A Reconfigurable Multi-mode Implementation of Hyperspectral Target Detection Algorithms

Đorđije Bošković[a], Milica Orlandić[a,*], Tor Arne Johansen[b]

[a] *Department of Electronic Systems*
Norwegian University of Science and Technology - NTNU
Trondheim, Norway
[b] *Centre for Autonomous Marine Operations and Systems*
(NTNU-AMOS), Department of Engineering Cybernetics
Trondheim, Norway

Abstract

Hyperspectral images obtained by imaging spectrometer contain a large data amount that requires techniques such as target detection for information extraction. The proposed multi-mode FPGA implementation combines matrix correlation and inversion matrix computations by using the Sherman-Morrison method to achieve real-time operation. The implementation supports Constrained Energy Minimization (CEM), Adjusted Spectral Matched Filter (ASMF) and modified Adaptive Cosine Estimator (ACE) detectors. The detection performance of the algorithms is evaluated using standard detection metrics. The proposed implementation has been realized on Zynq family SoCs and verified against the MATLAB reference software. The detection results for different fixed-point data types and target detection algorithms are reported. Finally, the proposed implementation is compared with state-of-the-art designs in terms of both throughput and resource utilization.

Keywords: hyperspectral imaging, target detection, Adaptive Cosine Estimator (ACE), Constrained Energy Minimization (CEM), Adjusted Spectral Matched Filter (ASMF), FPGA, real-time processing

*Corresponding author
Email address: milica.orlandic@ntnu.no (Milica Orlandić)

1. Introduction

The hyperspectral imaging (HSI) combines remote sensing and spectrometry by capturing hundreds of contiguous bands in a certain wavelength range of the electromagnetic spectrum. In contrast to true color imaging which adjusted to the human eye's spectral sensitivity, hyperspectral imaging includes an abundance of wavelengths referred to as bands. Recent drone and satellite missions tend to incorporate HSI sensors with increased spectral, spatial and temporal resolution causing dramatic growth of hyperspectral data dimensionality and bringing in new challenges in the data analysis and information extraction. In addition, the available drone and satellite down-link bandwidths do not follow this trend [1]. This leads to the incorporation of high-performance computing platforms such as graphic processing units (GPUs) and field programmable gate arrays (FPGAs) [2] into onboard hyperspectral processing stages. Compared to GPUs, FPGAs offer smaller size and weight, as well as substantially lower power consumption. The flexibility of FPGA through reconfigurability offers on-the-fly upgrades which can extend the life span of remote sensing satellites. Over the years, FPGAs have become one of the preferred choices for fast processing of hyperspectral data, due to their reconfigurability, parallelization properties and low power consumption.

The hyperspectral data processing chain consists usually of a number of stages such as binning, optics and sensor corrections, radiometric corrections, geo-referencing and registration, motion blur correction, super-resolution, atmospheric correction, dimensionality reduction, after which data analytic stages, such as target detection or classification, are used to extract useful information. The dimensionality reduction precedes usually the target detection stage and its effects on various target detection algorithms are surveyed in [3]. Finally, the processed data are compressed for the downlink operation. The objective of target detection is to find signatures of interest in the hyperspectral image and algorithms are usually affiliated with spectral processing techniques where each pixel corresponds to a ground-resolution cell containing a spectrum used

to determine specific materials [4]. Their performance in the context of various applications can be limited by factors such as:

- Spatial extent of a target in the hyperspectral image
- Number of spectral bands used for target detection
- Appropriate usage of radiance or reflectance domain in target detection
- Estimation and modeling of the image background for optimal detection
- Selection of threshold for automated target detection
- Spectral variability of the target due to spectral mixing in a sub-pixel target

A push-broom HSI camera captures a frame with one spatial and one spectral dimension by registering intensity at each wavelength channel within one line in the targeted scene. Lines with new spatial information are captured as the camera moves. The resulting image is represented as a three-dimensional data cube (having two spatial and one spectral dimension) as shown in Fig. 1, where a pixel is a set of samples extracted from a certain spatial location and can be plotted by the spectral reflectance curve as a function of wavelength. An intensity image in one spectral channel after applying corrections for the illumination spectrum represents the spatial distribution of reflectance. There are three common scanning methods of hyperspectral image cubes: band interleaved by pixel (BIP), band interleaved by line (BIL) and band sequential (BSQ). In BIP format, all spectral components of a pixel are scanned in subsequent locations, followed by another pixel in a frame. In BIL format, data are stored band by band for each frame. Finally, the image in BSQ format is created by storing the data cube band by band. Each of these scanning orders provides different computational complexity for the processing of hyperspectral data [5, 6], in particular when the data streaming order differs from the processing order. The proposed work, built upon the hardware/software codesign presented in [7], introduces the design and integration of a fully FPGA-based solution through

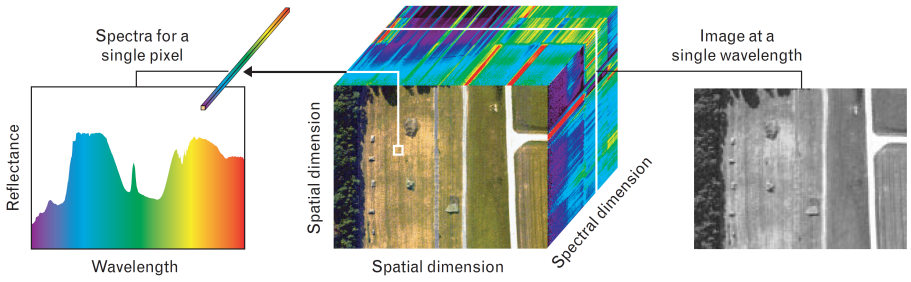


Figure 1: Illustration of a hyperspectral data cube (in the center), along with spectra plot (on the left) for a single pixel and a single spectral channel (band) intensities (on the right) [4].

the implementation of the correlation matrix and inverse matrix computations. Accuracy and performance analysis of the fixed-point implementation, performance and resource utilization analysis and comparison to the state of the art implementations are also presented.

The remainder of the paper is organized as follows. Section 2 describes target detection algorithms and state-of-the-art target detection algorithm implementations on FPGA platforms.Section 3 describes the proposed hardware implementation of a number of target detection algorithms. Section 4 discusses the results of FPGA implementations with respect to hardware performance, resource utilization and detection performance. Finally, section 5 concludes with guidelines for future development.

2. Background

The observed spectra in remote sensing applications exhibit inherent variability due to illumination, atmospheric conditions, sensor noise, ground-cell position, material mixing and spatial resolution. In addition, spectral mixing occurs when spatial HSI resolution is lower than the spatial variability of ground-cell materials. Such spectrum is regarded as a mixed pixel (sub-pixel), whereas a pure pixel contains only one material, the endmember. The resulting reflectance is, therefore, a combination of reflectances corresponding to the

area each material covers in the ground cell. The geometrical representation used for describing spectral mixing, similarity and change transformation due to illumination and environmental conditions is not an adequate model of the stochastic nature of hyperspectral data. Instead, for describing hyperspectral images a statistical approach using mean vector $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$ and correlation matrix $\mathbf{R}$ is employed.

2.1. Target detection algorithms

Many target detection algorithms are based on the estimation of image background to distinguish between target and non-target pixels. These algorithms consist of a mapping stage where the input pixel vector $\mathbf{x}$ is mapped onto a scalar detection statistic value $y = D(\mathbf{x})$ and threshold selection stage which constructs a background statistic model with a threshold value η. The value $y = D(\mathbf{x})$ corresponds to the probability of the input pixel to be a designated target and, when compared with the threshold value η, it determines if a pixel under test contains a designated target. It is the task of the threshold selection system to set the optimum threshold so that present targets are detected and the false alarm rate is kept below a certain value.

The mathematical framework for development and analysis of target detection originates from binary hypothesis testing in statistical signal processing. Hypothesis H_0, also called the null hypothesis, asserts that the pixel being tested is not a target. An alternative hypothesis H_1 asserts that the pixel under test contains the target. Each target detection algorithm is based on the different models for both hypotheses.

2.1.1. Spectral Angle Mapper (SAM)

The detection problem is modelled as follows:

$$\begin{aligned} H_0 &: \mathbf{x} = \mathbf{b}, \\ H_1 &: \mathbf{x} = \alpha\mathbf{s} + \mathbf{b}, \end{aligned} \tag{1}$$

where $\mathbf{b}$ is a combination of background and inherent random noise, $\mathbf{s}$ is the known spectral signature and α is a scaling factor of the intensity of the signal $\mathbf{s}$.

