Dual-Mode FPGA Implementation of Target and Anomaly Detection Algorithms for Real-Time Hyperspectral Imaging

Bin Yang, Minhua Yang, Antonio Plaza, *Fellow, IEEE*, Lianru Gao, *Member, IEEE*, and Bing Zhang, *Senior Member, IEEE*

***Abstract*—Target and anomaly detection are important techniques for remotely sensed hyperspectral data interpretation. Due to the high dimensionality of hyperspectral data and the large computational complexity associated to processing algorithms, developing fast techniques for target and anomaly detection has received considerable attention in recent years. Although several high-performance architectures have been evaluated for this purpose, field programmable gate arrays (FPGAs) offer the possibility of onboard hyperspectral data processing with low-power consumption, reconfigurability and radiation tolerance, which make FPGAs a relevant platform for hyperspectral processing. In this paper, we develop a novel FPGA-based technique for efficient target detection in hyperspectral images. The proposed method uses a streaming background statistics (SBS) approach for optimizing the constrained energy minimization (CEM) and Reed-Xiaoli (RX) algorithms, which are widely used techniques for target and anomaly detection, respectively. Specifically, these two algorithms are implemented in streaming fashion on FPGAs. Most importantly, we present a dual mode that implements a flexible datapath to decide in real time which one among these two algorithms should be used, thus allowing for the dynamic adaptation of the hardware to either target detection or anomaly detection scenarios. Our experiments, conducted with several well-known hyperspectral scenes, indicate the effectiveness of the proposed implementations.**

***Index Terms*—Field programmable gate arrays (FPGAs), hyperspectral imaging, real-time processing, streaming background statistics (SBS), target and anomaly detection.**

Manuscript received September 03, 2014; revised December 01, 2014; accepted December 24, 2014. Date of publication January 22, 2015; date of current version July 30, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 41325004 and in part by the Key Research Program of the Chinese Academy of Sciences under Grant KZZD-EW-TZ-18. *(Corresponding author: Bing Zhang.)*

B. Yang is with the School of Geosciences and Info-Physics, Central South University, Changsha 410083, China, and also with the Key Laboratory of Digital Earth Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China (e-mail: 14348066@qq.com).

M. H. Yang is with the School of Geosciences and Info-Physics, Central South University, Changsha 410083, China (e-mail: yangmhua@163.com).

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

L. R. Gao and B. Zhang are with the Key Laboratory of Digital Earth Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China (e-mail: gaolr@radi.ac.cn; zb@radi.ac.cn).

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.2015.2388797

I. Introduction

HYPERSPECTRAL imaging provides significant information about the spectral characteristics of materials in the surface of the Earth [1]–[3]. Due to the increased spatial, spectral, and temporal resolution of hyperspectral data sets, the analysis of these data sets now requires improved computing infrastructure in order to cope with the required processing times in many time-sensitive applications [4]. In the last few years, many techniques have been designed for hyperspectral data exploitation in areas such as classification [5] or spectral unmixing [6]. Hyperspectral imagery has also been widely used in reconnaissance and surveillance applications, where targets of interest need to be detected and characterized [7], [8].

The process of detecting and characterizing a target in hyperspectral imagery can be considered as consisting of two main stages:

1) Usually an anomaly detector [9], [10], which identifies spectral vectors that have significant spectral differences from their surrounding background pixels. In [11], a spectral anomaly detection algorithm was developed for detecting targets of unknown spectral distribution against a background with unknown spectral covariance. This algorithm, now commonly referred to as the Reed-Xiaoli (RX) anomaly detector, has been successfully applied to many hyperspectral target detection applications [7], [10], [12], [13] and is considered as the benchmark anomaly detection algorithm for hyperspectral data.
2) To identify whether or not the anomaly is a target or natural clutter (background). This stage can be achieved if the spectral signature of the target is known. Such signature can be obtained from a spectral library [8]. A successful algorithm for this purpose is (among several others [8], [10]) the constrained energy minimization (CEM) [14]. It detects the desired target signal source using a unity constraint while suppressing noise and unknown signal sources by minimizing the average output power. Combining techniques for anomaly and target detection (in joint or separate fashion) offers great advantages in practical applications.

The improved spatial, spectral, and temporal resolutions provided by hyperspectral instruments demand fast computing solutions that can accelerate the efficient exploitation of hyperspectral data sets. Techniques based on hardware accelerators such as clusters [15], [16], distributed platforms [17], and

specialized devices such as commodity graphics processing units (GPUs) [18]–[20] or field programmable gate arrays (FPGAs) [21] have been widely used for accelerating hyperspectral imaging algorithms. Due to their reconfigurability, low-power consumption, the availability of rad-hard developments, and their capacity to be installed onboard airborne and satellite instruments, FPGAs represent a very interesting architecture for this purpose.

An FPGA [22] can be roughly defined as an array of interconnected logic blocks. One of the main advantages of these devices is that both the logic blocks and their interconnections can be (re)configured by their users as many times as needed in order to implement different combinational or sequential logic functions. This characteristic provides FPGAs with the advantages of both software and hardware systems, in the sense that FPGAs now exhibit more flexibility and shorter development times than application specific integrated circuits (ASICs) but, at the same time, are able to provide much more competent levels of performance, closer to those offered by GPUs (but with much lower power consumption). In fact, the power and energy efficiency of FPGAs have significantly improved during the last decade. FPGA vendors have achieved this goal by improving the FPGA architectures, including optimized hardware modules, and taking advantage of the most recent silicon technology. For instance, manufacturing companies such as Xilinx[1] or Altera[2] have reported a 50% reduction in the power consumption when moving from their previous generation of FPGAs [21]. This feature, together with the availability of FPGAs with increased tolerance to ionizing radiation in space, has consolidated FPGAs as the current standard choice for on-board hyperspectral remote sensing [23]–[25].

In the past, there have been several developments toward the implementation of target and anomaly detection algorithms in FPGA architectures [26]. Specifically, Chapter 15 in [26] generally discusses the use of FPGAs in detection applications and provides specific application case studies. Chapter 16 in [26] describes FPGA implementations of techniques for hyperspectral target detection applications. Chapter 17 in [26] 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 [27]. Other implementations based on software optimizations for target and anomaly detection algorithms have also been explored in [28] and [29]. Despite the availability of several FPGA implementations of target and anomaly detection algorithms, few contributions have described the possibility of exploiting different algorithms adaptively using the same FPGA board or the joint use of anomaly and target detection techniques using the same FPGA device.

In this work, we develop a dual-mode parallel FPGA-based hardware platform able to realize real-time hyperspectral target and anomaly detection using the same hardware board. Two highly representative algorithms: RX and CEM (considered as reference techniques for signature-based anomaly and target detection, respectively) are optimized and integrated in a newly designed parallel hardware module. By implementing a flexible data path, these two detection modes can be combined for their exploitation in different detection conditions. Due to the similarities of certain parts of the RX and CEM algorithms, these algorithms can share most of the processing resources in the FPGA, thus allowing us to greatly optimize the design while reducing power consumption. Specifically, our design has been implemented on a Kintex-7 FPGA KC705 Evaluation Kit and tested using two real-hyperspectral data sets, respectively, collected by the HyMap and AVIRIS instruments over Cook City, Montana, and the World Trade Center in New York City. Our experimental results demonstrate that the proposed hardware platform can significantly outperform an equivalent software version, being able to provide target and anomaly detection results in real time, which makes our dual-mode system highly appealing for on-board hyperspectral data exploitation.

This paper is organized as follows. Section II briefly describes the CEM and RX algorithms used for target and anomaly detection purposes, respectively. Section III details the software optimizations carried out for real-time processing using the aforementioned algorithms. Section IV describes the FPGA design and implementation carried out in this work. Section V presents our experimental results. Finally, Section VI concludes the paper with some remarks and hints at plausible future research lines.

II. Related Algorithms

In this section, we provide a short overview of the CEM and RX algorithms, which have been widely used for signature-based target and anomaly detection purposes, respectively.

