# HW/SW Implementation of Hyperspectral Target Detection Algorithm

Đorđije Bošković, Milica Orlandić
Department of Electronic Systems
NTNU
Trondheim, Norway
Email: dordijeb@stud.ntnu.no, milica.orlandic@ntnu.no

Sivert Bakken, Tor Arne Johansen
Centre for Autonomous Marine Operations and Systems
(NTNU-AMOS), Department of Engineering Cybernetics
NTNU
Trondheim, Norway
Email: sivertba@stud.ntnu.no, tor.arne.johansen@ntnu.no

***Abstract*—Hyperspectral images obtained by imaging spectrometer contain a vast amount of data which require techniques such as target detection to extract useful information. This article presents an implementation of the target detection method Adaptive Cosine Estimator (ACE) for hyperspectral images. The algorithm is implemented as hardware-software partitioned system on Zynq-7000 development platform. The computationally intensive operations are accelerated on FPGA with the speed-up factor of $28.54$. The timing analysis presents results for the partitioned system as well as for the software implementation on Zynq processing system used for comparison. The detection performance of the implemented algorithm is tested and verified using publicly available hyperspectral scenes with ground truth data.**

***Index Terms*—hyperspectral imaging, target detection, Adaptive Cosine Estimator (ACE), FPGA**

## I. Introduction

Hyperspectral imaging combines digital imaging and spectrometry; its main goal is capturing and processing of images consisting of many spectral components. As each object has its own spectral signature defined by varying reflectance as a function of wavelength, it is essential to measure the distribution of electromagnetic radiation in certain spectral bands. Opposed to typical true color imaging which is adjusted to human spectral sensitivity, hyperspectral imaging can also include abundance of wavelengths outside the visible spectrum. In satellite remote sensing applications, the data obtained from the imaging spectrometer provides significant information about the spectral characteristics of surfaces and materials of the Earth [1].

The data from the hyperspectral camera can be represented as a spectral cube, consisting of two spatial and one spectral axis. Therefore, a pixel in a hyperspectral cube is an array of captured spectral intensities for certain spatial coordinates. Depending on the underlying architecture, the cube is usually stored and accessed in BSQ, BIP or BIL format [2]. It is also useful to plot spectra in the spectral space, where each of the spectral bands can be considered as one dimension in $N$-dimensional space. Therefore, the point (or vector) in spectral space actually represents one spectrum. Since all materials exhibit some variability in their reflectance spectra, it is expected that a particular material will be empirically characterized as a cloud of points in spectral space.

The increasing amount of spatio-spectral data obtained by modern hyperspectral imagers has contributed to creating new challenges in hyperspectral data processing, especially in scenarios which require real-time operation [3]. Considering satellite hyperspectral missions, another important issue is the constrained down-link bandwidth from a satellite to the ground stations [4]. To fulfill this, intensive on-board processing performed on powerful platforms can be used. The systems with FPGAs have become an integral part of satellite remote sensing missions [5], characterized by low power consumption, inherent reconfigurability, high parallelization capabilities and commercial availability. The on-board hyperspectral data processing tasks are usually pipelined, consisting mainly of the following stages: binning, optical and sensor corrections, radiometric corrections, geo-referencing and registration, motion blur correction, super-resolution, atmospheric correction and dimensionality reduction, etc. The reduced data cube can then be processed by a target detection system.

A number of target detection algorithms, such as constrained energy minimization (CEM) [6], adaptive cosine estimator (ACE) [7] and spectral angle mapper (SAM) [1] have been widely used for hyperspectral images. Target detection is involved in many civilian and military applications [8], such as detection of vehicles, infrastructure, vegetation, pollution or harmful algae species in oceans. As such, the objects of interest can be scarcely populated in the scene or constitute a significant portion of the image [1]. The spectral signature of the target is required for mentioned algorithms, and it can be obtained from a spectral library or extracted from the scene for testing purposes.

