current set of endmembers as follows:

$$V\left(\mathbf{E}_1^{(k)}, \mathbf{E}_2^{(k)}, \ldots, \mathbf{E}_p^{(k)}\right) = \frac{\left|det\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{E}_1^{(k)} & \mathbf{E}_2^{(k)} & \cdots & \mathbf{E}_p^{(k)} \end{bmatrix}\right|}{(p-1)!}. \tag{2}$$

4) *Replacement*. For each pixel vector $\mathbf{X}(i,j)$ in the input hyperspectral data, recalculate the volume by testing the pixel in all p endmember positions, i.e., first calculate $V(\mathbf{X}(i,j), \mathbf{E}_2^{(k)}, \ldots, \mathbf{E}_p^{(k)})$, then $V(\mathbf{E}_1^{(k)}, \mathbf{X}(i,j), \ldots, \mathbf{E}_p^{(k)})$, and so on, until $V(\mathbf{E}_1^{(k)}, \mathbf{E}_2^{(k)}, \ldots, \mathbf{X}(i,j))$. If none of the p recalculated volumes is greater than $V(\mathbf{E}_1^{(k)}, \mathbf{E}_2^{(k)}, \ldots, \mathbf{E}_p^{(k)})$, then no endmember is replaced. Otherwise, the combination with maximum volume is retained. Let us assume that the endmember absent in the combination resulting in the maximum volume is denoted by $\mathbf{E}_j^{(k+1)}$. In this case, a new set of endmembers is produced by letting $\mathbf{E}_j^{(k+1)} = \mathbf{X}(i,j)$ and $\mathbf{E}_i^{(k+1)} = \mathbf{E}_i^{(k)}$ for $i \neq j$. The replacement step is repeated in an iterative fashion, using as many iterations as needed until there are no more replacements of endmembers.

To our best knowledge, and despite the importance of the N-FINDR algorithm in the hyperspectral unmixing community, there are no available implementations of this algorithm on reconfigurable hardware in the literature. In the following section, we provide a possible hardware implementation of this algorithm.

IV. FPGA Implementation of the N-FINDR Algorithm

This section is organized as follows. In Section IV-A, we provide an overview of the determinant properties that will be used to design our hardware version of N-FINDR for FPGAs. Section IV-B describes such implementation, explaining the performance of each designed hardware module. Finally, Section IV-C describes one of such modules (the N-FINDR hardware module) in terms of FPGA design language.

A. Use of Determinant Properties

The most time-consuming part of the N-FINDR algorithm is the *volume calculation* step. The limited available resources in a small or medium FPGA to calculate determinants of large order (embedded multipliers, look-up tables, and slices) complicates the development of an efficient hardware implementation of the algorithm. The calculation of a determinant is a succession of additions and subtractions of the product of the main diagonal elements for all permutations of the matrix columns. To find a determinant of order $p > 3$, it is not advisable to use this development as sum of $p!$ terms, where each of them is a product of p elements. This decomposition requires a large number of operations. However, the use of the determinant properties can simplify the calculations. Specifically, the development of a determinant by the elements of a row or column generally does not provide a means to solve the problem effectively by numerical methods. To calculate the determinants, it is advisable to transform the determinants into others which are increasingly easier to calculate, down to one which is trivial. In this paper, we use the matrix triangulation method which allows us to calculate determinants of sufficient order to extract the necessary number of endmembers in most of hyperspectral images using a small FPGA.

In the following, we provide a description of the matrix triangulation method adopted in this work (see Algorithm 1 for additional details). To find the determinant of a square

[3]http://www.pacificspectral.com

matrix $\mathbf{A} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \mathbf{E}_1^{(k)} & \mathbf{E}_2^{(k)} & \dots & \mathbf{E}_p^{(k)} \end{bmatrix}$ with size $p \times p$, the following steps can be applied.

1) If the first element of the matrix $a_{11} \neq 0$, go to step (2). If $a_{11} = 0$ and $a_{l1} \neq 0$ for some l, switch the rows first and l th, so that the determinant changes sign and the element a_{11} is different from zero. If all a_{l1} values are zero, the problem is solved because the determinant of $\mathbf{A}$ is equal to 0, and the process is finished.
2) Let us assume that our goal is to find the determinant of order p (denoted by Δ_p) of matrix $\mathbf{A}$, whose element a_{11} is nonzero. To simplify notation, we will denote the elements in the first column of $\mathbf{A}$ by $[\alpha_1, \alpha_2, \dots, \alpha_p]^T$, with $\alpha_1 \neq 0$. We now subtract the result of multiplying the first row times α_l/α_1 to the the l th row and perform this operation for $l = 2, 3, \dots, p$. This operation will not alter the value of the determinant, which is still Δ_n, and the elements of the first column become $[\alpha_1, 0, \dots, 0]^T$. Therefore, the determinant Δ_p can now be expressed in the form $\Delta_p = \alpha_1 \Delta_{p-1}$, where Δ_{p-1} is the determinant of order $p-1$, i.e., the result of deleting the first row and the first column of Δ_{p-1}.
3) Using the determinant Δ_{p-1}, we can now repeat the process and reduce the calculation to a determinant Δ_{p-2}. By reiterating this process, we can arrive at a determinant of second order, which is calculated trivially.

Algorithm 1 Matrix triangulation method applied to a square matrix $\mathbf{A}$ with size p. In this algorithmic description, we will assume for simplicity and standard notation throughout the paper that $\mathbf{A}[i][j] \equiv a_{ij}$.

```
for (i=0; i<p; i++) {//p denotes the size of the square matrix A
  //Step 1
  if(A[i][i] == 0) {
      for (j = i + 1; j < p; j++) {
          if(A[j][j]! = 0) {
          aux_row = A[i]; //A[i] denotes the full i th row
                A[i] = A[j];A[j] = aux_row;
                break;
          } end if
      } end for
  } end if
  if(A[i][i] == 0) return 0;
  //Step 2
  for (j = i + 1; j < p; j++) {
A[j] = A[j] − A[i] ∗ (A[j][i]/A[i][i]);
} end for
} end for
```

In our particular case, we iterate until obtaining a triangular matrix and then multiply the main diagonal elements. Furthermore, we exploit the property that the determinant of $\mathbf{A}$ is equal to the determinant of $\mathbf{A}^T$ and take into account that the change of the determinant sign when we switch two rows does not affect our calculation because we are looking for the absolute value. With these design choices in mind, we will see in the following subsection that our FPGA design can extract up to $p = 21$ endmembers using a small FPGA as the Xilinx Virtex-4 XC4VFX60. If the necessary number of endmembers is lower than this figure, we can execute the calculation of the determinants in parallel in the FPGA, thus reducing the execution time even more.

