FPGA Implementation of an Algorithm for Automatically Detecting Targets in Remotely Sensed Hyperspectral Images

Carlos González, Sergio Bernabé, Daniel Mozos, and Antonio Plaza

***Abstract*—Timely detection of targets continues to be a relevant challenge for hyperspectral remote sensing capability. The automatic target-generation process using an orthogonal projection operator (ATGP-OSP) has been widely used for this purpose. Hyperspectral target-detection applications require timely responses for swift decisions, which depend upon (near) real-time performance of algorithm analysis. Reconfigurable field-programmable gate arrays (FPGAs) are promising platforms that allow hardware/software codesign and the potential to provide powerful onboard computing capabilities and flexibility at the same time. In this paper, we present an FPGA implementation for the ATGP-OSP algorithm. 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-7 XC7VX690T FPGA and tested using real hyperspectral data collected by NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) over the Cuprite mining district in Nevada and the World Trade Center in New York. Experimental results demonstrate that our hardware version of the ATGP-OSP algorithm can significantly outperform a software version, which makes our reconfigurable system appealing for onboard hyperspectral data processing.**

***Index Terms*—Automatic target-generation process (ATGP), field-programmable gate arrays (FPGAs), hyperspectral imaging, 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 airborne visible infrared imaging spectrometer (AVIRIS) [2]. This system is now able to cover the wavelength region from 400 to 2500 nm using 224 spectral channels, at nominal spectral resolution of 10 nm. 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 wealth of spectral information available from latest-generation hyperspectral imaging instruments, which has substantially increased their spatial, spectral, and temporal resolutions, has quickly introduced new challenges in the analysis and interpretation of hyperspectral data sets. This often leads to the requirement of hardware accelerators to speedup computations, in particular, in analysis scenarios with real-time constraints in which onboard processing is generally required [3]. It is expected that, in future, hyperspectral sensors will continue increasing their spatial, spectral, and temporal resolutions (images with thousands of spectral bands are currently in operation or under development [4]). Such wealth of information has opened ground-breaking perspectives in several applications [5] (many of which with real-time processing requirements) such as environmental modeling and assessment for Earth-based and atmospheric studies, risk/hazard prevention and response including wildland fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination, target detection for military and defense/security purposes, urban planning and management studies [6].

Reconfigurable hardware solutions such as field-programmable gate arrays (FPGAs) have consolidated during the past years as one of the preferred choices for the fast processing of hyperspectral remotely sensed images [9]–[14] due to the following three main reasons. First, because of their smaller size, weight, and power consumption when compared with other high-performance computing systems, such as clusters of computers, multicore processors, and/or graphical processing units (GPUs) [15], [16]. The second reason becomes motivated by the current availability of FPGA devices with increased levels of tolerance to ionizing radiation in space, which converts them into the—nowadays—most widely used solution for onboard processing at earth observation satellites [17]. The last but never the least reason comes from the fact that FPGAs have the inherent ability to change their functionality through partial or full reconfiguration [18]. As an example, we have recently published the implementation of

Manuscript received July 06, 2015; revised October 26, 2015; accepted November 19, 2015. Date of publication January 11, 2016; date of current version September 30, 2016. This work was supported in part by the Spanish Ministry of Science and Innovation under reference READAR (TIN2013-40968-P), in part by the Extremadura Local Goverment and the European Union through the FEDER Fund (projects EI-14-0004-1 and EI-14-0007-1), and in part by the program Formación Posdoctoral under reference FPDI-2013-16280 from the Spanish Ministry of Economy and Competitiveness (MINECO).

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

A. Plaza is with the Department of Computer Technology and Communications, Polytechnic School of Cáceres, University of Extremadura, E-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/JSTARS.2015.2504427

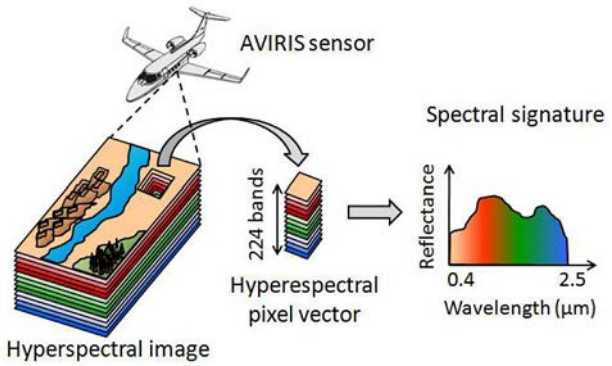

Fig. 1. Concept of hyperspectral imaging.

the HySime algorithm to estimate the number of endmembers [19], in which we parallel the most of every stage of the algorithm and apply reconfiguration between each stage. Thus, this last characteristic extends the useful life of remote sensing autonomous systems, since FPGAs permit changes to the usage model and the data-processing paradigm in space rather than hard-coding of all components prior to launch.

The special properties of hyperspectral data have significantly expanded the domain of many analysis techniques. In particular, algorithms for detecting (moving or static) targets, or targets that could expand their size (such as propagating fires) often require timely responses for swift decisions that depend upon high computing performance of algorithm analysis [7]. These algorithms are considered very important tasks for hyperspectral data exploitation in defense and security applications [8]. In the past, there have been several developments toward the implementation of target-detection algorithms in FPGA architectures [6]. Specifically, Chapter 15 in [6] generally discusses the use of FPGAs in detection applications and provides specific application case studies. Chapter 16 in [6] describes FPGA implementations of techniques for hyperspectral target-detection applications. Chapter 17 in [6] describes an on-board real-time processing technique for fast and accurate target detection and discrimination in hyperspectral data. Real-time implementations of several popular target detection and classification algorithms for hyperspectral imagery have also been discussed in [20]. Other implementations based on software optimizations for target-detection algorithms have also been explored in [21] and [22]. Although several techniques such as the automatic target-generation process (ATGP) [23] have been recurrently and successfully used for target-detection purposes in hyperspectral imagery, a full FPGA implementation of this algorithm is not yet available to the community. In a recent work [24], a comparison of different target and anomaly detection algorithms for real-time hyperspectral imaging has been conducted, but the ATGP algorithm was not included in the comparison. In this paper, we develop an FPGA-based hardware version of the automatic target-generation process based on an orthogonal subspace projector (ATGP-OSP) [10] using the pseudoinverse operation, which allows us to detect the necessary number of targets in a hyperspectral image. The results demonstrate that the proposed architecture, when mapped onto a Xilinx Virtex-7 XC7VX690T FPGA device, can significantly outperform a software version.