For random, zero-mean and normally distributed background $\mathbf{b}$ with variance σ, the SAM detector is defined as:

$$D_{SAM}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{x})^2}{(\mathbf{s}^T\mathbf{s})(\mathbf{x}^T\mathbf{x})}. \tag{2}$$

2.1.2. Constrained Energy Minimization (CEM)

This detector uses a finite impulse response (FIR) filter given as:

$$D_{CEM}(\mathbf{x}) = \mathbf{h}^T\mathbf{x}, \tag{3}$$

where $\mathbf{h}$ is a weight vector used to minimize the output power of a filter [8]. The detection statistic for CEM detector is given as:

$$D_{CEM}(\mathbf{x}) = \frac{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x}}{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{s}}, \tag{4}$$

where $\mathbf{R}$ is the sample correlation matrix computed as $\mathbf{R} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T$. The CEM algorithm is a small target detector, where low-probability distribution of target pixels is assumed. The estimation of the sample correlation matrix can be contaminated when the number of target pixels surpasses a certain limit.

2.1.3. Adaptive coherence/cosine estimator (ACE)

The formulation of the ACE algorithm is based on the following binary hypotheses:

$$\begin{aligned} H_0 &: \mathbf{x} = \beta\mathbf{b}, \\ H_1 &: \mathbf{x} = \alpha\mathbf{s} + \beta\mathbf{b}, \end{aligned} \tag{5}$$

where the noise scaling factor β is introduced. For zero-mean, white random vector process $\mathbf{b} \sim N(\mu, \Sigma)$, the ACE algorithm is derived as:

$$D_{ACE}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{\Sigma}^{-1}\mathbf{x})^2}{(\mathbf{s}^T\mathbf{\Sigma}^{-1}\mathbf{s})(\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x})}. \tag{6}$$

The sample covariance matrix $\mathbf{\Sigma}$ is computed as $\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i^T - \boldsymbol{\mu})$, where $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i)$ is the global sample mean vector. The ACE estimator does not meet real-time requirements since the covariance matrix is estimated using de-meaned hyperspectral data. This requires acquisition of the entire

hyperspectral cube prior to processing. A modified ACE detector, ACE-R, is adjusted to use the inverse of the correlation matrix $\mathbf{R}^{-1}$ as follows:

$$D_{ACE-R}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x})^2}{(\mathbf{s}^T\mathbf{R}^{-1}\mathbf{s})(\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x})}. \tag{7}$$

2.1.4. Adjusted Spectral Matched Filter (ASMF)

This detector [9] is constructed by the use of the simplified Reed-Xiaoli (RX) anomaly detector [10] in order to adjust the output of CEM detector and by introducing a non-target pixel suppression factor A defined as:

$$A = \Big|\frac{\mathbf{x}^T\mathbf{R}^{-1}\mathbf{s}}{\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x}}\Big|. \tag{8}$$

The factor A with the variable power n is then combined with CEM detector as follows:

$$D_{ASMF}(\mathbf{x}) = \frac{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x}}{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{s}} \cdot \Big|\frac{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x}}{\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x}}\Big|^n, \tag{9}$$

where variable power n amplifies or suppresses different spectral features and its selection affecting detection performance depends on the scenes used for testing and the designated targets and their properties.

2.2. State-of-the-art FPGA implementations of target detection algorithms

In recent period, there have been proposed a few FPGA implementations of hyperspectral target detection algorithms [11, 12, 13] to achieve real-time operation. The real-time performance is usually limited by background estimation techniques in global target detectors requiring the collection of the entire data set. The local CEM target detector in [11] uses streaming background statistics (SBS) on a sliding window of pixels with fixed size and requires the inverse correlation matrix to compute detection statistics. Rather than inverting the correlation matrix for each subset of pixels, the Sherman-Morrison formula [14] is used to gradually update the inverse of the correlation matrix as the window slides through the image. The problems introduced by this approach are analyzed in [12]. Namely, the inverse of the correlation matrix update phase is a computationally intensive task performed twice for the incoming and outgoing

pixel of the window. In addition, the updating step creates data dependency since a new update cannot proceed until the previous update is finished. This causes stalls in the processing pipeline and degrades considerably the overall computational performance. In [12], a new optimization of SBS method, non-sliding window, has been proposed. The non-sliding window introduces independent inverse matrix computations by adding incoming pixels into the window and keeping outgoing pixels in background statistic estimation. This leads to the substantial reduction in the number of computations required to update the correlation matrix and a speed-up of the system. However, new issues such as increased resource utilization and lowered data accuracy due to dynamic range of inverted correlation matrix are created. In [15], QR-decomposition is implemented for the matrix inversion problem by using Coordinate Rotation Digital Computer (CORDIC) [16]. The implementation of an automatic target-generation process using an orthogonal projection operator (ATGP-OSP) [13] includes the Gauss-Jordan elimination method for matrix inversion for non real-time operations. The ATGP implementation using the Gram-Schmidt method for orthogonal projection is proposed in [17]. In [5], various detection algorithms are analyzed for their real-time implementation depending on the availability of data and data scanning formats. The use of the sample correlation matrix which does not require data mean removal and allows intermediate data analysis instead of the sample covariance matrix is also discussed. In addition, it is stated that the statistics with an appropriately chosen percentage of pixels does not affect the detection performance. An FPGA implementation of Collaborative-Representation-based Detector (CRD) in [18] provides matrix inversion operation in real-time by using the Sherman-Morrison formula. The detection performance results of the algorithm implementation are not compared due to different hyperspectral datasets used for evaluation. Nevertheless, the algorithm shows promising results.

2.3. Matrix inversion

The inverse matrix computation is a computationally intensive task usually performed directly or iteratively. The choice of the method affects the performance and precision of the solution. Direct methods for inverting matrices, such as Gauss-Jordan elimination, LU decomposition, and Cholesky decomposition requiring an a-priori created matrix ready for inversion, are less suited to operate with real-time constraints. In that case, the iterative methods are preferred, such as the Sherman-Morrison formula:

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 + \mathbf{v}^T\mathbf{A}\mathbf{u}}. \tag{10}$$

where $\mathbf{u}$ and $\mathbf{v}$ are two arbitrary column vectors and $\mathbf{A}$ is an invertible square matrix. To invert the matrix $\mathbf{A}$ using Sherman-Morrison formula, the following iteration scheme is used:

$$\mathbf{X}_i = \mathbf{X}_{i-1} - \frac{\mathbf{X}_{i-1}(\mathbf{a}_i - \mathbf{i}_i)\mathbf{i}_i^T\mathbf{X}_{i-1}}{1 + \mathbf{i}_i^T\mathbf{X}_{i-1}(\mathbf{a}_i - \mathbf{i}_i)} \tag{11}$$

where $\mathbf{X}_0 = \mathbf{I}$, $\mathbf{a}_i$ is the i-th column of matrix $\mathbf{A}$, and $\mathbf{i}_i$ is the i-th column of the identity matrix. The inverse $\mathbf{X}_n = \mathbf{A}^{-1}$ is obtained after n iterations, where n is the number of columns of matrix $\mathbf{A}$.

For a system with no real-time constraint, a full data set is used to compute the correlation matrix. For the correlation matrix computation of an image with N pixels and K spectral components, it is required to perform $N \cdot K^2$ multiplications. The inverse matrix is computed using a method with general complexity of $O(K^3)$ multiplications. Using Sherman-Morrison formula, $N \cdot (3K^2 + K)$ multiplications are needed to obtain the final inverse matrix. Moreover, the Sherman-Morrison formula enables the system to estimate certain spectral features on-the-fly as pixels are being acquired. For the Guass-Jordan elimination method under real-time constraints, K^2 and $O(K^3)$ multiplications are required for R matrix updating and inverse matrix computations, respectively, for each incoming pixel, in comparison to the Sherman-Morrison formula which requires only $3K^2 + K$ multiplications.

2.4. Adaptation of inverse matrix calculation for FPGA implementation

The FPGA-based ACE detection accelerator in [7] can be used for subsequent runs of the target detection algorithms in the scenes for which the correlation matrix $\mathbf{R}$ is extensively trained. Otherwise, the correlation matrix needs to be estimated by using the captured hyperspectral image. In order to eliminate the constant N, the matrix $\mathbf{S}$ can be defined as $\mathbf{S} = N \cdot \mathbf{R}$. The matrix inverse $\mathbf{S}_i^{-1}$ for the pixel $\mathbf{x}_i$ is updated by the use of $\mathbf{S}_{i-1}^{-1}$ as follows:

$$\mathbf{S}_i^{-1} = (\mathbf{S}_{i-1} + \mathbf{x}_i \mathbf{x}_i^T)^{-1} = \mathbf{S}_{i-1}^{-1} - \frac{\mathbf{S}_{i-1}^{-1} \mathbf{x}_i \mathbf{x}_i^T \mathbf{S}_{i-1}^{-1}}{1 + \mathbf{x}_i^T \mathbf{S}_{i-1}^{-1} \mathbf{x}_i}. \tag{12}$$

The initial inverse matrix $\mathbf{S}_0^{-1}$ is either set to:

- an identity matrix $\mathbf{I}$ multiplied with a scaling factor β [11] or
- a (pseudo) inverse of correlation matrix for a certain percentage of pixels.