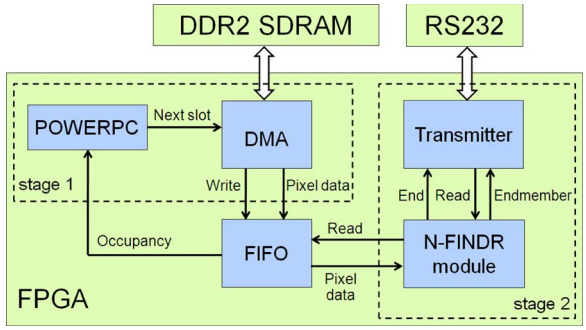

Fig. 4. Hardware architecture adopted to implement the complete system.

B. Hardware Implementation

Fig. 4 shows the architecture of the hardware used to implement the N-FINDR algorithm, along with the I/O communications. For data input, we use a DDR2 SDRAM and a direct memory access (DMA) (controlled by a PowerPC) with a first in first out (FIFO) to store the pixel data. The N-FINDR module is used to implement our version of the N-FINDR algorithm. Finally, a transmitter is used to send the endmembers via a RS232 port. The general structure of the hardware used to implement the N-FINDR algorithm can be seen as a pipelined architecture. We can distinguish two stages which are communicated using a FIFO structure: the first stage provides the necessary data for the system, and the second stage carries out the endmember extraction process and finally sends the endmembers via a RS232 port. Therefore, all stages are working in parallel.

On the other hand, Fig. 5 shows the architecture of the hardware used to implement the N-FINDR algorithm. We use registers to store the pixels vector selected as endmembers until the current moment, their positions in the image and their volume, the current pixel vector data, its position, its greatest volume and the index inside the matrix where it is obtained, and finally the module to calculate the absolute value of the determinant. To be able to provide a new pixel vector (a row of the matrix) every clock cycle, we use registers to store the current set of endmembers and a combinational circuit consisting of multiplexers, where all these modules are managed by the control unit. For simplicity, the hardware needed for the replacement step is not shown in Fig. 5. To implement this step, we use a comparator between the volume registers, and, if the volume with the new pixel is greater, the replacement is carried out by the control unit.

Fig. 6 describes the architecture of the module used to calculate the absolute value of the determinant using the matrix triangulation process. We can clearly distinguish three units: The *circular queue*, with a small control unit, is responsible

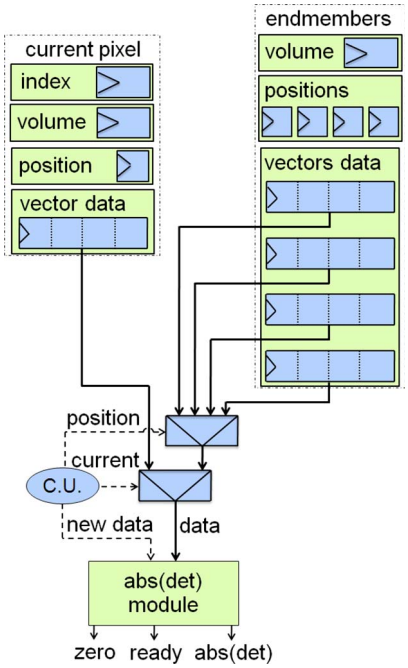

Fig. 5. Hardware architecture adopted to implement the N-FINDR algorithm.

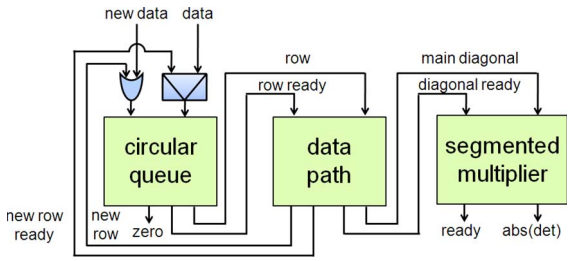

Fig. 6. Hardware architecture adopted to implement the absolute value of the determinant, i.e., module abs(det) in Fig. 5.

for providing the different rows to the *data path* in the matrix triangulation process and to switch the new rows from the *data path* to have the element $a_{ii} \neq 0$ (step 1 of the algorithm). The *data path* receives the last m rows of the matrix $(p \times p)$, stores the first one, and performs the zeroing of the elements of place $p - m + 1$ in the remaining rows by subtracting the stored row multiplied by the result of the division between the $m - p + 1$ elements of the current and stored row (step 2 of the algorithm). Finally, the segmented multiplier calculates the multiplication of the main diagonal elements of the triangular matrix and obtains the absolute value. In the following, we describe the behavior of these units in more detail. First, for $j = 2, \ldots, p$ we take a multiple a_{j1}/a_{11} of the first row and subtract it to the jth row, to make $a_{j1} = 0$. Thus, we have knocked out all elements of $\mathbf{A}$ below the "pivot" element a_{11} in the first column. Now, for $j = 3, \ldots, n$, we take a multiple a_{j2}/a_{22} of the second row and subtract it to the jth row. When we have finished this operation, all subdiagonal elements in the second column are zero, and we are ready to process the third column. By applying this process to columns $i = 1, \ldots, p - 1$ (there are no subdiagonal elements to knock out in the p th column), we complete the matrix triangulation process and $\mathbf{A}$ can be reduced to an upper triangular form. It is important to note that, since we eliminate

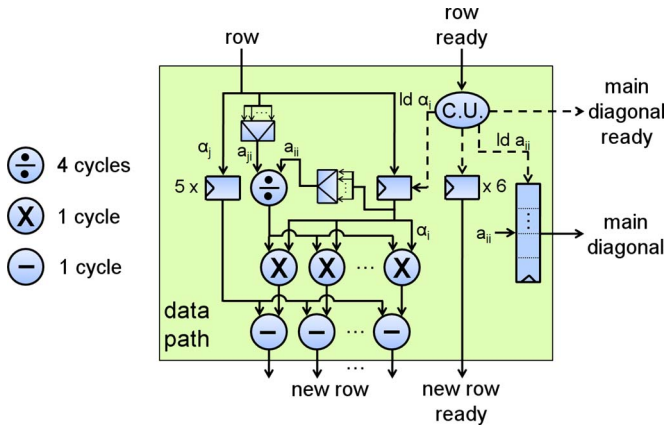

Fig. 7. Hardware architecture adopted to implement step (2) of the matrix triangulation process.

subdiagonal elements in column by column fashion beginning with the first column, the zeroed elements remain zero; while we are subtracting a multiple a_{ji}/a_{ii} of row i to row $j (j > i)$, to knock out a_{ji}, we are just subtracting multiples of zero to zero in columns 1 to $i - 1$.