Algorithm 1. Pseudocode of ATGP-OSP

1: **inputs**: $\mathbf{F} \in \mathbf{R}^n$ and t; // $\mathbf{F}$ denotes an n-dimensional hyperspectral image with r pixels, and t denotes the number of targets to be detected
2: $\mathbf{U} = [x_0|0|, \cdots, |0]$; // $\mathbf{x}_0$ is the pixel vector with maximum length in $\mathbf{F}$
3: **for** $(i = 0; i < t; i++)\{$
4: $P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$; // $P_{\mathbf{U}}^{\perp}$ is a vector orthogonal to the subspace spanned by the columns of $\mathbf{U}$
5: $v = P_{\mathbf{U}}^{\perp}\mathbf{F}$ // $\mathbf{F}$ is projected onto the direction indicated by $P_{\mathbf{U}}^{\perp}$
6: $i = \arg\max_{\{1,\cdots,r\}} v\,[:, i]$; // The maximum projection value is found, where r denotes the total number of pixels in the hyperspectral image and the operator ':' denotes 'all elements'
7: $x_i \equiv \mathbf{U}\,[:, i+1] = \mathbf{F}\,[:, i]$; // The target matrix is updated
8: }**end for**
9: **outputs**: $\mathbf{U} = \{x_0, x_1, \cdots, x_{t-1}\}$

The remainder of the paper is organized as follows. Section II describes our implementation of the ATGP-OSP algorithm. Section III describes its parallel implementation on a Xilinx Virtex-7 XC7VFX690T FPGA. Section IV provides an experimental assessment of both target-detection accuracy and processing performance of the proposed FPGA-based algorithm, using well-known hyperspectral data sets (with quality ground-truth) collected by the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [2] over the Cuprite mining district in Nevada and over the World Trace Center in New York. Finally, Section V concludes with some remarks and hints at plausible future research lines.

II. Automatic Target-Generation Process

In this section, we briefly describe the original version of the ATGP target-detection algorithm that will be efficiently implemented using reconfigurable hardware in this work. This version is the ATGP algorithm that uses an orthogonal subspace projector (ATGP-OSP) for calculating the orthogonal projection using the pseudoinverse operation.

ATGP was initially developed to find spectral signatures using orthogonal projections [25]. Let $\mathbf{x}_0$ be an initial target signature (i.e., the pixel vector with maximum length in the original n-dimensional hyperspectral image $\mathbf{F} \in \mathbf{R}^n$). This algorithm uses an orthogonal projection operator which is given by the following expression:

$$P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T \tag{1}$$

where $\mathbf{U}$ is a matrix of spectral signatures, $\mathbf{U}^T$ is the transpose of this matrix, and $\mathbf{I}$ is the identity matrix. This orthogonal projection operator is applied to all image pixels, with $\mathbf{U} = [x_0]$. It then finds a target signature, denoted by $\mathbf{x}_1$, with the maximum projection in $\langle \mathbf{x}_0 \rangle^{\perp}$, which is the orthogonal complement space linearly spanned by $\mathbf{x}_0$. A second target signature $\mathbf{x}_2$ can then be found by applying another orthogonal subspace projector $P_{\mathbf{U}}^{\perp}$ with $\mathbf{U} = [x_0, x_1]$ to the original image, where the

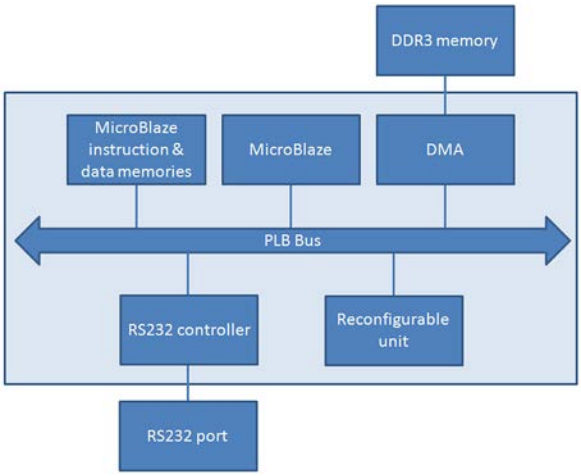

Fig. 2. Hardware architecture used to implement the complete system.

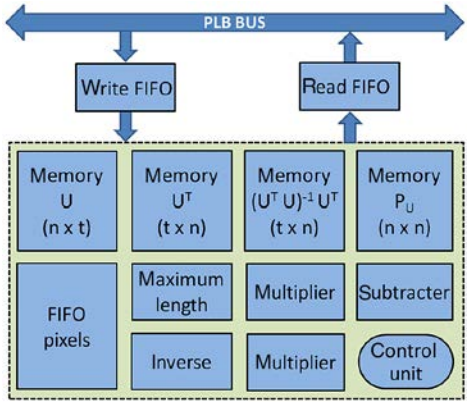

Fig. 3. Modules used to implement the ATGP-OSP algorithm.

target signature that has the maximum orthogonal projection in $\langle \mathbf{x}_0, \mathbf{x}_1 \rangle^{\perp}$ is selected as $\mathbf{x}_2$. The above procedure is repeated until a set of target pixels $\{x_0, x_1, \ldots, x_{t-1}\}$ is extracted, where t is an input parameter to the algorithm. This method is summarized in Algorithm 1.

III. FPGA Implementation of the ATGP-OSP Algorithm

Fig. 2 describes the general architecture of the hardware used to implement the ATGP-OSP algorithm, along with the I/O communications. For data input, we have used a DDR3 SDRAM and a direct memory access (DMA) controlled by the MicroBlaze using a prefetching approach. The reconfigurable unit is used to implement our version of the ATGP-OSP algorithm. Finally, an RS232 controller is used to send the estimated number of endmembers via an RS232 port. During the development, we have also used this port for debugging.