2.4.1. Sliding-window adaptation

As described in [11], for a sliding window of P pixels surrounding a pixel under test $\mathbf{x}_i$, the matrix $\mathbf{S}_i$ is given as:

$$\mathbf{S}_i = \sum_{n=i-P/2}^{i+P/2} \mathbf{x}_n \mathbf{x}_n^T \tag{13}$$

and the update step is performed as follows:

$$\mathbf{S}_i = \mathbf{S}_{i-1} + (\mathbf{x}_{i+P/2} \mathbf{x}_{i+P/2}^T - \mathbf{x}_{i-P/2} \mathbf{x}_{i-P/2}^T). \tag{14}$$

The inverse of the matrix $\mathbf{S}_i^{-1}$ is then obtained by the Sherman-Morrison formula in two steps as follows:

$$\mathbf{T}^{-1} = \mathbf{S}_{i-1}^{-1} - \frac{\mathbf{S}_{i-1}^{-1} \mathbf{x}_{i+P/2} \mathbf{x}_{i+P/2}^T \mathbf{S}_{i-1}^{-1}}{\mathbf{x}_{i+P/2}^T \mathbf{S}_{i-1}^{-1} \mathbf{x}_{i+P/2} + 1} \tag{15}$$

$$\mathbf{S}_i^{-1} = \mathbf{T}^{-1} - \frac{\mathbf{T}^{-1} \mathbf{x}_{i-P/2} \mathbf{x}_{i-P/2}^T \mathbf{T}^{-1}}{\mathbf{x}_{i-P/2}^T \mathbf{T}^{-1} \mathbf{x}_{i-P/2} - 1}. \tag{16}$$

This adaptation is feasible for real-time operation at the cost of increased computational complexity due to the sliding-window.

2.4.2. Running Sherman-Morrison update

For non-sliding window, the matrix $\mathbf{S}_i$ is expressed as:

$$\mathbf{S}_i = \frac{1}{\beta} \cdot \mathbf{I} + \mathbf{x}_1\mathbf{x}_1^T + \mathbf{x}_2\mathbf{x}_2^T + \cdots + \mathbf{x}_i\mathbf{x}_i^T, \tag{17}$$

and the inverse matrix $\mathbf{S}_i^{-1}$ is computed as in Eq. 12. The adoption of this method reduces the execution time, but the dynamic range of values of the matrix $\mathbf{S}_i^{-1}$ is increased over the number of iterations causing the need for increased utilization of the FPGA resources.

2.5. Detection Performance Metrics

Detection performance of target detection algorithms is evaluated by using available hyperspectral scenes with ground truth and metrics for performance evaluation. The performance of a binary classifier is evaluated by a confusion matrix. Due to the binary decision, there are four elements defined in a confusion matrix: true positives tp, false positives fp, false negatives fn and true negatives tn. True elements of the matrix are correctly classified pixels: true positives are present targets regarded as detected, while true negatives are confirmed background pixels. Undetected targets are false negatives, and background pixels classified as targets are false positives.

Based on the confusion matrix elements, Matthews correlation coefficient (MCC) metric is defined as:

$$MCC = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}, \tag{18}$$

where the MCC score is in the range $(-1, 1)$ and a successful detection of targets without false positives or negatives is defined by $MCC = 1$.

The visibility metric [19] is a measure of the algorithm's ability to separate background pixels and target pixels which evaluates the robustness of a detection algorithm as:

$$Visibility = \frac{\mid T_t - T_b \mid}{T_{max} - T_{min}} \tag{19}$$

where T_t is the average detection statistic for target pixels, T_b is the average detection statistic for non-target pixels and factors T_{max} and T_{min} are the

maximum and minimum evaluated detection statistics in the scene for a given algorithm, respectively.

3. Implementation

The proposed FPGA implementation combines correlation matrix and inversion matrix computations by using the Sherman-Morrison formula to achieve real-time operation. The implementation supports ACE-R, CEM and ASMF detectors, which are feasible for real-time processing since the correlation matrix is used to estimate the background statistics. In the proposed implementation, these detectors are integrated with the Sherman-Morrison inverse updating method, as follows:

$$D_{CEM-RT}(\mathbf{x}_i) = \frac{\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{x}_i}{\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{s}}, \tag{20}$$

$$D_{ACE-RT}(\mathbf{x}_i) = \frac{(\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{x}_i)^2}{(\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{s})(\mathbf{x}_i^T \mathbf{S}_{i+k}^{-1} \mathbf{x}_i)}, \tag{21}$$

$$D_{ASMF-RT}(\mathbf{x}_i) = \frac{\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{x}_i}{\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{s}} \cdot \left| \frac{\mathbf{s}^T \mathbf{S}_{i+k}^{-1} \mathbf{x}_i}{\mathbf{x}_i^T \mathbf{S}_{i+k}^{-1} \mathbf{x}_i} \right|^n, \tag{22}$$

where the target detection results are delayed by k pixels, allowing the initial k updates of the inverse matrix $\mathbf{S}^{-1}$. The experimental results demonstrate that the delay k can be set equal to the number of bands in a pixel K.

3.1. Input logic

The initialization of the target detection core is performed by enabling the core and initializing the matrix $\mathbf{S}^{-1}$ and target spectrum $\mathbf{s}$. For hyperspectral datasets with a large number of spectral bands, the dedicated BRAM blocks are used. The matrix storage is organized such that the matrix row is written in parallel in one clock cycle instead of writing each element sequentially.

3.2. Processing logic

The proposed implementation consistently uses Eq. 12 to update the previous estimate of the inverse matrix $\mathbf{S}_{i-1}^{-1}$ (elements of the matrix are noted as $r_{i,j}$). The computation of the Sherman-Morrison inverse matrix update is divided into three stages.

3.2.1. Stage 1

The first stage of Sherman-Morrison inverse matrix update produces a vector $\mathbf{S}_{i-1}^{-1}\mathbf{x}_i$ by using an array of dot product units as shown in Fig. 2, where intermediate registers *IP*, *IR* and *IM* have sizes of the input sample width, correlation data width and product data width.

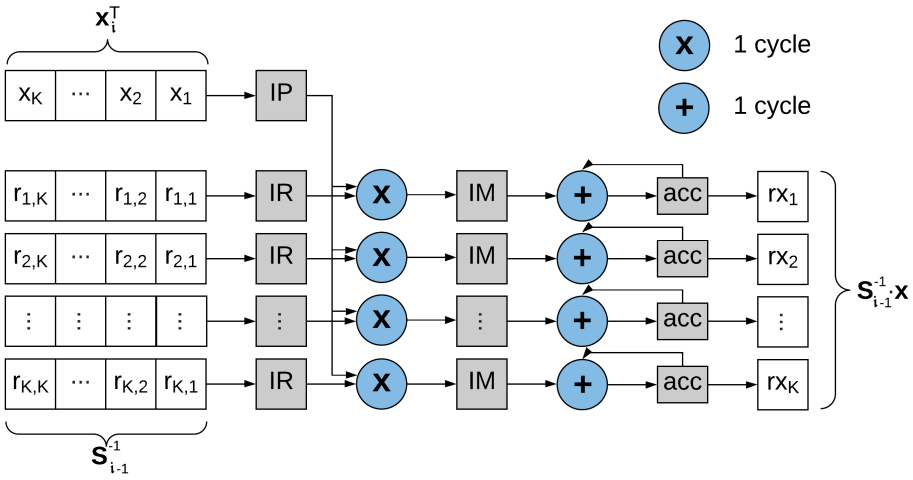


Figure 2: RTL design of the Stage 1

3.2.2. Stage 2

The block diagram of the second stage with data inputs $\mathbf{S}_{i-1}^{-1}\mathbf{x}_i$ and $\mathbf{x}_i$ is shown in Fig. 3. After the computation of $\mathbf{S}_{i-1}^{-1}\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_{i-1}^{-1}$ and $d = 1+\mathbf{x}_i^T\mathbf{S}_{i-1}^{-1}\mathbf{x}_i$, the inverse $p = \frac{1}{d}$ is obtained, where the division operation is implemented by using available Xilinx AXI divider IP. The total execution time is $K+D+delay$ clock cycles, where K is the number of spectral bands, *delay* is pipeline latency

and D is divider latency. As stage 2 requires the result of stage 1 to commence

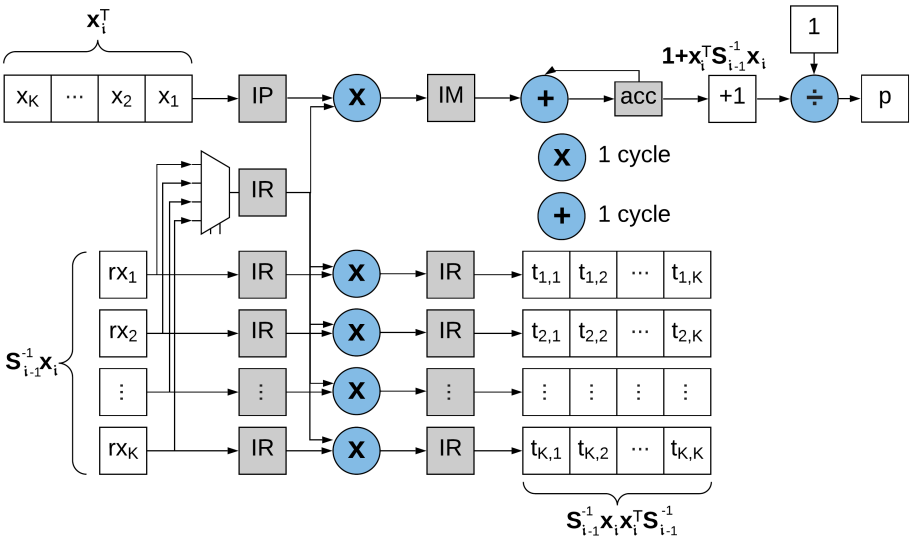


Figure 3: RTL design of the Stage 2

the operation, these stages are scheduled sequentially.