Fig. 7 shows the architecture of the *data path* used in our design to implement step 2 of the matrix triangulation process. Basically, this data path stores the first row of each iteration in a register and subsequently (for each of the remaining rows) calculates a_{ji}/a_{ii}, multiplies the result by the stored row, and finally subtracts it to the appropriate row. It also stores the main diagonal elements and alerts when all elements are obtained. Obviously, if one of the diagonal pivots a_{ii} is zero, we cannot use a_{ii} to knock out the elements below it; we cannot change a_{ji} by subtracting any multiple of $a_{ii} = 0$ to it. We must switch row i with another row l below it (switching with a row above i would destroy some of the zeros introduced earlier), which contains a nonzero element a_{li} in the i th column. Now, the new pivot a_{ii} is nonzero, and we can continue the matrix triangulation process. If $a_{li} = 0$ for $l = i, \ldots, p$, then it will not be satisfactory to switch row i with any of rows below it as all the potential pivots are zero, and therefore the determinant of $\mathbf{A}$ will also be zero (Fig. 8).

To perform the switching of rows, we use a modified circular queue. In addition to using a pointer to indicate the first element place (*head*), we use a second pointer to indicate the location of the first row (with nonzero pivot) for the next iteration (*first nonzero*) and another one to indicate where the last element is placed below the previous pointer (*tail*). Once the matrix rows are stored, they are sent successively to the *data path*. In this way, we receive in order the new rows that will be stored in the position indicated by the *first nonzero* or the *last* pointer, depending on whether a row with nonzero pivot was previously detected for the next iteration. The correct update of the pointers is carried out by a small control unit. In the case that the pivot is zero in all new rows, the zero signal is activated indicating that the value of the determinant is zero. In the following, we provide an example of how the aforementioned operation is performed. After five matrix rows are stored [see Fig. 6(a)], rows r_1, r_2, and r_3 are sent, and the new rows r'_2 and r'_3 are received, but, in both cases, the pivot element for the

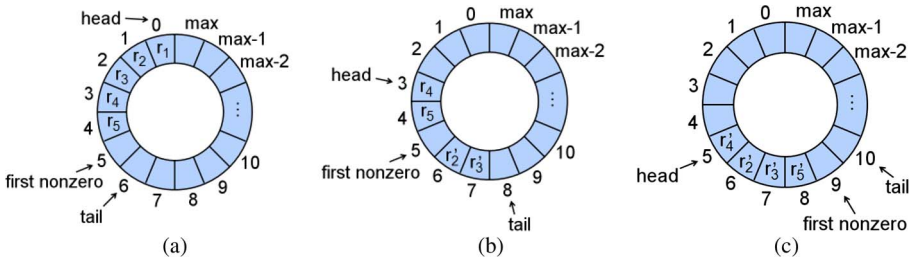

Fig. 8. Hardware architecture adopted to implement step (1) of the matrix triangulation process.

next iteration is zero, so they have been stored in the position indicated by the *last* pointer [see Fig. 6(b)]. After sending the other rows, we get the new row r'_4 whose pivot is nonzero and has therefore been stored in the position indicated by the *first nonzero* pointer. Finally, we get the new row r'_5 which, regardless of the value of the pivot, has been stored in the position indicated by the *last* pointer [see Fig. 6(c)]. In this way, the rows are ready for the next iteration.

When the matrix is reduced to an upper triangular form, the segmented multiplier calculates the product of the main diagonal elements. Nowadays, the vast majority of FPGAs contain hardware multipliers. This allows multiplication to be done inside the FPGA without the use of large numbers of look-up tables and also at a low power cost. Our pipelined multiplier is able to multiply n input data in $\log_2(n)$ clock cycles using n multipliers.

To conclude this section, we provide a step-by-step description of how the proposed architecture performs the extraction of a set of p endmembers from a hyperspectral image.

- First, the PowerPC randomly selects an initial set of p endmembers and sends an order to the DMA to write them in the FIFO.
- Afterwards, the control unit reads these endmembers and sends them to the registers where they are stored. Then, the volume of the initial set of endmembers is calculated, and the result is stored in a register.
- After the PowerPC has written the initial set of endmembers, it sends an order to the DMA to start copying a piece of the image from the DDR2 SDRAM to the 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 FIFO and sends a new order to the DMA every time that it detects that the FIFO is half empty. This time, the DMA will bring a piece of the image that occupies half of the FIFO total capacity.
- When the data of the first pixel has been written in the FIFO, the N-FINDR module starts working. The pixel data is stored, and every clock cycle, a new diagonal is sent to the module in charge of calculating the absolute value of the determinant by the control unit. When the volume is calculated, the control unit compares it with the volume stored, and, if the first one is greater, the volume and position registers are updated. Then, we recalculate the volume by testing the pixel in the next endmember positions in the same way. This volume is calculated for all p endmember positions. Finally, if the highest recalculated volume is greater than the total volume, the control unit updates the set of endmembers and the volume and position registers. This step is repeated several times depending on the total number of pixels in the hyperspectral image.
- Finally, the transmitter extracts the endmembers from the registers and sends them via a RS232 port.