The remainder of the paper is organized as follows. Section II describes target detection algorithms as well as metrics used to estimate their detection performance. In section III, analysis of detection performance of considered algorithms for FPGA implementation is briefly explained. Section IV describes hardware-software codesigned implementation of modified Adaptive Cosine Estimator (ACE) on ZedBoard *Zynq-7000* System-on-Chip. Section V provides analysis of the proposed FPGA-based implementation using hyperspectral scenes with provided ground truth data. Finally, section VI concludes with guidelines for future development.

## II. Target Detection Algorithms

The objective of target detection algorithms is to find an object of interest in the hyperspectral image. The algorithms inspected in this paper are based on the statistical approach, where spectral reflectance features are exploited to identify the target. A typical target detection system consists of target detection algorithm and threshold selection system, as shown in Fig. 1.

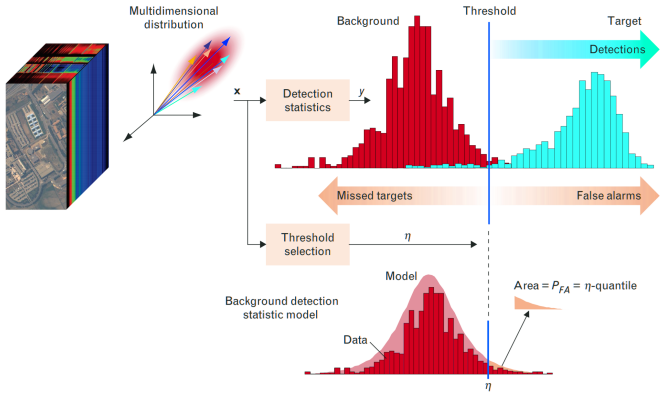



Fig. 1. Hyperspectral target detection system, adapted from [8].

The target detection algorithm maps input pixel vector $\mathbf{x}$ onto a scalar value $y = D(\mathbf{x})$, where $y$ is called detection statistic. In other words, the target detection algorithm provides the system with a numerical value which is related to the probability of the inspected pixel to be a designated target. Afterwards, the detection statistic $y$ is compared to a threshold value $\eta$ in order to determine if the input pixel contains target signature. The threshold provided by threshold selection system is based on the estimated background image data. The optimum threshold is set so that a significant amount of present targets are detected and the false alarm rate is kept below a certain value. Threshold selection system is out of scope of this paper.

In statistical signal processing, target detection is regarded as binary hypothesis testing between a null and alternative hypothesis. Null hypothesis $H_0$ asserts that the pixel being tested is not a target, whereas the alternative hypothesis $H_1$ asserts the observed pixel as target. Modelling the signals under both hypothesis is characteristic for each target detection algorithm. A number of target detection algorithms are selected for further analysis.

### A. Spectral Angle Mapper

The detection problem can be presented 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 background clutter and noise, $\mathbf{s}$ is the known target vector, and $\alpha$ represents a parameter influenced by illumination and sub-pixel mixing. In the case of the random, zero-mean and normally distributed background $\mathbf{b}$, the spectral angle mapper [1] is defined as:

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

An equivalent detection statistic is given as:

$$D_{SAM}(\mathbf{x}) = -\cos^{-1}\frac{\mathbf{s}^T\mathbf{x}}{\sqrt{(\mathbf{s}^T\mathbf{s})(\mathbf{x}^T\mathbf{x})}} \tag{3}$$

which represents an angle between a reference spectrum and a pixel under test [1].

