Low-Power Hyperspectral Anomaly Detector Implementation in Cost-Optimized FPGA Devices

Julián Caba, María Díaz, Jesús Barba, Raúl Guerra, Soledad Escolar, and Sebastián López, *Senior Member, IEEE*

***Abstract*—Onboard data processing for on-the-fly decision-making applications has recently gained momentum in the field of remote sensing. In this context, hyperspectral anomaly detection has received special attention since its main purpose lies in the identification of abnormal events in an unsupervised manner. Nevertheless, onboard real-time hyperspectral image processing still poses several challenges before becoming a reality. This is why there is an emerging trend toward the development of hardware-friendly algorithmic solutions embedded in reconfigurable devices. In this context, this work contributes to a hardware architecture that ensures a progressive line processing in time-sensitive applications limited by the scarcity of hardware resources. In this sense, we have implemented the state-of-the-art hardware-friendly line-by-line fast anomaly detector for hyperspectral imagery (HW-LbL-FAD) detector on a reconfigurable hardware for a real-time performance. Specifically, we have selected a cost-optimized field-programmable gate array (ZC7Z020-CLG484) to implement our solution whose results draw up a good tradeoff between the following three features: time performance, energy consumption, and cost. The experimental results indicate that our hardware component is able to process hyperspectral images of 825x1024 pixels and 160 bands in 0.51 s with a power budget of 1.3 W and costs around 150€. Regarding detection performance, the HW-LbL-FAD algorithm outperforms other state-of-the-art algorithms.**

***Index Terms*—Anomaly detection, field-programmable gate arrays (FPGAs), high-level synthesis (HLS), hyperspectral imaging, line-by-line performance, low power, real time.**

I. Introduction

In THE recent years, anomaly detection has been extensively studied in the field of hyperspectral data analysis [1]. Its popularity is further enhanced by the ability to spot abnormal events or man-made targets in an unsupervised manner. Generally speaking, anomalies are seen as not abundant pixels in a scene whose spectrum is significantly dissimilar to their surroundings or to the predominant background pattern. Hence, one basis behind the anomaly detection issue is the identification of desired targets that are unknown in advance and whose existence could be indicative of a suspicious behavior. This lack of previous knowledge turns the detection of anomalous spectra into an essential matter in military and civilian applications, such as defense and surveillance, environmental monitoring, rare natural disaster detection, agriculture studies, among others [2], [3].

From the foregoing, it is easy to conclude that anomaly detection is a very challenging task whose success relies on accurate modeling of unknown and heterogeneous background. To do so, several algorithmic solutions have been proposed in the literature. The most widely studied are based on statistical approaches under the assumption of Gaussian multivariate distribution. In this sense, the Reed-Xiaoli (RX) [4] method is regarded as the benchmark in the anomaly detection field and several upgraded variants can be found in the literature [5]–[9]. Nonetheless, the emergence of sparsity [10]–[13], compressed sensing, and collaborative-representation-based [14] models has made possible to successfully develop other non-RX-based methods that do not assume a normal probability function in very heterogeneous background pattern [15]. Other proposals are based on dimension reduction and feature extraction in order to remove interband correlations among spectral bands [16]. Additionally, recent advances in the deep learning and tensor theory have also drawn increasing attention [17], [18].

Most of the aforementioned algorithmic solutions reach excellent levels of accuracy in the detection results, but at an expense of raising the computation complexity of the involved operations. It is usually reflected in intensive memory requirements, high implementation costs, and nonscalability [19]–[23]. This fact has prevented their full incorporation in earth observation systems for onboard image processing, such as unmanned aerial vehicles (UAVs). These aerial platforms are actually still limited in terms of computational capacities, power budget, and data storage. Therefore, on-Earth processing has been the mainstream solution for the hyperspectral data handling. Nevertheless, the ever-growing acquisition data rates of the newest generation of hyperspectral cameras, and the bottleneck around the

Manuscript received January 16, 2022; revised February 17, 2022; accepted February 25, 2022. Date of publication March 11, 2022; date of current version March 25, 2022. This work was supported in part by the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI) of the Conserjería de Economía, Industria, Comercio y Conocimiento of the Gobierno de Canarias, in part by the European Social Fund (FSE) [POC2014-2020, Eje 3 Tema Prioritario 74 (85%)], and in part by Grant TALENT (PID2020-116417RB-C4, subprojects 2 and 4) funded by MCIN/AEI/10.13039/501100011033 and the EU's Horizon 2020 programme under project SHAPES (GA No. 857159). *(Julián Caba and María Díaz contributed equally to this work.) (Corresponding authors: María Díaz; Julián Caba.)*

Julián Caba, Jesús Barba, and Soledad Escolar are with the School of Computer Science, University of Castilla-La Mancha, 13071 Ciudad Real, Spain (e-mail: julian.caba@uclm.es; jesus.barba@uclm.es; soledad.escolar@uclm.es).

María Díaz, Raúl Guerra, and Sebastián López are with the Institute of Applied Microelectronics (IUMA), 35017 Las Palmas de Gran Canaria, Spain (e-mail: mdmartin@iuma.ulpgc.es; rguerra@iuma.ulpgc.es; seblopez@iuma.ulpgc.es).

Digital Object Identifier 10.1109/JSTARS.2022.3157740

transmission of huge volumes of data make it increasingly unfeasible the realization of low-latency decision-making applications [24], [25]. Accordingly, the onboard processing has become a potential solution for these scenarios [26]–[28].

Additionally, the causality inherent to push-broom/whisk-broom-based frameworks must be also met for paving the way to a real-time performance. Nonetheless, there is a lack in the literature of nonglobal algorithm definitions that are able to independently process blocks of image pixels. In this context, the hardware-friendly line-by-line fast anomaly detector for hyperspectral imagery (HW-LbL-FAD) [29], [30] emerged to resolve the aforementioned requirements in the field of anomaly detection.

The HW-LbL-FAD algorithm is a subspace-based anomaly detector that follows an orthogonal projection strategy for estimating the orthogonal subspace spanned by the background distribution in which anomalous spectra are easily identified. This algorithm was particularly designed to meet the constraints imposed by nowadays remote sensing applications based on pushbroom/whiskbroom scanners. In this sense, this proposal processes blocks of image pixels independently, ruling out any spatial alignment restriction. Therefore, the methodology followed by the HW-LbL-FAD algorithm is well aligned with the needs imposed by the aforementioned applications since the detection of anomalous entities is conducted in a line-by-line fashion. Apart from this, the progressive line processing also offers other benefits derived from the low volume of data to be stored and processed at once, such as a reduction in the required hardware resources and the speed-up of the data handling processes.

In addition, the HW-LbL-FAD follows a mathematical strategy oriented to the reuse of the operations involved in its different computing stages for both the modeling of the background pattern and the detection of the anomalous spectra. This feature makes it possible the fully implementation of the HW-LbL-FAD on low-cost and low-power embedded systems ruling out any process of reconfiguration due to the lack of hardware resources. This approach also derives in a saving in the time-to-market and the invested human endeavours since allows to focus on the enhancement of the performance and the design of a datapath and control logic that enabled the time sharing of the hardware modules and the optimal use of the resources.

Against this backdrop, the main motivation of this work is to demonstrate the aforementioned assertions about the HW-LbL-FAD but under operating conditions. Therefore, the goal is to contribute to the scientific community with a verified hardware structure that ensures a progressive line processing in time-sensitive applications limited by the scarcity of hardware resources. In particular, we present a low-power, energy efficient and cost-optimized solution for a complex, multistaged anomaly detection algorithm for hyperspectral imaging targeting a field-programmable gate array (FPGA).

From the engineering point of view, this work makes it the most of the use of high-level synthesis (HLS) technologies and tools, by applying best practices and efficient design flows to the development of a custom HW-LbL-FAD hardware accelerator. Working with models at a higher abstraction level is key to boost engineers productivity within an HLS-based workflow since it enables the easier reuse of previous work. In this sense, the proposed architecture extends and improves a set of operators developed for a totally different application: hyperspectral imaging compression [31]. As it is the case of the software world, the adaptation of legacy code and its fine tuning to match the new functional and not functional requirements shorten the development time and effort and allows engineers to focus more on optimization and integration.

All of the aforementioned, together with the fact that FPGA-based implementations can provide much more competent levels of performance while sustaining lower power consumption compared with graphical processing units (GPUs) [32] contributes to endorse the use of the reconfigurable technology in the field of high-performance remote sensing applications.

The rest of this article is organized as follows. Section II includes a brief outline about the mathematical method behind the computing stages addressed by the HW-LbL-FAD. Section III includes a comprehensive explanation about the implementation model proposed for the execution of the HW-LbL-FAD in the targeted FPGA system-on-chip (SoC). Section IV evaluates the proposed anomaly detector by analyzing the hardware architecture implemented on a ZC7Z020 FPGA device. Section V collects a discussion about the performance and hardware resource utilization of the proposed hardware architecture compared with other state-of-the-art anomaly detectors also implemented on reconfigurable hardware. Finally, Section VI concludes this article.

II. Methods I: Algorithm Description

In this section, it is described the general outline of the selected anomaly detection method that will help us to face the challenges introduced previously in Section I. In this regard, the HW-LbL-FAD algorithm shows some unique characteristics that make it a good candidate for the aforementioned purposes.

1) *Low computational complexity and high degree of parallelism of the involved operations:* The different computing stages defined in the HW-LbL-FAD algorithm uses a set of core operations (see Section II-B) that does not perform computationally complex matrix operations, such as inverse matrix calculation or the extraction of eigenvalues and eigenvectors. This facilitates the subsequent hardware implementation, in terms of engineering effort and time to market, and also reduces the amount of required hardware resources.
2) *Reduction in the computing hardware resources:* The HW-LbL-FAD algorithm is a subspace-based anomaly detector whose different computing stages can be grouped into the following two differentiated processes:
   a) the modeling of the background distribution;
   b) the detection of potential anomalous spectra.

   Both processes are based on the same mathematical method, and they actually share the aforementioned set of core operations. Consequently, the resources devoted to the implementation of the hardware operators can be time shared by these two processes. To this

end, a good design of the interconnection datapath and a precise synchronization of the involved operations require an extra effort.

3) *Line-by-line performance:* The HW-LbL-FAD algorithm is able to process blocks of image pixels with not taking into consideration any spatial alignment requirement. Also, the algorithm analyzes the blocks as if they were independent of each other. Hence, the method followed by the HW-LbL-FAD algorithm is well aligned with the needs imposed by nowadays remote sensing applications based on pushbroom/whiskbroom scanners since the detection of anomalous pixels could be conducted in a line-by-line fashion. In addition, it also leads to a reduction in the amount of data to be stored and managed at one time, thereby minimizing the required memory resources and also speeding up the process of data analysis.
4) *Fixed-point notation:* The set of core operations can be seamlessly implemented using both floating-point and fixed-point notation, without incurring on a significant loss of accuracy in the detection results, as it was analyzed in [31] and [33]. Therefore, the HW-LbL-FAD can be properly implemented using the most used computing platforms, such as FPGAs and GPUs, and still make it the most of the underlying architecture-specific characteristics, maintaining excellent performance levels.

The mathematical background behind the HW-LbL-FAD algorithm is described in the following sections. In particular, the introduction of the aforementioned set of core operations is done in Section II-B, while the algorithm description is addressed in Section II-C.

