electronics

Article

An Efficient FPGA Implementation of Richardson–Lucy Deconvolution Algorithm for Hyperspectral Images

Karine Avagian and Milica Orlandić *

Department of Electronic Systems, NTNU, 7491 Trondheim, Norway; karine.avagian@gmail.com
* Correspondence: milica.orlandic@ntnu.no

Abstract: This paper proposes an implementation of a Richardson–Lucy (RL) deconvolution method to reduce the spatial degradation in hyperspectral images during the image acquisition process. The degradation, modeled by convolution with a point spread function (PSF), is reduced by applying both standard and accelerated RLdeconvolution algorithms on the individual images in spectral bands. Boundary conditions are introduced to maintain a constant image size without distorting the estimated image boundaries. The RL deconvolution algorithm is implemented on a field-programmable gate array (FPGA)-based Xilinx Zynq-7020 System-on-Chip (SoC). The proposed architecture is parameterized with respect to the image size and configurable with respect to the algorithm variant, the number of iterations, and the kernel size by setting the dedicated configuration registers. A speed-up by factors of 61 and 21 are reported compared to software-only and FPGA-based state-of-the-art implementations, respectively.

Keywords: hyperspectral imaging (HSI); field-programmable gate arrays (FPGA); image degradation; deconvolution; Richardson–Lucy algorithm; boundary conditions

Citation: Avagian, K.; Orlandić, M. An Efficient FPGA Implementation of Richardson–Lucy Deconvolution Algorithm for Hyperspectral Images. *Electronics* **2021**, *10*, 504. https://doi.org/10.3390/electronics10040504

Academic Editor: Gustavo Marrero Callico

Received: 18 January 2021
Accepted: 13 February 2021
Published: 21 February 2021

1. Introduction

A hyperspectral imaging (HSI) system combines spectroscopy and digital imaging such that an image is formed by splitting the detected light into narrow spectral bands [1]. Due to the interaction of light with different materials, the amount of detected radiance changes. Radiance includes illumination, the measurement position, and atmospheric effects, whereas reflectance is the ratio of reflected radiation to incident radiation and represents an intrinsic property of the material [2]. The measured reflectance, as a function of wavelength, is also called the *spectral signature*.

Hyperspectral remote sensing is performed by an HSI camera placed on an aircraft or a satellite. A pushbroom HSI scanner, containing a linear array of detector elements in the cross-track direction, can be used for data acquisition, as presented in Figure 1. The two-dimensional (2-D) data matrix, referred to as a frame, is produced in each time instance dictated by the defined frame rate. The frame width is equal to the number of detector elements N in the cross-track direction, whereas the height corresponds to the number of spectral bands P. A vector of size P for a spatial pair (x, y) is referred to as a pixel. The result of the acquisition process is a three-dimensional (3-D) data structure, a hyperspectral data cube $\mathbf{Y} \in \mathbb{R}^{N\times M\times P}$, where M is the number of temporal measurements. For each spectral band, $p \in \mathbb{R}^P$, there is a grayscale image $\mathbf{Y}^p \in \mathbb{R}^{N\times M}$ with two spatial dimensions $(x, y \in \mathbb{R}^{N\times M})$.

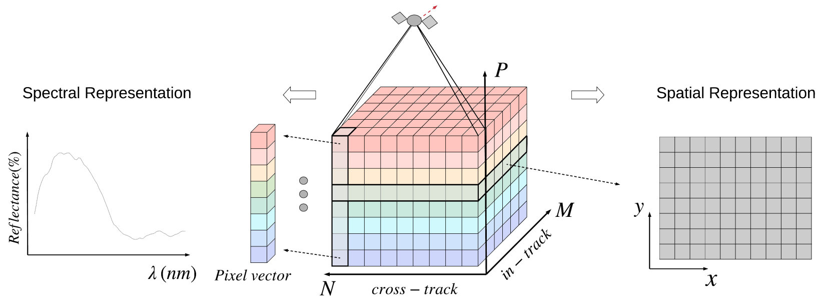


Figure 1. A representation of a hyperspectral data cube acquired by a pushbroom scanner and a pixel vector with the corresponding spectral signature.

Each sample is represented by a finite number of bits Q, which defines the radiometric resolution of an image. The HSI systems are also characterized by their spectral and spatial resolutions. The spatial resolution refers to the minimum distance at which two objects are distinguishable, whereas the spectral resolution is the minimum spectral distance between the recorded bands. A common measure of spatial resolution in remote sensing is the ground sampling distance (GSD), which defines the size of the grid elements projected onto Earth. To reduce the costs of HSI imagers, a trade-off between spectral and spatial resolutions is often performed at the cost of spatial resolution.