3.2.3. Stage 3

The final stage in the Sherman-Morrison updating procedure is shown in Fig. 4. The output of the divider p is multiplied with each column in the matrix $\mathbf{S}_{i-1}^{-1}\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_{i-1}^{-1}$ by the use of a multiplier array. Simultaneously, the corresponding column of the matrix $\mathbf{S}_{i-1}^{-1}$ is subtracted from the product column.

3.3. Resource sharing and target detection algorithms

Due to the data dependencies, idle states and decreased processing efficiency can be provoked such as in the case of multiplier array in stage 1 and dot product array in stages 2 and 3. These idle states are used for resource sharing between stages of the Sherman-Morrison inverse matrix update and the user-selected target detection algorithm. The sharing of multi-cycle hardware blocks, such as DSPs, results in significant resource savings at the cost of increasing complexity

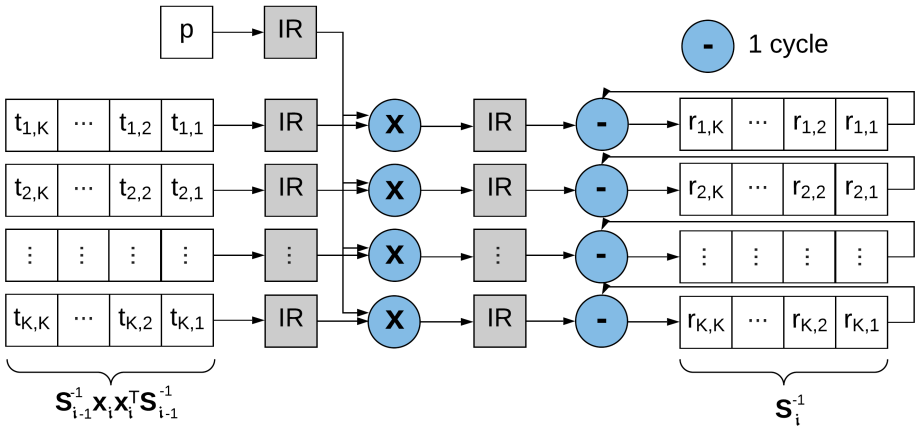


Figure 4: RTL design of the Stage 3

of operation schedule. Fig. 5 shows the timeline of operations, where memory and computation resources with identical tags are the same physical resources used at different time points. As such, one multiplier array is used in both stages 2 and 3 of Sherman-Morrison updating. The dot product array is used in stage 1 of Sherman-Morrison updating, and during stage 2 and 3 to obtain the products required for target detection, $\mathbf{S}_{i-1}^{-1}\mathbf{s}$ and $\mathbf{S}_{i-1}^{-1}\mathbf{x}_{i-k}$, respectively. The timing diagram in Fig. 6 shows the initialization phase and processing of three pixels by Sherman-Morrison inverse matrix updating. After system initialization, pixels are read simultaneously during stage 3 of the Sherman-Morrison update process. Additionally, the delayed pixel stream is input in target detection step 1. The detection statistics computation overlaps with Sherman-Morrison updating. This leads to a significant reduction of the number of clock cycles required for processing of each pixel, where the number of required clock cycles per pixel is $3K+D+3$.

3.4. Controller design

The controller monitoring the inputs and outputs of processing blocks enables resource sharing and its state machine is shown in Fig. 7. The initialization process consists of four control steps, *Idle, Initialize, WaitForStart* and

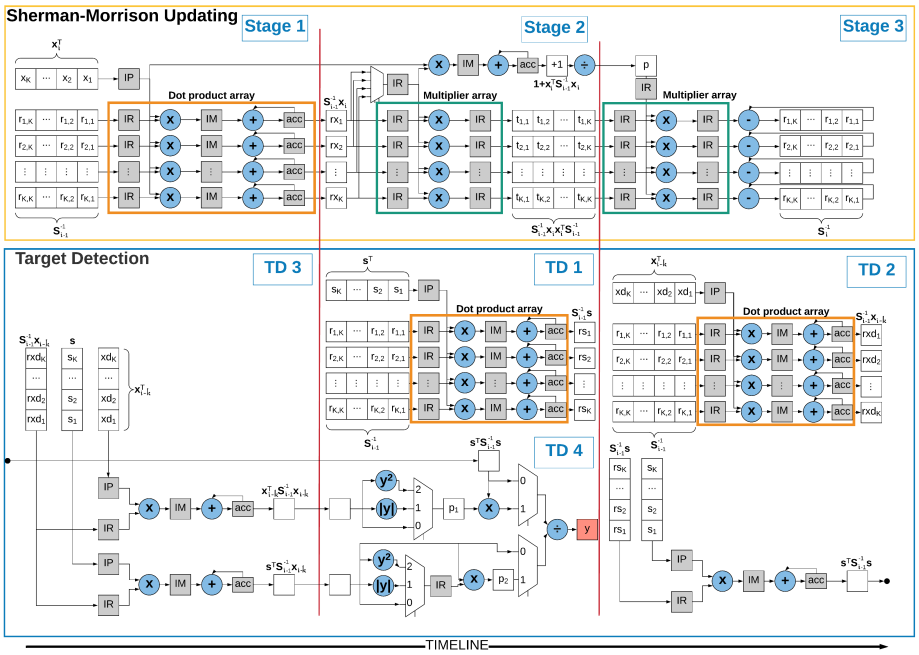


Figure 5: Timeline of operation stages in Sherman-Morrison updating and target detection.

WriteVector. The initialization begins in state *InitializeMatrix* with parallel transfer of K elements of the initial matrix $\mathbf{S}_0^{-1}$ to BRAM dedicated for storing matrix $\mathbf{S}^{-1}$ and ends when K^2 elements are stored in the initialization register, after which the upload of target pixel $\mathbf{s}$ follows. After the initialization phase, the controller iterates through states *Step1Fetch* to *Step3* for each incoming pixel, where each step in the state machine corresponds to the stages of Sherman-Morrison updating and target detection algorithms. As the last pixel component is received and processed, the controller returns to *Idle* state. The *Fetch* and *Wait* states are designed to accommodate latency of BRAM blocks and cores such as divider. For instance, in state *Step2Wait*, the controller waits D clock cycles until the divider produces a valid result.

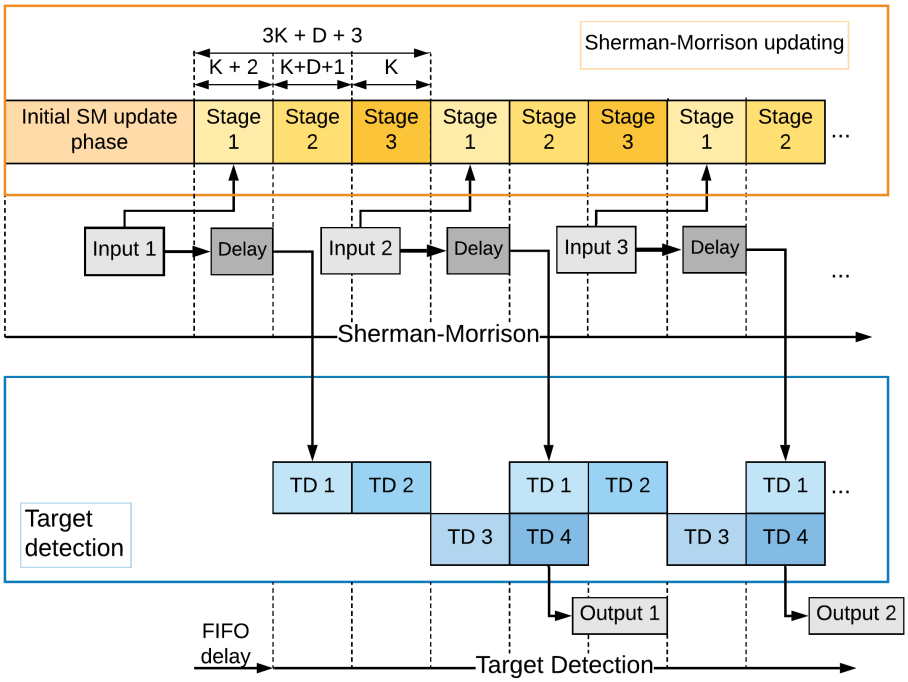


Figure 6: Timing diagram for Sherman-Morrison updating and target detection hardware.