We have a general architecture to make hardware/software codesign with a fixed part (MicroBlaze and its memory, PLB bus, DMA and RS232 controller) and a reconfigurable part (the reconfigurable unit). This reconfigurable unit can be used to completely implement an algorithm or some parts of it (the parts that are best suited to a hardware implementation). MicroBlaze is a low-performance and low-consumption soft-core embedded processor. The main advantage of having the MicroBlaze is that it facilitates the reconfiguration process and the adaptation of the system for different I/O (SD Card, Ethernet, PCI bus, etc.).

Fig. 3 shows the modules used to implement the ATGP-OSP algorithm together with the I/O communications toward the PLB bus. In the following, we explain each of the modules and provide a step-by-step description of how the proposed architecture performs the target detection from a hyperspectral image.

The first decision that was taken about the ATGP-OSP algorithm was how to calculate the inverse of the result of $\mathbf{U}^T\mathbf{U}$ matrix multiplication. We tried to implement a method that would take advantage of the hardware features, such as parallelization, i.e., the possibility of performing simultaneously separate and independent operations, to reduce the execution time. Considering this, we tested several methods to calculate the inverse of a matrix [26]: adjoint method, method of partitions, frame's method, Newton's formula, and Gauss–Jordan elimination. Gauss–Jordan elimination method was finally selected to be used in the ATGP-OSP algorithm because it has desirable features in terms of its ulterior hardware implementation (as we will see below, it is highly parallelizable).

The Gauss–Jordan method can be stated as follows: If we have matrix $\mathbf{A}$, and a sequence of elementary row operations $\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_k$ which will reduce $\mathbf{A}$ to $\mathbf{I}$, then, the same sequence of row operations will reduce $\mathbf{I}$ to $\mathbf{A}^{-1}$.

Thus, to calculate the inverse matrix, we need to accomplish the following steps.

1) Set the matrix (must be square) and append the identity matrix of the same dimension to it.
2) Reduce the left matrix to row echelon form using elementary row operations for the whole matrix (including the right one).
3) As a result of this process, we obtain the inverse calculated on the right.
4) If a determinant of the main matrix is zero, inverse does not exist.

This method is fully parallelizable considering that one elementary row operation can be done at once and different elementary row operations can be done at the same time. Using these parallel properties, we rewrite the Gauss–Jordan method in the way shown in Algorithm 2.

Fig. 4 describes the architecture used to calculate the inverse of a matrix using the Gauss–Jordan elimination method following Algorithm 2. As we can see, it has two memories to store a matrix and its inverse, two data paths to calculate the inverse and two control units to read and write into the memories.

Fig. 5 shows the architecture of the *data path* used in our design to implement the inverse 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.

Fig. 6 describes the architecture of the *maximum length* module used in our design to find the initial target signature (in

Algorithm 2. Pseudocode of the Gauss–Jordan method. In this algorithmic description, we will assume for simplicity and standard notation throughout the paper that $\mathbf{A}[i][j] \equiv a_{ij}$.

row $= [0, \ldots, \mathrm{n}-1]$ // n denotes the size of the square matrix **A**
$\mathbf{A}^{-1} = \mathbf{I}$

//Forward Elimination to build an upper triangular matrix
for$(i = 0; i < n; i++)\{$
 if(**A**[row$[i]][i] == 0)\{$
  for$(j = i+1; j < n; j++)\{$
   if(**A**[row$[j]][j]! = 0)\{$
    row $[i]$ = row$[j]$;
    row$[j]$ = row$[i]$; // This operation is done in parallel with the previous one
    break;
   }end if
  }end for
 }end if
if(**A**[row $[i]][i] == 0)$ **error** "Matrix is singular";
for$(j = i+1; j < n; j++)\{$
  A[row$[j]]$ = A[row$[j]]$ − A[row$[i]]$ ∗ (A[row$[j]][i]$/ A[row$[i]][i]$);
  $\mathbf{A}^{-1}[row[j]] = \mathbf{A}^{-1}[row[j]] - \mathbf{A}^{-1}[row[i]] * (\mathrm{A}[\mathrm{row}[j]][i]/\mathrm{A}[\mathrm{row}[i]][i])$; // This operation is done in parallel with the previous one
 }end for
}end for

//Backward Elimination to build a diagonal matrix
for$(i = n-1; i > 0; i--)\{$
 for$(j = i-1; j >= 0; j--)\{$
  A[[row$[j]]$ = A[[row$[j]]$ − [A[[row$[i]]$ ∗ (A[[row$[j]][i]$/ A[[row$[i]][i]$);
  $\mathbf{A}^{-1}[[row[\mathrm{j}]] = \mathbf{A}^{-1}[[row[j]] - \mathbf{A}^{-1}[[row[i]] * (A[[\mathrm{row}[j]][i]/A[[\mathrm{row}[i]][i])$; // This operation is done in parallel with the previous one
 }end for
}end for

//Last division to build an identity matrix
for$(i = 0; i < n; i++)\{$
  $\mathbf{A}^{-1}[row[i]] = \mathbf{A}^{-1}[row[i]] * (1/\mathrm{A}[\mathrm{row}[i]][i])$;
\}**end for**

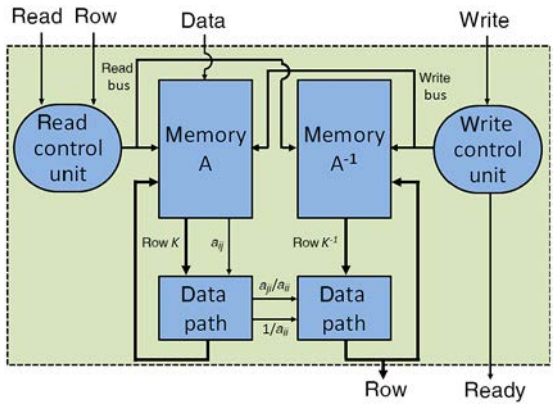

Fig. 4. Hardware architecture used to calculate the inverse of a matrix by Gauss–Jordan elimination.