A. General Notation

Before starting with the HW-LbL-FAD algorithm description, it is needed to define some variables and terminology employed thorough the remainder of this manuscript. A hyperspectral image, $\mathbf{HI} = \{\mathbf{F}_i, i = 1, \ldots, nr\}$, is a sequence of nr hyperspectral frames or lines of pixels, $\mathbf{F}_i$, comprised by nc pixels with nb spectral bands. Pixels within $\mathbf{HI}$ are grouped in blocks of BS pixels, $\mathbf{M}_k = \{\mathbf{r}_j, j = 1, \ldots, BS\}$, being BS usually equal to nc, or multiple of it, and k spans from 1 to $\frac{nr \cdot nc}{BS}$ hyperspectral frames. $\hat{\boldsymbol{\mu}}$ is the average pixel or *centroid* of each $\mathbf{M}_k$ block. $\mathbf{C} = \{\mathbf{c}_j, j = 1, \ldots, BS\}$ represents the centralized version of the input image block, $\mathbf{M}_k$. $\mathbf{E}_k = \{\mathbf{e}_n, n = 1, \ldots, p\}$ saves the p most different hyperspectral pixels extracted from each $\mathbf{M}_k$ block, while $\mathbf{B}* = \{\mathbf{E}_k, k = 1, \ldots, n_f\}$ retains the subset of selected pixels $\mathbf{E}_k = \{\mathbf{e}_n, n = 1, \ldots, p\}$ within the first captured n_f $\mathbf{M}_k$. $\mathbf{V}_k = \{\mathbf{v}_n, n = 1, \ldots, p\}$ comprises p vectors of BS elements where each $\mathbf{v}_n$ vector corresponds to the projection of the BS pixels within $\mathbf{M}_k$ onto the corresponding n extracted pixel, $\mathbf{e}_n$. $\mathbf{Q} = \{\mathbf{q}_n, n{=}1, \ldots, p\}$ and $\mathbf{U} = \{\mathbf{u}_n, n{=}1, \ldots, p\}$ save p pixels of nb bands that are orthogonal among them.

B. Set of Core Operations

The HW-LbL-FAD algorithm is based on the concept of orthogonal subspace projections for the selection of the most different pixels in a subset of image spectra, as well as for the estimation of the amount of spectral information that can be represented by them. For doing this, the HW-LbL-FAD algorithm orchestrates a set of core operations that actually carries out the orthogonalization process defined by the well-known state-of-the-art Gram–Schmidt method [34].

These operations are shown in the pseudocode displayed in Algorithm 1 and can be applied to the entire image or independently to a block of BS hyperspectral pixels, $\mathbf{M}_k$.

1) *Average pixel calculation:* The first characteristic pixel, $\mathbf{e}_1$, to be extracted is that with the highest deviation from the average pixel, $\hat{\boldsymbol{\mu}}$. Hence, the first step is to compute $\hat{\boldsymbol{\mu}}$ for the input image block, $\mathbf{M}_k$, (see line 1).
2) *Centralization:* The second step is to centralize the input image block, $\mathbf{M}_k$, subtracting $\hat{\boldsymbol{\mu}}$ from each image pixel and obtaining the centralized version of the input image, $\mathbf{C}$, (see line 2).
   Afterwards, the p most representative pixels within $\mathbf{M}_k$ are sequentially extracted by lines 3–18 of Algorithm 1. To do so, the orthogonal projection of each image pixel with respect to the selected pixels, $\mathbf{e}_n$, is addressed implementing the aforementioned Gram–Schmidt method. Once the first representative pixel has been selected, the following operations are in charge of sequentially extracting new characteristic pixels by selecting the pixels with the largest orthogonal projections to the pixels already extracted. At this point, image pixels just retain the information that is not contained by previously selected pixels, and thus, that is orthogonal to them. With it, it is achieved to select the most different pixels in each iteration, understanding it, those pixels that cannot be well represented by previously selected pixels. For doing so, the operations listed in the following are repeated until a certain stop condition is met (see line 8).
3) *Brightness calculation:* The selected pixels are those with the highest dot product with itself in each iteration (see lines 4–6). In the remainder of this document, it is referred to as brightness of a pixel. The selected pixels, $\mathbf{e}_n$, are those from $\mathbf{M}_k$ corresponding to the highest brightness in $\mathbf{C}$ (see line 11). Then, their orthogonal projection counterparts in $\mathbf{C}$, $\mathbf{q}_n$, and their normalized version, $\mathbf{u}_n$, are accordingly obtained as shown in lines 12 and 13, respectively.
4) *Projection:* After that, all vectors within $\mathbf{C}$ are projected onto $\mathbf{u}_n$, obtaining the projection vector, $\mathbf{v}_n$, (see line 14).
5) *Subtraction:* After the information that can be spanned by the defined $\mathbf{q}_n$ and $\mathbf{u}_n$ vectors is stored in the projection image vector, $\mathbf{v}_n$, it is subtracted from $\mathbf{C}$ (see line 15). Hence, $\mathbf{C}$ now only retains the spectral information that is orthogonal to the already selected spectra. For this reason, pixels within $\mathbf{Q}$ and $\mathbf{U}$ are orthogonal among each other.

Although the aforementioned set of core operations are used for the background modeling and the detection of anomalous spectra in the HW-LbL-FAD, Díaz [30] also probed that other hyperspectral analysis techniques can be addressed using them, such as lossy compression, target detection, unmixing, etc. This feature enables the coexistence of multiple applications at the same time with the advantage of sharing the most

Algorithm 1: Set of Core Operations.

Inputs: $\mathbf{M}_k = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{BS}], \alpha$
Outputs: $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p]$ {Characteristic pixels }; $\hat{\boldsymbol{\mu}}$ { Average Pixel }; $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p]$ { Orthogonal vectors }; $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p]$ { Orthonormal vectors }; τ { Threshold}
Algorithm:

Average pixel: $\hat{\boldsymbol{\mu}}$;
Centralization: $\mathbf{C} = \mathbf{M}_k - \hat{\boldsymbol{\mu}}$;
while exit $= 0$ **do**
  for $j = 1$ **to** BS **do**
    Brightness: $\mathbf{b}_j = \mathbf{c}'_j \cdot \mathbf{c}_j$;
  end for
  Maximum Brightness: $j_{\max} = \text{argmax}(\mathbf{b}_j)$;
  if $\frac{b_{j\max}}{(\mathbf{r}_{j\max} - \hat{\boldsymbol{\mu}})' \cdot (\mathbf{r}_{j\max} - \hat{\boldsymbol{\mu}})} \cdot 100 < \alpha$ **then**
    Stop Condition: exit $= 1$
  else
    Extracted pixels: $\mathbf{e}_n = \mathbf{r}_{j_{\max}}$;
    $\mathbf{q}_n = \mathbf{c}_{j_{\max}}$;
    $\mathbf{u}_n = \mathbf{q}_n / b_{j_{\max}}$;
    Projection: $\mathbf{v}_n = \mathbf{u}'_n \cdot \mathbf{C}$;
    Subtraction: $\mathbf{C} = \mathbf{C} - \mathbf{q}_n \cdot \mathbf{v}_n$;
    $\tau = b_{j_{\max}}$
  end if
end while

computationally costly operations. From a productivity point of view, it also leads to a reduction in the time to market since just a single mathematical approach has to be analyzed and implemented. To give an example, the HyperLCA algorithm [33] is a transform-based lossy compressor whose spectral transform uses the aforementioned set of core operations for the decorrelation of the data. In [31] was evaluated its suitability for line-by-line scenarios through the design of a hardware architecture implemented on a cost-optimized FPGA. The advantage of sharing the same mathematical model is that the HLS operators that instantiate the functionality of the core operators for the data correlation are for the most part reused in this article for anomaly detection.

C. HW-LbL-FAD Algorithm

The HW-LbL-FAD algorithm is a subspace-based anomaly detector that is based on the concept of the linear mixing model (LMM) to deal with the anomaly detection issue. Built on the premise that the anomalous targets and the background signals lie into a different lower dimensional subspace, the LMM can be rewritten as

$$\mathbf{r}_\mathrm{j} = \sum_{n=1}^{p} \mathbf{b}_\mathrm{n} \cdot a_{\mathrm{j,n}} + \mathbf{s} \cdot a_{\mathbf{s}\mathrm{j}} + \mathbf{n}_\mathrm{j} \tag{1}$$

where $\mathbf{b}_\mathrm{n}$ represents the n background signal, $\mathbf{s}$ is the desired target signature, $a_\mathbf{s}$ is the abundance factor of $\mathbf{s}$ in the pixel $\mathbf{r}_\mathrm{j}$, and $\mathbf{n}_\mathrm{j}$ represents the noise contained in the image pixel $\mathbf{r}_j$.

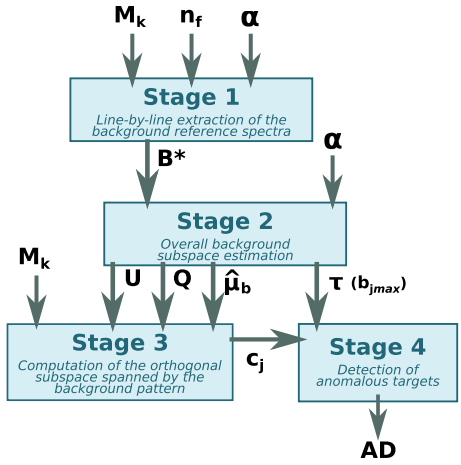

Fig. 1. Dataflow of the HW-LbL-FAD computing stages.

Nonetheless, anomalous spectra are not known beforehand, but a distinguishing feature is that they do no match the background pattern, and thus, they cannot be accurately represented by the background samples. Therefore, subspace-based anomaly detection is based on the idea that anomalous entities are better detectable in the orthogonal subspace spanned by the background distribution. In this context, the projection separation index for an image $\mathbf{r}_\mathrm{j}$ is calculated as [35]

$$\mathbf{d} = (\mathbf{r}_\mathrm{j} - \hat{\boldsymbol{\mu}}_b)' \cdot \mathbf{P} \cdot (\mathbf{r}_\mathrm{j} - \hat{\boldsymbol{\mu}}_b) \tag{2}$$

where $\hat{\boldsymbol{\mu}}_b$ is the estimated average pixel of the background samples and $\mathbf{P}$ is the matrix that projects the data onto the orthogonal subspace to one spanned by the background samples.

On this basis, the issue around the detection of anomalous targets can be split in two main problems, one is the modeling of the background pattern and the other is the computation of the orthogonal subspace spanned by it in order to mark those pixels with the highest projection as potential anomalies. Therefore, the functionality of the HW-LbL-FAD algorithm is described in four computing stages in order to address the two aforementioned tasks. Fig. 1 shows the dataflow diagram of the HW-LbL-FAD computing stages, in which the inputs and outputs are also depicted. The two first stages are performed sequentially, while the two last are performed in parallel. These computing stages are further analyzed in the following sections and summarized in Algorithm 2.

1) Modeling of the Subspace Spanned by the Background Pattern: The first two computing stages of the HW-LbL-FAD algorithm, referred to as *Stages 1* and *2*, are in charge of the background estimation. In this sense, each hyperspectral frame, understood as a line of hyperspectral pixels captured by a push-broom scanner per shot, is independently processed using the set of core operations introduced in Section II-B. These two algorithm computing stages are further analyzed as follows.

a) Stage 1: Line-by-line extraction of the background reference spectra: The first stage of the HW-LbL-FAD algorithm consists in the feature selection of a set of reference spectra representative of the background distribution. To do so, the first n_f hyperspectral frames; $\mathbf{M}_k$, $k = [1, \ldots, n_f]$; captured

Algorithm 2: The HW-LbL-FAD Algorithm.