Super-resolution aims to improve the spatial image resolution by combining information from several low-resolution images into a high-resolution image. Super-resolution consists of image registration and deblurring. In image registration, the pixel information from several low-resolution images is placed into their correct positions in a high-resolution grid, whereas the deblurring step removes the distortions that occur during image formation. These distortions produced in the spatial domain can appear due to optical, motion, or detector blur. Optical blur results from imperfections in optical lenses, motion blur is caused by the relative motion between the camera and the scene's objects, whereas detector blur is produced by the nonuniformity in detector responses. Restoration methods, such as Richardson–Lucy [3,4] or Wiener reconstruction filtering use information about the direction and size of the blur in the deconvolution kernel. Classical 2-D restoration algorithms such as Wiener filtering [5] and constrained least squares filtering [6] have been further developed to solve 3-D restoration problems. Separability between the spectral and spatial domains is assumed in [7], such that Wiener filtering is performed independently on each 2-D band of the HSI data set. The presented methods are characterized by slow execution times and increased complexity due to the individual models for each band. In [8], a modified Wiener filtering is performed without the separability assumption. An alternative solution based on a multi-channel least squares filtering is presented in [9]. In both works, it is stated that, in the presence of strong cross-channel correlations, the multi-channel model outperforms the single-channel model. The drawback of the described solutions is inefficiency to restore the frequencies beyond the point spread function (PSF) bandwidth [10], resulting in ringing artifacts due to the generation of negative-value pixels in the deconvolved image.

The Richardson–Lucy deconvolution (RL deconvolution) algorithm achieves both image restoration and super-resolution, where super-resolution refers to the restoration of high-frequency components [11]. For hyperspectral images acquired by a *short-wave infrared* (SWIR) pushbroom imaging system, a 3-D RL deconvolution method [12] enhances the spatial resolution in both the cross- and in-track directions with the cross-channel displacement corrections. The spectral image is modeled as a 3-D convolution between an undistorted image with a spatially variant response function. The authors state that the reduction of displacement and blur can be achieved simultaneously by using 3-D RL deconvolution when the response function is known. The spatial resolution is enhanced, with the largest effect in the bands with the lowest acquired resolution.

There are several hardware implementations for the 2-D RL deconvolution algorithm. A space-invariant PSF is considered in [13–15], whereas a space-variant PSF is assumed in [16]. In [13], a whole scene is affected by the same degradation kernel and the RL deconvolution is performed by the use of Fourier transformations. The presented architecture is digital signal processor (DSP)-based and uses a Virtex-4 field-programmable gate array (FPGA) as a co-processor. The maximum processing frequency is 100 MHz. The proposed system executes the algorithm on images of size 64×64. In [14], a fully ported hardware implementation of the RL deconvolution algorithm is presented. The architecture supporting kernels from 3×3 to 9×9 is implemented to process a fixed amount of iterations (equal to 10). The architecture is tested on Stratix V, and the maximum processing frequency is 61 MHz. An accelerated RL deconvolution, with β values between 1 to 3 in the first iteration, is implemented in [15]. The kernel is assumed to be space-invariant and separable, whereas a fixed number of RL deconvolution iterations is set to 2. The algorithm is implemented as a continuous datapath, which simplifies the control system. The algorithm is implemented on a Xilinx Virtex 3 XC2VP50, the maximum frequency is 63 MHz, and the achieved throughput is 60 MP/s (megapixels per second). In [16], the PSF is assumed to be shift-variant. To ease the computational complexity and memory requirements, PSFs are described by sparse matrices. The proposed architecture is implemented on Altera Stratix V. Both the motion blur and the lens distortion are modeled by Gaussian PSF. A unique PSF is associated with every image sample. The maximum processing frequency is not however stated.

The proposed work, built upon the hardware/software co-design presented in [17], introduces the design and a full real-time FPGA implementation of an RL deconvolution algorithm. The proposed work is performance-driven in terms of processing, speed and it supports generic image sizes and configurations of the parameters such as algorithm variants, the number of iterations, and the kernel size prior to algorithm execution. In this way, the implementation provides flexibility towards its use in different applications and sensors.

The rest of this paper is organized as follows. Section 2 introduces the image registration concept and describes the Richardson–Lucy algorithm. The analysis, which defines the configuration setup for implementation, is presented in Section 3. The proposed hardware implementation is described in Section 4. The influence of the configuration choices for the chosen architectural solutions on the logic use, timing, and power are analyzed in Section 5. Finally, the conclusions are given in Section 6.