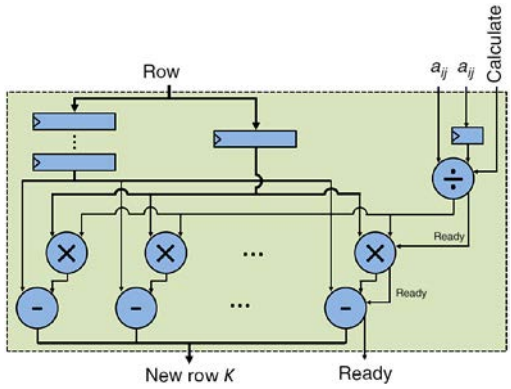

Fig. 5. Data path to implement the inverse process.

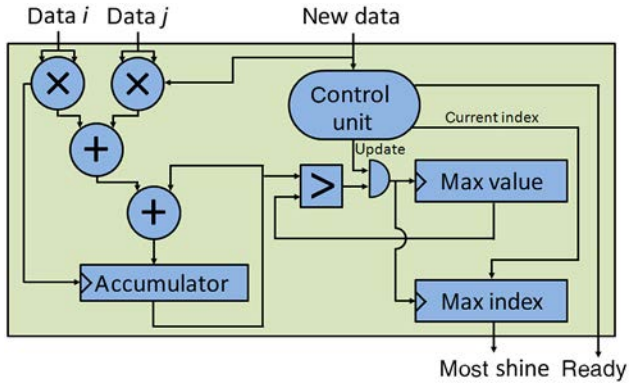

Fig. 6. Hardware architecture used to find the pixel with maximum length.

our case, the pixel vector with maximum length in the original n-dimensional hyperspectral image) and the pixel vector with maximum length after applying the orthogonal projection operator in each iteration. To find the length of a pixel $\mathbf{f}_i$, we calculate the dot-product of $\mathbf{f}_i$ with itself using the expression $\sum_{k=1}^{N} \mathbf{f}_i(k) \times \mathbf{f}_i(k)$. Taking into account that we can read from external memory two-pixel components at the same time and the multiplication $P_{\mathbf{U}}^{\perp}\mathbf{f}_i$ are performed in a way that we also obtain two-pixel components at the same time (it is explained later), the hardware needed to compute the dot-product which is composed of two multipliers, two adders, and a register. When the length of a new pixel is calculated (control unit knows the number of bands n), the result of the dot-product is compared with the previous maximum value. If the result is a new maximum, it will be stored for future comparisons together with its corresponding index into the hyperspectral image.

Fig. 7 shows the architecture of the *multiplier* used in our design to perform the dot-product of two given vectors. It is composed of a row of multipliers and a tree of adders so that for two vectors $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, b_1, \ldots, b_n]$, we first calculate the product element by element resulting $[a_0 \times b_0, a_1 \times b_1, \ldots, a_n \times b_n]$. Then, each of these multiplications is added together 2 to 2, resulting $[a_0 \times b_0 + a_1 \times b_1, a_2 \times b_2 + a_3 \times b_3, \ldots, a_{n-1} \times b_{n-1}$

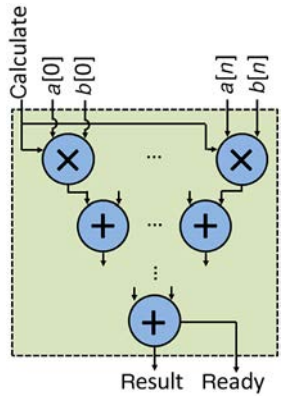

Fig. 7. Hardware architecture used to multiply two vectors.

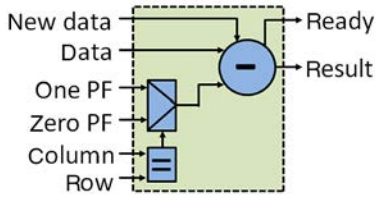

Fig. 8. Hardware arquitecture used to implement I-A operation.

$+ a_n \times b_n]$ at the first level. The addition 2 to 2 is repeated through different levels until we obtain the dot-product. Vectors can be of any dimension less than or equal to a maximum. This module is used for matrix multiplication so we have vectors with size n (the maximun) and t (the number of targets). We use a pipeline architecture so we can perform row by column multiplications consecutively cycle by cycle.

Fig. 8 describes the module responsible to perform the subtraction in the operation $\mathbf{I} - \mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$. When a new element a_{ij} is calculated in the last multiplication $\mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$, this module calculates $1 - a_{ij}$ when $i = j$, i.e., it is a diagonal element, and $0 - a_{ij}$ for the rest of the matrix elements.

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 ATGP-OSP module is processing a set of data, the DMA is fetching the following set and storing it in the FIFO.

Fig. 9 describes the architecture used for pixel input, composed of n FIFOs and a control unit. With this architecture, pixel components are written by twos (we can read from external memory two pixel components at the same time) and we can read one complete pixel. We implement a prefetching technique to hide the latency of the input communication.

Following the same idea, we implement the rest of the memories but with only one data input. Therefore, we write the elements of a matrix one by one but we can read one complete row or column (two in the case of the $P_{\mathbf{U}}^{\perp}$ memory). Memories U and U^T are really implemented using two physical memories, one accesible by rows and the other by columns, to cover all matrix multiplications of the algorithm.

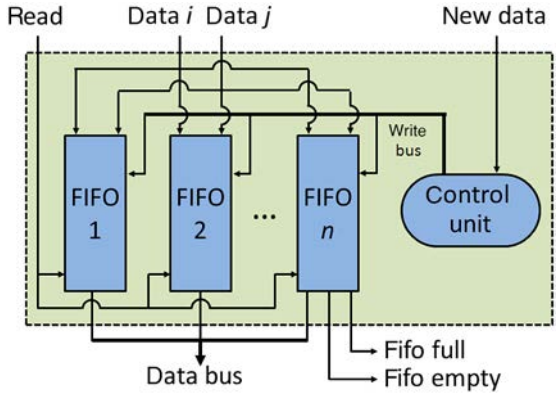

Fig. 9. Hardware architecture used for pixel input.