A. CEM Algorithm

The CEM algorithm [14] is commonly used in situations in which the spectral signature of the desired target is known. Given a finite set of observations $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}$ is a sample pixel vector, let us suppose that the desired signature $\mathbf{d}$ is also known *a priori*. The objective of CEM is to design a linear finite impulse response (FIR) filter $\mathbf{w}$ to minimize the filter output power subject to the constraint $\mathbf{d}^T\mathbf{w} = 1$. The solution was shown in [31] and called CEM detector, with the weight vector given by

$$\mathbf{w}^* = \frac{\mathbf{R}^{-1}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}} \tag{1}$$

where $\mathbf{R} = \frac{1}{N}\left[\sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T\right]$ is the sample autocorrelation matrix of $\mathbf{X}$.

B. RX Algorithm

In certain scenarios, prior knowledge about the target signatures is not available. The RX algorithm [11] was developed to address this particular scenario. Assuming that a single

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

pixel target is the observation test vector, the result of the RX algorithm is given by the filter

$$RX(\mathbf{x}) = (\mathbf{x} - \mathbf{u}_b)^T \sum^{-1} (\mathbf{x} - \mathbf{u}_b) \tag{2}$$

where $\mathbf{u}_b$ is the estimated background clutter sample mean, and $\sum$ is the estimated background clutter covariance. An important variant of the RX algorithm was introduced in [28] by replacing the sample covariance matrix by the sample correlation matrix. As a result, the RX algorithm can also be written as

$$RX(\mathbf{x}) = \mathbf{x}^T \mathbf{R}^{-1} \mathbf{x} \tag{3}$$

which uses $\mathbf{x}$ and $\mathbf{R}^{-1}$ in place of $(\mathbf{x} - \mathbf{u}_b)$ and $\sum^{-1}$, respectively. This version of the RX algorithm has been widely used in real-time scenarios for hyperspectral image processing [13], [29].

C. Local Variants of CEM and RX Algorithms

The classic CEM and RX algorithms can be considered as global target detectors because the correlation matrix is computed using all the pixels of the image, and the background is defined with reference to the full image. As a result, both the CEM and RX calculate the detection results after collecting the whole image data. Several variants of these algorithms have been constructed by applying the same concept to a local region comprising the neighborhood of each image pixel in order to better capture the local statistics. This region is defined by a sliding window centered on the pixel under test [29], or by the image line in which the pixel is located [26]. These methods are considered as the local versions of CEM and RX. However, they are computationally more expensive than the original algorithms because they involve the computation of a correlation (or covariance) matrix and its inverse for every local region, as opposed to the standard versions of the algorithm (which perform the same calculation for the whole image). Although several optimization techniques such as the QR-decomposition [26] can reduce the complexity of the inverse matrix computation, a significant amount of inverse matrix calculations makes it difficult to realize real-time processing in practice. In Section III, we present an optimization of the CEM and RX algorithms (and their local versions) for real-time processing.

III. Optimization of CEM and RX for Real-Time Processing

A. Streaming Background Statistics

In the aforementioned local detection algorithms, for each pixel under test, the algorithm needs to calculate a new local correlation matrix to produce the FIR filter. Even after using the recurrence relation among the correlation matrices associated to neighboring pixels, there are a significant number of pixels to replace. In order to reduce the run-time of every independent correlation matrix computation, we present a streaming background statistics (SBS) method to simplify the calculation of the local background statistics. The method can be simply described as follows. For the nth pixel under test $\mathbf{x}_n$, let the correlation matrix $\mathbf{R}_{\mathrm{n}}$ be computed by a set of previously collected pixels. Then, $\mathbf{R}_{\mathrm{n}}$ can be expressed as

$$\mathbf{R}_{\mathrm{n}} = (1/K) \left[\sum\nolimits_{i=n-K}^{n-1} \mathbf{x}_i \mathbf{x}_i{}^{\mathrm{T}} \right]. \tag{4}$$

Let us now define an SBS matrix $\mathbf{S}_{\mathrm{n}} = K \cdot \mathbf{R}_{\mathrm{n}}$, which can be expressed as $\mathbf{S}_{\mathrm{n}} = \sum_{i=n-K}^{n-1} \mathbf{x}_i \mathbf{x}_i^{\mathrm{T}}$. Thus, for a new pixel under test $\mathbf{x}_{n+1}$, the new relative SBS matrix will be expressed as

$$\mathbf{S}_{n+1} = \mathbf{S}_n + \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} - \mathbf{x}_{n-K} \mathbf{x}_{n-K}^{\mathrm{T}}. \tag{5}$$

From this expression, it is easy to infer that, for the local background, we do not need to calculate the correlation matrix by traversing all the neighboring pixels. Instead, we only need to add a new pixel and remove the oldest one. Most importantly, based on this streaming structure we can easily perform an inverse matrix real-time update by using a Sherman–Morrison formula [31]. Then, this SBS matrix can be used to implement in real time in the CEM and RX algorithms by replacing the correlation matrix by the SBS matrix.

B. Real-Time Processing of Matrix Inversion

To perform a real-time matrix inversion, we first introduce the Sherman–Morrison formula. Let us assume that $\mathbf{A}$ is an invertible square matrix, and $\mathbf{u}$ and $\mathbf{v}$ are two different vectors. Then the Sherman–Morrison formula states that

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^{\mathrm{T}})^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}}{1 + \mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{u}}. \tag{6}$$

Here, $\mathbf{u}\mathbf{v}^{\mathrm{T}}$ is the outer product of two vectors $\mathbf{u}$ and $\mathbf{v}$. The general form shown here is the one presented by Bartlett [32]. If the inverse of $\mathbf{A}$ is already known, the formula provides a computationally affordable way to compute the inverse of $\mathbf{A}$ corrected by the matrix $\mathbf{u}\mathbf{v}^{\mathrm{T}}$. This computation is relatively cheap, because the inverse of $\mathbf{A} + \mathbf{u}\mathbf{v}^{\mathrm{T}}$ does not have to be computed from scratch (which in general is expensive) but can be simply computed by correcting $\mathbf{A}^{-1}$ instead. Obviously, the Sherman–Morrison formula can be performed in (5) by computing the inverse of the SBS matrix. If a previous inverse matrix $\mathbf{S}_n^{-1}$ is known, we can easily derive the inverse of the next SBS matrix $\mathbf{S}_{n+1}^{-1}$ by the following two steps:

Step 1) Calculate the transitional inverse matrix $\mathbf{C}^{-1}$

$$\mathbf{C}^{-1} = (\mathbf{S}_n + \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}})^{-1} = \mathbf{S}_n^{-1} - \frac{\mathbf{S}_n^{-1}\mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \mathbf{S}_n^{-1}}{\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{x}_n + 1}. \tag{7}$$

Step 2) Calculate the inverse of matrix $\mathbf{S}_{n+1}$

$$\begin{aligned} \mathbf{S}_{n+1}^{-1} &= (\mathbf{C} - \mathbf{x}_{n-K}\mathbf{x}_{n-K}^{\mathrm{T}})^{-1} \\ &= \mathbf{C}^{-1} - \frac{\mathbf{C}^{-1}\mathbf{x}_{n-K}\mathbf{x}_{n-K}^{\mathrm{T}}\mathbf{C}^{-1}}{\mathbf{x}_{n-K}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x}_{n-K} - 1}. \end{aligned} \tag{8}$$

At this point, it is important to note that we have not yet obtained the inverse of the first SBS matrix $\mathbf{S}_{K+1}$. This means that we still need to rebuild a computing unit to calculate such

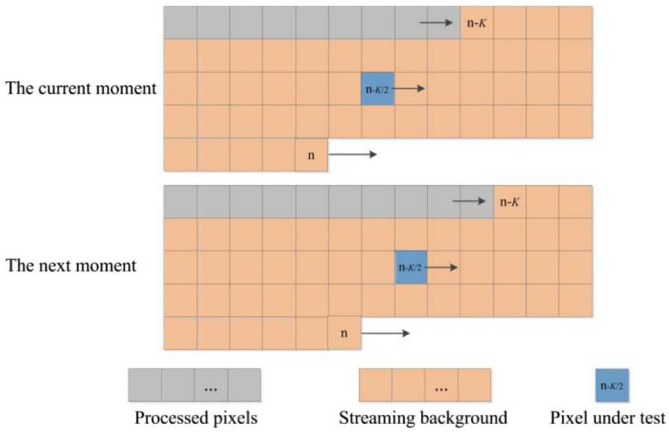

Fig. 1. Structure and processing flow of the SBS method.