C. N-FINDR Hardware Module

In this subsection, we provide some details about the N-FINDR hardware module. This module is the most important component of our system, and we took special care in its definition using VHDL[4] as the hardware description language. Here, we follow a structural description style to illustrate the behavior of the main module as a set of instances of different hardware components such as registers, arithmetic-logic units, multiplexors, etc. The structural description exactly matches the architecture shown in Fig. 5. Some of these modules are in fact quite complex themselves. Hence, to illustrate their behavior, we also use a structural description scheme. The only exception is the control unit, which has been described as a state machine.

Since the requirements of each hyperspectral image can be very different (for instance, the optimal number of endmembers may change form one image to another), a flexible design was needed. For this purpose, we have designed a fully customizable module using several *generic* parameters which allow us to instantiate the design according to the characteristic of the considered hyperspectral images. Such instantiation is performed without the need to change the VHDL code. To customize our design, we just need to set the appropriate values for each generic parameter. These parameters are: 1) the number of endmembers to be extracted, 2) the number of pixels in the hyperspectral data, 3) the number of bits needed to represent the radiance/reflectance values, and 4) the number of bits for encoding the spatial position within the image data. For instance, adapting our design for a different number of endmembers only involves adjusting the corresponding *generic* value and synthesizing the module. Hence, our design can be used to search for any number of endmembers provided that enough hardware resources are available.

Fig. 9 provides a structural description of the N-FINDR hardware module in VHDL language. For simplicity, we have only included the instantiations needed for the main components, and we have removed some components and signal definitions,

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

```
entity NFINDRmodule is
    generic(endmembers : integer := 16;
            numBitsData : integer := 16;
            numPixels : integer := 350*350;
            numBitsPositions : integer := 17);
    port(clk, rst, newVectorData: in std_logic;
         din : in std_logic_vector(endmembers*numBitsData-1 downto 0);
         position : in std_logic_vector(numBitsPositions-1 downto 0 );
         ready : out std_logic;
         doutPositions : out std_logic_vector(endmembers*numBitsPositions-1 downto 0));
end NFINDRmodule;

architecture NFINDRmoduleArch of NFINDRmodule is

  -- Component definitions

  -- Signals definition

begin

  endmembersPositionsAndData : for I in 0 to endmembers-1 generate

    data: register generic map (endmembers*numBitsData) port map (clk, rst,
    ldEndmembers(I), dinEndmemberData, doutData(I*numBitsData+numBitsData-1 downto
    I*numBitsData));

    positions: register generic map (endmembers*numBitsPositions) port map (clk, rst,
    ldEndmembers(I), position, doutPositions(I*numBitsPositions+numBitsPositions-1
    downto I*numBitsPositions));

  end generate;

  endmembersVolume : register generic map (numBitsData)
    port map (clk, rst, ldEndmembersVolume, absDet, doutEndmembersVolume);

  pixelIndex : register generic map (numBitsPositions)
    port map (clk, rst, ldPixelIndex, dinPixelIndex, doutPixelIndex);

  pixelVolume : register generic map (numBitsData)
    port map (clk, rst, ldPixelVolume, absDet, doutPixelVolume);

  pixelPosition : register generic map (numBitsPositions)
    port map (clk, rst, ldPixelPosition, position, doutPixelPosition);

  pixelData : register generic map (endmembers*numBitsData)
    port map (clk, rst, ldPixelData, din, doutPixelData);

  mux01 : mux generic map (endmembers*numBitsData, endmembers)
    port map (doutData, position, doutMux01);

  mux02 : mux generic map (endmembers*numBitsData, 2)
    port map (doutMux01&doutPixelData, current, data);

  absDetModule : generic map (endmembers, numBitsData, numBitsCounter)
    port map (newData, data, zero, ready, absDet);

  controlUnit : generic map (endmembers, numBitsData, numPixels, numBitsPositions)
    port map (newData, current, position, ready, ...); -- Rest of signals

end NFINDRmoduleArch;
```

Fig. 9. N-FINDR hardware module description in terms of FPGA design language (VHDL).

as well as some interconnection details. As shown by Fig. 9, all the hardware modules use one of several *generic* parameters. Hence, they can be adaptively customized according to the characteristics of the hyperspectral image to be processed. Further, the VHDL description in Fig. 9 provides an example of how to instantiate a variable number of elements of a given component. This is done right after the first `begin` sentence in the code, where the `for generate` structure is used to adapt our data path to the number of endmembers to be searched for in the considered hyperspectral image.

V. Experimental Results

This section is organized as follows. In Section V-A, we describe the FPGA board used in our experiments. Section V-B describes the hyperspectral data set that will be used for demonstration purposes. Section V-C evaluates the endmember extraction accuracy of the considered implementation. Finally, Section V-D shows the resources used for our hardware implementation and performs a comparison of our proposed FPGA design with an equivalent software version of the N-FINDR algorithm.

A. FPGA Architecture

The hardware architecture described in Section IV has been implemented using VHDL language for the specification of the N-FINDR module. Further, we have used the Xilinx ISE environment and the Embedded Development Kit (EDK) environment[5] to specify the complete system. The full system has been implemented on a ML410 board (see Fig. 10), a low-cost reconfigurable board with a single Virtex-4 XC4VFX60

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

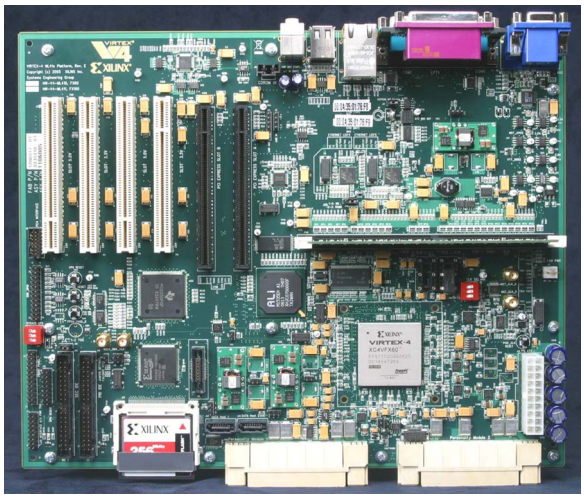

Fig. 10. Xilinx ML410 board with a Virtex-4 XC4VFX60 FPGA component.

FPGA component, a DDR2 SDRAM DIMM slot which holds up to 2 GB, a RS232 port, and some additional components not used by our implementation. We use a Xilinx Virtex-4 XC4VFX60 FPGA because it is based on the same architecture as other FPGAs [32] 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 immediately implement our design on it.