To conclude this section, we provide a step-by-step description of how the proposed architecture performs the target detection from a hyperspectral image.

1) We read from the *write FIFO* all pixels and calculate the pixel with maximun length in the original hyperspectral image.
2) The index of the pixel with maximun length is written in the read FIFO.
3) Microblaze reads this index and writes the pixel in the write FIFO. Then, it is read and stored in $\mathbf{U}$ and $\mathbf{U}^T$ memories. Henceforth, all pixels are written in write FIFO using a prefetching approach and stored in the data FIFO.
4) We multiply $\mathbf{U}^T\mathbf{U}$ storing the result in the $\mathbf{A}$ memory of the inverse module and in parallel we initialize $\mathbf{A}^{-1}$ matrix with the identity matrix.
5) We calculate the inverse of $\mathbf{U}^T\mathbf{U}$ following Algorithm 2.
6) We multiply $\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$ storing the result in the $\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$ memory.
7) We multiply $\mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$, perform $I - \mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T$, and store the result in the $P_{\mathbf{U}}^{\perp}$ memory.
8) We read a complete pixel from the data FIFO, multiply $P_{\mathbf{U}}^{\perp} f$, and calculate its length. When the length of all pixels is calculated, we iterate from step 2) until all targets are detected.

IV. Experimental Results

This 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 target-detection accuracy of the considered implementation. Finally, Section IV-D shows the resources used for our hardware implementation and performs a comparison of our proposed FPGA design with an equivalent software version.

A. FPGA Architecture

The hardware architecture described in Section III has been implemented using VHDL language for the specification of the ATGP-OSP algorithm. Moreover, we have used the

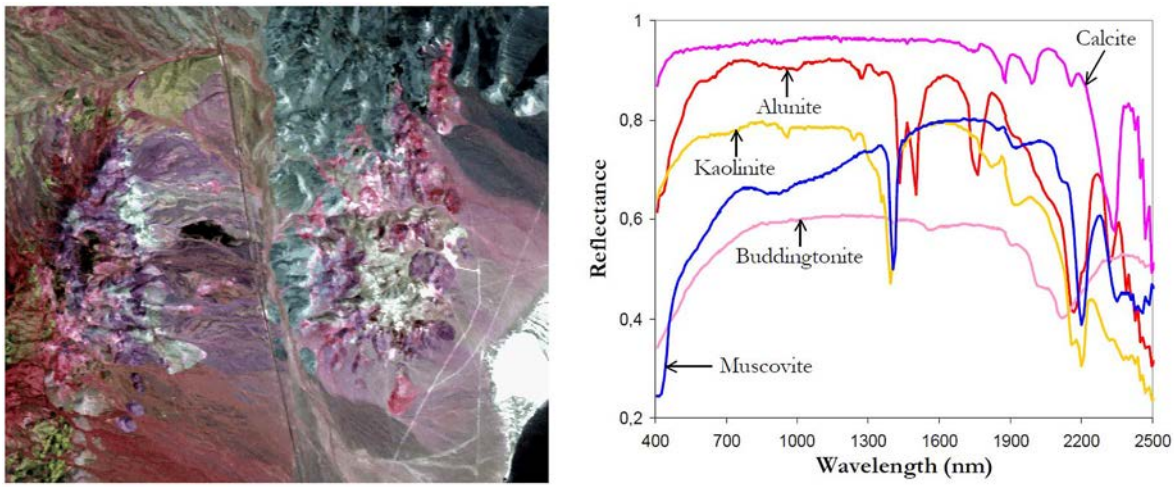

Fig. 10. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada. (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

Fig. 11. (a) False color composition of an AVIRIS hyperspectral image collected by NASAs Jet Propulsion Laboratory over lower Manhattan on September 16, 2001. (b) Location of thermal hot spots in the fires observed in World Trade Center area, available [online]: http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif.

Xilinx ISE environment and the Embedded Development Kit (EDK) environment[1] to specify the complete system. The full system has been implemented on a VC709 board, a reconfigurable board with a single Virtex-7 XC7VX690T, two DDR3 SDRAM DIMM slots which holds up to 4 GB each one, an RS232 port, and some additional components not used by our implementation.

B. Hyperspectral Image Data Sets

The hyperspectral data sets used in these experiments are the well-known AVIRIS Cuprite scene [see Fig. 10(a)], available online in reflectance units,[2] and the AVIRIS World Trade Center (WTC) scene [see Fig. 11(a)]. These scenes have been widely used to validate the accuracy and performance of target-detection algorithms.

The AVIRIS Cuprite scene comprises a relatively large area (350 lines by 350 samples and 20-m pixels) and 224 spectral bands between 0.4 and 2.5 μm, with nominal spectral resolution of 10 nm. 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),[3] will be used to assess ATGP-OSP algorithm in this paper.

For the second image, the instrument was flown by NASA's Jet Propulsion Laboratory over the World Trade Center

[1] Available: [Online] http://www.xilinx.com/ise/embedded/edk_pstudio.htm

[2] Available: [Online] http://aviris.jpl.nasa.gov

[3] Available: [Online] http://speclab.cr.usgs.gov/spectral-lib.html

TABLE I
SPECTRAL ANGLE VALUES (IN DEGREES) BETWEEN TARGET PIXELS AND KNOWN GROUND TARGETS FOR ATGP-OSP ALGORITHM OVER THE WTC SCENE

	A	B	C	D	E	F	G	H
ATGP-OSP	0.00°	14.74°	0.00°	27.58°	20.47°	7.29°	4.33°	31.51°

TABLE II
SPECTRAL ANGLE VALUES (IN DEGREES) BETWEEN TARGET PIXELS AND KNOWN GROUND TARGETS FOR ATGP-OSP ALGORITHM OVER THE CUPRITE SCENE

	Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite
ATGP-OSP	4.91°	4.49°	9.73°	10.89°	5.43°