4. Results

4.1. Detection performance analysis

Four hyperspectral datasets with known ground truth (Salinas, Pavia, Indian Pines [20] and HyMap Cooke City [21]) are used for detection performance analysis and functional verification of the proposed implementation. Endmembers for Salinas, Pavia and Indian Pines are extracted from the scene by averaging the ground truth pixels containing a certain material [3], whereas a standard spectral library with the targets is provided with HyMap Cooke City date set. The performance metrics are computed for each endmember of the corresponding scene by iterating through 10000 threshold values ranging from the obtained minimum to the maximum of the probability image. Final MCC values for each endmember are selected as the highest MCC values obtained

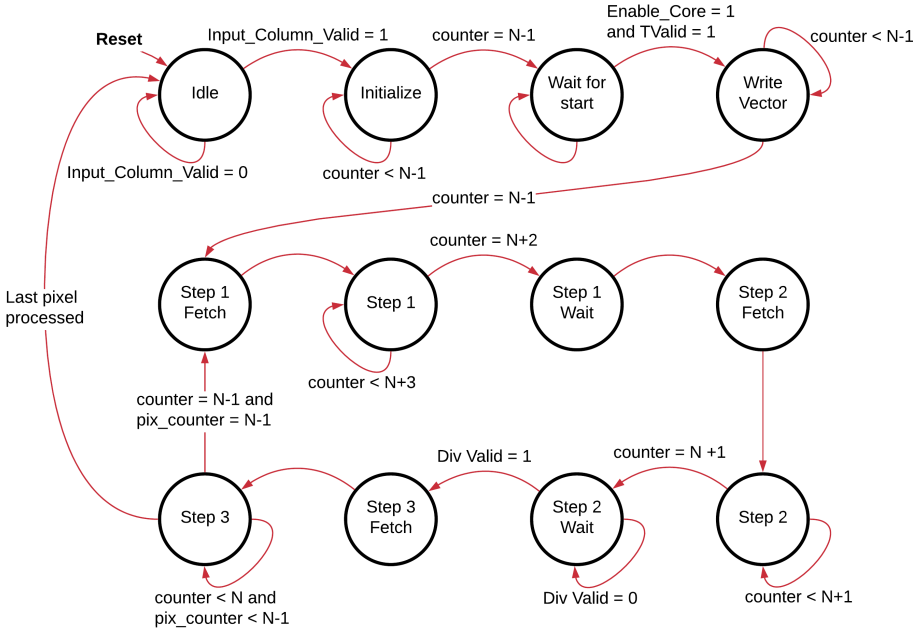


Figure 7: State machine for Sherman-Morrison updating and target detection hardware.

over threshold values. The plotted MCC and visibility values are the average of all known signatures in the scene.

The Salinas scene collected by AVIRIS is a near-homogeneous scene with a width of 512 pixels, a height of 217 pixels, 224 spectral bands and 16 endmembers of which many are similar crops planted at a different time as shown in Fig. 8(a). The performance of ACE, ACE-R, ASMF ($n = 1$ and $n = 2$), CEM and SAM detectors is shown in Fig. 9(a). The ACE, ACE-R, ASMF and CEM detectors achieve a high MCC score, whereas the visibility is drastically degraded for the CEM and SAM detectors. The adapted ACE-R achieves the highest visibility score, whereas ASMF with $n = 2$ reaches the highest average MCC score.

In Fig. 8(b), Indian Pines scene is a dataset dominated by agricultural areas and plants collected by the AVIRIS sensor. The ACE-R and ASMF detectors achieve the highest MCC and visibility score as shown in Fig. 9(b). The SAM detector with the lowest MCC and visibility scores is unable to distinguish between different endmembers and the background. The visibility of ASMF decreases as n increases reducing the robustness of the algorithm for threshold selection.

Pavia University scene acquired by ROSIS sensor has 610×340 pixels with 103 spectral bands. The scene is characterized by 9 distinct endmembers and with a non-classified area regarded as background. The detection performance results for this scene are shown in Fig. 9(c). None of the endmembers can be considered small targets, which reduces the performance of algorithms with background statistics estimation. Nevertheless, the ASMF and ACE-R achieve the highest MCC and visibility values.

The HyMap Cooke City [22], specifically designed for testing target detection algorithms, contains target ground truth locations, with a small number of target pixels compared to the image size. The data set includes atmospherically compensated hyperspectral images with 800×180 pixels and 126 spectral bands. There are four types of colored panels (Fig. 10) placed in the scene. The data set includes the exact positions in the form of region of interest (RoI) files, as

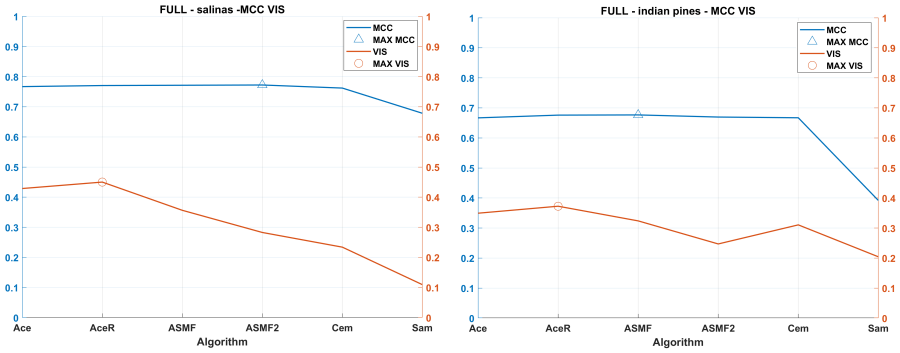
(a) Salinas scene (b) Indian Pines scene (c) Pavia scene

Figure 8: Ground truth map

well as standard spectral library files of the targets. The RoI of size 90×90 is shown in Fig. 11(a) and the spectral signatures of the targets are plotted in Fig. 11(b). Panel pixels in RoI are regarded as target pixels. The ground truth is formed based on full-pixel and sub-pixel candidates. In this sense, panels F1 and F2 can form near-full pixel targets due to their size and the ground resolution, whereas F3 and F4 panels are sub-pixel targets. The performance estimation results for Cooke City scene are shown in Fig. 9(d). The ACE, ACE-R and ASMF detectors achieve high MCC and visibility scores, whereas CEM and SAM detectors are not able to distinguish many spectral features in the scene.

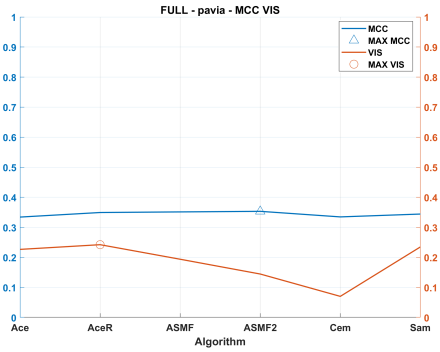
4.2. Overall system with target detection module

Based on the previous analysis, a multi-mode FPGA implementation of ACE-R, ASMF and CEM target detectors is proposed, where each of these target detection algorithms can be selected by the user. The detectors are highly integrated with Sherman-Morrison updating to achieve high performance and to lower resource utilization. The proposed multi-mode target detection implementation is described by the VHDL language, and the Vivado tool is used for synthesis, implementation, testing and verification on a low-end development board Zedboard SoC with a Zynq-7020 FPGA. In addition, the core is synthesized for larger Zynq-7035 FPGA for the comparison with the state of the art in terms of resource utilization and performance. The proposed solution is tested on ZedBoard SoC with ARM Cortex-A9 processor and programmable logic as