2. Background

2.1. Image Degradation

The HSI optical system collects incoming light from the object plane and forms an image in the image plane through a set of lenses, mirrors, and beam splitters. The transition between the two planes through the optical system and introduced distortions are described and modelled by the HSI's response function. The light reflected by the surface is spatially incoherent if the phase and the amplitude of the light wave fluctuate randomly. An imaging system is isoplanatic if the response function is space-invariant. For a spatially incoherent and isoplanatic imaging system, the intensity measurement $f(s)$ can be described as a linear shift-invariant system as follows:

$$f(s) = H(s) \circledast g(s) \tag{1}$$

where the function $H(s)$ is the instrument's response function or the *point spread function* (PSF) of the optical system [18], $g(s)$ is the intensity of the object at position s, $\circledast$ denotes convolution, and $f(s)$ is the intensity measurement of that object formed by the instrument [18]. The variable s corresponds to a sample (x, y, p) in a 3-D hyperspectral data cube. It is assumed that the neighbouring spectral bands do not affect each other and are modeled independently; thus, the 3-D HSI data cube is a composite of P separate 2-D images of size $N \times M$. The 2-D observation model within a spectral band p is given as follows:

$$\mathbf{Y}^p = \mathbf{H}_D^p \circledast \mathbf{X}^p + \mathbf{N}^p. \tag{2}$$

where $\mathbf{Y}^p \in \mathbb{R}^{N \times M}$ is the acquired 2-D image in pth band of the observed HSI data cube, $\mathbf{X}^p \in \mathbb{R}^{N \times M}$ is the corresponding 2-D ideal image without distortions, $\mathbf{H}_D^p \in \mathbb{R}^{W_y \times W_x}$ is the degradation kernel of size $W_y \times W_x$, and $\mathbf{N}^p \in \mathbb{R}^{N \times M}$ is a signal-independent Gaussian noise [10].

A 2-D convolution is a computation-intensive operation with a computational complexity $\mathcal{O}(MNW_xW_y)$. To decrease computational complexity, a separability property is introduced. The property implies that a 2-D kernel $\mathbf{H}_{x,y}$ of size $W_y \times W_x$ can be decomposed into two 1-D kernels as follows:

$$\mathbf{H}_{x,y} = \mathbf{H}_y \cdot \mathbf{H}_x, \tag{3}$$

where $\mathbf{H}_y$ and $\mathbf{H}_x$ are vertical and horizontal kernels of sizes $Wy \times 1$ and $1 \times W_x$, respectively. The 2-D separable convolution performs first convolution of the input image with the vertical kernel and then that of the intermediate result with the horizontal kernel. For convolution with the separable kernel, the computational complexity is decreased to $\mathcal{O}(MN(W_x + W_y))$, resulting in a speed-up of $W_xW_y/(W_x + W_y)$ when compared to the standard 2-D convolution.

In optical blur modeling, separable Gaussian PSFs given as

$$\mathbf{H}_{x,y} = \mathbf{H}_x \cdot \mathbf{H}_y = \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right] \cdot \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right], \tag{4}$$

are often used, where the standard deviation σ determines the width of the PSF and controls the amount of blur in the image.

2.2. Image Restoration

The goal of image restoration is to find the best approximation of an original image from a blurred and noisy observation. The restoration process can be non-blind, semi-blind, or blind depending on a level of knowledge about the response function. If the response function is known, the image restoration process is a *non-blind* restoration. A generalized model for image degradation and restoration is shown in Figure 2, where an ideal image $\mathbf{X}^p$ is filtered with a degradation function $\mathbf{H}_D^p$ and further distorted by additive noise $\mathbf{N}^p$, resulting in a degraded image $\mathbf{Y}^p$. The degraded image is then convolved with a restoration filter $\mathbf{H}_R^p$ to produce a restored image $\hat{\mathbf{X}}^p$.

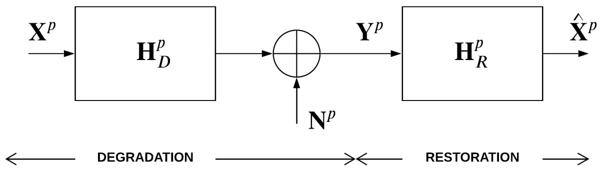


Figure 2. A general block diagram representing degradation and restoration model.