### B. Constrained Energy Minimization

The target detection algorithm can also be designed as a FIR linear filter $\mathbf{h} = [h_1, h_2, ..., h_K]^T$ [6] with detection statistic given as:

$$D(\mathbf{x}) = \mathbf{h}^T\mathbf{x}. \tag{4}$$

The vector $\mathbf{h}$ is optimized so that the detection statistic better separates the background clutter and the target. The optimization is performed by minimizing the background energy under the following constraint:

$$min(\mathbf{h}^T\mathbf{R}\mathbf{h}) \quad subject \quad to \quad \mathbf{h}^T\mathbf{s} = 1 \tag{5}$$

where $\mathbf{R}$ is the sample correlation matrix. The resulting detection statistic of constrained energy minimization is then given as:

$$D_{CEM}(\mathbf{x}) = \frac{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x}}{\mathbf{s}^T\mathbf{R}^{-1}\mathbf{s}}. \tag{6}$$

### C. Adaptive Cosine Estimator

A model of the detection hypotheses given as:

$$\begin{aligned} H_0 &: \mathbf{x} = \beta\mathbf{b} \\ H_1 &: \mathbf{x} = \alpha\mathbf{s} + \beta\mathbf{b} \end{aligned} \tag{7}$$

leads to the Adaptive Cosine Estimator (ACE) [7], where the parameter $\beta$ is the newly introduced scaling factor of the combination of noise and background clutter $\mathbf{b} \sim N(\mu, \sigma)$. The detector is characterized by the following equation:

$$D_{ACE}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{\Sigma}^{-1}\mathbf{x})^2}{(\mathbf{s}^T\mathbf{\Sigma}^{-1}\mathbf{s})(\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x})} \tag{8}$$

where all factors are mean centered and the covariance matrix $\mathbf{\Sigma}$ can be estimated from the sampled image data.

To satisfy real-time performance requirements and to obviate the need for mean centering of the hyperspectral data, an adaptation of ACE algorithm is proposed. The adaptation consists of replacement of the covariance matrix with the correlation matrix as follows:

$$D_{ACE-R}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x})^2}{(\mathbf{s}^T\mathbf{R}^{-1}\mathbf{s})(\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x})}, \tag{9}$$

where the correlation matrix $\mathbf{R}$ is estimated from the given dataset or its subset.

### D. Target Detection Algorithm Performance Metrics

In order to choose the algorithm to implement and later verify the implementation, performance of the target detection algorithms is evaluated using the Matthews correlation coefficient (MCC) and visibility metric [9]. The performance of the algorithm is usually visualized using a confusion matrix that contains the number of true positives, true negatives, false positives and false negatives. True positives represent correctly detected targets, while false negatives are present targets which are not detected by the algorithm. On the other side, some pixels might be regarded as targets even if they are part of the background which belongs to false positives count (Fig. 1). In contrast with that, true negatives are correctly classified background pixels.

*1) MCC metric:* MCC metric is defined as:

$$MCC = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \quad (10)$$

where $tp$ are true positive, $tn$ are true negative, $fp$ are false positive and $fn$ are false negative counts. MCC score is a value in range from $-1$ to 1, where $MCC = 1$ means successful detection of all hyperspectral targets without false positives or negatives by target detection algorithm for a given threshold. On the other side, $MCC = -1$ indicates that the algorithm always gives the opposite class in case of binary classification. This metric involves all four quadrants of the confusion matrix.

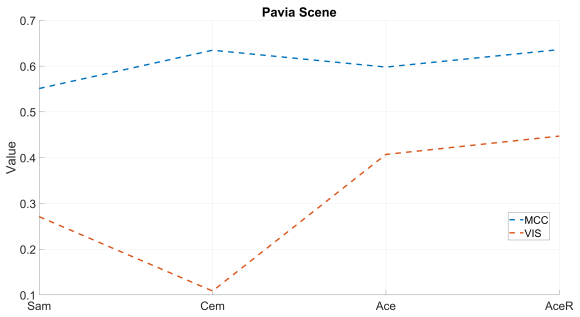
*2) Visibility metric:* The robustness of an algorithm is evaluated using the visibility metric, which is a measure of the algorithm's ability to separate background clutter and target. It is given as:

$$Visibility = \frac{\mid T_t - T_b \mid}{T_{max} - T_{min}} \quad (11)$$

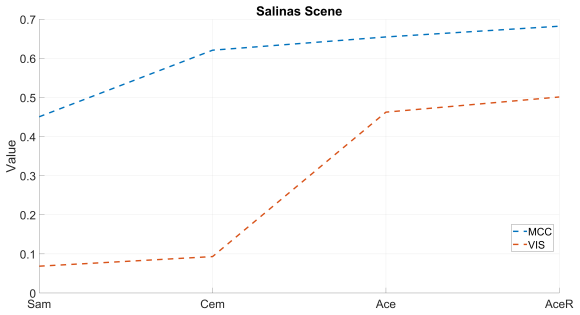
where $T_t$ is the average detection statistic for target pixels, and $T_b$ is the average detection statistic for non-target pixels based on the ground truth data. Factors $T_{max}$ and $T_{min}$ are the maximum and minimum evaluated detection statistics in the scene for a given algorithm. The best and the maximum score of visibility is 1, and the lowest score is 0.

## III. Analysis and Adaptation of Algorithms

The algorithms are tested on two hyperspectral datasets, namely, Pavia University scene and Salinas scene, which are publicly available on [10]. The testing of each algorithm is performed on full image dimensionality (all spectral bands), as well as in pipeline with dimensionality reduction technique - Principal component analysis (PCA). For Pavia scene, *Painted Metal sheets* and *Meadows* signatures are used as target signatures, while tests on Salinas scene are performed using *Lettuce romaine 4th and 5th week* signatures. For testing purposes, the thresholds are generated from a linearly spaced array with values between maximum and minimum detection statistic for a given scene and corresponding spectral signature. Thus, the MCC scores are obtained as the maximum achievable value over a range of thresholds.




(a) Pavia scene, average maximum MCC and visibility score for Painted Metal sheets and Meadows endmembers




(b) Salinas scene, average maximum MCC and visibility score for Lettuce romaine 4th and 5th week endmembers

Fig. 2. MCC and visibility scores for Pavia and Salinas datasets

The results for ACE, ACE-R, CEM and SAM algorithms for both scenes are shown in Fig. 2. Although the SAM algorithm shows good performance, it is characterized by low visibility. This makes the algorithm non-robust and shows its inability to separate background clutter from the possible targets. The CEM algorithm shows slightly lower MCC and visibility score over almost all scenes. Finally, it can be observed that the adapted ACE-R algorithm has the same performance or outperforms CEM, ACE and SAM algorithms. The targets are extracted from the provided scenes and especially ACE(-R) algorithm performs well in these conditions, which is consistent with the literature [1]. In particular, ACE-R proves to have high both MCC score and visibility on the used datasets.

Target detection with dimensionality reduction by PCA is also performed on Pavia and Salinas scenes. After the testing, it can be concluded that lowering the number of dimensions does not drastically degrade the performance metric values. In certain cases such as Salinas scene, MCC and visibility scores for ACE and ACE-R algorithms are improved. One of the reasons for the improvement can be found in elimination of the noise in discarded dimensions. Compared to full image dimensionality, dimensionality reduction shows advantages in both substantially shorter computing time as well as improvement of the performance scores for certain algorithms and scenes. Based on the analysis, ACE-R is chosen to be implemented on an FPGA-based HW/SW partitioned system.

## IV. Implementation of the Target Detection Core

In this paper, on-board target detection system for hyperspectral images is prototyped on *ZedBoard*. This board contains *Zynq-7000* System-on-Chip consisting of processing system (PS) with *ARM Cortex-A9* CPU and programmable logic (PL). The sensor data is stored in DDR3 memory, which can be accessed by both PS and PL of *Zynq SoC*. The programmable logic can communicate directly with DDR memory through AXI interface and AXI Direct Memory Access (DMA) [11]. As a step towards the full FPGA implementation, HW/SW codesign implementation of ACE-R target detection algorithm has been proposed. The overview of the system is shown in Fig. 3.