B. Hyperspectral Image Data Sets

Four different hyperspectral data sets have been used in our experiments.

- The first one is the well-known AVIRIS Cuprite scene [see Fig. 11(a)], collected in the summer of 1997 and available online in reflectance units after atmospheric correction.[6] The portion 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 MB. Water absorption and low SNR bands were removed prior to the analysis. 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. 11(b)], available in the form of a U.S. Geological Survey library (USGS)[7] will be used to assess endmember signature purity in this work.
- The second data set corresponds to a EO-1 Hyperion data set available in radiance units (i.e., it is not atmospherically corrected). The data were collected over the same Cuprite mining district as the aforementioned AVIRIS scene in the summer of 2001. In this case, we used a full EO-1 Hyperion flightline with much larger dimensions, i.e., 6479 × 256 pixels and 242 spectral bands out of which water absorption and low SNR bands were also removed prior to the analysis. Since the scene is not atmospherically corrected, it is not possible to establish comparisons between image endmembers and USGS spectral signatures, although we will use this scene to illustrate processing performance with a much larger data set (with a total size of around 800 MB).
- The third and fourth data sets correspond to two AVIRIS data sets of the Jasper Ridge Biological Preserve in California. The data set s are available in both radiance (uncorrected) and reflectance (atmospherically corrected) units. Each of the data sets, acquired on April 1998, consists of 512 × 614 pixels and 224 spectral bands (for a total size of around 140 MB 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. 12 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. 12. 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 miscalibration between field spectra and image spectra.

C. Endmember Extraction Accuracy Evaluation

In this subsection, we evaluate the endmember extraction accuracy of our proposed N-FINDR implementation using different hyperspectral scenes. First of all, we emphasize that our hardware version gives exactly the same results as our software implementation, which we have validated using N-FINDR's commercial implementation to guarantee that the three versions provide exactly the same results with the considered scenes. An important issue in endmember extraction is whether the hyperspectral data has been atmospherically corrected or not. For this purpose, we have used both atmospherically corrected and uncorrected data in our experiments to evaluate the impact of this process on the final endmember extraction results. It should be noted that, if the endmember extraction process is to be conducted onboard the imaging instrument, it would be necessary to decide if an onboard atmospheric correction module should be applied prior to endmember extraction. This may have an important impact on the full processing chain (particularly if we are targeting real-time analysis of hyperspectral images). Although this discussion is outside the scope of this paper, our experimental results below reveal that the N-FINDR can extract high-quality spectral endmembers from both atmospherically corrected and uncorrected data sets.

[6]http://aviris.jpl.nasa.gov
[7]http://speclab.cr.usgs.gov/spectral-lib.html

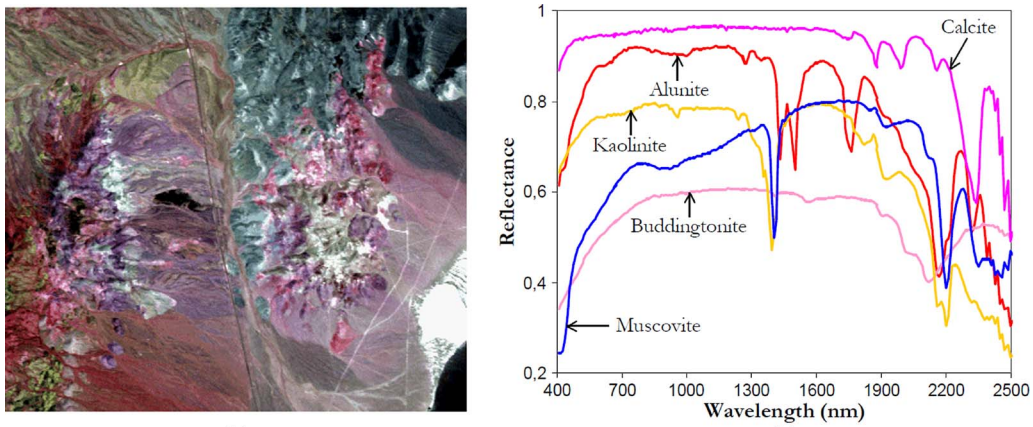

Fig. 11. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada. (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

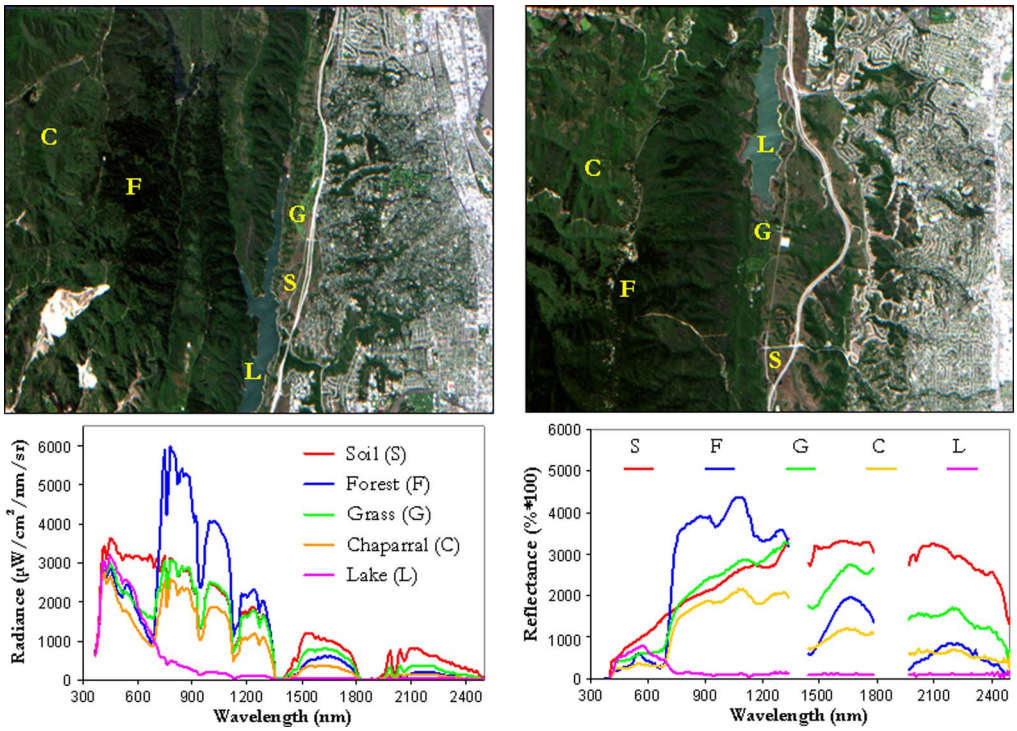

Fig. 12. AVIRIS hyperspectral images collected over Jasper Ridge Biological Preserve in radiance (left) and reflectance (right) units, along with the spectral signatures and spatial location of representative endmembers in the two considered scenes.