Inputs: $\mathbf{HI} = [\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{\frac{nr \cdot nc}{BS}}], n_f, \alpha$
Outputs: $\mathbf{AD} = [\mathbf{x}_{11}, \mathbf{x}_{12}, \ldots, \mathbf{x}_{kj}]$
Algorithm:
Stage 1:
1: **for** $\mathrm{k} = 1$ **to** n_f **do**
2: $\mathbf{E}_\mathrm{k} = \mathbf{Algorithm1}(\mathbf{M}_\mathrm{k}, \alpha)$
3: $\mathbf{B}* = [\mathbf{B}*, \mathbf{E}_\mathrm{k}]$;
4: **end for**
Stage 2:
5: $[\hat{\boldsymbol{\mu}}_b, \mathbf{Q}, \mathbf{U}, \tau] = \mathbf{Algorithm1}(\mathbf{B}*, \alpha)$
Stage 3: Applied to each new received frame, $\mathbf{M}_k$, $k > n_f$
6: **for** $\mathrm{j} = 1$ **to** BS **do**
7: Centralization: $\mathbf{c}_j = \mathbf{r}_j - \hat{\mu}_b$
8: **for** $\mathrm{n} = 1$ **to** p **do**
9: Projection: $v = \mathbf{U}'_n \cdot \mathbf{c}_j$
10: Subtraction: $\mathbf{c}_j = \mathbf{c}_j - \mathbf{Q}_n \cdot v$
11: **end for**
Stage 4:
12: Brightness AD: $\mathbf{d}_j = \mathbf{c}'_j \cdot \mathbf{c}_j$
13: **if** $\mathbf{d}_j \leq 1.5 \cdot \tau$ **then** $(\mathbf{x}_{kj} = 0)$
14: **else** $(\mathbf{x}_{kj} = 1)$
15: **end if**
16: **end for**

by the hyperspectral scanner are independently processed in order to select the most different spectra within them, $\mathbf{B}* = [\mathbf{E}_1, \ldots, \mathbf{E}_{\mathrm{n_f}}]$ (see lines 1–4 of Algorithm 2). It is done under the assumption that these image blocks are fully representative of the background distribution, and hence, free of anomalies. For this reason, it is critical that enough n_f hyperspectral frames are taken to cover all spectral variability of the background pattern. In addition, background samples obtained in previous flights may be used instead of obtaining them from the first n_f frames.

b) Stage 2: Overall background subspace estimation: Since the hyperspectral frames are independently processed in the *Stage 1* ruling out any spatial alignment restriction, $\mathbf{B}*$ comprises several alike spectra. As a consequence, it is required to obtain a subset of the most dissimilar pixels within $\mathbf{B}*$ that better define the background distribution, which is done in the *Stage 2* of the HW-LbL-FAD algorithm (see line 5 of Algorithm 2). To do so, the set of core operations is applied once again, though the input matrix, $\mathbf{M}_k$, is now replaced by $\mathbf{B}*$ whose columns collect the background reference vectors extracted from each first n_f frames. As outputs, we obtain the background subspace comprised of orthogonal vectors, $\mathbf{Q}$ and $\mathbf{U}$, whose definition prevents the implicit computation of the orthogonal projection matrix, $\mathbf{P}$, in *Stage 3* of the algorithm, which is a very computationally demanding task.

Additionally, $\mathbf{C}$ in Algorithm 1 retains the spectral information lost if the background is reconstructed using the p reference vectors selected in this *Stage 2*. On this basis, the remaining maximum brightness in $\mathbf{C}$, $\tau = b_{j_{\max}}$ (see line 16 of Algorithm 1), could be potentially used as a benchmark to identify anomalous pixels in *Stage 4*.

2) Computation of the Orthogonal Subspace Spanned by the Background Pattern and Detection of Anomalous Targets: The last two computing stages of the HW-LbL-FAD algorithm, referred to as *Stages 3* and *4*, are in charge of the estimation of the orthogonal subspace to the one spanned by the background pattern computed in the two previous stages and the detection of the anomalous pixels. For doing so, each hyperspectral pixel, $\mathbf{r}_j$, within new sensed hyperspectral frames, $\mathbf{M}_K$, $k > n_f$, is independently processed using some of the already mentioned set of core operations, as it is analyzed in more details in the following lines.

a) Stage 3: Computation of the orthogonal subspace to the one spanned by the background pattern: The third stage of the HW-LbL-FAD algorithm focuses on the computation of the orthogonal subspace matrix, $\mathbf{P}$. Nevertheless, this calculation is computationally expensive since it implies a matrix inverse computation whose dimension directly depends on the number of background samples, p. On this basis, the HW-LbL-FAD algorithm takes into consideration this issue and offers an optimized alternative that leads to a posterior more easy-handled hardware acceleration of the computing stage. Consequently, the orthogonal projection matrix, $\mathbf{P}$, is not explicitly calculated.

In this regard, the projection separation index, d, denoted in (2), actually matches with the brightness of each image pixel, $\mathbf{r}_j$, after being subtracted all the information belonging to the background, and hence, that is orthogonal to the subspace spanned by the background distribution. Hence, it is also equivalent to applied the Gram–Schmidt orthogonalization method in order to obtain the orthogonal component of each image pixel, $\mathbf{r}_j$, to the subspace spanned by the $\mathbf{Q}$ and $\mathbf{U}$ vectors outputted in *Stage 2*, which are indeed an orthogonal representation of the background distribution (see lines 6–11 of Algorithm 2). As it can be noticed, following this strategy permits to reuse the already defined set of core operations, which leads to an optimization of the human and hardware resources and lower time-to-market.

b) Stage 4: Detection of anomalies: Finally, d score computation is actually equal to the brightness calculation of the remaining spectral information contained in each image pixel, $\mathbf{r}_j$, after *Stage 3*. Then, if d is higher than a threshold, it is labeled as an anomaly. In this sense, the HW-LbL-FAD also defines an automatic thresholding method that enables the on-the-fly line-by-line segmentation of the anomalous targets. For doing so, it used the τ estimated in *Stage 2*. Hence, a potential anomalous pixel must derive in a $d > 1.5 \cdot \tau$ (see lines 12–16 of Algorithm 2).

III. Methods II: Hardware Implementation

The HW-LbL-FAD method has been implemented on an FPGA-based computing platform by using an HLS tool to define independent specialized hardware accelerators for each type of operations involved in the algorithm computing stages. HLS technology allows engineers to reduce the development time and complexity of custom-hardware designs by introducing high-level languages (HLLs), such as C or C++, to describe the

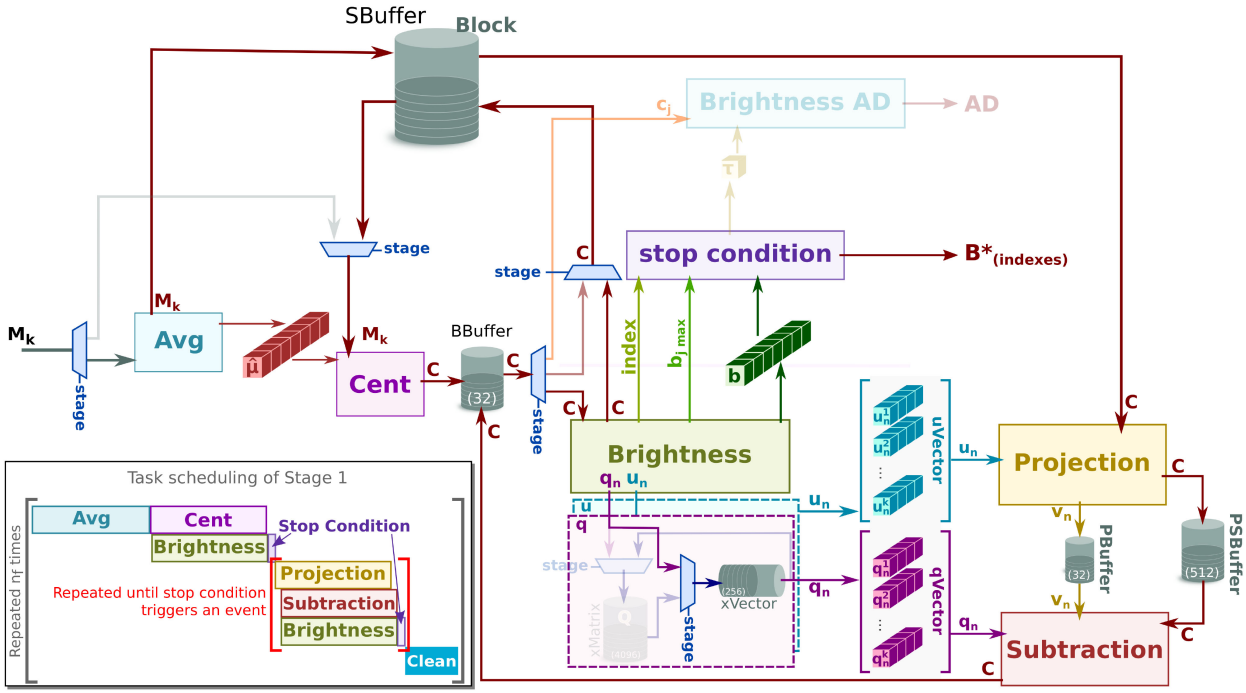

Fig. 2. Stage 1: Line-by-line extraction of the background reference spectra.

functionality that is later translated into a lower level register transfer level (RTL) model [36]. HLS frameworks provide the developer with the means to either optimize and improve legacy HLLs models wrote by nonhardware engineers or initiate new project from scratch.

In this work, we leverage the use of HLS tools to rise the productivity and shorten the development cycles. To this end, the HLS models of some core operations used by the HW-Lbl-FAD algorithm were inherited from a previous development [31] targeting the accelerated version of a hyperspectral imaging compression application. For this legacy components, the main work to be performed was their adaptation so that they match the functional and nonfunctional requirements of the HW-Lbl-FAD implementation. This component-tailoring process is shorter and less laborious than approaching the design of a new one, which resulted into the reduction of the effort and completion time of the project. However, the process is not completely free from fine-tuning and optimization tasks in order to reduce latency and increase the throughput.

Therefore, HW-Lbl-FAD datapath is a combination of reenginered legacy modules and new ones, implementing the processing of the hyperespectral data. Once such modules were modeled and tested, it was necessary to orchestrate their functioning in order to implement the behavior of the different stages of the anomaly detector. Also, the synchronization between stages needed of a supra entity responsible for routing the data through the hardware modules. The approach followed to provide a solution to this two challenges was to implement a finite state machine (FSM) in VHDL that activates the corresponding selectors (blue trapezoids in Figs. 2–4), and also each specialized hardware accelerator at the right time. The management of hardware modules is driven by the standard handshake protocol of Vitis HLS. This protocol defines four control signals to the hardware interface of the modules; it contains an input control signal, *start*, which is used to activate the module. In addition to this signal, the protocol also includes three output control signals that report the status of the modules; the *done* signal is used to indicate that the module has finished its execution, the *idle* signal states that the module is at an idle state, and the *ready* signal marks the time at which the module can receive another input data.

From the performance point of view, the arithmetic has been adapted to fixed-point precision in order to be suitable for FPGA technology, which is more efficient dealing with integer operations. In previous works [31], [33], it was demonstrated that the simplification of the arithmetic operations by employing integer arithmetic and bit shifting [37] did not affected significantly to the accuracy of the results, even increasing the quality of the output in some configurations of the input. Thus, applying the lessons learnt from previous works, the size of hyperspectral information has been set to a maximum of 12 bits, which is the most common bit size in remote sensing applications [38], [39] and also provides good accuracy results as is shown in [31].