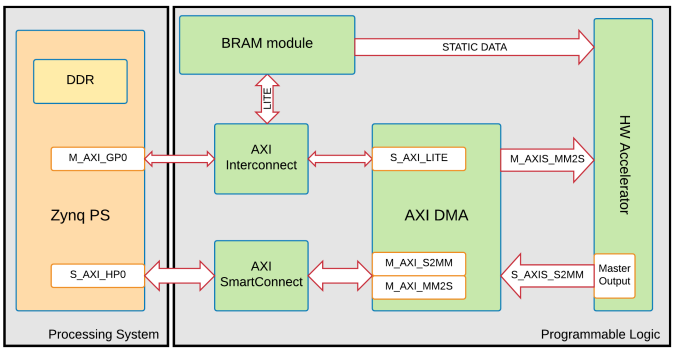



Fig. 3. System design of target detection accelerator

The operations of the algorithm are partitioned on the heterogeneous platform between processing system and programmable logic with a special consideration of background estimation. Since the correlation matrix can be estimated from the obtained image or already known from the previous runs, the correlation matrix and its inversion are computed in software and transferred to the hardware accelerator for the further algorithm steps. The computationally intensive matrix and vector operations such as dot product are then performed in hardware.

The block diagram of the implemented FPGA accelerator is shown in Fig. 4. The accelerator contains three pipelined HW stages. Initially, the inverted correlation matrix $\mathbf{R}^{-1}$ is uploaded to the BRAM and serves during the whole algorithm execution.

In the first stage, each spectral component is streamed from DDR through DMA via AXI interfaces, and fed to the dot product (DP) modules which perform the following operation:

$$\begin{bmatrix} L_i(\lambda_1) & L_i(\lambda_2) & \dots & L_i(\lambda_K) \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1K} \\ r_{21} & r_{22} & \dots & r_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ r_{K1} & r_{K2} & \dots & r_{KK} \end{bmatrix}$$

where $r_{ii}$ is an element of the precomputed inverted correlation matrix $\mathbf{R}^{-1}$. To compute this vector-matrix product and produce a vector of dot product elements:

$$\begin{bmatrix} \mathbf{x}^T\mathbf{row}_1(\mathbf{R}^{-1}) & \mathbf{x}^T\mathbf{row}_2(\mathbf{R}^{-1}) & \dots & \mathbf{x}^T\mathbf{row}_K(\mathbf{R}^{-1}) \end{bmatrix},$$