Before describing the obtained results, we first describe the metric used for quantitative comparison in our experiments. To reduce the impact of atmospheric interferers in our assessment, we use the spectral angle (SA) [14] between the most similar endmember detected by our implementation and the reference spectral signatures available for each scene. It should be noted that the SA between a pixel vector $\mathbf{X}(i,j)$ selected by the N-FINDR and a reference spectral signature $\mathbf{S}_k$ available *a priori* can be simply computed as

$$\mathrm{SA}\left[\mathbf{X}(i,j),\mathbf{S}_k\right] = \cos^{-1}\frac{\mathbf{X}(i,j)\cdot\mathbf{S}_k}{\|\mathbf{X}(i,j)\|\cdot\|\mathbf{S}_k\|} \tag{3}$$

i.e., the SA measures the angle formed by n-dimensional vectors. As a result, low SA scores mean high spectral similarity between the compared vectors. This spectral similarity measure is invariant in the multiplication of $\mathbf{X}(i,j)$ and $\mathbf{S}_k$ by constants, and, consequently, it is invariant before unknown multiplicative scalings that may arise due to differences in illumination and angular orientation [11]. This can help us to compensate the different acquisition conditions for a pixel in the original image and for a spectral signature collected on the ground (as it is the case for the reference USGS signatures used for the AVIRIS Cuprite image).

Due to the unavailability of reference spectral signatures in the EO-1 Hyperion data, we have conducted an experiment-based cross examination of endmember extraction accuracy with the AVIRIS Cuprite and AVIRIS Jasper Ridge data sets to assess the spectral similarity between the endmembers derived by our N-FINDR implementation and the reference signatures

TABLE II
SPECTRAL ANGLE SIMILARITY SCORES BETWEEN THE ENDMEMBERS EXTRACTED BY N-FINDR FROM THE AVIRIS CUPRITE SCENE AND THE AVAILABLE USGS SPECTRAL LIBRARY SIGNATURES

	Alunite	**Buddingtonite**	**Calcite**	**Kaolinite**	**Muscovite**
Radians	0.084	0.073	0.089	0.138	0.092
Degrees	4.812	4.182	5.099	7.906	5.271

available for these scenes, which comprise a set of USGS spectral library signatures in the case of AVIRIS Cuprite (available in reflectance units) and a set of image pixels labeled as spectrally pure in the case of AVIRIS Jasper Ridge (available both in radiance and reflectance units). The only input parameter for N-FINDR algorithm is the number of endmembers to be extracted. In our experiments, this number was set to $p = 16$ for the AVIRIS Cuprite scene and to $p = 19$ for the AVIRIS Jasper Ridge scenes after estimating the dimensionality of the data using the *virtual dimensionality* [37] concept.

Tables II and III show the SA values between the endmembers detected by our FPGA-based implementation and the reference spectral signatures available for the AVIRIS Cuprite and AVIRIS Jasper Ridge scenes, respectively. It should be noted that the tables only display the smallest SA scores of all endmembers extracted in each case with respect to each reference signature. The results are displayed in the form of radians and degrees (the range of values for the SA is [0, 90] degrees). These results are consistent with those published before in the literature (e.g., in [19]). A comparison of the endmember extraction by N-FINDR with regard to those achieved by other popular endmember extraction algorithms is also available in [19]. Of particular importance are the results reported on Table III, which indicate that N-FINDR can successfully derive spectral endmembers from both radiance and reflectance data. This opens the question on whether the endmember extraction process should be conducted before or after atmospheric correction. Since the atmospheric correction process can also be applied once the endmember pixel locations have been identified, our speculation is that the endmember extraction process can be performed in real time at the same time as the data are collected at the sensor, and without the need for a previous atmospheric correction module that should also be implemented in hardware to fully comply with real-time requirements. This topic will be subject to investigation in our future research.

D. Performance Evaluation

In this subsection, we conduct an experimental evaluation of the computational performance of our proposed FPGA implementation. For illustrative purposes, Table IV shows the resources used for our hardware implementation of the proposed N-FINDR algorithm design for different values of p (numbers of endmembers to be extracted), conducted on the 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 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 N-FINDR algorithm together with the DSP48Es multipliers.

As shown by Table IV, the percentage of hardware utilization increases as the number of endmembers to be identified increases. For the maximum value used in our experiments $(p = 21)$, the percentage of total hardware utilization is 97.40% thus reaching almost full hardware occupancy. However, other values of p tested in our experiments, e.g., $p = 19$, still leave room in the FPGA for additional algorithms. In turn, values of p superior to those used in our experiments are quite unfeasible due to our experiments with different scenes and previous results published in the endmember extraction literature (i.e., the number of endmembers is generally lower than the number of spectral bands). Hence, we believe that our hardware implementation will be able to cope with many different analysis scenarios. In any event, this value could be significantly increased in more recent FPGA boards, but we have decided to report results only with a board which is similar to a space-certified one 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. Hence, we have paid special attention to this issue. To reduce the I/O overheads, we have included DMA, and we have applied a prefetching approach to hide the communication latency. Basically, while the N-FINDR module is processing a set of data, the DMA is fetching the following set and storing it in the 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 N-FINDR module never needs to stop. 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 FIFO by the PowerPC instead of the DMA. In this version, the processing time was increased more than an order of magnitude so we can conclude that the resources used for the DMA (see Table IV) are well spent.