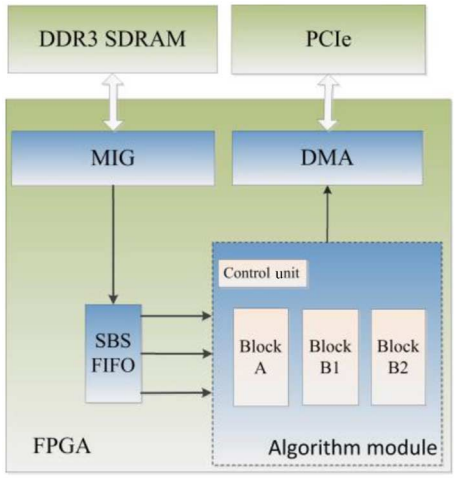

Fig. 2. Hardware architecture to implement the complete system.

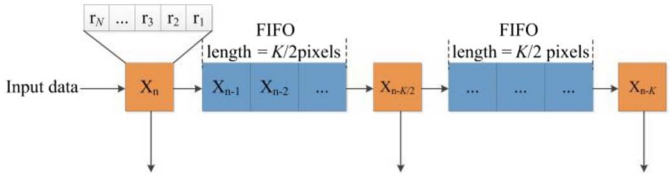

Fig. 3. Block RAM structure of SBS FIFO.

inverse matrix. To avoid this problem, we use β times the identity matrix $\mathbf{I}$ as the initial inverse matrix $\mathbf{S}^{-1}$. As the inverse of $\mathbf{I}$ is itself, the inverse matrix $\mathbf{S}_{K+1}^{-1}$ can be calculated in pixel-by-pixel fashion through the first K pixels, using the same Sherman–Morrison formula. This means that we have added an additional matrix $(1/\beta)\cdot\mathbf{I}$ to the SBS matrix; this procedure can be expressed as

$$\mathbf{S}_{K+1} = (1/\beta)\cdot\mathbf{I} + \mathbf{x}_1\mathbf{x}_1^{\mathrm{T}} + \mathbf{x}_2\mathbf{x}_2^{\mathrm{T}} + \cdots + \mathbf{x}_K\mathbf{x}_K^{\mathrm{T}}. \quad (9)$$

Most importantly, the additional matrix $(1/\beta)\cdot\mathbf{I}$ will not affect the performance of the detectors. Instead, it has the capacity to make the detection results more stable [33].

C. Real-Time CEM and RX Algorithms

This section introduces the real-time versions of CEM and RX algorithms. As the local versions of CEM and RX use the individual correlation matrix to compute the detection results for each pixel, our proposed SBS matrix can be simply applied in the CEM and RX algorithms for real-time purposes. Since the SBS matrix $\mathbf{S}$ is K times correlation matrix $\mathbf{R}$, the inverse of the SBS matrix is $\mathbf{S}^{-1} = (1/K)\mathbf{R}^{-1}$. With these considerations in mind, our SBS-based versions of the CEM and RX algorithm can be simply described as follows:

$$\textit{SBS-CEM}(\mathbf{x}) = \frac{K(\mathbf{x}^T\mathbf{S}^{-1}\mathbf{d})}{K(\mathbf{d}^T\mathbf{S}^{-1}\mathbf{d})} = \frac{\mathbf{x}^T\mathbf{S}^{-1}\mathbf{d}}{\mathbf{d}^T\mathbf{S}^{-1}\mathbf{d}} \quad (10)$$

$$\textit{SBS-RX}(\mathbf{x}) = K(\mathbf{x}^T\mathbf{S}^{-1}\mathbf{x}). \quad (11)$$

Furthermore, in our implementations, the current SBS matrix $\mathbf{S}_n$ is used to calculate the detection result of the middle pixel $\mathbf{x}_{n-K/2}$ instead of the current pixel $\mathbf{x}_n$, as shown in Fig. 1. In this way, each pixel under test is located in the middle region of the background, which can improve the performance of target detection. As a result, our proposed methods can also be expressed as

$$\textit{SBS-CEM}(\mathbf{x}_{n-K/2}) = \frac{\mathbf{x}_{n-K/2}^T\mathbf{S}_n^{-1}\mathbf{d}}{\mathbf{d}^T\mathbf{S}_n^{-1}\mathbf{d}} \quad (12)$$

$$\textit{SBS-RX}(\mathbf{x}_{n-K/2}) = K(\mathbf{x}_{n-K/2}^T\mathbf{S}_n^{-1}\mathbf{x}_{n-K/2}). \quad (13)$$

In Section IV, we describe a dual-mode FPGA design and implementation for the aforementioned CEM and RX algorithms in FPGA architectures. It is important to reiterate that the proposed implementation is based on the previously described software optimizations, which allowed us to simplify the design of CEM and RX algorithms and adapt them to real-time processing scenarios in order to facilitate their onboard exploitation.

IV. Dual-Mode FPGA Design for Target Detection

In this section, we develop an FPGA hardware design for both the CEM and RX algorithms that uses the SBS and real-time matrix inversion updating method described in Section III. The design mainly involves four modules, depicted in Fig. 2. First, a double data rate type three synchronous dynamic random access memory (DDR3 SDRAM) is used to store the whole hyperspectral image. For data input, we use a memory interface generator (MIG) to generate memory controllers and interfaces. With a buffer unit, the MIG has been designed to transfer pixels to the next module only when the previous pixel has been processed. Second, two block random access memories (RAMs) implemented as two first in first out (FIFO) memories are used for registering the received pixels. As shown in Fig. 3, two block RAMs with the same size are connected in serial fashion, with three pixel registers ($\mathbf{x}_n$, $\mathbf{x}_{n-K/2}$, and $\mathbf{x}_{n-K}$)—highlighted in Fig. 3 in orange color—used to extract data for the subsequent calculation. Then, the algorithm module is designed to implement our parallel version of both the CEM and RX algorithm, which is the most complicated part of the design and will be introduced

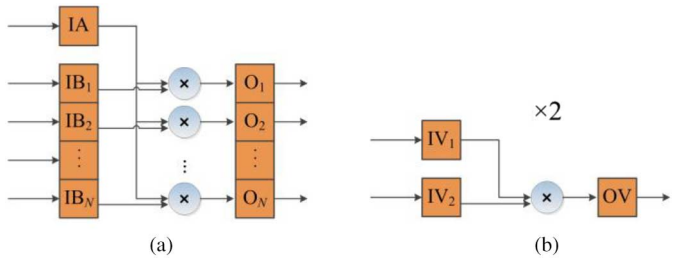

Fig. 4. Hardware structure of (a) Block A in Fig. 2. (b) Block B1 and Block B2 in Fig. 2.

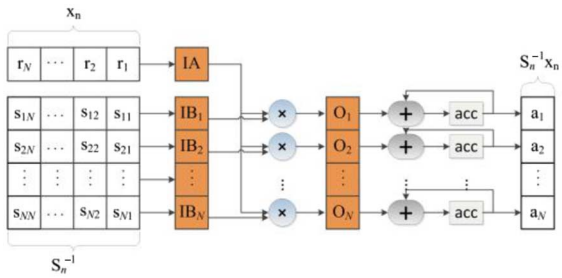

Fig. 5. Parallel structure of Block A in step 1) operation.

hereinafter. Finally, a direct memory access (DMA) module is used to send the detection results via the peripheral component interconnect (PCI) express bus.