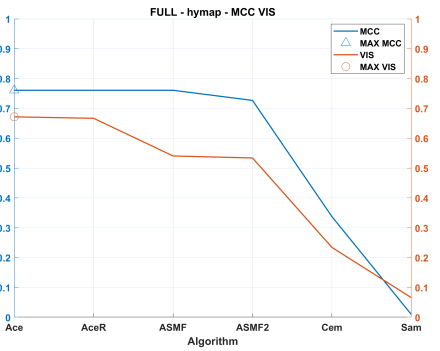


(a) Salinas scene

(b) Indian Pines scene

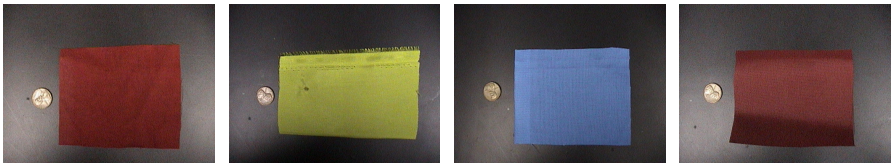


(c) Pavia scene



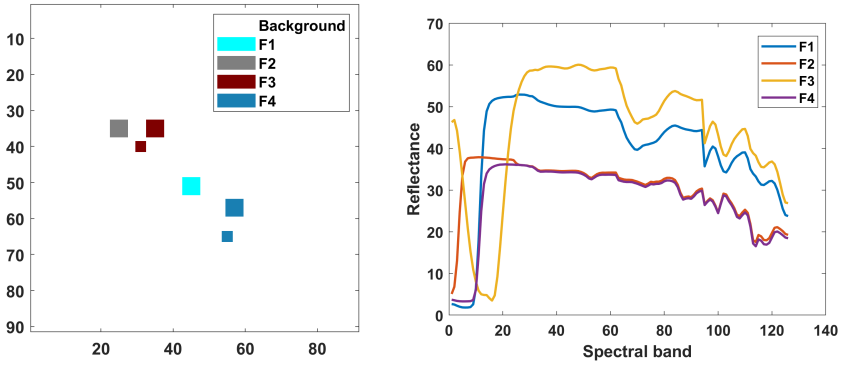
(d) Cooke City scene

Figure 9: *MCC* and visibility values



(a) F1 (b) F2 (c) F3 (d) F4

Figure 10: Targets in Cooke City scene - F1, F2, F3 and F4



(a) RoI (b) Target signatures

Figure 11: Region of interest and target signatures in HyMap Cooke City scene

shown in Fig. 12, where the configuration parameters are uploaded through a dedicated AXI-lite register file in the initialization module and the direct memory access (DMA) module is used to stream the hyperspectral data to/from the FPGA with minimal CPU intervention in BIP scanning order. The communication with memory is tested by the use of AXI DMA [23] and CubeDMA [24] cores, where the operating frequency of the proposed system is constrained to 100 MHz due to limitations of DMA cores. The single stream coming from memory via the DMA core is replicated into multiple outbound interfaces. The first interface is connected directly to DMA, whereas the other receives pixels delayed by FIFO block. For the delay $k = K$, the FIFO requires depth of at least K^2 elements.

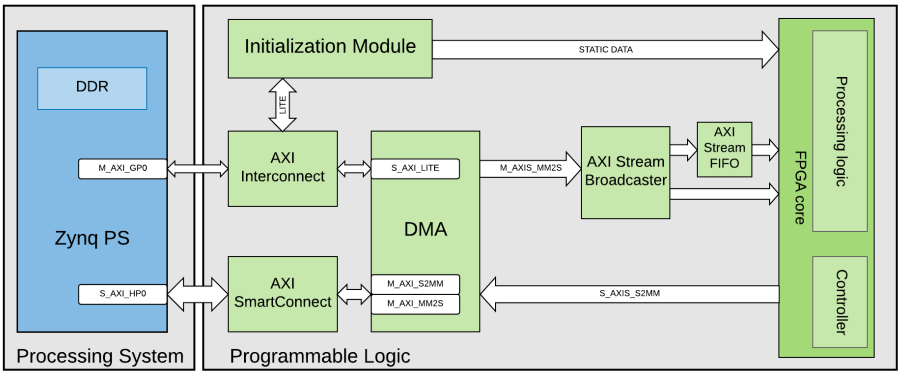


Figure 12: Block design of the FPGA solution for target detection.

4.3. Fixed-point considerations

The optimal fixed-point data types are obtained for the generic inputs to the VHDL design, enabling the implemented module to adapt to various hyperspectral datasets with different degrees of precision. The fraction lengths based on simulation results are proposed for the specified desired word length. The simulation uses all the pixels in the scene obtaining maximum and minimum simulation values for input, output and local variables of the function. The proposed fixed-point data types for HyMap scene are presented in form of (*total bits*, *integer part*, *fractional part*) in Table 1. The 16-bit elements of input vector $\mathbf{x}$ have 1-bit integer part and 15-bit fractional part. The 30, 35, 38, 42 and 52-bit word lengths for intermediate results are chosen according to the number of DSP blocks. The integer part does not change, whereas the fractional part is extended for longer words. In this manner, the fractional part of $\mathbf{S}^{-1}\mathbf{x}\mathbf{x}^T\mathbf{S}^{-1}$ can be extended by $4-7$ bits without inferring additional DSP blocks. To prevent dramatic degradation of target detection performance which can occur for smaller fractional parts, the scaling factor β for computing initial inverse matrix S_0^{-1} is set to 1000.

Table 1: Fixed-point data types for Sherman-Morrison implementation

Intermediate data	30-bit word length	35-bit word length	38-bit word length	42-bit word length	52-bit word length
DSP blocks	4	**4**	5	**5**	**9**
$\mathbf{x}$	(16,1,15)	(16,1,15)	(16,1,15)	(16,1,15)	(16,1,15)
$\mathbf{S}^{-1}$	(30,11,19)	(35,11,24)	(38,11,27)	(42,11,31)	(52,11,41)
$\mathbf{S}^{-1}\mathbf{x}$	(30,11,19)	(35,11,24)	(38,11,27)	(42,11,31)	(52,13,41)
$\mathbf{S}^{-1}\mathbf{x}\mathbf{x}^T\mathbf{S}^{-1}$	(30,22,8)	(35,22,13)	(38,22,16)	(42,22,20)	(52,22,30)
d=$1+\mathbf{x}^T\mathbf{S}^{-1}\mathbf{x}$	(30,18,12)	(35,18,17)	(38,18,20)	(42,18,24)	(52,18,34)
p=$1/d$	(30,5,25)	(35,5,30)	(38,5,33)	(42,5,37)	(52,5,47)
$(\mathbf{S}^{-1}\mathbf{x}\mathbf{x}^T\mathbf{S}^{-1})\cdot p$	(30,11,19)	(35,11,24)	(38,11,27)	(42,11,31)	(52,11,41)

4.4. Resource utilization

The intermediate data precision and number of spectral bands impact resource utilization, in particular DSP blocks. Thus, post-synthesis resource utilization for 32 spectral bands after PCA-based dimensionality reduction with 32-bit intermediate data is presented in Table 2.

4.5. Detection performance analysis for different fixed-point data types

The detection performance of the proposed FPGA implementation with Sherman-Morrison inverse matrix updating is analyzed by using Salinas and HyMap scene. The analysis has been performed using MATLAB fixed-point tools, as well as the ZedBoard prototyping platform. Detection performance evaluation of adapted ACE-R is performed on Salinas scene and *Lettuce romaine 4th week* endmember as a designated target. Fig. 13 shows the progress of the target detection in real time on the scene with 111104 pixels reduced to 20 spectral bands after PCA-based dimensionality reduction. The target is detected with high detection accuracy for 40-bits fixed-point word length. The detection results for different fixed-point data types and target detection algorithms are reported in Table 3. The presented results show that the detection performance increases proportionally with word length. The significant

Table 2: Resource utilization report, 32 bands, 32-bit intermediate data, full FPGA solution

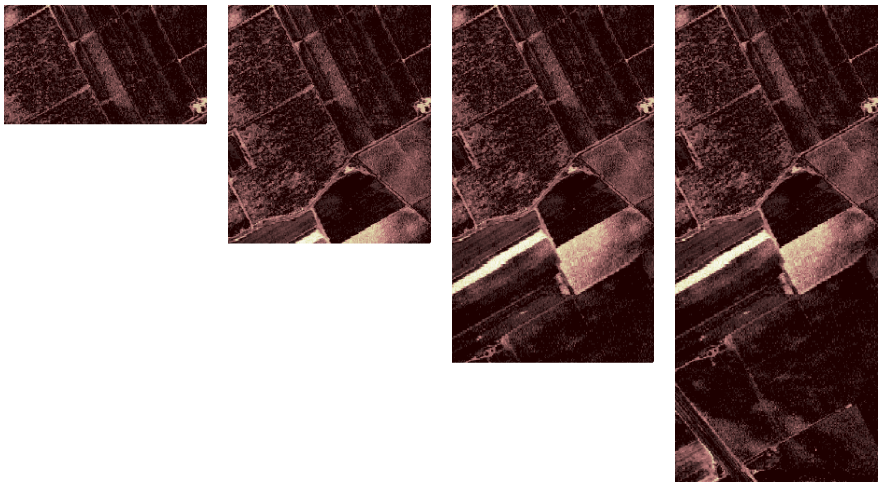
Module	Slice LUTs	Slice Reg	DSP	BRAM
PS-PL system	16018	20074	198	35
Input logic	2945	4080	0	2
AXI DMA	1669	2140	0	2
Init. module	1276	1940	0	0
Processing logic	9267	11048	198	32
Dot Product Array	2983	2950	64	0
DP controller	7	6	0	0
DP datapath	43	69	2	0
Multiplier Array	1567	580	128	0
Mult. controller	34	5	0	0
Mult. datapath	59	18	4	0
AXI Divider	2439	7133	0	0
Matrix storage	117	0	0	14.5/17.5
Controller	275	39	0	0
Output logic	79	297	0	0

degradation are reported for 32-bit intermediate data due to the high number of underflows occurring during Sherman-Morrison updating. The high dynamic range of data requires long words for processing and two solutions can be implemented to relieve this problem. First solution proposes to change fixed-point data type during execution allowing the growth of the fractional part, and thus, proving higher detection accuracy. Second solution is to use a subset of pixels for the Sherman-Morrison method, thus to additionally limit the dynamic range of intermediate data. In [5], it is claimed that if the subset is appropriately selected, the detection performance will remain unchanged. In its favor, experiments have shown that when using 10% randomly selected pixels from the image, the detection performance changes negligibly. The improvement when using two different fixed-point types is presented in Table 4 for ACE-R detector. Firstly, the previously described fixed-point types are used (annotated in table as FP1), with extended fractional part for $\mathbf{S}^{-1}\mathbf{x}\mathbf{x}^T\mathbf{S}^{-1}$. After K pixels are processed, the hardware switches to new fixed-point types for intermediate

data (annotated as FP2) which retain the word length, but with an increased fractional part.