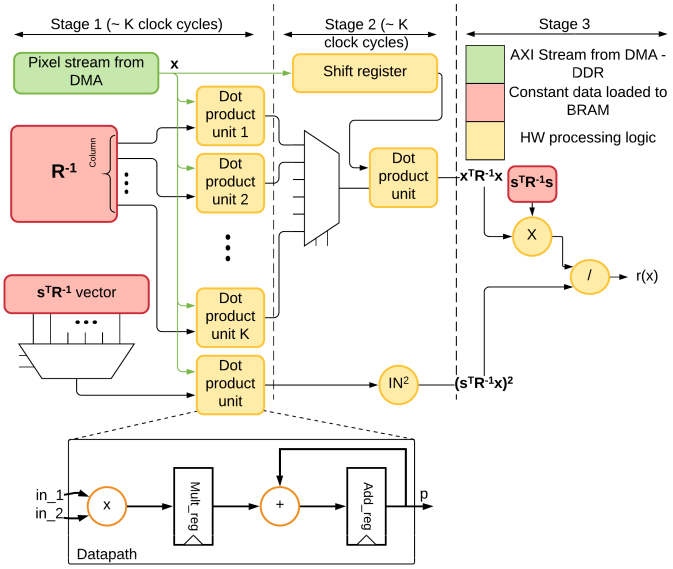



Fig. 4. Block diagram of accelerator's processing logic

it takes $K + delay$ clock cycles where $K$ is number of spectral bands and $delay$ corresponds to the pipeline delay. The resulting vector is then stored in intermediate registers in stage 2. In this stage, incoming stream from DMA is stored in shift register and used for computation of dot product with computed vector from stage 1. This results in computation of $\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x}$. Additionally, in stage 1 product $\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x}$ is calculated, where $\mathbf{s}^T\mathbf{R}^{-1}$ vector is stored in BRAM and used for each incoming pixel. The result of this operation is squared in stage 2. The final stage 3 can either be implemented in dedicated HW or in SW, while for HW implementation, a fixed-point divider is required, such as AXI IP divider [12] provided by Xilinx. This substantially adds to the latency of the output, however it can achieve significant speedup. On the other side, when performed in SW, the pre-processed detection statistics coming from FPGA accelerator are multiplied and divided as being written to DDR memory. Finally, to stream the output data and communicate with DMA, Master Output module has been designed. It acts as an AXI stream master interface which are not memory-mapped and allow data-burst mode.

## V. Results

The FPGA accelerator shown in Fig. 4 has been implemented in VHDL targeting ZedBoard development platform. Performance analysis of the design and individual sub-modules has been performed with post-synthesis results. The results are shown in Table I with annotated bit widths used in each module.

The FPGA accelerator implementation is constrained by dot product datapath module speed. Although stages 1 and 2 can operate at the same maximum frequency, other parts of the design are able to operate at significantly higher frequencies than the core datapath of the design. Thus, the dot product module has been deeply pipelined for the optimal performance.

TABLE I
PERFORMANCE ANALYSIS OF HW MODULES

| Module | Minimum period | Maximum frequency |
|---|---|---|
| PS-PL system (32 x 16) | $6.835ns$ | $146.3MHz$ |
| FPGA accelerator (32 x 16) | $5.745ns$ | $174.06MHz$ |
| Stage 1/2 (32 x 16) | $5.745ns$ | $174.06MHz$ |
| DP controller (16 bands) | $1.644ns$ | $608.27MHz$ |
| DP datapath (32 x 16) | $5.745ns$ | $174.06MHz$ |
| Master Output (16 packets) | $2.628ns$ | $380.52MHz$ |
| BRAM module (32) | $4.663ns$ | $214.45MHz$ |

Most importantly, the speed-up of the accelerator is 28.54 times when FPGA fabric is clocked with frequency of 100MHz. The speed-up is evaluated in comparison with *ARM Cortex-A9* processor running on 666.67MHz, while assuming that the correlation matrix is uploaded once and reused for consequent algorithm runs to detect a target signature. Compared to full processing time of $4.00s$ in software for Salinas scene (reduced to 16 spectral bands using PCA), where ACE-R algorithm without correlation matrix calculation takes $598.28ms$, the full computation time achieved with the use of the accelerator is $3.29s$, while ACE-R takes $20.96ms$.

Post-synthesis resource utilization has been presented in Tables II and III for 18-bit and 16-bit input hyperspectral data samples, respectively. Number of bands is set to 16 for both cases. It is important to note that the dedicated DSP block on *ZYNQ PL* has inputs that are 25 and 18 bits wide. However, since the design must accommodate different bit widths, those parameters are generic and affect the performance of the accelerator. In that sense, resource utilization for two sets of bit width parameters is presented, with the resource-optimized parameter set (25,18), and non-optimal set (32,16) providing higher precision. Therefore, the BRAM elements are of 25 and 32 bits, respectively. It should be noted that Top Level includes stage 1 and 2 of the FPGA accelerator as well as corresponding Master Output modules. All modules were synthesized out-of-context with default Vivado settings. The bit widths of the BRAM elements and input samples in Table II correspond to recommended input bit width for DSP blocks on *ZYNQ* PL. It can be observed that the DP datapath uses exactly one block for this setup and no other programmable logic to create a multiplier or an adder.