A. Inverse Matrix Updating

The main function of the algorithm module is the calculation and updating of the matrix inversion operation, i.e., the computation of (7) and (8). In order to improve the capability of parallel computation with less logic resources, we arrange a block with N (number of spectral bands) hard core multipliers to realize the parallel matrix calculation (Block A in Fig. 2) and two blocks (Block B1 and Block B2 in Fig. 2) with independent multiplier for vector product calculation. A control unit is designed to control the input/output of the blocks, so that we can reuse these calculation blocks to implement the whole computation. The construction of these blocks is graphically illustrated in Fig. 4.

Next, we describe how we have implemented the calculation of (7) by using the aforementioned blocks, which can be summarized in three steps.

Step 1) As shown in Fig. 5, the Block A is first used to calculate the product $\mathbf{S}_n^{-1}\mathbf{x}_n$ using as inputs the matrix $\mathbf{S}_n^{-1}$ and the vector $\mathbf{x}_n$. It is worth noting that the register of matrix $\mathbf{S}_n^{-1}$ should be set to β times the identity matrix when the system is initialized.

Step 2) Then, we calculate the numerator $\mathbf{S}_n^{-1}\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}$ in (7) by reusing the Block A, as shown in Fig. 6(a). It is important to note that the input vector $\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}$ is equal to the transpose of $\mathbf{S}_n^{-1}\mathbf{x}_n$ because of the symmetry of matrix $\mathbf{S}$. At the same time, the Block B1 is used to calculate the product $\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{x}_n$ of two vectors. Further, a divider is connected to the output of Block B1 to calculate the factor $1/(\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{x}_n+1)$, as shown in Fig. 6(b).

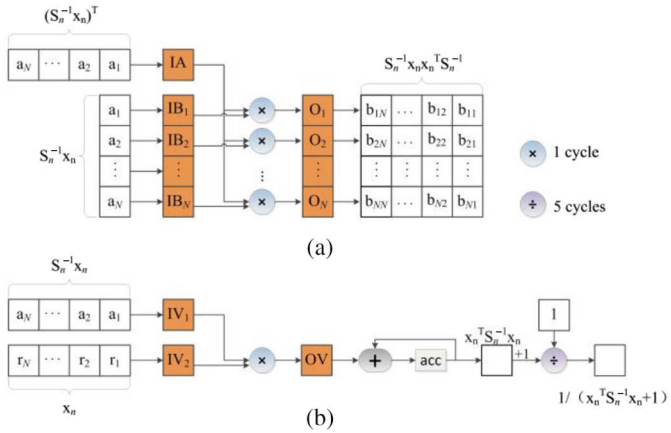

Fig. 6. Data flow of (a) Block A in step 2). (b) Block B1 in step 2).

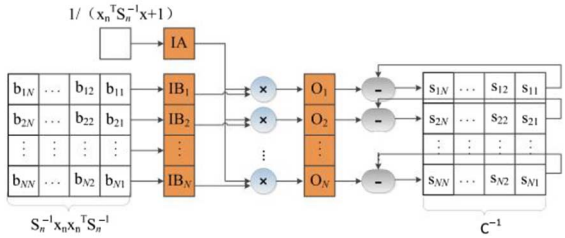

Fig. 7. Reusing the Block A for calculating the matrix $\mathbf{C}^{-1}$ in step 3).

Step 3) Finally, we reuse the Block A to achieve a parallel division calculation without additional dividers (which usually needs more hardware resources than multipliers). As shown in Fig. 7, the Block A calculates the matrix $\mathbf{C}^{-1}$, which can share the same memory space with matrix $\mathbf{S}^{-1}$ in the chip.

As described above, we calculate the inverse matrix $\mathbf{C}^{-1}$ by reusing two calculation blocks. Interestingly, we can implement the calculation of (8) by adopting the same design with only one difference: to replace $\mathbf{x}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{x}+1$ by $\mathbf{x}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{x}-1$ in step 2). Thus, by repeating steps 1) to 3), we can perform the real-time update of the inverse matrix of SBS by using a simple hardware design.

There are two kinds of data that need to be stored in this processing. One is the vector data, e.g., the pixel vector $\mathbf{x}_n$. This is implemented by the registers in FPGA chip. Another is the matrix data. There are two matrices that need to be considered in our design, i.e., the inverse matrix $\mathbf{S}_n^{-1}$ and the temporal matrix $\mathbf{S}_n^{-1}\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}$. For parallel computing purposes, each matrix is implemented by a number of simple dual port RAMs according to the number of spectral bands; thus, we can read and write every line of the matrix at the same time. At last, the control unit generates the signals for the control of the storage units and calculation blocks.

Another important aspect in our hardware implementation is the issue of arithmetical precision. We have paid special attention to this issue in our design. Although FPGA can implement floating-point operations, the fixed-point structure is a much simpler way with less processing time and less logic resources. Fig. 8 shows the structure of our fixed-point data. For the input reflectance image, we use 16-bits binary format in which the

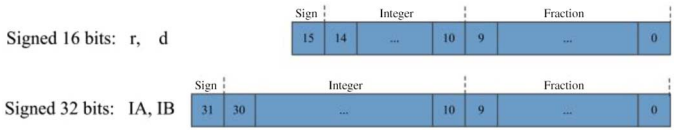

Fig. 8. Data structure in the design.

lower 10 bits express the fraction part, whereas the upper bits express the integer part. The intermediate data and detection results are expressed using 32 bits format, including a lower 10 bits fractional part. Since we reuse the same computing modules (multipliers) to perform the calculation, all the data are designed as signed numbers. It should be noted that, when we calculate the reciprocal $1/(\mathbf{x}_n^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{x}_n + 1)$, we use binary 1×2^{30} as the numerator; as a result, we perform a right shift operation in the subsequent computing.

B. Dual-Mode Implementation

After updating the current inverse matrix, and according to a simple mode command, the control unit is chosen at the next step to calculate the current detection output. When the mode command is set to "1," the system executes the steps corresponding to the target detection mode, according to (12). Again, we reuse the blocks to perform the remainder computations. First, we calculate the product of matrix $\mathbf{S}^{-1}$ and vector $\mathbf{d}$ by using Block A. This process for calculating vector $\mathbf{S}_n^{-1}\mathbf{d}$ is similar to step 1) and designated as step CEM1. Second, the Block B1 calculates the product $\mathbf{x}_{n-K/2}^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{d}$ in a step designated as step CEM2. At the same time, the Block B2 is used to calculate the denominator $\mathbf{d}^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{d}$ in (12). Finally, in step CEM3, the outputs of the Blocks B1 and B2 are used to calculate the target detection result by a single divider. Similarly, when the mode command is set to "0," the system executes the steps corresponding to the anomaly detection mode. We first calculate the product $\mathbf{S}_n^{-1}\mathbf{x}_{n-K/2}$ by using the Block A in a step designated as RX1. Then, we calculate $\mathbf{x}_{n-K/2}^{\mathrm{T}}\mathbf{S}_n^{-1}\mathbf{x}_{n-K/2}$ by using the Block B1 in step RX2. Considering that the RX results can be visualized as a grayscale image in which, the higher the probability of detecting a target, the higher the value of the pixel, we can remove the constant K in (13) without affecting the detection performance.

It is important to note that all the illustrated compute steps reuse the same blocks shown in Fig. 4. For each block operation, the input time of an N-dimensional vector is N times of clock cycle and the delay of the multiplier is 1 clock cycle. Thus, the run-time of these blocks is consistent with $N + 1$ times of the system clock period. In addition, five extra clock cycles are needed for finishing the division stage, so the run-time of step 2) is $N + 6$ clock cycles. For illustrative purposes, Fig. 9 shows a timing diagram of our implementation of CEM and RX algorithms. The run-time of each step is shown in Fig. 9. It should be noted that, when we finish the calculation of step CEM1 or step RX1, the Block A is released and the algorithm module is already for processing the next pixel. Therefore, for each pixel, the run-time of both the target and anomaly detection processes in our system is consistently the same and comprising a total of $8N + 17$ clocks.