For the sake of clarity, the following sections describe the hardware implementation of the four stages that compose the HW-LbL-FAD algorithm (see Algorithm 2) and explain the decision-making process followed in each stage. Furthermore, the FPGA-based architecture is only one, but it has been divided into three figures, each of them showing how the core of operations carries out a particular stage of the HW-LbL-FAD. To do so, only the modules and connections active during one stage are highlighted, while the others are watermarked.

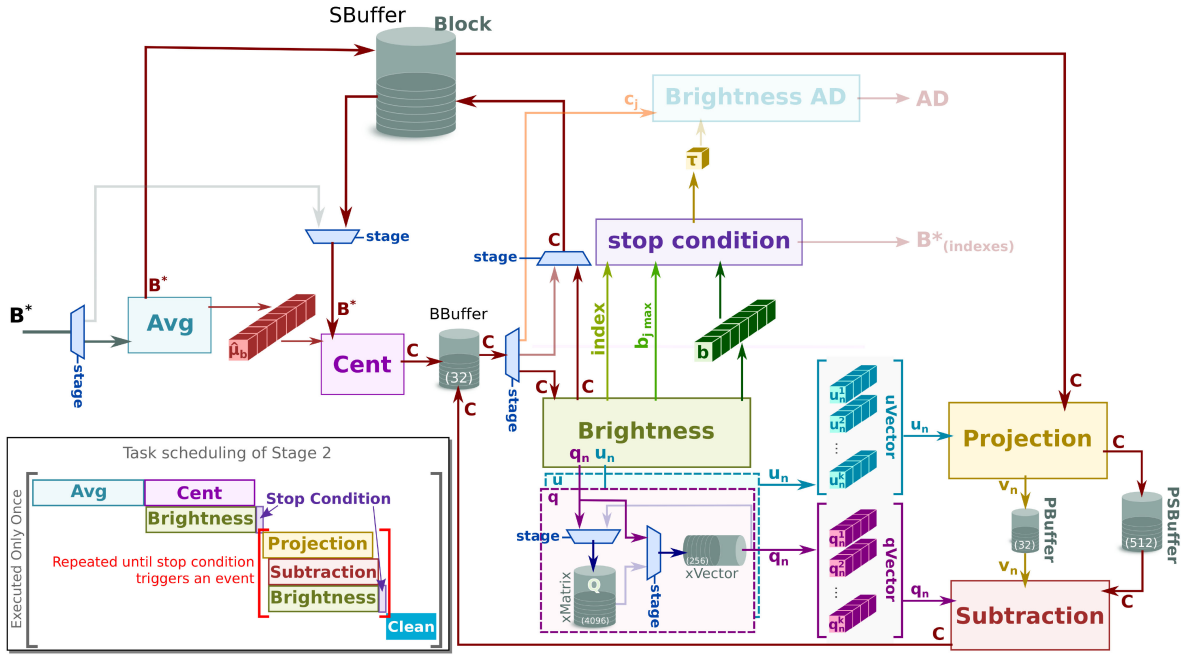

Fig. 3. Stage 2: Overall background subspace estimation.

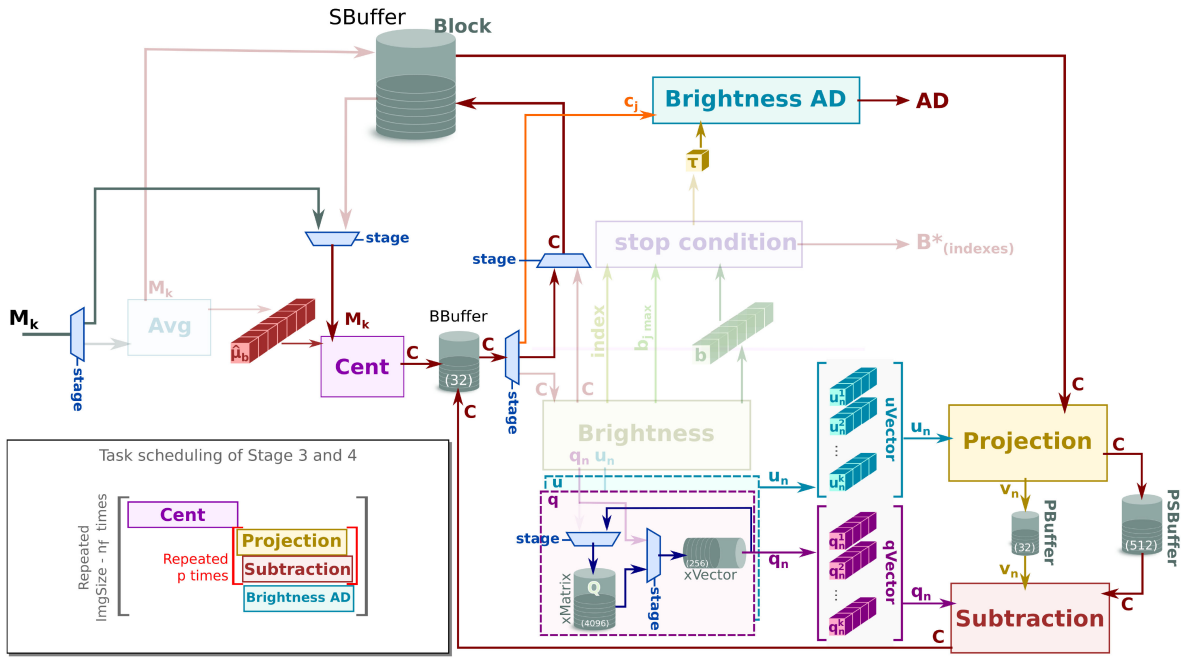

Fig. 4. Stage 3 and 4: Computation of the orthogonal subspace spanned by the background pattern and detection of anomalous targets.

A. *Stage 1*

Fig. 2 highlights the operations performed in the first stage of the HW-LbL-FAD algorithm to extract the background reference spectra. First, the original hyperspectral block, $\mathbf{M}_k$, is stored in an internal memory (*SBuffer*) and its centroid or average pixel ($\hat{\boldsymbol{\mu}}$) is obtained. Both operations are performed by the *Avg* module, which is able to compute several bands of a hyperspectral block in parallel (line 1 of Algorithm 1). Once the *Avg* module finishes, the *Cent* module can start the centralization of the original hyperspectral block by reading it from the *SBuffer*, i.e., this operation consists in subtracting the average pixel, which was calculated by the *Avg* module, from each hyperspectral pixel of the original block, $\mathbf{M}_k$ (line 2 of Algorithm 1). Therefore, the *SBuffer* first in, first out (FIFO) must guarantee enough depth to store a hyperspectral block.

The result of the *Cent* module is written into the *BBuffer* FIFO, whose depth is tiny in contrast to the *SBuffer*. This small depth

is due to the fact that, as soon as the output of the *Cent* module is ready, the next module can consume it. Hence, the loop in Algorithm 1 can start without waiting for the completion of the block centralization step.

Unlike the *Avg* and *Cent* modules, the HLS model of the *Brightness* component had to be revisited in order to implement the dynamic stop condition required by the HW-Lbl-FAD algorithm. This feature was not supported by the version developed in [31] since the number of iterations was fixed at design time, given a target compression rate.

The number of times the *Brightness*, *Projection*, and *Subtraction* steps are executed depend on the brightest hyperspectral pixel selected during the current iteration, its brightness in the first iteration, and the α input parameter (line 8 of Algorithm 1). All these values, but the α parameter that is fixed at design time, are provided by the *Brightness* module. Therefore, the *Brightness* module stores all brightness values calculated in the first iteration in a BRAM (**b** element in green of Fig. 2) and provides to the *Stop condition* module the value of the brightest hyperspectral pixel ($b_{j\max}$) and its index inside the hyperspectral block ($index$), which is currently being processing. The *Stop condition* module determines the number of iterations to be carried out, i.e., it implements the control of *exit* variable of Algorithm 1 and the *if* statement. In this regard, the *if* condition implies that at least the *brightness* module is executed twice, since in the first iteration, the condition is evaluated to false because the same brightness is being compared and there is no difference. The *Stop condition* module is also responsible for providing the indexes of the hyperspectral pixels that correspond to the background reference spectra ($\boldsymbol{B}^*$), thus it returns one index per iteration whenever the condition of *if* statement is evaluated to false.

Provided that the *Stop condition* does not trigger the loop exit, the rest of the operations inside the loop of Algorithm 1 can be carried out. Thus, the orthogonal projection vectors $\boldsymbol{q_n}$ and $\boldsymbol{u_n}$ are accordingly obtained using the brightest pixel computed by the *Brightness* module (lines 12 and 13 of Algorithm 1) and stored in separate FIFOs (depicted as *xVector* in Fig. 2): *qVector* and *uVector*, respectively. From a performance point of view, the *Brightness* module has been optimized, introducing a handmade ping-pong buffer that allows to obtain better time results compared to the initial inherited implementation in [31], thanks to the fact that the initial interval (II) between hyperspectral pixels has been halved. This new implementation maintains the independence of the location of the brightest hyperspectral pixel.

Although the *Projection* and *Subtraction* modules are represented by separate boxes in Fig. 2, it must be mentioned that both perform their computations in parallel. The *Projection* module reads a hyperspectral block from *SBuffer*, which has been previously written by the *Brightness*. In the first iteration, the *Brightness* module is fed with the output of the *Cent* module (that is, the centralized block) and consumes the output of the *Subtraction* module during the second and following iterations. Therefore, in *SBuffer* is always present a copy of the hyperspectral block that must be processed. The *Projection* module obtains the projected image vector according to line 14 of Algorithm 1. In parallel, the hyperspectral pixel is written in *PSBuffer*, which can store two hyperspectral pixels that is the minimum depth to prevent the dataflow from stalling. It is because the *Subtraction* module needs the same hyperspectral pixel read by the *Projection* module and the projections returned by such module (written in *PBuffer*) to obtain the subtracted image according to line 15 of Algorithm 1. The *Projection* and *Subtraction* modules read the orthogonal projection vectors $\boldsymbol{u_n}$ and $\boldsymbol{q_n}$, respectively. However, we have improved this operation by implementing a handmade array partition of these vectors, thus several bands of a hyperspectral block can be operated in parallel.

From the point of view of the dependencies among the different modules that compose the implementation of the HW-LbL-FAD algorithm, the *Cent* module cannot start until the centroid, $\hat{\boldsymbol{\mu}}$, is calculated by the *Avg* module, so these hyperspectral operations are executed one after the other. However, the *Brightness* module can start just when the first output of the *Cent* module is ready. Then, the *Stop condition* module must determine whether the loop *Projection*, *Subtraction*, and *Brightness* modules must be executed (operations inside the red box found at the bottom left corner in Fig. 2). The three hardware modules can perform its operations in parallel; the projected and subtracted image tasks have not dependencies with the next brightness operation, i.e., intraloop dependencies have been solved. The FSM must be informed when it should stop the loop, thus, the *Stop condition* module must report such event to it. Finally, the FSM cleans the internal memories that contain invalid values for the following iterations or stages. This is due to the fact that the last iteration stores the hyperspectral block in *SBuffer* and new orthogonal vectors are calculated but not used, that is the *xVector* FIFOs contain data. All of these tasks are repeated n_f times or, in other words, n_f hyperspectral blocks are needed to obtain the background reference spectra, besides, the tasks must be scheduled as shown in the bottom box of Fig. 2.