Table 3: Comparison of detection performance scores for ACE-R, ASMF and CEM algorithms using different fixed-point types for Salinas scene

Algorithm	MCC score				Visibility			
	32	36	40	Global	32	36	40	Global
ACE-R	0.0154	0.6167	0.8284	0.8191	$4.14 \cdot 10^{-5}$	0.7114	0.7052	0.7078
ASMF	0.0200	0.6167	0.8284	0.8191	$5.72 \cdot 10^{-5}$	0.3614	0.3619	0.4569
CEM	0.2237	0.6312	0.6507	0.6811	$11.2 \cdot 10^{-5}$	0.0487	0.0356	0.0268



(a) 25% (b) 50% (c) 75% (d) 100%

Figure 13: Resulting image after real-time detection using Salinas scene and *Lettuce romaine 4th week* as the target signature

The proposed implementation is also tested by using target signatures *F1* and *F4* from HyMap Cooke City scene dataset. The scene is cropped to 90 by 90 pixel image area containing all implanted target pixels. The detection results for implemented ACE, ASMF, ASMF-2 and CEM detectors with 32- and 42-bit input data widths for Sherman-Morrison updating are compared with their global detector counterparts which compute inverse matrix R^{-1} on the complete cube and then perform target detection. The results are shown in Fig. 14 for

target signature *F4*. Table 5 and Table 6 present *MCC* and visibility scores for various data widths of Sherman-Morrison implementation intermediate data when detecting signatures *F1* and *F4*, respectively. The detectors maintain high detection performance also with the short word length on small image sizes. In the majority of cases, the resulting scores are very close to those obtained using global floating-point detectors. Similarly, as with the Salinas scene, changing the fixed-point types to increase the fractional part, proportionally boosts the detection performance. This effect on both *MCC* and visibility metrics is shown in Fig. 15.

Table 4: Comparison of detection performance scores using different combinations of fixed-point types in ACE-R algorithm for Salinas scene

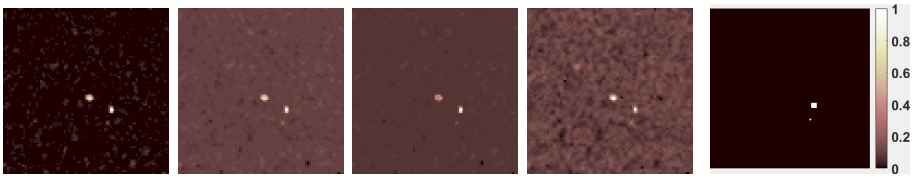
Algorithm	MCC score			Visibility		
	32	36	40	32	36	40
ACE-R FP1	0.0154	0.6167	0.8284	$4.14 \cdot 10^{-5}$	0.7114	0.7052
ACE-R FP2	**0.6063**	**0.8286**	**0.8328**	**0.7083**	0.7049	**0.7285**

Table 5: Comparison of detection performance scores using different fixed-point types for detecting target signature *F1* in HyMap Cooke City scene

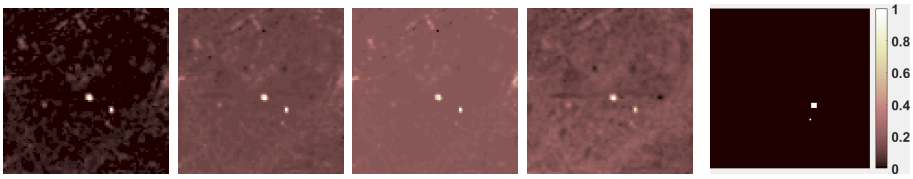
Algorithm	MCC score					Visibility				
	30	35	38	42	Global	30	35	38	42	Global
ACE-R	0.94	1	1	1	0.95	0.64	0.76	0.79	0.81	0.76
ASMF	0.94	1	1	1	0.95	0.61	0.68	0.71	0.71	0.67
ASMF-2	0.88	0.88	0.90	0.95	0.95	0.61	0.57	0.55	0.53	0.81
CEM	0.89	0.90	0.90	0.90	0.84	0.53	0.53	0.53	0.53	0.49

Table 6: Comparison of detection performance scores using different fixed-point types for detecting target signature *F4* in HyMap Cooke City scene

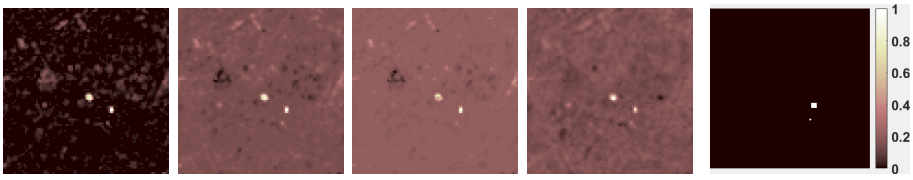
Algorithm	MCC score					Visibility				
	30	35	38	42	Global	30	35	38	42	Global
ACE-R	0.35	0.45	0.45	0.55	0.57	0.32	0.39	0.43	0.45	0.43
ASMF	0.35	0.45	0.45	0.55	0.58	0.30	0.34	0.38	0.39	0.39
ASMF-2	0.32	0.47	0.63	0.56	0.63	0.26	0.31	0.36	0.34	0.36
CEM	0.36	0.48	0.48	0.47	0.54	0.27	0.24	0.24	0.24	0.31



(a) Global ACE-R (b) Global ASMF (c) Global ASMF-2 (d) Global CEM (e) *F4* ground truth



(f) ACE-RT 32-bit (g) ASMF-RT 32-bit (h) ASMF-2-RT 32-bit (i) CEM-RT 32-bit (j) *F4* ground truth



(k) ACE-RT 42-bit (l) ASMF-RT 42-bit (m) ASMF-2-RT 42-bit (n) CEM-RT 42-bit (o) *F4* ground truth

Figure 14: Detection results (probability images) for *F4* target signature from HyMap Cooke City scene obtained using the proposed FPGA implementation with Sherman-Morrison updating.

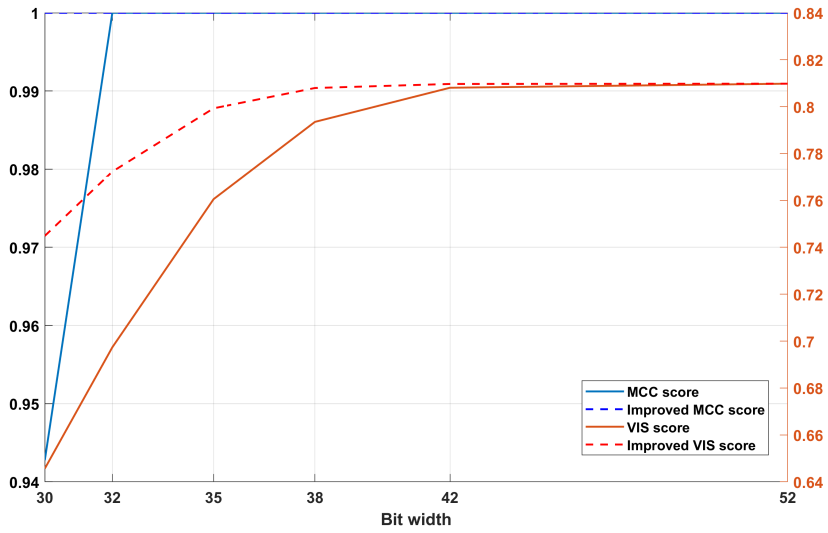


Figure 15: Comparison of detection performance metrics when increasing fractional part of fixed-point types for adapted ACE-R detector.