Finally, Table V reports the processing times measured by the considered FPGA implementation and by an equivalent software version developed in C language and executed on a PC with AMD Athlon 2.6 GHz processor and 512 Mb of RAM, for the hyperspectral considered data sets. 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 the table. In all cases, we report the value of p (number of endmembers to be extracted), which is scene dependent, the total size of the image in megabytes, and the speedup of the hardware implementation with regard to our software version.

To conclude this section, we emphasize that our reported FPGA processing times are still quite far from real-time

TABLE III
SPECTRAL ANGLE SIMILARITY SCORES BETWEEN THE ENDMEMBERS EXTRACTED BY N-FINDR FROM THE AVIRIS JASPER RIDGE SCENES (IN RADIANCE AND REFLECTANCE UNITS) AND THE AVAILABLE PURE SPECTRAL SIGNATURES IN BOTH SCENES

	Radiance data					**Reflectance data**				
	Soil	Forest	Grass	Chaparral	Lake	Soil	Forest	Grass	Chaparral	Lake
Radians	0.077	0.065	0.044	0.050	0.033	0.028	0.025	0.022	0.020	0.019
Degrees	4.411	3.724	2.521	2.864	1.890	1.604	1.432	1.260	1.145	1.088

TABLE IV
SUMMARY OF RESOURCE UTILIZATION FOR THE FPGA-BASED IMPLEMENTATION OF THE N-FINDR ALGORITHM FOR DIFFERENT NUMBERS OF ENDMEMBERS

Component	Number of endmembers (p)	Number of DSP48Es	Number of slice flip flops	Number of 4 input LUTs	Number of slices	Percentage of total	Maximum operation frequency (MHz)
N-FINDR module	9	92	6322	11031	6231	24.65	43.1
	16	128	12036	20779	11700	46.23	42.9
	18	128	15095	24247	14056	55.6	42.8
	19	128	16646	26763	17577	69.53	42.6
	21	128	20077	34155	24622	97.40	42.3
RS232 Transmitter	-	0	69	128	71	0.28	208
DMA Controller	-	0	170	531	367	1.45	102

TABLE V
PROCESSING TIMES MEASURED FOR THE HARDWARE IMPLEMENTATION AND FOR AN EQUIVALENT SOFTWARE VERSION FOR THE CONSIDERED HYPERSPECTRAL IMAGES

	AVIRIS Cuprite	**EO-1 Hyperion**	**AVIRIS Jasper Ridge**
Number of endmembers (p)	16	21	19
Total size (Megabytes)	50	800	140
Time software version (seconds)	502.06	9110.24	1851.34
Time hardware version (seconds)	13.46	239.11	49.35
Speedup	37.29	38.10	37.50

performance. For instance, the cross-track line scan time in AVIRIS, a push-broom instrument, is quite fast (8.3 to collect 512 full pixel vectors). This introduces the need to process a hyperspectral data set with 614×512 pixels (such as our AVIRIS Jasper Ridge scenes) in 4.98 s to fully achieve real-time performance. In our experiments, the processing times achieved in the considered FPGA board for the same volume of data are on the order of ten times this figure. Although we believe that the inclusion of more recent FPGA boards could significantly improve our reported processing times, we have decided to report results on a board which is very close to those certified for space operation in the hope of illustrating the real challenges to be faced when porting hyperspectral imaging algorithms into certified FPGA hardware. In future developments, we will also focus on improving our proposed implementation to achieve a better utilization of hardware resources and reduce the reported processing times, which in any event are considered to be acceptable in many remote sensing applications.

VI. CONCLUSION AND FUTURE RESEARCH LINES

One of the most important challenges to be addressed in hyperspectral imaging is the computational complexity of algorithm analysis resulting from the ever-increasing spatial, spectral, and temporal resolution of new hyperspectral remote sensing missions, many of them based on spaceborne platforms. With the recent advances in reconfigurable computing, particularly in FPGAs, hyperspectral imaging algorithms can now be implemented in high-performance FPGAs to accelerate their response times. An important technique for hyperspectral data exploitation is spectral unmixing, of which endmember extraction is a fundamental task. In this paper, we have discussed the role of FPGAs in hyperspectral remote sensing missions and further developed the first FPGA implementation of the N-FINDR algorithm for endmember extraction, which is selected here as a case study to illustrate the pros and cons of reconfigurable technology in the context of airborne and spaceborne remote sensing missions. Our experimental results, conducted on a Virtex-4 XC4VFX60 FPGA (a platform with the same architecture and similar area than radiation-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 results with compact size, which make our reconfigurable system appealing for onboard hyperspectral data processing.

Although our reported processing times are still far from real-time performance, further developments in both algorithm design and hardware availability lead us to believe that the goal of onboard real-time hyperspectral image processing in space will be accomplished in the near future. In this regard, the reconfigurability of FPGA systems opens many innovative perspectives, ranging from the appealing possibility of being able to adaptively select the data processing algorithm to be applied on board, out of a pool of available algorithms, from a control station on Earth immediately after the data are collected by the sensor, to the possibility of providing a real-time response in remote sensing applications with real-time requirements. As future work, we are investigating FPGA implementations of techniques for estimating the number of endmembers in the scene, such as the virtual dimensionality concept in [37], as well as techniques for estimating endmember abundances to provide a full spectral unmixing chain. We are also investigating other aspects, such as the possibility of keeping different configurations of the N-FINDR implementation with different number of endmembers in the same FPGA, to cope with the variable number of endmembers expected in different applications, in addition to an analysis of the sensitivity of the N-FINDR algorithm to the random initial values (in this work, all compared versions start from exactly the same set of random pixels although in the future more intelligent strategies for initialization could be applied). Future work should also perform comparisons between our hardware implementation of the matrix triangular method for volume determination with other techniques available in the literature that implement the same method in hardware using singular value decomposition [38], [39]. Finally, we are also evaluating other specialized hardware platforms for onboard hyperspectral data exploitation, such as commodity graphics processing units (GPUs). It should be noted that GPUs represent an interesting alternative for real-time implementation, but this type of hardware is still not certified for aerospace operation mainly due to its high power consumption when compared to FPGAs.

ACKNOWLEDGMENT

The authors gratefully thank Dr. Robert O. Green at NASA/JPL and Dr. Stephen G. Ungar at NASA/GSFC for, respectively, providing the AVIRIS and EO-1 Hyperion data used in our experimental assessment. Last but not least, the authors gratefully thank the three anonymous reviewers for their careful assessment of our manuscript and for providing very interesting recommendations for improving its technical quality and presentation.