B. Stage 2

The architecture of the HW-LbL-FAD hardware accelerator is an evolution of the proposed design for the hyperspectral image compression use case presented in [31]. As it was previously mentioned, HLS tools and design workflow provides a neat, easy, and fast way to customize already available models and integrate them in new designs. In this work, concerning *Stage 1*, the focus of the effort has been put on optimizing the legacy FPGA-based architecture and components, reducing the bottleneck points in order to achieve a higher performance and making them suitable for the new application scenario (i.e., new functionality added and interface accommodation at the RTL level.

Nevertheless, the HW-LbL-FAD algorithm involves new steps, not covered in the compression of hyperspectral images, that must be carried out by the accelerator. Taking as the stating point, the Stage 1 enhanced datapath, the new functionality to be supported implies, mainly, the reengineering of the routing resources and intermodule connection infrastructure. The HW-LbL-FAD algorithm was envisioned, from the conception phase, as a solution targeting small FPGA devices with a

low-power consumption, limited-resource. Thus, the modeling of the background distribution and the detection of potential anomalous spectra were designed to use the same set of core operations that, when it comes to physical implementation, results into shared computing resources. Behavioral architectural variations are done by small selectors (governed by the FSM control signals) whose input or output channel depends on the current stage. Thereby, the same hyperspectral hardware operators can be reused time multiplexed throughout the different stages.

The input hyperspectral block, $\boldsymbol{B}^*$, for this second stage is the background reference spectra outputted during *Stage 1*. In contrast to this previous stage, the second one only operates with a single hyperspectral block instead of $\boldsymbol{n_f}$ blocks. However, it is highly likely that the block size will be different in both stages. Implementing a solution where the block size is variable causes an increase of hardware resources and breaks potential hardware optimizations, so the proposed solution maintains the block size across all stages to 1024 hyperspectral pixels. Thus, the hyperspectral block processed by the second stage must be extended up to 1024 pixels by repeating the data of the hyperspectral background block in order.

Fig. 3 shows an overview of the FPGA-based solution for the second stage. It is worth noticing that the hardware modules are the same than in the previous stage, while the datapath has been slightly modified by introducing two main architecture variations. First, the orthogonal projection vectors $\boldsymbol{u_n}$ and $\boldsymbol{q_n}$ extracted from the background hyperspectral block, $\boldsymbol{B}^*$, are stored into internal memories, *uMatrix* and *qMatrix*, and are also consumed by *Projection* and *Subtraction* modules, respectively. In other words, the outputs of the *Brightness* module feed the following modules like it was in *Stage 1*, and besides, a copy of these outputs are stored to be used by the next stages. The number of orthogonal projection vectors, $\boldsymbol{u_n}$ and $\boldsymbol{q_n}$, representative of the background distribution is also stored. Second, the other change concerns the *Stop condition* module that does not provide any output in this case. Now, the *Stop condition* module stores in a register the remaining maximum brightness, τ, of the background hyperspectral block, $\boldsymbol{B}^*$.

Similarly to *Stage 1*, when the background hyperspectral block, $\boldsymbol{B}^*$, is processed, some internal memories are cleaned up. These memories store the four outputs that line 5 of Algorithm 2 states; the *qMatrix* and *uMatrix* FIFOs, depicted as *xMatrix* in Fig. 3, stores the $\boldsymbol{Q}$ and $\boldsymbol{U}$ vectors, respectively, the centroid of the background hyperspectral block, $\hat{\boldsymbol{\mu}}_{\boldsymbol{b}}$, and the remaining maximum brightness, τ, used as a threshold in the subsequent *Stage 4*. The bottom box in Fig. 3 shows the scheduling of the tasks, which coincides with the previous stage, so there are no differences in terms of data dependencies.

C. Stages 3 and 4

Stage 3 is the most different from the others, because it uses only three hyperspectral operators (see Fig. 4). First, the new received hyperspectral block, $\boldsymbol{M_k}$, is centralized using the centroid obtained from *Stage 2*, $\hat{\boldsymbol{\mu}}_{\boldsymbol{b}}$. Thus, the centroid is not calculated any more when a new hyperspectral block is processed. Hence, the *Avg* module is bypassed and the *Cent* module is actually the first operation to be performed (line 7 of Algorithm 2).

Once the hyperspectral block is centralized, it is stored in *SBuffer* FIFO for being processed later by the *Projection* and *Subtraction* modules, which form the new loop body of this stage (lines 9 and 10 of Algorithm 2). Now, the number of iterations is not dynamic but it is known in advance and matches the number of orthogonal vectors extracted in the *Stage 2*, p. Therefore, neither *Brightness* nor *Stop condition* modules are enabled at this stage, and thus, the *Brightness* module does not generate any new orthogonal vectors, $\boldsymbol{u_n}$ and $\boldsymbol{q_n}$. Anyway, the *Projection* and *Subtraction* modules obtain the projected image, $\mathbf{v}_n$, and the subtracted image, $\mathbf{C}$, respectively, from the centralized hyperspectral block, using the orthogonal vectors that were calculated in *Stage 2* and stored in *qMatrix* and *uMatrix*. It is worth mentioning that these orthogonal vectors must be remembered for the following blocks. To address such feature, the *qMatrix* and *uMatrix* memories have been customized to store the data being read, i.e., the data are kept in the same memory space and order (see dark blue arrows of Fig. 4). In addition, to be compatible with the previous stages, the orthogonal vectors, $\boldsymbol{u_n}$ and $\boldsymbol{q_n}$, are also copied into *uBuffer* and *qBuffer*, respectively. Thus, the data are injected through the same channel to the *Projection* and *Subtraction* modules.

Unlike *Stages 1* and *2*, the subtracted image obtained in the last iteration of the loop is not discarded. Therefore, the output generated by the *Subtraction* module in the last iteration feeds *Stage 4* (see orange arrow of Fig. 4). This stage is composed by a unique module called *Brightness AD*, that is responsible to obtain the anomaly map by identifying the anomalous pixels. The *Brightness AD* module processes the $\boldsymbol{c_j}$ block to obtain the brightness of each hyperspectral pixel that compose the last subtracted image and compare them with the threshold, $\boldsymbol{\tau}$, obtained in *Stage 2* (lines 12 and 13 of Algorithm 2, respectively). The output of this module is a map of anomalies, where each bit reports the state of a pixel; 1 states that is an anomalous pixel (the brightness value is greater than the threshold) and 0 indicates that the pixel has not anomalous spectra.

Stages 3 and *4* can perform their tasks in parallel once the hyperspectral block is centralized. The bottom box in Fig. 4 shows the scheduling of these tasks, the *Projection* and *Subtraction* modules maintain the dependence between them. As a novelty, the internal memories are not cleared here since it is the *Brightness AD* module that is in charge of reading the subtracted image generated by the *Subtraction* module in the last iteration. This fact is possible due to the number of iterations is known in advance (p times) and the FSM manages it by a counter.

IV. Results

This section evaluates the proposed anomaly detector by analyzing the hardware architecture implemented on a ZC7Z020-CLG484 FPGA device, as well as, the performance achieved by comparing the obtained results.

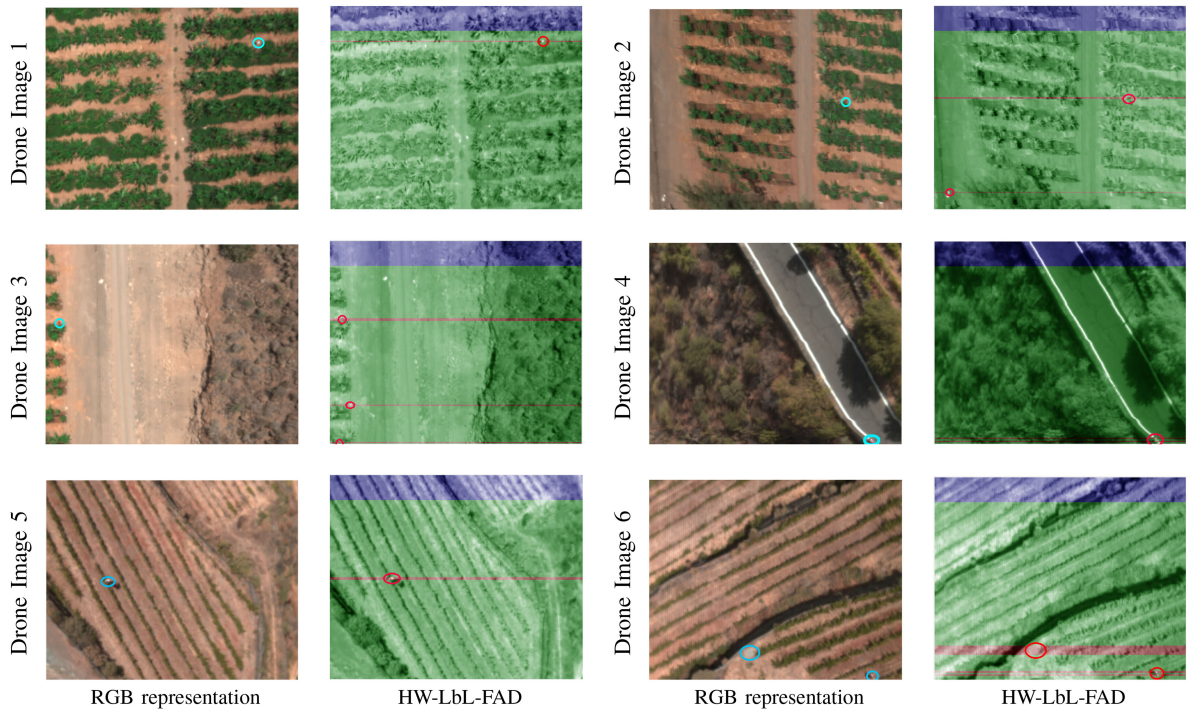

Fig. 5. Anomaly detection results obtained by the HW-LbL-FAD.

A. *Detection Performance Analysis*

The accuracy of the detection performance has been carried out in software due to the flexibility to modify the algorithm input parameters such as, the block size (BS) and the number of image blocks used to estimate the background pattern (n_f). This analysis was previously published in [29] and [40], in which three state-of-the-art anomaly detectors were selected to compare their output with the software version of the HW-LbL-FAD algorithm. In particular, the chosen algorithms were the OSPRX [41], the LSMAD [11], and the PLP-KRXD [42]. The analysis concluded that the HW-LbL-FAD algorithm features a high detection capability in all scenes, significantly outperforming the other state-of-the-art methods.

This work extends previous studies and also analyzes the detection accuracy achieved by the proposed hardware architecture of the HW-LbL-FAD algorithm to determine whether there are differences with respect to the software-implemented counterpart. To do so, it has processed a set of six HSIs taken by a custom acquisition platform over different farming areas on the island of Gran Canaria (Spain). The first flight campaign was carried out over a plantation of bananas in the South-West of the Island ($27°52'17.4''$N $15°45'44.2''$W), while the second and third flight campaigns were carried over some vineyard areas in the center of the island ($27°59'35.6''$N $15°36'25.6''$W and $27°59'15.2''$N $15°35'51.9''$W). The dataset was collected by a custom aerial platform that mounts a *Specim FX10* pushbroom hyperspectral camera on a DJI Matrice 600 drone [43]. The image sensor works in the range of the electromagnetic spectrum between 400 and 1000 nm using 1024 spatial pixels per scanned cross-track line and 224 spectral bands. Nevertheless, the hyperspectral images used in the experiments only retain the spectral information of 160 spectral bands; the first 20 and the last 45 spectral bands have been discarded because of the bands near to the sensor limits do not provide accurate spectral response. The input hyperspectral images have 825 blocks of 1024 pixels and 160 spectral bands, where the first 100 blocks are used to calculate the background spectra ($\boldsymbol{n_f}$) in *Stage 1* and the following blocks (725) are analyzed for anomalies.