TABLE II
RESOURCE UTILIZATION REPORT 25X18, 16 BANDS, 32 BIT OUTPUT

| Module | Slice LUTs | Slice Registers | DSP blocks | BRAM tiles |
|---|---|---|---|---|
| PS-PL system | 6371 | 9935 | 32 | 10.5 |
| Top Level | 537 | 1772 | 23 | 0 |
| Stage 1 | 8 | 5 | 17 | 0 |
| Stage 2 | 8 | 22 | 3(6) | 0 |
| DP controller | 5 | 5 | 0 | 0 |
| DP datapath | 2 | 0 | 1 | 0 |
| Master Output (16 pkt) | 83 | 298 | 0 | 0 |
| BRAM wrapper (25 bit) | 133 | 300 | 0 | 8 |

Table III shows utilization results for bit width values which are not optimal for this architecture, resulting in usage of 2 DSP blocks to create a multiplier and additional logic elements to synthesize the wider accumulator. Overall, the resource utilization is higher in this case, emphasizing the importance of correct use of dedicated FPGA functional units.

TABLE III
RESOURCE UTILIZATION REPORT 32X16, 16 BANDS, 32 BIT OUTPUT

| Module | Slice LUTs | Slice Registers | DSP blocks | BRAM tiles |
|---|---|---|---|---|
| PS-PL system | 7567 | 11838 | 54 | 10.5 |
| Top Level | 1470 | 2937 | 40 | 0 |
| Stage 1 | 905 | 1178 | 34 | 0 |
| Stage 2 | 57 | 125 | 6 | 0 |
| DP controller | 5 | 5 | 0 | 0 |
| DP datapath | 53 | 69 | 2 | 0 |
| Master Output (16 pkt) | 83 | 298 | 0 | 0 |
| BRAM wrapper (32 bit) | 148 | 362 | 0 | 8 |

The part of the accelerator pipeline is shown in Fig. 5 with annotated bit widths. The accumulator in a dot product module is 52 bit wide (for 16 bands) and truncated to 32 bits at the end of both stages. To estimate the precision loss for computation of $\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x}$, the module has been simulated as well as signals debugged on ZedBoard and compared with floating point values in MATLAB. Then, relative root mean squared error is used as follows:

$$RRMSE = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(SW(i) - HW(i))^2}}{\frac{1}{N}\sum_{i=1}^{N} SW(i)} \cdot 100\% \quad (12)$$

for produced data in HW (simulated and on-board) and obtained data from MATLAB. Resulting RRMSE for computation of factor $\mathbf{x}^T\mathbf{R}^{-1}\mathbf{x}$ for 1000 samples is $0.2692\%$, whereas resulting RRMSE for factor $(\mathbf{s}^T\mathbf{R}^{-1}\mathbf{x})^2$ for the same number of samples is $0.6134\%$ due to square operation (64 bits).

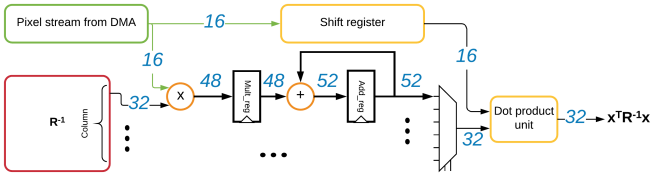
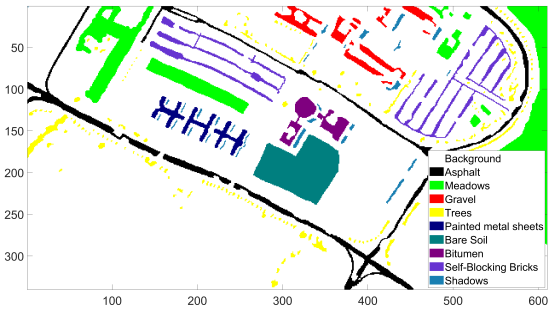
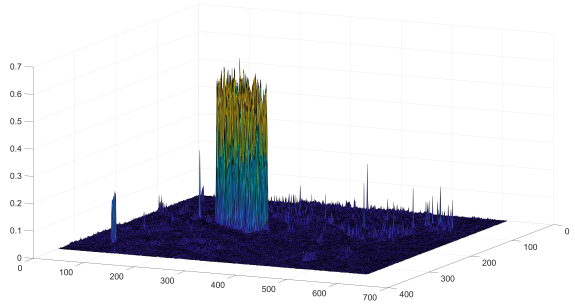