4.6. Real-time processing performance analysis and comparison with the state-of-the-art implementations

The performance of the proposed implementation is analyzed by using post-synthesis results. The maximum operating frequency on Zedboard SoC is 123MHz. The design requires $3K+D+3$ clock cycles to process one pixel with K spectral bands. Table 7 presents the comparison of incoming data rates from three hyperspectral sensors (HSI, HyMap and AVIRIS) and the processing data rate of the FPGA implementation for the provided sensor parameters (number of spectral bands, frame rate, frame size and data width). Since the achieved throughput is significantly higher than throughput required by each sensor, the proposed target detection implementation provides real-time processing capability for these sensors. The maximum processing data rate of the proposed system is theoretically limited to 66.67MB/s when clocked with 100MHz and 16-bit components.

The performance and resource utilization comparisons with the current state-of-the-art SBS-CEM [11] and DPBS-CEM [12] implementations are also pre-

Table 7: Sensors and FPGA processing data rate for the proposed FPGA solution

Sensor	Frame rate fps	Spectral bands	Frame size ppf	Data Width	Sensor througput	Achieved throughput
HSI	32	100	1216	16	7.78MB/s	49.63MB/s
HyMap [25]	16	126	512	16	1.97MB/s	49.96MB/s
AVIRIS [26]	100	224	512	16	22.93MB/s	57.80MB/s

sented. Table 8 shows the number of clock cycles, operating frequencies and processing times for SBS-CEM [11], DPBS-CEM [12], ACE-R codesign [7] and the proposed implementation for processing the full HyMap image data set of 224000 pixels. For a fair comparison, the proposed target detection core is synthesized for a larger Zynq-7035 SoC with Kintex-7 FPGA technology. The DPBS-CEM implementation requires the lowest number of clock cycles. However, as presented in Table 9, the high resource utilization in DPBS-CEM limits the choice of FPGA platforms which can accommodate the algorithm implementation.

Table 8: Comparison of data processing speed for the FPGA implementations

Implementation	Clock cycles	Freq. [MHz]	Proc. Time [s]	Technology
SBS-CEM [11]	229607996	200	1.148	Virtex-7
DPBS-CEM [12]	31360557	200	0.1568	Virtex-7
ACE-R Codesign [7]	-	666	3.29	ARM Cortex A9
Proposed	100774324	200	0.5	Zynq-7035 (Kintex-7)

Table 9: Resource utilization comparison

Implementation	Slice LUTs	Slice Registers	DSP	BRAM
SBS-CEM	21730	28245	265	120
DPBS-CEM	111073	217958	1396	379
Proposed (32-bit)	36233	28719	762	135
Proposed (35-bit)	45900	31035	762	136

5. Conclusion

This paper investigates target detection algorithms and feasibility of the real-time FPGA implementation of ACE-R, ASMF and CEM target detectors. To meet real-time constraint, the proposed FPGA solution uses the Sherman-Morrison method for target detection algorithms in hyperspectral imagery. The work builds upon the hardware/software co-design implementation of the ACE-R target detection algorithm. The simplicity of the proposed FPGA solution is maintained in order to facilitate its integration in the onboard satellite processing systems while satisfying all critical timing constraints. The flexibility introduced by the multi-mode design of target detection has negligible additional resource costs. However, there is still room for potential parallelization and performance improvements. The end goal is integration of the developed FPGA solution with other modules in the hyperspectral processing pipeline, for example, dimensionality reduction techniques.

Acknowledgement

This work was supported by the Research Council of Norway (RCN) through MASSIVE project, grant number 270959, and AMOS project, grant number 223254.

References

[1] R. Trautner, ESA's roadmap for next generation payload data processors, Proc. of DASIA Conference.

[2] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, J. Chanussot, Hyperspectral remote sensing data analysis and future challenges, IEEE Geoscience and Remote Sensing Magazine 1 (2) (2013) 6–36. doi:10.1109/mgrs.2013.2244672.

[3] S. Bakken, M. Orlandic, T. A. Johansen, The effect of dimensionality reduction on signature-based target detection for hyperspectral imaging, CubeSats and SmallSats for Remote Sensing III.

[4] D. Manolakis, D. Marden, G. A. Shaw, Hyperspectral image processing for automatic target detection applications, Lincoln laboratory journal 14 (1).

[5] Q. Du, R. Nekovei, Fast real-time onboard processing of hyperspectral imagery for detection and classification, Journal of Real-Time Image Processing 4 (3) (2008) 273–286. doi:10.1007/s11554-008-0106-9.

[6] J. Fjeldtvedt, M. Orlandic, T. A. Johansen, An Efficient Real-Time FPGA Implementation of the CCSDS-123 Compression Standard for Hyperspectral Images, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 11 (10) (2018) 3841–3852. doi:10.1109/jstars.2018.2869697.

[7] Đ. Bošković, M. Orlandić, S. Bakken, T. A. Johansen, HW/SW Implementation of Hyperspectral Target Detection Algorithm, in: 2019 8th Mediterranean Conference on Embedded Computing (MECO), IEEE, 2019, pp. 1–6.

[8] J. C. Harsanyi, Detection and classification of subpixel spectral signatures in hyperspectral image sequences, University of Maryland Baltimore County, 1993.

[9] L. Gao, B. Yang, Q. Du, B. Zhang, Adjusted spectral matched filter for target detection in hyperspectral imagery, Remote Sensing 7 (6) (2015) 6611–6634. doi:10.3390/rs70606611.

[10] C.-I. C. Chang, H. Ren, S.-S. Chiang, Real-time processing algorithms for target detection and classification in hyperspectral imagery, IEEE Transactions on Geoscience and Remote Sensing 39 (4) (2001) 760–768. doi:10.1109/36.917889.

[11] B. Yang, M. Yang, A. Plaza, L. Gao, B. Zhang, Dual-Mode FPGA Implementation of Target and Anomaly Detection Algorithms for Real-Time Hyperspectral Imaging, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 8 (6) (2015) 2950–2961. doi:10.1109/jstars.2015.2388797.

[12] J. Lei, Y. Li, D. Zhao, J. Xie, C.-I. Chang, L. Wu, X. Li, J. Zhang, W. Li, A Deep Pipelined Implementation of Hyperspectral Target Detection Algorithm on FPGA Using HLS, Remote Sensing 10 (4) (2018) 516. doi:10.3390/rs10040516.

[13] C. Gonzalez, S. Bernabe, D. Mozos, A. Plaza, FPGA Implementation of an Algorithm for Automatically Detecting Targets in Remotely Sensed Hyperspectral Images, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9 (9) (2016) 4334–4343. doi:10.1109/jstars.2015.2504427.

[14] J. Sherman, W. J. Morrison, Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix, The Annals of Mathematical Statistics 21 (1) (1950) 124–127. doi:10.1214/aoms/1177729893.

[15] J. Wang, C.-I. Chang, M. Cao, FPGA design for constrained energy minimization, Chemical and Biological Standoff Detectiondoi:10.1117/12.518559.

[16] J. E. Volder, The cordic trigonometric computing technique, IRE Transactions on electronic computers (3) (1959) 330–334.

[17] S. Bernabe, S. Lopez, A. Plaza, R. Sarmiento, P. G. Rodriguez, FPGA Design of an Automatic Target Generation Process for Hyperspectral Image Analysis, 2011 IEEE 17th International Conference on Parallel and Distributed Systemsdoi:10.1109/icpads.2011.64.

[18] J. Wu, Y. Jin, W. Li, L. Gao, B. Zhang, FPGA implementation of collaborative representation algorithm for real-time hyperspectral target de-

tection, Journal of Real-Time Image Processing 15 (3) (2018) 673–685. doi:10.1007/s11554-018-0823-7.

[19] X. Jin, S. Paswaters, H. Cline, A comparative study of target detection algorithms for hyperspectral imagery, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVdoi:10.1117/12.818790.

[20] Hyperspectral Remote Sensing Scenes, http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes (2018).

[21] Target Detection Blind Test, http://dirsapps.cis.rit.edu/blindtest/ (2009).

[22] T. Cocks, R. Jenssen, A. Stewart, I. Wilson, T. Shields, The HyMap airborne hyperspectral sensor: The system, calibration and performance, Proceedings of 1st EARSEL Workshop on Imaging Spectroscopy, Zurich (1998) 37–42.

[23] AXI DMA v7.1 LogiCORE IP, https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf (2018).

[24] J. Fjeldtvedt, M. Orlandic, CubeDMA – Optimizing three-dimensional DMA transfers for hyperspectral imaging applications, Microprocessors and Microsystems 65 (2019) 23–36. doi:10.1016/j.micpro.2018.12.009.

[25] F. Kruse, J. Boardman, A. Lefkoff, J. Young, K. Kierein-Young, T. Cocks, R. Jenssen, P. Cocks, HyMap: An Australian hyperspectral sensor solving global problems - Results from USA HyMap data acquisitions.

[26] Airborne visible/infrared imaging spectrometer, https://aviris.jpl.nasa.gov (2019).