The selected dataset contains some artifacts that are considered to be anomalous pixels. Their locations within the test bench have been highlighted in the RGB representation of each scene in Fig. 5 using blue circles. These artifacts are people walking among the crop fields in *Drone Image 1*, *Drone Image 2*, *Drone Image 3*, and *Drone Image 5*. In *Drone Image 4*, the anomalous entities are two people standing next to the road. Finally, *Drone Image 6* contains a person walking through the vineyard area and a concrete construction.

The HSIs acquired by the UAV-based aerial platform have been used as the inputs to both the software implementation of the HW-LbL-FAD algorithm and the hardware architecture proposed in this work. Doing so, we are able to check whether the behavior of the hardware component differs from its software counterpart. Fig. 5 displays the detection maps produced by the hardware implementation (column of pictures labeled as *HW-LbL-FAD*), which have been superimposed on a panchromatic representation of the analyzed scenes. Lines in red color indicate spatial hyperspectral frames corrupted by anomalous signatures. As it can be observed, the outputs of the hardware and software implementations are identical, validating the operation of the proposed architecture.

TABLE I
HARDWARE UTILIZATION OF HW-LBL-FAD ALGORITHM FOR A XILINX ZYNQ-7020 PROGRAMMABLE SOC AFTER POSTSYNTHESIS PHASE

PEs	BRAM18K	DSP48E	FlipFlops	LUTs
1	214 *(76.43%)*	14 *(6.36%)*	8,073 *(7.59%)*	6,744 *(12.68%)*
2	183 *(65.36%)*	22 *(10%)*	8,624 *(8.11%)*	7,470 *(14.08%)*
4	191 *(68.21%)*	38 *(17.27%)*	9,981 *(9.38%)*	9,115 *(17.13%)*
5	193 *(68.93%)*	46 *(20.9%)*	10,656 *(10.01%)*	9,940 *(18.68%)*
8	197 *(70.36%)*	70 *(31.82%)*	12,666 *(11.9%)*	12,411 *(23.33%)*
10	205 *(73.21%)*	86 *(39.09%)*	14,787 *(13.9%)*	14,856 *(27.92%)*
16	193 *(68.93%)*	134 *(60.91%)*	18,433 *(17.32%)*	18,724 *(35.2%)*
20	197 *(70.36%)*	166 *(75.45%)*	23,071 *(21.68%)*	23,493 *(44.16%)*

B. Hardware Analysis

Although there is a wide variety of FPGA devices and technologies (e.g., Artix, Kintex, or UltraScale+), the proposed architecture has been implemented on a ZC7Z020-CLG484 FPGA device (Artix architecture) due to its low-cost and low-power consumption, features that make it attractive for embedded applications. It is worth noting that the ZynQ SoCs have a balanced performance per watt ratio, which is a very important metric for space applications. However, the FPGA resources in this kind of devices are limited, posing a new challenge in the hardware design process. The reuse of already available functionality in previously implemented hyperspectral hardware operators and adapting them to be compatible across stages has been the key to meet the constraints set by the physical platform. Replication of modules was not feasible in this scenario. The choice of a device of this manufacturer implies the use of its tools. In particular, the Vitis Design Suite has been used to prototype the proposed architecture.

Table I summarizes the resources required by each configuration of the hardware accelerator, which have been extracted from postsynthesis reports. The proposed architecture sets the hyperspectral block size in 1024 hyperspectral pixels (BS) with 12-bits depth and 160 bands, while the number of processing elements (PE) is variable and it depends on the number of bands that contains the hyperspectral block. The PE parameter determines the number of hyperspectral bands that can be processed concurrently by the *Brightness*, *Brightness AD*, *Projection*, and *Subtraction* modules. PE must be a divisor of the number of bands so the HLS model for these operators can keep the compliance with the recommended coding style conventions. It is worth mention that the available resources of the ZC7Z020-CLG484 FPGA device admits a maximum configuration of 20 PEs since the 32 PEs version requires 262 digital signal processing units (DSPs) but only 220 are available. Therefore, the DSPs are the limiting factor that handicaps the scalability of the architecture together with the summation tree when using a map-reduce model as a solution [31]. In contrast, the BRAM resources do not depend on the number of PEs, it varies according to the block size (BS). Hence, all configurations use roughly 70% of the BRAM resources, while the FlipFlops and thelookup tables (LUTs) are not critical.

Performance results have been measured as the number of frames that can be processed per second (FPS). Table II lists the FPS achieve by the hardware accelerator according to the number of PEs instantiated and the configuration of the clock frequency. In this sense, the fastest clock frequency is really an overclocked version of the slowest clock. In this regard, the hyperspectral operators that have been described with the HLS were configured for a clock frequency of 100 MHz, so the Vitis HLS tool gets better scheduling and binding of hardware resources, which means less hardware resources. Then, the resulting architecture is overclocked with a clock frequency of 143 MHz. Performance results draw a linear behavior, i.e., the performance scales accordingly with the number of PEs. In addition, the fastest versions (143 MHz) achieve better performance results, roughly 40%, than the slowest ones (100 MHz). It is worth mentioning that a higher clock frequency, such as 200 MHz, is not possible because the synthesis tools are not capable of fulfilling the timing restrictions for this architecture.

TABLE II
FPS ACHIEVED BY THE DIFFERENT VERSIONS OF THE HW-LBL-FAD ACCELERATOR

PEs	1	2	4	5	8	10	16	20
100 MHz	84	169	338	424	592	697	948	1077
143 MHz	130	259	518	606	900	1055	1423	1609

TABLE III
DATASET SIZES AND PROCESSING TIME OF THE STATE-OF-THE-ART AND THE HW-LBL-FAD DETECTORS

Proposal	Data Set	Lines	Block Size	Bands	Freq. (MHz)	Time (s)
[47]	HyMap	614	512	126	200	1.09
	WTC	614	512	224	200	2.71
[48]	HyMap	100	300	126	100	0.375
	Hydice Forest	64	64	169	100	0.062
	Hydice Urban	80	100	175	100	0.124
[49]	San Diego	100	100	189	200	0.528
	Urban-Beach 1	100	100	207	200	0.553
	Urban-Beach 2	100	100	191	200	0.513
	Urban-Beach 3	100	100	205	200	0.553
	Urban-Beach 4	150	150	188	200	1.176
	El Segundo	250	300	224	200	7.856
ours	Drone (ours)	825	1024	160	100 143	0.76 0.51

TABLE IV
SUMMARY OF AVAILABLE HARDWARE RESOURCES OF THE DEVICES USED IN THE STATE-OF-THE-ART AND THE HW-LBL-FAD

Device (Proposal)	BRAM18K	DSP48E	FlipFlops	LUTs	Price (€)
XC7VX980T ([48])	3,000	3,600	1,224,000	612,000	30,949.10
XC7VX690T ([47])	2,940	3,600	866,400	433,200	16,577.93
XC7K325T ([49])	890	840	407,600	203,800	2,103.98
ZC7Z020 (ours)	280	220	106,400	53,200	154.70

Considering the features of the employed hyperspectral camera, the *Specim FX10* has a maximum frame rate of 327 FPS, each frame or captured line being 1024 pixels with 224 bands (full range) or/and 514 FPS for 1024 pixels and 140 bands per pixel [44]. On top of that, the smallest version of the HW-LbL-FAD accelerator capable of processing the data that comes from the image sensor is the one with 4 PEs and the clock frequency configured at 143 MHz. On the basis of the results listed in Tables I and II and the runtime requirements of the hyperspectral camera, the proposed architecture fits even in a smaller ZynQ device (XC7Z014S), which is also based on Xilinx's Artix FPGA architecture.

Fig. 6 graphically shows the power consumption obtained by the FPGA-based implementation for several versions of the

TABLE V
HARDWARE UTILIZATION OF STATE-OF-THE-ART AND HW-LBL-FAD ALGORITHMS

Proposal	Device	Data set	BRAM18K	DSP48E	FlipFlops	LUTs	Freq. (MHz)	On-chip Power (W)
[47]	XC7VX690T	HyMap	240 *(26.97%)*	265 *(31.55%)*	28,245 *(6.93%)*	21,730 *(10.66%)*	200	0.641
		WTC	284 *(9.47%)*	459 *(12.75%)*	43,544 *(3.56%)*	33,997 *(5.56%)*	200	0.897
[48]	XC7VX980T	HyMap	4 *(0.13%)*	1064 *(29.56%)*	772,164 *(63.09%)*	269,751 *(44.08%)*	100	2.72
		Hydice Forest	3 *(0.10%)*	460 *(12.78%)*	277,048 *(22.63%)*	140,362 *(22.93%)*	100	1.333
		Hydice Urban	3 *(0.10%)*	580 *(16.11%)*	414,673 *(33.88%)*	146,977 *(24.02%)*	100	1.619
[49]	XC7K325T	San Diego	216 *(7.35%)*	12 *(0.33%)*	49,964 *(5.77%)*	33,969 *(7.84%)*	200	0.614
		Urban-Beach 1	282 *(9.59%)*	12 *(0.33%)*	57,962 *(6.69%)*	43,890 *(10.13%)*	200	0.687
		Urban-Beach 2	198 *(6.73%)*	12 *(0.33%)*	46,370 *(5.35%)*	31,681 *(7.31%)*	200	0.593
		Urban-Beach 3	282 *(9.59%)*	12 *(0.33%)*	57,929 *(6.69%)*	43,874 *(10.13%)*	200	0.686
		Urban-Beach 4	282 *(9.59%)*	12 *(0.33%)*	58,593 *(6.76%)*	44,906 *(10.37%)*	200	0.691
		EI Segundo	525 *(17.86%)*	12 *(0.33%)*	83,713 *(9.66%)*	67,591 *(15.60%)*	200	0.911
ours	ZC7Z020	Drone (ours)	197 *(70.36%)*	166 *(75.45%)*	23,071 *(21.68%)*	23,493 *(44.16%)*	100 143	0.7 1.077

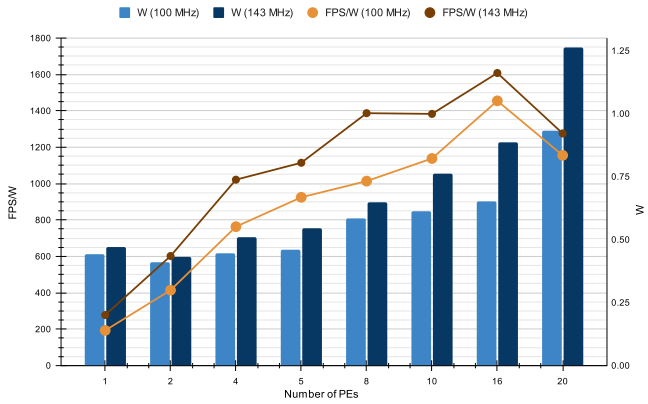

Fig. 6. Power tradeoff of different versions of the HW-LbL-FAD accelerator.

accelerator on a ZynQ-7020 device. From the point of view of energy consumption, the proposed architecture is highly efficient (right vertical axis, light, and dark blue bars), ranging from 0.5 W, when a PE is instantiated, to 1 and 1.3 W, when the architecture instantiates 20 PEs in 100 and 143 MHz, respectively. The consumption of sensed data draws an exponential behavior instead of a linear, because of the number of hardware resources and clock regions that must be fed. Despite these good results, the power consumption is not completed as the hardware accelerator depends on other resources, such as a DDR controller. For instance, energy budget is increased by 2.3 W when all components are assembled into a TE0720 SoC module.