(WTC) area in New York City on September 16, 2001, just 5 days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. The data set consists of 614×512 pixels, 224 spectral bands, and a total size of (approximately) 140 MB. The spatial resolution is 1.7 m/pixel. The leftmost part of Fig. 11 shows a false color composite of the data set selected for experiments using the 1682, 1107, and 655 nm channels, displayed as red, green, and blue, respectively. Vegetated areas appear green in the leftmost part of Fig. 11, while burned areas appear dark gray. Smoke coming from the WTC area (in the red rectangle) and going down to south Manhattan appears bright blue due to high spectral reflectance in the 655 nm channel. Extensive reference information, collected by U.S. Geological Survey (USGS), is available for the WTC scene. In this work, we use a U.S. Geological Survey thermal map which shows the target locations of the thermal hot spots at the WTC area, displayed as bright red, orange, and yellow spots at the rightmost part of Fig. 11 and labeled as capital letters (A–H) in the same figure. Here, some of the targets are full-pixel in nature (meaning that they occupy several pixels, such as A and C) while other targets are subpixel in nature and, hence, more difficult to detect. The map is centered at the region where the towers collapsed, and the temperatures of the targets range from 700 to 1300 F. The thermal map displayed in the rightmost part of Fig. 11 will be used in this work as ground-truth to validate the target-detection accuracy of the proposed parallel algorithms and their respective serial versions.

C. Target-Detection Accuracy Evaluation

It is important to emphasize that our hardware version of the ATGP-OSP algorithm provides exactly the same results as a software version of the algorithm, implemented using the Intel ICC compiler and optimized via compilation flags to exploit data locality and avoid redundant computations [29].

Table I shows the spectral angle distance (SAD) [27] values (in degrees) between the most similar target pixels detected by ATGP-OSP algorithm, and the pixel vectors at the known target positions, labeled from "A" to "H," in the AVIRIS World Trade Center image. It should be noted that the SAD between a target t_i detected by the ATGP-OSP and a reference spectral signature s_j is given by $SAD(t_i, s_j) = cos^{-1}(t_i \cdot s_j / \|t_i\| \cdot \|s_j\|)$. For this scene, the number of target pixels to be detected was set to $t = 30$ after calculating the virtual dimensionality (VD) of the data [28]. As shown in Table I, the ATGP-OSP algorithm extracted targets that were similar, spectrally, to the known ground-truth targets (these versions were able to perfectly detect the targets labeled as "A" and "C," and had more difficulties in detecting very small targets).

Table II shows the SAD values (in degrees) between the most similar target pixels detected by ATGP-OSP algorithm, and the pixel vectors at the known positions of the target minerals in the AVIRIS Cuprite image. In all cases, the number of target pixels to be detected was set to $t = 19$ after calculating the VD. As shown in Table II, SAD values are low, so again the ATGP-OSP-algorithm-extracted targets were similar, spectrally, to the known ground-truth targets. Here, it should be noted that the target detection performance in this example is uneven for different minerals. This is related to different aspects such as the atmospheric correction conducted on the Cuprite scene and the spectral similarity between the reference (ground) signatures and the endmembers extracted from the image. However, in all cases, the SAD values are quite low, indicating good detection performance for all considered targets.

D. Performance Evaluation

In this section, we conduct an experimental evaluation of the computational performance of our proposed FPGA implementation. For illustrative purposes, Table III shows the resources used for our proposed hardware implementation of the ATGP-OSP algorithm. The FPGA design was implemented on the Xilinx Virtex-7 XC7VX690T FPGA of the VC709 board. This FPGA has a total of 866 400 slice registers, 433 200 slice look-up tables (LUTs), and 134 381 LUT-FF pairs. In addition, the FPGA includes some heterogeneous resources such as 3600 DSP48E1s and 1470 distributed block RAMs. In our implementation, we took advantage of these resources to optimize the design. Block RAMs are used to implement the FIFOs and the memories, so a vast majority of the slices are used for the implementation of the ATGP-OSP algorithm together with the DSP48E1s.

Finally, Table IV reports the processing times obtained for the hardware implementation of the ATGP-OSP algorithm on the considered FPGA architecture and for the two considered hyperspectral scenes. Software version was executed in the Intel Xeon CPU E5-2695 v3 processor with 14 cores running at 2.30 GHz, with 64 GB of DDR3 RAM memory and implemented using the Intel ICC compiler with optimization flag –O3 to exploit data locality and avoid redundant computations. This software version have been implemented using a modified strategy performed in [29] and OpenMP. In previous versions, the potential bottleneck in the implementation was to calculate the projections, where the reduction process was included. In this paper, the projections are calculated without using locking routine (omp_set_lock) and (omp_unset_lock) improving the scalability of the algorithm with regard to [29]. In all cases, we report the total size of the image in megabytes and the speedup of the hardware implementation with regard to the

TABLE III
SUMMARY OF RESOURCE UTILIZATION FOR THE FPGA-BASED IMPLEMENTATION OF THE ATGP-OSP ALGORITHM TO PROCESS UP TO 256 BANDS AND 32 TARGETS

	Number of BRAMs	Number of DSP48E1s	Number of slice LUTs	Number of LUT-FF pairs	Number of slice registers	Maximum operation frequency (MHz)
Multiplicator	0	1278	97527	8160	16830	80
Inverse	65	320	30415	4456	6950	79
Maximum length	0	5	567	110	143	76
Subtracter	0	5	427	108	129	85
FIFO pixels	128	0	5150	2056	4624	355
Memory U	256	0	24276	0	0	355
Memory U^T	256	0	24276	0	0	355
Memory $(U^TU)^{-1}U^T$	32	0	3092	0	0	355
Memory $P_U^{\perp}$	256	0	24276	0	0	355
Total (%)	994 (68%)	2874 (80%)	132802 (31%)	12521 (12%)	22962 (3%)	72

TABLE IV
PROCESSING TIMES MEASURED FOR THE HARDWARE IMPLEMENTATION AND FOR OUR OPENMP SOFTWARE IMPLEMENTATION OF THE ATGP-OSP ALGORITHM

	AVIRIS Cuprite	AVIRIS WTC
Total size (MB)	50	140
ATGP-OSP (SW 1 thread) (s)	20.14	541.51
ATGP-OSP (SW 2 threads) (s)	10.26	270.39
ATGP-OSP (SW 4 threads) (s)	5.33	138.08
ATGP-OSP (SW 8 threads) (s)	3.04	71,88
ATGP-OSP (HW) (s)	3.31	15.62
Speedup	0.92	4.6

software version using eight threats. In our experiments, the processing times achieved in the FPGA board show a similar time when processing the AVIRIS Cuprite scene and a speedup over 4.6 when it comes to the WTC AVIRIS scene. It is also remarkable that the obtained speedups increase with the image size.