V. Experimental Results

The hardware architecture described in Section IV has been implemented on a Kintex-7 FPGA KC705 Evaluation Kit. The KC705 board provides features common to many embedded processing systems, including 1 GB DDR3 SODIMM memory, an eight-lane PCI Express interface. The architecture has been implemented using the Verilog language. To better compare the performance of the proposed algorithms implemented on the FPGA to an optimized software implementation on a computer, the simulations were also conducted using MATLAB and C language in Win7 system, using an Intel Core2 Quad CPU with 2.5 GHz and 4 GB of main memory. The remainder of the section is organized as follows. First, we describe the hyperspectral data sets used in experiments. Then, we provide an assessment of the developed implementations in terms of target and anomaly detection accuracy. Finally, we provide an evaluation of the proposed FPGA implementations in terms of computational performance as compared to an optimized algorithm version and the aforementioned MATLAB/C implementations.

A. Hyperspectral Image Data Set

Two real-hyperspectral data sets collected by different instruments have been used for experimental evaluation of the target and anomaly detection in different real scenarios. The two considered scenes contain ground-truth information, which has been used for the evaluation. In the following, we provide a description of the two considered data sets.

1) HyMap Data Over Cook City, Montana: This hyperspectral data set is provided as part of a publicly available self-test exercise for target detection purposes [34] by the Digital Imaging and Remote Sensing Group, Center for Imaging Science, Rochester Institute of Technology. The data set includes HyMap reflectance images of Cook City in Montana, USA, covering an area of 280×800 pixels with 126 spectral bands between 0.4 and $2.4\ \mu\mathrm{m}$. The ground resolution of the data is approximately 3 m. This data set also comprises the exact locations and SPL files of a set of test targets, which has made it one of the standard test sites for evaluating the accuracy of hyperspectral target and anomaly detection algorithms. There is a grass region located in the data set and four real fabric panels (see Table I) arranged as targets in this region, as shown in Fig. 10(a) in which the area of interest is highlighted by a yellow box. The spectral signature of each target is obtained and preprocessed by the project-equipped SPL files. By rescaling the SPL spectra to the true reflectance data and resampling the SPL spectra according to the data set wavelength, we obtain the prior target spectra, as given in Fig. 10(b).

2) AVIRIS Data Over the World Trade Center in New York City: This data set was collected by the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), operated by NASA's Jet Propulsion Laboratory, over the World Trade Center (WTC) area in New York, on September 16, 2001 (just five days after the terrorist attacks that collapsed the two main towers in the WTC complex) [29]. The full data set selected for experiments consists of 614×512 pixels with 224 spectral bands between

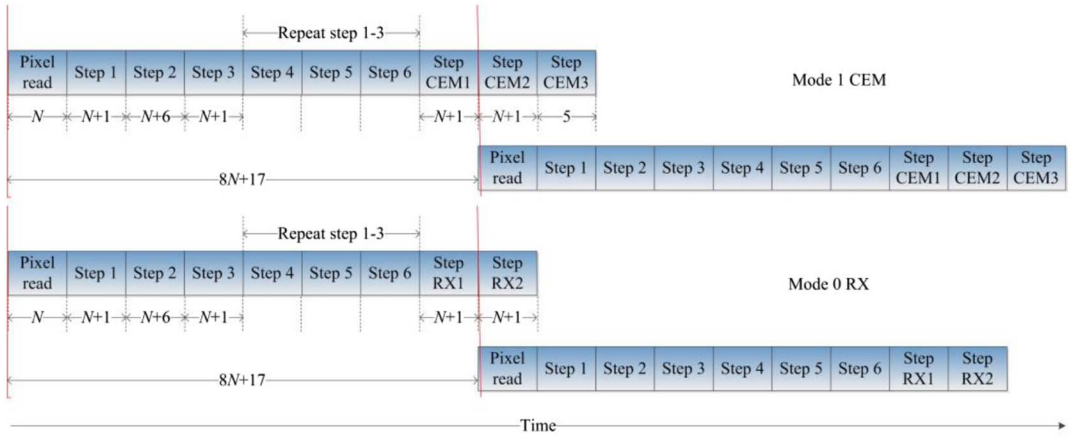

Fig. 9. Timing diagram with algorithm module for the CEM and RX algorithm.

TABLE I
CHARACTERISTICS OF REAL TARGETS

Name	Size (m)	Type
F1	3 × 3	Red cotton
F2	3 × 3	Yellow nylon
F3a	2 × 2	Blue cotton
F3b	1 × 1	Blue cotton
F4a	2 × 2	Red nylon
F4b	1 × 1	Red nylon

0.4 and 2.5 μm. This area covered the hot spots corresponding to latent fires at the WTC, which can be considered as anomalies in this scene. Fig. 11 shows a true color representation of the portion selected for experiments and displays a ground-truth data, highlighted by a yellow box, which shows the spatial location of the hot spots provided by the United States Geological Survey (USGS). Thus, we use this data set to test the performance of our proposed anomaly detection implementation.

B. Analysis of Target and Anomaly Detection Accuracy

In this section, we evaluate the detection accuracy of the proposed real-time implementations of target and anomaly detection algorithms using the real hyperspectral data sets described above. In our hardware design, the coefficient β is set to 10 000 and the length K of the SBS FIFO is set to N^2 (square of the number of spectral bands). The global and local versions of CEM and RX algorithms are evaluated together with our SBS-based approaches. For the global algorithms, the correlation matrix is computed with reference to the full image. For the local versions, the size of local window around each pixel under test is set to 23×23, as set in [29]. A regularization term $\mathbf{0.0001} \cdot \mathbf{I}$ is also added to the local correlation matrix for each pixel that forces the filter coefficients of local algorithms to shrink and get better performance [33].

1) Target Detection Using the HyMap Data: First, we focus on the detection results obtained for the HyMap data set. Since we have the spectra of four fabric panels as a prior knowledge, we adopt the target detection mode in this work. Fig. 12 shows the two-dimensional (2-D) plots of the F1 detection results obtained by the Global-CEM, Local-CEM, and proposed SBS-CEM for the HyMap data set. It should be pointed out that the SBS-CEM is implemented by traversing the image pixels from left to right and from top to bottom. The first K pixels are used to calculate the initial inverse matrix and the detection output is zero. After receiving the $K + 1$th pixel, the system output results one-by-one. In order to show the detection results more clearly, all of the 2-D plots are linearly enhanced from 0 to the max. For illustrative purposes, we amplified the grass region where the targets are located. As shown in Fig. 12, the target F1 (located in the middle of the yellow box area in Fig. 10) can be detected precisely by our proposed method. As we expected, compared with the global and local CEM algorithm, the SBS-CEM method has the same, or even better performance in terms of target detection accuracy. This observation is confirmed by the detection results of F2–F4 shown in Fig. 13. The receiver operating characteristic (ROC) curves corresponding to the detection results are given in Fig. 14. This type of curves establishes a one-to-one correspondence between the true positive rate and the false alarm rate in the detection process. Thus, they provide an essential metric for quantitative evaluation of detection performance. The algorithm with the best performance is indicated by a curve nearest to the upper left, which indicates the highest detection rate under the same false alarm rate. As we can see, in the detection of F1, F2, and F3, the SBS-CEM demonstrates the best performance. Furthermore, the area under curve (AUC) is given in Table II. The higher the AUC, the better the detection results.