The power efficiency in terms of FPS per watt (brown and yellow series) is also shown in Fig. 6. The information that draws this figure is too relevant for applications under power budget constraints, such as applications embedded in UAVs to onboard real-time processing. In this kind of applications, it is essential to extend the battery life due to unknown or unplanned scenarios, in which extra consumption is necessary to continue the mission. For example, an unexpected gust of wind means stabilizing the drone by increasing the speed of its motors, and hence, the energy consumption is greater than in a regular scenario. In this sense, this figure of merit can help to schedule a mission with the guarantee of a successful completion of the mission. Although the implementation at the highest clock frequency is slightly greater than the slower one, it is counterbalanced by the number of FPS processed. Therefore, the relation FPS/W is better for the fastest version (143 MHz) of the proposed accelerator.

V. DISCUSSION

The performance and hardware resource utilization of the HW-LbL-FAD algorithm has been compared with three state-of-the-art anomaly detectors also implemented on reconfigurable hardware. All of these detectors have been particularly developed for the 7-series FPGAs from Xilinx. In [45], the RX algorithm, optimized by a streaming background statistics approach, has been developed in a Kintex-7 XC7K325 T FPGA (KC705 Evaluation Kit). Another interesting anomaly detection algorithm is the collaborative-representation-based setector (CRD), which has been implemented on a Virtex-7 XC7VX980 T FPGA in [46]. Finally, the anomaly detector presented in [47] is based on morphological reconstruction and a simplified guided filter (Fast-MGD) that has been implemented on a Virtex-7 XC7VX690 T.

Each of the aforementioned state-of-the-art anomaly detectors uses its own dataset, which are listed in Table III. The size of each hyperspectral cube is defined by the number of lines (Y-spatial axis), block size (X-spatial axis), and the number of bands (spectral axis). The clock frequency of each solution and the time to process the hyperspectral cube are also shown in Table III. In addition, the HW-LbL-FAD detector also uses its own dataset, which corresponds to the sensed by the custom UAV described previously. Due to the variety of datasets, the time comparison has been done by the number of elements within the hyperspectral cube regardless of the anomaly detector, defined by the multiplication of the two spatial axis and the spectral one. Fig. 7 graphically shows the time (right vertical axis, light blue bars) required to compute the number of elements of the hyperspectral cube (left vertical axis, dark blue bars). In addition, Fig. 7 also draws the number of elements processed per second (right vertical axis, orange line). It is worth mentioning that the proposed architecture has a better time performance than the other proposals; the HW-LbL-FAD detector processes five and seven times more data than the second better proposal (see[45]) in its slowest (100 MHz) and fastest (143 MHz) versions, respectively. Although proposals presented in [46] and [47] employ tiny hypersepectral cubes, both achieve worse time-performance balance.

From the point of view of hardware utilization in terms of FPGA technology, Table V shows the hardware utilization by each anomaly detector in number of resources an its percentage according to the device and the dataset. Perhaps, this table

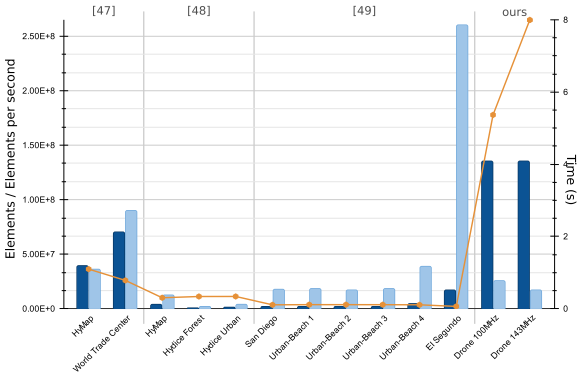

Fig. 7. Element-time processing tradeoff of different state-of-the-art anomaly detectors and the HW-LbL-FAD detector.

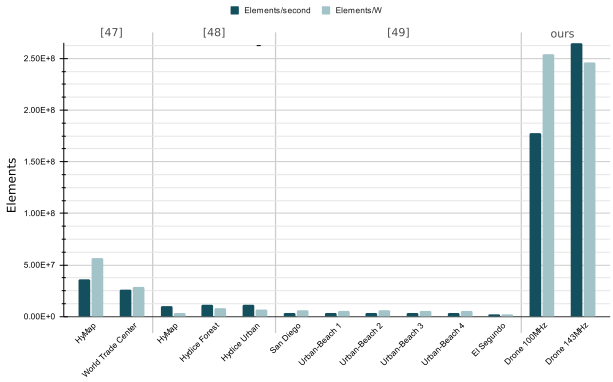

Fig. 8. Power tradeoff of state-of-the-art and HW-LbL-FAD detectors.

misleads the reader when try to interpret it, because one can observe that the state-of-the-art detectors use a small percentage of available hardware resources. However, this is not the case, the state-of-the-art proposals have selected large FPGA devices that make their results seem very promising, but the fact is that most implementation do not fit in a cost-optimized FPGA. The use of midrange and large-range devices (Kintex and Virtex architectures, respectively) instead of cost-optimized ones (Artix architecture) introduces a new dimension to a project: the cost of the final product. Table IV displays the price of each FPGA device and the summary of the number of available hardware resources. The architectures based on Virtex-7 devices, [46] and [47], have a high economic cost that may be unaffordable for embedded systems, while the one based on a Kintex-7 device has a better price but its performance is so far to process large hyperspectral cubes. However, our proposal is a low-cost solution that exploits the FPGA technology as much as possible, obtaining the best tradeoff between the performance and cost.

Analyzing the state-of-the-art architectures, we can observe that, like our proposal, [45] and [46] use fixed-point precision instead of floating-point notation, because this arithmetic is more suitable for FPGA technology providing better results. By contrary, [47] adopts floating-point arithmetic, whose performance is the worst of the evaluated proposals. On top of that, the FPGA-based architecture presented in [45] instantiates several FIFOs that play the role of data buffering, and hence, the BRAM resources are notably present in the summary of hardware utilization. Unfortunately, this architecture does not provide a high parallelism, despite having enough hardware resources to do so. The main problem is that the II between pixels is roughly 17 cycles, which is an aggravating factor in performance terms. In [46], the proposed architecture is not explained, and hence, it cannot be deeply analyzed. It can be deduced that authors try to avoid the use of BRAM and implement their solution by using other faster resources for data buffering purposes (FlipFlops and LUTs). Finally, the architecture described in [47] has been implemented by the use of HLS in which the stages are pipelined. However, designing a fully HLS-based architecture is not a good idea, as current tools are not able to decouple all data dependencies. In this sense, our FPGA-based architecture employs a mixed strategy by instantiating HLS-based modules connected through FIFOs, which has been generated as BRAMs, and VHDL-based FSM to synchronize the stages and tasks of the HW-LbL-FAD algorithm. This fact features our solution with a high degree of parallelism, where the II is one cycle, except in the first pixel of those tasks that depend on the previous one in which II is the number of bands, i.e., 160 cycles to start processing data.

From the energy point of view, Table V lists in its last column the on-chip power depicted in watts of each proposal, which has been estimated with the Xilinx Power Estimator tool (XPE) as the aforementioned proposals do not provide this information. For the sake of clarity, Fig. 8 draws the elements per second (dark green bars) and the processed elements per watt (light green bars). We can conclude that our solution beats the other state-of-the-art proposals in the power tradeoff, where slowest and fastest version of the HW-LbL-FAD are between 5 and 107 times better than the anomaly detectors presented in the literature. It is worth mentioning that both versions of the HW-LbL-FAD detector have a similar number of processed elements per watt (see Fig. 8).

VI. Conclusion

The main challenge faced in this work was the design of a low-power, energy efficient solution for a complex, multistaged anomaly detection algorithm for hyperspectral imaging targeting a reconfigurable logic platform. Together with the resource and power budget constraints, high performance requirements were also mandatory to be met.

The initial implementation of the core set of hyperspectral operations is inherited from previous works [31], [33], where the suitability of the FPGA technology for this type of applications was tested. So, this strategy has, therefore, saved a significant amount of time and effort, allowing the authors to focus on the enhancement of the performance and the design of a datapath and control logic that enabled the time sharing of the hardware modules and the optimal use of the resources. We also have gone one step further and provided minimal functional modifications to the operators so they became versatile and easily reused during the different stages of the algorithm. As a result, the core set of hardware hyperspectral operators not only is valid for image compression [31] but for anomaly detection task while not incurring in any overhead.

The reuse of resources was mandatory given the features of the FPGA platform the accelerator has been deployed on. On the one hand, the cost of the solution must be kept within a reasonable margin. The lower the cost of the reconfigurable chip, the lower the number of DSPs or in-chip memory available. On the other hand, this proposal aims to target embedded hyperspectral imaging processing systems such as the ones on board of UAVs or other autonomous, battery-powered platforms. In this sense, the size of the reconfigurable chip, the number of clock regions, and the working clock frequency have a direct impact on energy consumption. Thus, the proposed architecture contributes to overcome these challenges by performing the hyperspectral operations with the same hardware accelerators through the different stages by fundamentally modifying the routing of the data at run time.

In terms of detection performance, the HW-LbL-FAD achieves the same precision as that of its software version. It means that the accuracy of the anomalous pixel detection is high and greater than that achieved by the state-of-the-art algorithms. Regarding the time processing tradeoff, the proposed architecture processes large hyperspectral cubes in a short period of time with a small power budget compared to the state-of-the-art proposals. Moreover, the production cost of our architecture, by using FPGA devices, is attractive because of its low cost.

Although we focus in this work on a high data-rate scenario characterized by a push-broom scanner mounted on an UAV, the conclusions drawn from this work can be extrapolated to other fields where remotely sensed hyperspectral images ought to be processed in real time (e.g., spaceborne missions that carry space-grade FPGAs).

REFERENCES

[1] H. Su, Z. Wu, H. Zhang, and Q. Du, "Hyperspectral anomaly detection: A survey," *IEEE Geosci. Remote Sens. Mag.*, to be published, doi: 10.1109/MGRS.2021.3105440.

[2] S. Roy, S. Pathak, and S. N. Omkar, "Automated mineralogical anomaly detection using a categorization of optical maturity trend at lunar surface," *Int. J. Remote Sens.*, vol. 42, no. 21, pp. 8262–8297, 2021.

[3] J. Kuester, W. Gross, and W. Middelmann, "Hyperspectral anomaly detection of hidden camouflage objects based on convolutional autoencoder," *Proc. SPIE*, vol. 11870, pp. 98–106, 2021.

[4] I. 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.

[5] S. Matteoli, M. Diani, and G. Corsini, "Improved estimation of local background covariance matrix for anomaly detection in hyperspectral images," *Opt. Eng.*, vol. 49, no. 4, 2010, Art. no. 46201.

[6] C.-I. Chang and S.-S. Chiang, "Anomaly detection and classification for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 6, pp. 1314–1325, Jun. 2002.

[7] N. M. Nasrabadi, "Regularization for spectral matched filter and RX anomaly detector," *Proc. SPIE*, vol. 6966, 2008, Art. no. 696604.

[8] Q. Guo, B. Zhang, Q. Ran, L. Gao, J. Li, and A. Plaza, "Weighted-RXD and linear filter-based RXD: Improving background statistics estimation for anomaly detection in hyperspectral imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2351–2366, Jun. 2014.