Fig. 5. Part of FPGA accelerator pipeline with annotated bit widths for 16-bit stream

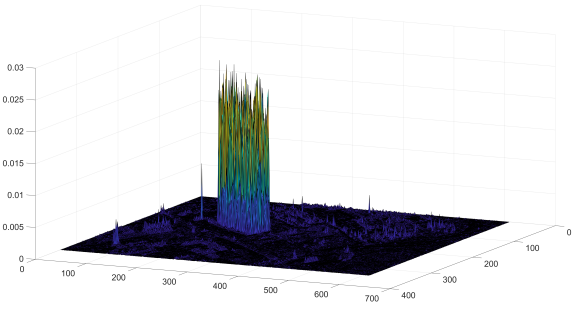
Detection results on Pavia scene using ACE-R algorithm are shown in Fig. 6 where ground truth map is given in Fig. 6(a). The detection results obtained using HW/SW codesigned implementation are presented in Fig. 6(b) and the error induced by fixed-point implementation is shown in Fig. 6(c). Table IV shows that the proposed fixed-point implementation does not significantly degrade the results in terms of MCC and visibility score.




(a) Pavia - ground truth



(b) Pavia 3D plot of detection results - Painted Metal Sheets endmember



(c) Pavia 3D plot of detection error compared to fixed point

Fig. 6. Pavia - Painted Metal Sheets detection results

TABLE IV
DETECTION PERFORMANCE

| Implementation | MCC score | Visibility score | AUC |
|---|---|---|---|
| SW model | 0.81439 | 0.62253 | 0.81946 |
| HW/SW | 0.81379 | 0.61804 | 0.81727 |

## VI. CONCLUSION

In this paper, HW/SW codesigned implementation of ACE target detection algorithm for hyperspectral data has been presented. A heterogeneous platform with processing system and programmable logic has been used along with specific communication interfaces. A number of algorithm implementations have been profiled for performance and resource utilization for HW implementation has been reported. The implemented design serves as a solid ground for the full HW implementation where it is required to implement correlation and inverse matrix calculation in the programmable logic in order to accelerate the execution of the algorithm to fulfill near real time requirement.

## 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] M. T. Eismann, *Hyperspectral remote sensing*. SPIE Press, 2012.
[2] Q. Du and R. Nekovei, "Fast real-time onboard processing of hyperspectral imagery for detection and classification," *Journal of Real-Time Image Processing*, vol. 4, no. 3, pp. 273–286, 2008.
[3] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geoscience and Remote Sensing Magazine*, vol. 1, no. 2, pp. 6–36, 2013.
[4] P. W. Fortescue, J. Stark, and G. Swinerd, *Spacecraft systems engineering, fourth edition*. Wiley, 2011.
[5] 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," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 698–722, 2013.
[6] J. C. Harsanyi, *Detection and classification of subpixel spectral signatures in hyperspectral image sequences*. University of Maryland Baltimore County, 1993.
[7] L. Scharf and L. McWhorter, "Adaptive matched subspace detectors and adaptive coherence estimators," *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers*, 1996.
[8] D. Manolakis, D. Marden, and G. A. Shaw, "Hyperspectral image processing for automatic target detection applications," *Lincoln laboratory journal*, vol. 14, no. 1, 2003.
[9] X. Jin, S. Paswaters, and H. Cline, "A comparative study of target detection algorithms for hyperspectral imagery," *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV*, 2009.
[10] "Hyperspectral remote sensing scenes," 2018. [Online]. Available: http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes
[11] "AXI DMA v7.1 LogiCORE IP," 2018. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf
[12] "Divider Generator v5.1 LogiCORE IP," 2018. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/div_gen/v5_1/pg151-div-gen.pdf