To conclude this section, we emphasize that our reported FPGA processing times are still quite far from real-time performance. For instance, the cross-track line scan time in AVIRIS, a push-broom instrument, is quite fast (8.3 ms to collect 512 full-pixel vectors). This introduces the need to process the WTC scene in less than 5.09 s (and the cuprite scene in less than 1.98 s) to fully achieve real-time performance. 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.

V. CONCLUSION AND FUTURE RESEARCH LINES

In this paper, we have developed an FPGA implementation of the automatic target-generation process using an orthogonal subspace projector (ATGP-OSP), a popular approach to detect targets in remotely sensed hyperspectral data. Our experimental results, conducted on a Virtex-7 XC7VX690T FPGA, demonstrate that our hardware implementation can significantly outperform (in terms of computation time) a software version, with compact size, which make our reconfigurable system appealing for onboard hyperspectral data processing. Furthermore, the existence of radiation-hardened FPGAs offers the appealing possibility of adaptively selecting a hyperspectral processing algorithm to be applied onboard (out of a pool of available algorithms) from a control station on Earth.

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. As future work, we will study the implementation on FPGA of an automatic target-generation process including Gram–Schmidt method (ATGP-GS) for calculating orthogonal projections instead of using an orthogonal subspace projector that includes a very expensive operation, i.e., the calculation of the inverse, in the traditional implementation of this algorithm, to obtain real-time performance. Power consumption is also an important issue. Since FPGA tools currently do not provide accurate results in terms of power at the synthesis stage [30], [31], we have considered as a future issue to develop a measurement system that allows us to determine, once a complete hyperspectral unmixing has been physically mapped onto a FPGA device, not only the power consumption of the whole board but also the power consumption of its individual constituents.

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, 1985.
[2] R. O. Green *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, no. 3, pp. 227–248, 1988.
[3] A. Plaza, "Special issue on architectures and techniques for real-time processing of remotely sensed images," *J. Real-Time Image Process.*, vol. 4, no. 3, pp. 191–193, 2009.
[4] A. Plaza and C.-I. Chang, "Special issue on high performance computing for hyperspectral imaging," *Int. J. High Perform. Comput. Appl.*, vol. 22, pp. 363–448, 2008.

[5] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, no. 1, pp. 110–122, 2009.
[6] A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*. Boca Raton, FL, USA: CRC Press, 2007.
[7] A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral Imagery," *J. Parallel Distrib. Comput.*, vol. 66, no. 3, pp. 345–358, 2006.
[8] A. Plaza and C.-I. Chang, "Clusters versus FPGA for parallel processing of hyperspectral imagery," *Int. J. High Perform. Comput. Appl.*, vol. 22, no. 4, pp. 366–385, 2008.
[9] S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, and A. Plaza, "The Promise of Reconfigurable Computing for Hyperspectral Imaging Onboard Systems: A Review and Trends," *Proc. IEEE*, vol. 101, no. 3, pp. 698–722, Mar. 2013.
[10] S. Bernabe, S. Lopez, A. Plaza, R. Sarmiento, and P. G. Rodriguez, "FPGA Design of an Automatic Target Generation Process for Hyperspectral Image Analysis," *in Proc. IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2011, pp. 1010–1015.
[11] S. Lopez, P. Horstrand, G. M. Callico, J. F. Lopez, and R. Sarmiento, "A Novel Architecture for Hyperspectral Endmember Extraction by Means of the Modified Vertex Component Analysis (MVCA) algorithm," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 6, pp. 1837–1848, Dec. 2012.
[12] C. Gonzalez, D. Mozos, J. Resano, and A. Plaza, "FPGA implementation of the N-FINDR algorithm for remotely sensed hyperspectral image analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 2, pp. 374–388, Feb. 2012.
[13] C. Gonzalez, J. Resano, A. Plaza, and D. Mozos, "FPGA implementation of abundance estimation for spectral unmixing of hyperspectral data using the image space reconstruction algorithm," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 1, pp. 248–261, Feb. 2012.
[14] C. Gonzalez, J. Resano, D. Mozos, A. Plaza, and D. Valencia, "FPGA implementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis," *EURASIP J. Adv. Signal Process.*, vol. 2010, no. 969806, pp. 1–13, 2010.
[15] 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.
[16] C. Gonzalez, S. Sanchez, A. Paz, J. Resano, D. Mozos, and A. Plaza, "Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing," *Integr. VLSI J.*, vol. 46, no. 2, pp. 89–103, 2013.
[17] L. Sterpone, M. Porrmann, and J. Hagemeyer, "A Novel Fault Tolerant and Runtime Reconfigurable Platform for Satellite Payload processing," *IEEE Trans. Comput.*, vol. 62, no. 8, pp. 1508–1525, Aug. 2013.
[18] J. A. Clemente, C. Gonzalez, J. J. Resano, and D. Mozos, "A Hardware Implementation of a Run-Time Scheduler for Reconfigurable Systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 7, pp. 1263–1276, Jul. 2011.
[19] C. Gonzalez, S. Lopez, D. Mozos, and R. Sarmiento, "FPGA implementation of the HySime algorithm for the determination of the number of endmembers in hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2870–2883, Jun. 2015.
[20] Q. Du and R. Nekovei, "Fast real-time onboard processing of hyperspectral imagery for detection and classification," *J. Real-Time Image Process.*, vol. 4, no. 3, pp. 273–286, 2009.
[21] C.-I. Chang, H. Ren, and S. S. Chiang, "Real-time processing algorithms for target detection and classification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 4, pp. 760–768, Apr. 2001.
[22] J. M. Molero, E. M. Garzon, I. Garcia, and A. Plaza, "Analysis and optimizations of global and local versions of the RX algorithm for anomaly detection in hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 2, pp. 801–814, Apr. 2013.
[23] H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1232–1249, Oct. 2003.
[24] B. Yang, L. Gao, A. Plaza, M. Yang, and B. Zhang, "Dual mode FPGA implementation of target and anomaly detection algorithms for real-time hyperspectral imaging," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2950–2961, Jun. 2015.
[25] H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1232–1249, Oct. 2003.
[26] D. Greenspan, "Methods of matrix inversion," *Amer. Math. Mon.*, vol. 62, no. 5, pp. 303–318, 1955.
[27] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Norwell, MA, USA: Kluwer, 2003.
[28] Q. Du and C.-I. Chang, "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.
[29] S. Bernabe, S. Sanchez, A. Plaza, S. Lopez, J. A. Benediktsson, and R. Sarmiento, "Hyperspectral unmixing on GPUs and multi-core processors: A comparison," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 3, pp. 1386–1398, Jun. 2013.
[30] G. Iles, J. Jones, and A. Rose, "Experience powering Xilinx Virtex-7 FPGAs," *in Proc. Topical Workshop Electron. Particle Phys. (TWEPP)*, 2013, pp. 23–27.
[31] D. Ghringer, J. Obie, M. Hbner, and J. Becker, "Impact of Task Distribution, Processor Configurations and Dynamic Clock Frequency Scaling on the Power Consumption of FPGA-based multiprocessors," *in Proc. 5th Int. Workshop Reconfig. Commun. Centric Syst. Chip (ReCoSoC)*, 2010, pp. 13–20.