2) Anomaly Detection Using the AVIRIS Data: For the AVIRIS World Trade Center data set, we implement our system working in the anomaly detection mode to detect the fire hot spots. Similar to the target detection case, the global and local versions of RX are used to compare with our proposed SBS-RX method. Fig. 15 presents the detection results achieved by the compared algorithms. From this figure, it is remarkable that the proposed SBS-RX can properly detect small anomalies that cannot be identified by the Global-RX. However, our method also brings some false alarms, which in any event are

Fig. 10. (a) HyMap reflectance image of Cook City in Montana, USA, and locations of the real targets. (b) Spectral signatures of four targets.

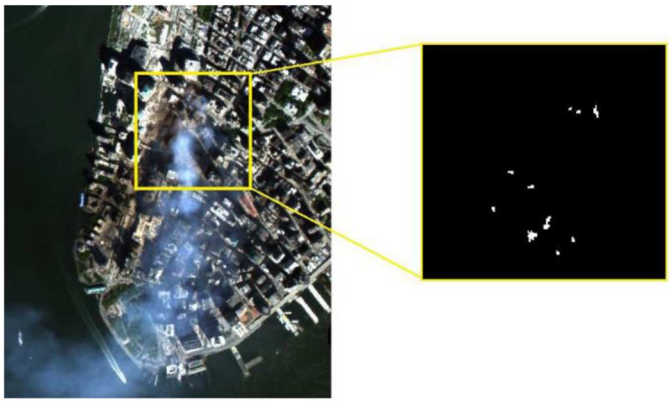

Fig. 11. AVIRIS image over the World Trade Center in New York City and ground-truth map indicating the spatial location of hot spot fires, available from the United States Geological Survey.

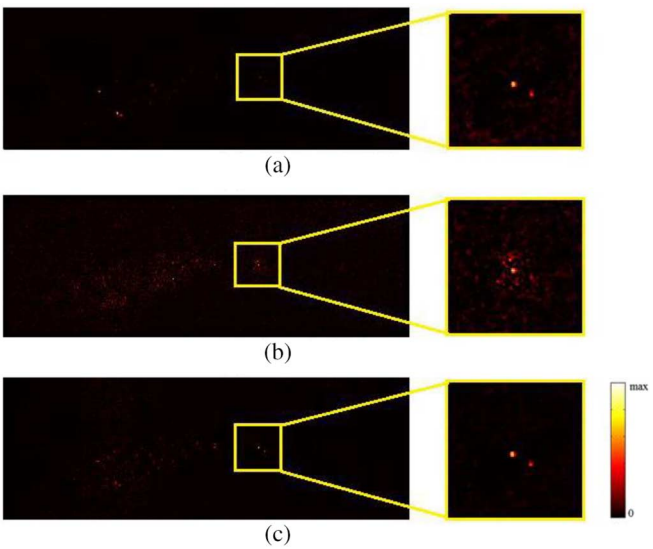

Fig. 12. Detection results for target F1 obtained by different algorithms: (a) Global-CEM; (b) Local-CEM; and (c) SBS-CEM.

much less than those introduced by the Local-RX. Fig. 16 shows the ROC curves corresponding to the results reported in Fig. 15. The AUC values corresponding to these curves are 0.9890 (Global-RX), 0.9208 (Local-RX), and 0.9833 (SBS-RX), respectively. As noted, the proposed method performs better than the Local-RX and similar than the Global-RX in this particular experiment.

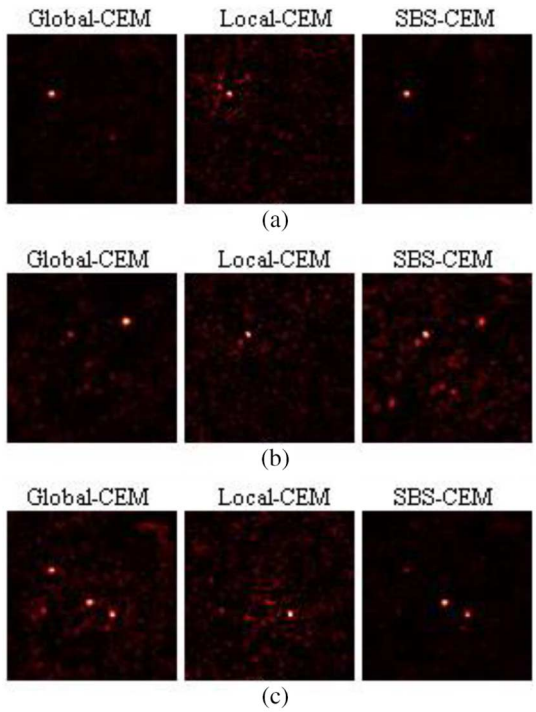

Fig. 13. Detection results obtained by different algorithms for targets: (a) F2; (b) F3; and (c) F4.

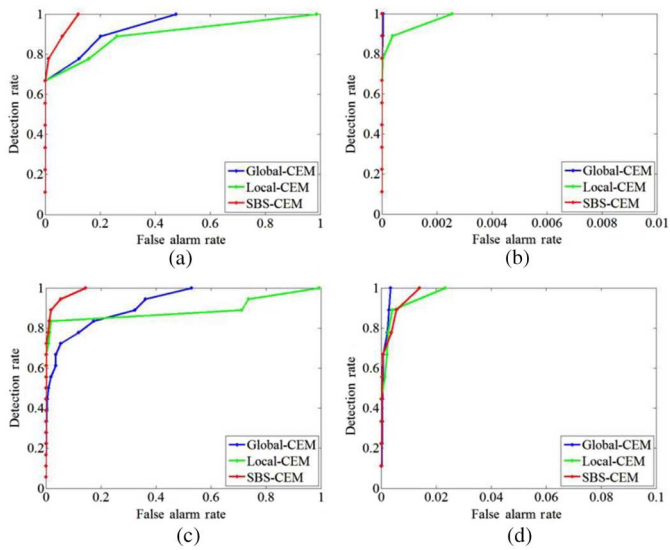

Fig. 14. ROC curve obtained by different algorithms for the detection of targets: (a) F1; (b) F2; (c) F3; and (d) F4.

TABLE II
AUC OBTAINED BY DIFFERENT ALGORITHMS FOR THE CONSIDERED TARGETS

	Global -CEM	Local -CEM	SBS-CEM
F1	0.9107	0.8435	**0.9783**
F2	**1**	0.9997	**1**
F3	0.9067	0.8618	**0.9862**
F4	**0.9987**	0.9962	0.9972

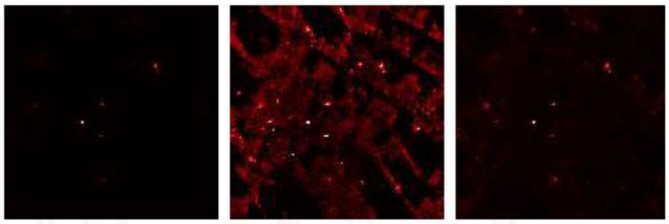

Fig. 15. Detection results obtained by (a) the Global-RX; (b) Local-RX; and (c) SBS-RX in the AVIRIS WTC data set.

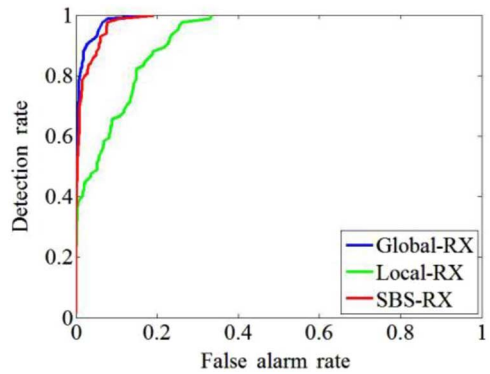

Fig. 16. ROC curve obtained by different anomaly detection algorithms in the AVIRIS WTC data set.