REFERENCES

[1] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for Earth remote sensing," *Science*, vol. 228, no. 4704, pp. 1147–1153, Jun. 1985.

[2] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, M. R. Olah, and O. Williams, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, no. 3, pp. 227–248, Sep. 1988.

[3] 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 Appl.*, 2006, pp. 1–6.

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

[5] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: A survey," *J. VLSI Signal Process. Syst.*, vol. 28, no. 1/2, pp. 7–27, May/Jun. 2001.

[6] D. A. Buell, T. A. El-Ghazawi, K. Gaj, and V. V. Kindratenko, "Guest Editors' introduction: High-performance reconfigurable computing," *IEEE Comput.*, vol. 40, no. 3, pp. 23–27, Mar. 2007.

[7] 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 Comput.*, vol. 41, no. 2, pp. 69–76, Feb. 2008.

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

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

[10] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 44–57, Jan. 2002.

[11] M. Faraklioti and M. Petrou, "Illumination invariant unmixing of sets of mixed pixels," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 10, pp. 2227–2234, Oct. 2001.

[12] O. Duran and M. Petrou, "Spectral unmixing with negative and superunity abundances for subpixel anomaly detection," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 1, pp. 152–156, Jan. 2009.

[13] M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," in *Proc. SPIE*, 1999, vol. 3753, pp. 266–275.

[14] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York: Kluwer, 2003.

[15] J. B. Adams, M. O. Smith, and P. E. Johnson, "Spectral mixture modeling: A new analysis of rock and soil types at the Viking Lander 1 site," *J. Geophys. Res.*, vol. 91, no. B8, pp. 8098–8112, 1986.

[16] A. Plaza, J. A. Benediktsson, J. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, J. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, no. Supplement 1, pp. S110–S122, Sep. 2009.

[17] M. E. Schaepman, S. L. Ustin, A. Plaza, T. H. Painter, J. Verrelst, and S. Liang, "Earth system science related imaging spectroscopy—An assessment," *Remote Sens. Environ.*, vol. 113, pp. 123–137, 2009.

[18] 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, no. 3, pp. 529–545, Mar. 2001.

[19] 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, no. 3, pp. 650–663, Mar. 2004.

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

[21] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. New York: Wiley, 2007.

[22] B. Neil and A. Dawood, "Reconfigurable computers in space: Problems, solutions and future directions," in *Proc. Military Aerosp. Appl. Programable Logic Devices Conf.*, 1999, pp. 1–2.

[23] M. A. Fischman, A. C. Berkun, F. T. Cheng, W. W. Chun, E. Im, and R. Andraka, "Design and demostration of an advanced on-board processor for the second-generation precipitation radar," in *Proc. IEEE Aerosp. Conf.*, 2003, vol. 2, pp. 1067–1075.

[24] Triscend Corporation, A White Paper Configurable Processors: An Emerging Solution for Embedded System Design, 1998, A White Paper.

[25] Altera Corporation, Intellectual Property Selector Guide, Mar. 2001.

[26] Xilinx. [Online]. Available: http://www.xilinx.com/products/ipcenter/ppc405_virtex4.htm

[27] J. Tabero, H. Mecha, J. Septién, S. Román, and D. Mozos, "A vertex-list approach to 2D Hw multitasking management in RTR FPGAs," in *Proc. DCIS*, 2003, pp. 545–550.

[28] S. Román, J. Septién, H. Mecha, and D. Mozos, "Constant complexity management of 2D HW multitasking in run-time reconfigurable FPGAs," *Reconfigurable Comput., Archit. Appl.*, vol. 3985, *Lecture Notes in Computer Science*, pp. 187–192, 2006.

[29] J. Resano, J. A. Clemente, C. González, D. Mozos, and F. Catthoor, "Efficiently scheduling runtime reconfigurations," *ACM Trans. Design Autom. Electron. Syst.*, vol. 13, no. 4, pp. 58–69, Sep. 2008.

[30] J. A. Clemente, C. González, J. Resano, and D. Mozos, "A task graph execution manager for reconfigurable multi-tasking systems," *Microprocess. Microsyst.*, vol. 34, no. 2–4, pp. 73–83, Mar. 2010.
[31] J. T. Thomson, *Rad Hard FPGAs*. [Online]. Available: http://esl.eng.ohiostate.edu/~rstheory/iip/RadHardFPGA.doc
[32] Xilinx. [Online]. Available: http://www.xilinx.com/publications/prod_mktg/virtex5qv-product-table.pdf
[33] M. E. Winter, "A proof of the N-FINDR algorithm for the automated detection of endmembers in a hyperspectral image," in *Proc. SPIE*, 2004, vol. 5425, pp. 31–41.
[34] A. A. Green, M. Berman, P. Switzer, and M. D. Craig, "A transformation for ordering multispectral data in terms of image quality with implications for noise removal," *IEEE Trans. Geosci. Remote Sens.*, vol. 26, no. 1, pp. 65–74, Jan. 1988.
[35] R. A. Schowengerdt, *Remote Sensing: Models and Methods for Image Processing*, 2nd ed. New York: Academic Press, 1997.
[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, no. 7, pp. 1480–1490, Jul. 2001.
[37] C.-I. Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 608–619, Mar. 2004.
[38] C. Bobda, K. Danne, and A. Linarth, "Efficient implementation of the singular value decomposition on a reconfigurable system," *Field Programmable Logic Appl.*, vol. 2778, *Lecture Notes in Computer Science*, pp. 1123–1126, 2003.
[39] A. Ahmedsaid, A. Amira, and A. Bouridane, "Improved SVD systolic array and implementation on FPGA," in *Proc. IEEE Int. Conf. FPT*, 2003, vol. 1, pp. 35–42.

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 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.

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

He is a permanent professor in 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.

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

Currently, he is Associate Professor at the Computer Engineering Department, Universidad of Zaragoza, 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 from IMEC-laboratory since 2002. His research has been focused in hardware/software codesign, task scheduling techniques, dynamically reconfigurable hardware, and field programmable gate arrays design.

Antonio Plaza (SM'05) 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. He 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 has served 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.

Dr. Plaza 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. He 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.