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

Currently, he is a Teaching Assistant with the Department of Computer Architecture and Automation, Universidad Complutense Madrid. As a Research Member of GHADIR group, he mainly focuses on applying runtime 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, and also the acceleration of artificial intelligence algorithms applied to games.

Dr. Gonzalez was the recipient of the Design Competition of the 2009 and 2010 IEEE International Conferences on Field Programmable Technology (FPT) and the second Prize in 2012, and the Best Paper Award of an Engineer under 35 years old at the 2011 International Conference on Space Technology.

Sergio Bernabé received the Computer Engineering degree and the M.Sc. degree in computer engineering from the University of Extremadura, Cáceres, Spain, in 2010, and the joint Ph.D. degree between the University of Iceland, Reykjavík, Iceland, and the University of Extremadura, in 2014.

He has been a Visiting Researcher at the Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, Las Palmas, Spain, and also at the Computer Vision Laboratory (LVC), Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil. He was a Postdoctoral Researcher (funded by FCT) with the Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, and is now a Postdoctoral Researcher (funded by the Spanish Ministry of Economy and Competitiveness) with the Complutense University of Madrid, Madrid, Spain. His research interests include the development and efficient processing of parallel techniques for different types of high-performance computing architectures.

Dr. Bernabé currently serves as an Active Reviewer of international conferences and international journals, including the *IEEE Journal of Selected Topics in Applied Earth Observation and Remote Sensing*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and the *IEEE Geoscience and Remote Sensing Letters*. He was the recipient of the 2013 Best Paper Award of the JSTARS journal and the Best Ph.D. Dissertation Award at the University of Extremadura, Cáceres, Spain, in 2015.

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

Currently, he is a Full-Professor with the Computer Architecture and Automation Department, Universidad Complutense de Madrid, where he leads the GHADIR Research Group on dynamically reconfigurable architectures. He has been Vice-Dean for students during seven years and currently he is the Dean of the Computer Science Faculty, Universidad Complutense de Madrid. His research interests include design automation, computer architecture, and reconfigurable computing.

Antonio Plaza (M'05–SM'07–F'15) was born in Caceres, Spain, in 1975. He is an Associate Professor (with accreditation for Full Professor) with the Department of Technology of Computers and Communications, University of Extremadura, Cáceres, Spain, where he is the Head of the Hyperspectral Computing Laboratory (HyperComp), one of the most productive research groups working on remotely sensed hyperspectral data processing worldwide. He has been the Advisor of 12 PhD dissertations and more than 30 M.Sc. dissertations. He was the Coordinator of the Hyperspectral Imaging Network, a European project with total funding of 2.8 million Euro. He has authored more than 500 publications, including 166 journal papers (115 in IEEE journals), 22 book chapters, and over 240 peer-reviewed conference proceeding papers (94 in IEEE conferences). He has edited a book *High-Performance Computing in Remote Sensing* (CRC Press/Taylor & Francis) and guest edited 9 special issues on hyperspectral remote sensing for different journals. His research interests include hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza served as Associate Editor for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING from 2007 to 2012. He is also an Associate Editor for *IEEE Access*, and was a member of the Editorial Board of the *IEEE Geoscience and Remote Sensing Newsletter* (2011–2012) and the *IEEE Geoscience and Remote Sensing Magazine* (2013). He was also a member of the steering committee of the *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (JSTARS). He served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) from 2011 to 2012, and has been currently serving as President of the Spanish Chapter of IEEE GRSS (since November 2012). He has served as a Proposal Evaluator for the European Commission, the National Science Foundation, the European Space Agency, the Belgium Science Policy, the Israel Science Foundation, and the Spanish Ministry of Science and Innovation. He has reviewed more than 500 manuscripts for over 50 different journals. He is currently serving as the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING journal. He is the recipient of the Recognition of Best Reviewers of the *IEEE Geoscience and Remote Sensing Letters* (in 2009), the Recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (in 2010), the Best Column Award of the IEEE Signal Processing Magazine in 2015, the 2013 Best Paper Award of the JSTARS journal, the Most Highly Cited Paper (2005–2010) in the *Journal of Parallel and Distributed Computing*, the Best Paper Awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology, and the Best Ph.D. Dissertation Award at the University of Extremadura, a recognition also received by six of his PhD students.