C. Computational Performance Analysis

In this section, we analyze the computational performance of the proposed FPGA implementations. As mentioned before, the FPGA design was implemented on the Kintex-7 FPGA KC705 Evaluation Kit. This FPGA has a total of 50 950 slices, 20 3800 six input look-up tables (LUTs), and 40 7600 slice flip flops available. In addition, the FPGA includes some heterogeneous resources such as 840 DSP48Es and 445 distributed 36 kb block RAMs. In our implementation, we took advantage of these resources to optimize the design. The block RAMs are used to implement the SBS FIFO and store the matrixes, so the vast majority of the slices are used for the implementation of the algorithms together with the DSP48Es multipliers. Table III shows the resources used for our FPGA implementation of the dual-mode detection process. It is important to note that the FPGA implementations need to be adapted to different instruments because of the different number of spectral bands. As shown in Table III, two different implementations (which refer to the considered hyperspectral data sets) are compared in our experiment. For the HyMap implementation, we construct 126 (spectral band number) 32×32 bits multipliers in parallel to form the block A, while for the AVIRIS implementation, the number is 224. Moreover, according to the size of matrix S and the depth of SBS FIFO, different RAM resources are used by the two implementations. As a result, the percentage of hardware utilization increases as the number of spectral band increases. The main bottleneck of our design is the use of a large number of multipliers. In our experiments, the Xilinx Kintex-7 series of FPGA can completely meet the requirements of our implementation. Also, our implementation can be performed in a smaller, even space-grade FPGA such as Virtex-4 XQR4VSX55, which has 512 DSP slices. The maximum operation frequency of each part is provided in the table; all the synthesized implementations can operate with a frequency above 200 MHz.

In the remainder of this section, we first report the computational times obtained by the classic RX and CEM algorithms in MATLAB (see Table IV). Then, we report the processing times measured for our SBS-based implementations in the FPGA as compared to the times obtained for the implementation of the same algorithms using both MATLAB and C (see Table V). As we can observe from Tables IV and V, the C implementations are slightly faster than the MATLAB implementations. This provides a representative comparison between two functionally equivalent hardware and software implementations.

It is important to reiterate that, although the processing time in the software implementations (C and MATLAB) for CEM and RX algorithm is different, the processing time of the two modes in our FPGA implementation is the same, as illustrated by the clock diagram shown in Fig. 9. In the processing of HyMap data set, when the FPGA clock frequency is set to 200 MHz, the processing time measured in the FPGA for both the CEM and RX algorithm is only 1.09 s, as compared to 71.35 for the CEM and 61.86 s for the RX required by the C implementations. Similarly, for the AVIRIS World Trade Center data set, the processing time measured in the FPGA is only 2.71 s, as compared to 152.68 and 132.37 s required for the C implementations. In both cases, we obtain significant advantages in terms of processing time with respect to the software implementations.

Most importantly, since the HyMap scanning rate is 12–16 Hz with 512 pixels per line, this means that in order to gather a full hyperspectral image with the same size of the considered data set, the HyMap sensor requires more than 27 s. Similarly, although the cross-track line scan time in AVIRIS is quite fast (8.3 ms to collect 512 full pixel vectors), this introduces the need to process the WTC scene (with 614×512 pixels, which represents the standard chunk size collected by AVIRIS before writing the data to disk onboard) in approximately 5.09 s to achieve similar time for processing and data acquisition. As a result, the proposed FPGA implementations are strictly in real time for the two considered instruments, as opposed to the software implementations. This represents a significant improvement for adequate exploitation of these algorithms in scenarios with real-time processing constraints.

At this point, we emphasize that other real-time processing techniques for CEM and RX were proposed in the literature [26]–[29]. In [28], the matrix QR-decomposition and its systolic array implementation was developed for matrix inverse calculation, which makes CEM a candidate for line-by-line

TABLE III
SUMMARY OF RESOURCE UTILIZATION FOR THE FPGA-BASED IMPLEMENTATION FOR TWO CONSIDERED HYPERSPECTRAL IMAGES

Component	HyMap implementation		AVIRIS implementation		DDR3 MIG		PCIe DMA	
	Units	Percentage	Units	Percentage	Units	Percentage	Units	Percentage
Number of DSP48Es	265	31	459	55	–	–	–	–
Number of block RAM	120	26	142	32	–	–	8	1
Number of slices	12 088	24	21776	43	4901	10	1374	3
Number of slice flip flops	28245	6	43544	11	7036	1	2961	1
Number of LUTs	21730	10	33997	17	11180	5	1937	1
Maximum frequency (MHz)	226.285		222.243		285.137		299.224	

TABLE IV
PROCESSING TIMES MEASURED FOR THE CEM AND RX ALGORITHMS IMPLEMENTED IN MATLAB

Algorithms	Global (s)		Local (s)		SBS (s)	
	CEM	RX	CEM	RX	CEM	RX
HyMap	22.79	21.61	8541.35	8268.50	75.78	72.58
WTC	47.71	46.42	20177.48	19911.49	161.32	153.47

TABLE V
PROCESSING TIMES MEASURED FOR SBS METHODS IN MATLAB, C, AND FPGA IMPLEMENTATIONS

Implementations	MATLAB (s)		C (s)		FPGA	
					Processing time (s)	Clock period
Algorithms	SBS-CEM	SBS-RX	SBS-CEM	SBS-RX	SBS-CEM/RX	SBS-CEM/RX
HyMap	75.78	72.58	71.35	61.86	1.09	229,607,996
WTC	161.32	153.47	152.68	132.37	2.71	568,699,710

processing in real time. Based on this method, an FPGA implementation of CEM was developed in [26], which uses systolic arrays architecture and the Coordinate Rotation Digital Computer (CORDIC) algorithm for implementing the computation of a matrix inverse. Although the CORDIC circuit and systolic arrays skillfully perform the matrix inverse calculation in real time, the system still needs to build the vector product module (similar to the Block A in our design) and FIR filter (similar to the Block B) to complete the detection. Compared with these studies, we simplify the FPGA implementation of CEM by using a simple matrix inverse updating method in conjunction with the proposed SBS. In this way, we perform a pixel-by-pixel real-time processing design using less hardware resources. Furthermore, our FPGA design takes advantage of the structural similarity between the target and anomaly detection algorithms to reuse the same hardware board so that it can be adapted to different detection circumstances. A GPU-based parallel implementation of RX has also been developed in [29], in which global and local versions of RX algorithm are optimized and performed in multicore platforms. As reported in this literature, for the same AVIRIS data set (shown in Fig. 11), the GPU processing time is 5.28 s for the global RX and 256.3 s for the local RX, while the processing time of our implementation is 2.71 s.

VI. CONCLUSION AND FUTURE LINES

In this paper, we have developed a new dual-mode FPGA implementation of the CEM and RX algorithms, which are standard techniques for hyperspectral target and anomaly detection. Our implementations are not only based on hardware developments, but also on algorithm optimizations. For instance, in order to meet the real-time processing requirements that are present in many application domains, a SBS method is developed to calculate the local background and noise. Further, we use a continuous update method to calculate the inverse of the background matrix. Our experimental results, conducted using a variety of hyperspectral scenes, demonstrate that our

algorithms and hardware implementation can successfully meet strict real-time target and anomaly detection requirements in two different case studies. Most importantly, our implementation illustrates a case in which both target and anomaly detection applications can be addressed using the same hardware board, which implies that they can be used in different detection circumstances. Although our hardware implementation has been carried out in a professional Kintex-7 FPGA, in the future, we will implement our methods in different FPGA architectures (particularly, in radiation-hardened FPGAs) in order to fully calibrate the possibility of porting the hardware modules developed to real earth observation missions, particularly in hardware modules, which can be radiation-hardened. Additional experiments with other data sets will also be conducted in future developments. Also, multi-FPGA platforms provide a good choice to perform future developments of our system. In the future, we are planning to continue this research line using multi-FPGA platforms to implement multitask hyperspectral processing systems including techniques such as hyperspectral image preprocessing, endmember extraction, and hyperspectral unmixing.