[9] H. Kwon and N. M. Nasrabadi, "Kernel RX-algorithm: A nonlinear anomaly detector for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 2, pp. 388–397, Feb. 2005.

[10] Y. Xu, Z. Wu, J. Li, A. Plaza, and Z. Wei, "Anomaly detection in hyperspectral images based on low-rank and sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 4, pp. 1990–2000, Apr. 2016.

[11] Y. Zhang, B. Du, L. Zhang, and S. Wang, "A low-rank and sparse matrix decomposition-based Mahalanobis distance method for hyperspectral anomaly detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1376–1389, Mar. 2016.

[12] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Sparse representation for target detection in hyperspectral imagery," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 3, pp. 629–640, Mar. 2011.

[13] J. Li, H. Zhang, L. Zhang, and L. Ma, "Hyperspectral anomaly detection by the use of background joint sparse representation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2523–2533, Jun. 2015.

[14] W. Li and Q. Du, "Collaborative representation for hyperspectral anomaly detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1463–1474, Mar. 2015.

[15] A. Schaum, "Hyperspectral anomaly detection beyond RX," *Proc. SPIE*, vol. 6565, 2007, Art. no. 656502.

[16] F. Li, X. Zhang, L. Zhang, D. Jiang, and Y. Zhang, "Exploiting structured sparsity for hyperspectral anomaly detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 7, pp. 4050–4064, Jul. 2018.

[17] P. S. Singh and S. Karthikeyan, "Salient object detection in hyperspectral images using deep background reconstruction based anomaly detection," *Remote Sens. Lett.*, vol. 13, no. 2, pp. 184–195, 2022.

[18] M. Yousefan, H. E. Najafabadi, H. Amirkhani, H. Leung, and V. Hajihashemi, "Deep anomaly detection in hyperspectral images based on membership maps and object area filtering," *Expert Syst. Appl.*, 2021, Art. no. 116200.

[19] V. Y. Pan and Z. Q. Chen, "The complexity of the matrix eigenproblem," in *Proc. 31st Annu. ACM Symp. Theory Comput.*, 1999, pp. 507–516.

[20] T. Elgamal and M. Hefeeda, "Analysis of PCA algorithms in distributed environments," 2015, *arXiv:1503.05214*.

[21] A. Bunse-Gerstner and W. B. Gragg, "Singular value decompositions of complex symmetric matrices," *J. Comput. Appl. Math.*, vol. 21, no. 1, pp. 41–54, 1988.

[22] A. K. Cline and I. S. Dhillon, "Computation of the singular value decomposition," in *Handbook of Linear Algebra*, L. Hogben, Ed. London, U.K.: Chapman & Hall, ch. 45, 2006.

[23] J. Dongarra *et al.*, "The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale," *SIAM Rev.*, vol. 60, no. 4, pp. 808–865, 2018.

[24] 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, Mar. 2011.

[25] A. Alcolea, M. E. Paoletti, J. M. Haut, J. Resano, and A. Plaza, "Inference in supervised spectral classifiers for on-board hyperspectral imaging: An overview," *Remote Sens.*, vol. 12, no. 3, pp. 534–562, 2020.

[26] 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.

[27] A. D. George and C. M. Wilson, "Onboard processing with hybrid and reconfigurable computing on small satellites," *Proc. IEEE*, vol. 106, no. 3, pp. 458–470, Mar. 2018.

[28] 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.

[29] M. Díaz, R. Guerra, P. Horstrand, S. López, and R. Sarmiento, "A line-by-line fast anomaly detector for hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8968–8982, Nov. 2019.

[30] M. Díaz, "A new approach for the effective processing of hyperspectral images: Application to pushbroom-based anomaly detection and compression systems," Ph.D. dissertation, Universidad de Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain, 2021.

[31] J. Caba, M. Díaz, J. Barba, R. Guerra, and J.A. López, "FPGA-based onboard hyperspectral imaging compression: Benchmarking performance and energy efficiency against GPU implementations," *Remote Sens.*, vol. 12, no. 22, 2020, Art. no. 3741. [Online]. Available: https://www.mdpi.com/2072-4292/12/22/3741

[32] Q. Du, B. Tang, W. Xie, and W. Li, "Parallel and distributed computing for anomaly detection from hyperspectral remote sensing imagery," *Proc. IEEE*, vol. 109, no. 8, pp. 1306–1319, Aug. 2021.

[33] R. Guerra, Y. Barrios, M. Díaz, A. Baez, S. López, and R. Sarmiento, "A hardware-friendly hyperspectral lossy compressor for next-generation space-grade field programmable gate arrays," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 12, pp. 4813–4828, Dec. 2019.

[34] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*. Manchester, U.K.: Manchester Univ. Press, 1992.

[35] N. M. Nasrabadi, "Hyperspectral target detection: An overview of current and future challenges," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 34–44, Jan. 2014.
[36] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 4, pp. 473–491, Apr. 2011.
[37] E. L. Oberstar, "Fixed-point representation & fractional math," *Oberstar Consulting*, Madison, WI, USA, p. 9, 2007.
[38] C. Hu *et al.*, "Dynamic range and sensitivity requirements of satellite ocean color sensors: Learning from the past," *Appl. Opt.*, vol. 51, pp. 6045–6062, Sep. 2012.
[39] A. Kumar, S. Mehta, S. Paul, R. Parmar, and R. Samudraiah, "Dynamic range enhancement of remote sensing electro-optical imaging systems," in *Proc. Symp. Indian Soc. Remote Sens.*, 2012.
[40] M. Díaz, R. Guerra, and S. López, "A hardware-friendly anomaly detector for real-time applications with push-broom scanners," in *Proc. 10th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens.*, 2019, pp. 1–5.
[41] C.-I. Chang, "Orthogonal subspace projection (OSP) revisited: A comprehensive study and analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 502–518, Mar. 2005.
[42] C. Zhao, W. Deng, Y. Yan, and X. Yao, "Progressive line processing of Kernel RX anomaly detection algorithm for hyperspectral imagery," *Sensors*, vol. 17, no. 8, 2017, Art. no. 1815. [Online]. Available: https://www.mdpi.com/1424-8220/17/8/1815
[43] P. Horstrand, R. Guerra, A. Rodríguez, M. Díaz, S. López, and J. F. López, "A UAV platform based on a hyperspectral sensor for image capturing and on-board processing," *IEEE Access*, vol. 7, pp. 66919–66938, 2019.
[44] Spectral Imaging Ltd., "Specim FX10 datasheet," Nov. 23, 2021. [Online]. Available: https://www.specim.fi/wp-content/uploads/2020/02/Specim-FX10-Technical-Datasheet-04.pdf
[45] B. Yang, M. Yang, A. Plaza, L. Gao, and B. Zhang, "Dual-mode FPGA implementation of target and anomaly detection algorithms for real-time hyperspectral imaging," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2950–2961, Jun. 2015.
[46] J. Wu, Y. Jin, W. Li, L. Gao, and B. Zhang, "FPGA implementation of collaborative representation algorithm for real-time hyperspectral target detection," vol. 15, no. 3, pp. 673–685, Oct. 2018. [Online]. Available: https://doi.org/10.1007/s11554-018-0823-7
[47] J. Lei, G. Yang, W. Xie, Y. Li, and X. Jia, "A low-complexity hyperspectral anomaly detection algorithm and its FPGA implementation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 907–921, Oct. 2021.

Julián Caba received the M.S. and Ph.D. degrees in computer science from the University of Castilla-La Mancha (UCLM), Ciudad Real, Spain, in 2009 and 2018, respectively.

He is currently an Assistant Professor with the Department of Information and Systems Technology, UCLM. His current research interests include hardware verification methodologies, embedded systems, high-level synthesis, run-time reconfigurable systems, and heterogeneous distributed systems.

Dr. Caba won the Ph.D. category in the Xilinx Open Hardware Contest in 2017.

María Díaz was born in Spain, in 1990. She received the Industrial Engineering degree from the University of Las Palmas de Gran Canaria (ULPGC), Las Palmas de Gran Canaria, Spain, in 2014, the master's degree in system and control engineering jointly from the Universidad Complutense de Madrid (UCM) and the Universidad Nacional de Educacin a Distancia (UNED), Madrid, Spain, in 2017 and the Ph.D. degree in telecommunication technologies and computer engineering from the ULPGC, Las Palmas de Gran Canaria, Spain, in 2021.

During Ph.D., she developed her research activities with the Integrated Systems Design Division, Institute for Applied Microelectronics. Additionally, she conducted a research stay with the GIPSA-lab, University of Grenoble Alpes, Grenoble, France, and with the University of Castilla-La Mancha, Ciudad Real, Spain. Her research interests include image and video processing, development of highly parallelized algorithms for hyperspectral images processing, and hardware implementation.

Jesús Barba received the M.S. and Ph.D. degrees in computer engineering from the University of Castilla-La Mancha (UCLM), Ciudad Real, Spain, in 2001 and 2008, respectively.

He has been working as an Associate Professor with the Department of Information and Systems Technology, since 2001 and a member of the ARCO Research Group, School of Computer Science, UCLM. His research interests include low-cost and low-power reconfigurable systems for ubiquitous computing, reconfigurable computing platforms for artificial intelligence algorithms, heterogeneous distributed embedded computing, and high-level synthesis tools.

Raúl Guerra was born in Las Palmas de Gran Canaria, Spain, in 1988. He received the Industrial Engineering degree from the University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, in 2012, the master's degree in telecommunications technologies from the Institute for Applied Microelectronics (IUMA), University of Las Palmas de Gran Canaria, in 2013, and the Ph.D. degree in telecommunications technologies from the University of Las Palmas de Gran Canaria, in 2017.

He was funded by the IUMA to do his Ph.D. research with the Integrated System Design Division. In 2016, he was a Researcher with the Configurable Computing Lab, Virginia Tech University. His current research interests include the parallelization of algorithms for multispectral and hyperspectral images processing and hardware implementation.

Soledad Escolar received the Ph.D. degree in computer science from the University Carlos III of Madrid, Madrid, Spain, in 2010.

She is currently an Associate Professor with the Department of Information Technologies and Systems, University of Castilla-La Mancha, Ciudad Real, Spain. Her current research interests include wireless sensor networks, highlighting energy management algorithms, network protocols, adaptive behavior, Internet of Things, and smart cities.

Sebastián López (Senior Member, IEEE) was born in Las Palmas de Gran Canaria, Spain, in 1978. He received the Electronic Engineering degree from the University of La Laguna, San Cristobal de La Laguna, Spain, in 2001, and the Ph.D. degree in electronic engineering from the University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, in 2006.

He is currently an Associate Professor with the University of Las Palmas de Gran Canaria, where he is involved in research activities with the Integrated Systems Design Division, Institute for Applied Microelectronics. He has coauthored more than 120 articles in international journals and conferences. His research interests include real-time hyperspectral imaging, reconfigurable architectures, high-performance computing systems, and image and video processing and compression.

Dr. López was the recipient of regional and national awards during his electronic engineering degree. He also serves as an Active Reviewer for different JCR journals and as a Program Committee Member of a variety of reputed international conferences. Furthermore, he acted as one of the program chairs of the IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS) in its 2014 edition and of the SPIE Conference of High Performance Computing in Remote Sensing, from 2015 to 2018. He is also an Associate Editor for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, *MDPI Remote Sensing*, and *Mathematical Problems in Engineering Journal*. He was an Associate Editor for the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, from 2008 to 2013. Moreover, he has been the Guest Editor of different special issues in JCR journals related with his research interests.