ACKNOWLEDGMENT

The authors would like to thank the Digital Imaging and Remote Sensing Group, Center for Imaging Science, Rochester Institute of Technology, for providing the target detection data sets used in our experiments. They also take this opportunity to gratefully acknowledge the editors and the anonymous reviewers for their outstanding comments and suggestions, which greatly helped us to improve the technical quality and presentation of the paper.

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] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. 110–122, 2009.

[3] J. M. Bioucas-Dias *et al.*, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geosci. Remote Sens. Mag.*, vol. 1, no. 2, pp. 6–36, Jul. 2013.

[4] A. Plaza, J. Plaza, A. Paz, and S. Sánchez, "Parallel hyperspectral image and signal processing," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 119–126, Apr. 2011.

[5] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proc. IEEE*, vol. 101, no. 3, pp. 652–675, Feb. 2013.

[6] J. Bioucas-Dias *et al.*, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 354–379, May 2012.

[7] M. T. Eismann, A. D. Stocker, and N. M. Nasrabadi, "Automated hyperspectral cueing for civilian search and rescue," *Proc. IEEE*, vol. 97, no. 6, pp. 1031–1055, Jun. 2009.

[8] D. Manolakis and G. Shaw, "Detection algorithms for hyperspectral imaging applications," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 29–43, Aug. 2002.

[9] S. Matteoli, M. Diani, and G. Corsini, "A tutorial overview of anomaly detection in hyperspectral images," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 25, no. 7, pp. 5–27, Aug. 2010.

[10] D. W. J. Stein *et al.*, "Anomaly detection from hyperspectral imagery," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 58–69, Aug. 2002.

[11] S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 38, no. 10, pp. 1760–1770, Oct. 1990.

[12] C. Willis, "Comparison of anomaly detection methods for hyperspectral imagery," in *Proc. SPIE*, Orlando, FL, USA, 2005, vol. 5988, pp. B1–B12.

[13] C.-I. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2007.

[14] Q. Du, H. Ren, and C.-I. Chang, "A comparative study for orthogonal subspace projection and constrained energy minimization," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 6, pp. 1525–1529, Aug. 2003.

[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, Aug. 2011.

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

[17] A. Plaza, D. Valencia, and J. Plaza, "An experimental comparison of parallel algorithms for hyperspectral analysis using homogeneous and heterogeneous networks of workstations," *Parallel Comput.*, vol. 34, no. 2, pp. 92–114, 2008.

[18] S. Sánchez, A. Paz, G. Martin, and A. Plaza, "Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units," *Concurrency Comput. Pract. Exper.*, vol. 23, no. 13, pp. 1538–1557, 2011.

[19] S. Bernabe, S. Lopez, A. Plaza, and R. Sarmiento, "GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 221–225, Mar. 2013.

[20] A. Paz and A. Plaza, "Clusters versus GPUs for parallel automatic target detection in remotely sensed hyperspectral images," *EURASIP J. Adv. Signal Process.*, vol. 2010, pp. 1–18, 2010, Article ID 915639.

[21] S. Lopez *et al.*, "The promise of reconfigurable computing for hyperspectral imaging on-board systems: Review and trends," *Proc. IEEE*, vol. 101, no. 3, pp. 698–722, Feb. 2013.

[22] S. Hauck, "The roles of FPGAs in reprogrammable systems," *Proc. IEEE*, vol. 86, no. 4, pp. 615–639, Apr. 1998.

[23] 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.

[24] 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.

[25] T. Cervero *et al.*, "A scalable and dynamically reconfigurable FPGA-based embedded system for real-time hyperspectral unmixing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, 2014, doi: 10.1109/JSTARS.2014.2347075, to be published.

[26] A. Plaza and C.-I. Chang, *High Performance Computing in Remote Sensing*. New York, NY, USA: Taylor & Francis, 2007.

[27] 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.

[28] 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.

[29] 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, May 2013.

[30] T. Cocks, R. Jenssen, A. Stewart, I. Wilson, and T. Shields, "The HyMap-TM airborne hyperspectral sensor: The system, calibration and performance," in *Proc. EARSEL Workshop Imag. Spectrosc.*, 1998, pp. 37–42.

[31] W. Hager, "Updating the inverse of a matrix," *SIAM Rev.*, vol. 31, no. 2, pp. 221–239, 1989.

[32] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 107–111, 1951.

[33] N. Nasrabadi, "Regularized spectral matched filter for target recognition in hyperspectral imagery," *IEEE Signal Process. Lett.*, vol. 15, pp. 317–320, Mar. 2008.

[34] D. Snyder *et al.*, "Development of a web-based application to evaluate target finding algorithms," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, 2008, vol. 2, pp. 915–918.

[35] M. S. Stefanou and J. P. Kerekes, "A method for assessing spectral image utility," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 6, pp. 1698–1706, May 2009.

Bin Yang received the M.S. degree in optical engineering from HuaZhong University of Science and Technology, Wuhan, China, in 2005. Currently, he is pursuing the Ph.D. degree in school of geosciences and info-physics at the Central South University, Changsha, China.

He is currently working with the Institute of Remote Sensing and Digital Earth, Chinese Academy of Science, Beijing, China. His research interests include hyperspectral remote sensing, hyperspectral image processing, and high-performance computing.

Minhua Yang received the M.S. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, in 1998, and the Ph.D. degree in resources and environment remote sensing from China Agricultural University, Beijing, China, in 2002.

He is currently a Professor with the Central South University, Changsha, China. He specializes in hyperspectral remote sensing and has more than 15 years of experience in studying and graduate education in this field. So far, he has been training 10 doctoral candidates, and more than 40 of postgraduates. His research interests include hyperspectral image processing, target detection, and vegetation information inversion.

Antonio Plaza (M'05–SM'07–F'14) 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, MD, USA; the Applied Information Sciences Branch, Goddard Space Flight Center, Greenbelt, MD, USA; and the AVIRIS Data Facility, Jet Propulsion Laboratory, Pasadena, CA, USA. 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. He was the Coordinator of the Hyperspectral Imaging Network, which is 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 has coedited a book on high-performance computing in remote sensing. He is the author or coauthor of around 300 publications on remotely sensed hyperspectral imaging, including more than 50 Journal Citation Report papers, 20 book chapters, and over 200 conference proceeding papers. His research interests include remotely sensed hyperspectral imaging, pattern recognition, signal and image processing, and efficient implementation of large-scale scientific problems on parallel and distributed computer architectures.

Dr. Plaza has 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 an Associate Editor of 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 papers submitted to more than 50 different journals, including more than 140 papers reviewed for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. He has served as a Chair for the IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing in 2011. He has also been serving as a Chair for the Society of Photo-Optical Instrumentation Engineers (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 was a recipient of the recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2009 and the recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING in 2010. He is currently serving as the Director of Education activities for the IEEE Geoscience and Remote Sensing Society.

Lianru Gao (M'12) received the B.S. degree in civil engineering from Tsinghua University, Beijing, China, in 2002, and the Ph.D. degree in cartography and geographic information system from the Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing, China, in 2007.

He is currently an Associate Professor with the Key Laboratory of Digital Earth Science, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences. He has authored over 60 papers in China and abroad. His research interests include spectral feature analysis, hyperspectral image processing, target detection, and image simulation.

Bing Zhang (M'11–SM'12) received the B.S. degree in geography from Peking University, Beijing, China, the M.S. and Ph.D. degrees in remote sensing from the Institute of Remote Sensing Applications, Chinese Academy of Sciences (CAS), Beijing, China.

Currently, he is a Professor and the Deputy Director of the Institute of Remote Sensing and Digital Earth, CAS. He specializes in hyperspectral remote sensing and has more than 17 years of experience in studying and graduate education in this field. His research interests include development of physics-based models and image processing software for the use of hyperspectral remote sensing data in solving problems in geology, hydrology, ecology, and botany.

Dr. Zhang received several Chinese National, Ministerial, and Provincial S&T Progress Awards for undertaking some innovative research and research-based development projects.