Richardson–Lucy algorithm (RL deconvolution) [3,4] is an iterative nonlinear deconvolution algorithm. Its convergence is slow, and the effects on the pixel values are the largest in the first iterations [19]. A method, referred to as a "multiplicative relaxation", introduced in [20], can accelerate the deconvolution process by using an additional parameter, β. The generalized RL deconvolution algorithm for 2-D data sets is given as follows:

$$\hat{\mathbf{X}}_{(k+1)}^p = \hat{\mathbf{X}}_{(k)}^p \left[\frac{\mathbf{Y}^p}{\mathbf{H}_R^p \circledast \hat{\mathbf{X}}_{(k)}^p} \odot \mathbf{H}_R^p\right]^\beta \tag{5}$$

where $\hat{\mathbf{X}}^p_{(k)}$ is the estimate of $\mathbf{X}^p$ after k iterations, $\odot$ is the correlation operator, $\circledast$ is the convolution operator, $\mathbf{H}^p_R$ is the restoration kernel, and $\beta > 1$ is the exponential correction factor. The accelerated RL deconvolution algorithm is characterized by $\beta > 1$. The standard RL deconvolution algorithm defined by $\beta = 1$ is presented in a block diagram in Figure 3 and is summarized in a number of steps as follows:

- decision on an initial estimate, $\hat{\mathbf{X}}^p{}_{(0)}$;
- computation of the residual, $residual = \dfrac{\mathbf{Y}^p}{\mathbf{H}^p_R \circledast \hat{\mathbf{X}}^p_{(k)}} = \dfrac{\mathbf{Y}^p}{\mathbf{Y}'^p_{(k)}}$;
- computation of the correction factor, $\phi_{(k)} = residual \odot \mathbf{H}^p_R$; and
- update of the initial estimate, $\hat{\mathbf{X}}^p{}_{(k+1)} = \hat{\mathbf{X}}^p{}_{(k)} \times \phi_{(k)}$.

The faster convergence reduces the number of iterations [21]. The improved convergence rate for $\beta > 1$ stated in [20] comes with a potential drawback of the lack of stability in the convergence [20]. In that sense, a maximum value of β in the accelerated algorithm is set to 2 to avoid divergence. In the case of additive noise, early termination of the restoration process can prevent noise amplification.

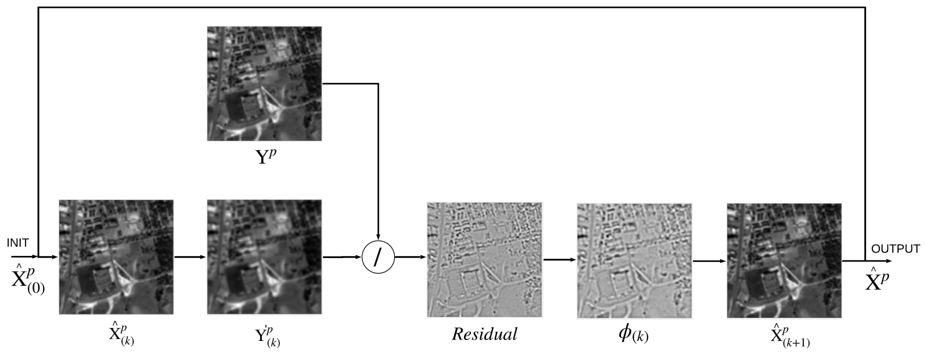


Figure 3. An illustration of one iteration of a Richardson–Lucy (RL) deconvolution algorithm.

2.3. Border Handling

The convolution process is depicted in Figure 4, where the blue, yellow, and green matrices illustrate an input image, a kernel, and the resulting image, respectively. When performing a convolution of an image and the kernel $\mathbf{H}^p_R$, the output of linear spatial filtering is an image of size $(N - (W_x - 1), M - (W_y - 1))$. The number of missing samples, marked grey, depends on the kernel and input image sizes and is equal to $(W - 1) \times (N + M - (W - 1))$ for $W = W_x = W_y$. The size of the input image can be preserved by extending input image with $(W - 1) \times (N + M + (W - 1))$ synthetic samples such that the convolution is also performed on the border samples of the original input image. The padded samples can be estimated by imposing prior boundary conditions (BCs) on the input image. The common BCs [22] are constant-BC (C-BC), periodic-BC (P-BC), and reflective-BC (R-BC), presented in Figure 5. In C-BCs, the estimated border samples are assumed to be constant. Periodic-BCs assumes repetition of the samples near the boundaries in each direction. In R-BCs, estimated border samples are mirrored versions of the samples near the boundary. Constant-BCs are the simplest solution but are the least reliable. Periodic-BCs allow the use of cyclic convolution, which can be implemented via *fast Fourier transform* (FFT) [22], whereas reflective-BCs preserve the boundary continuities.

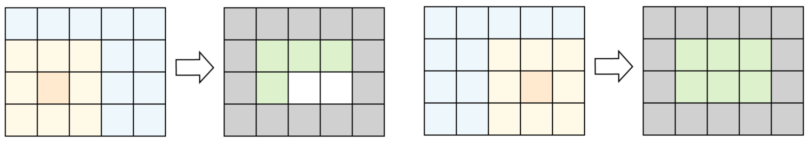
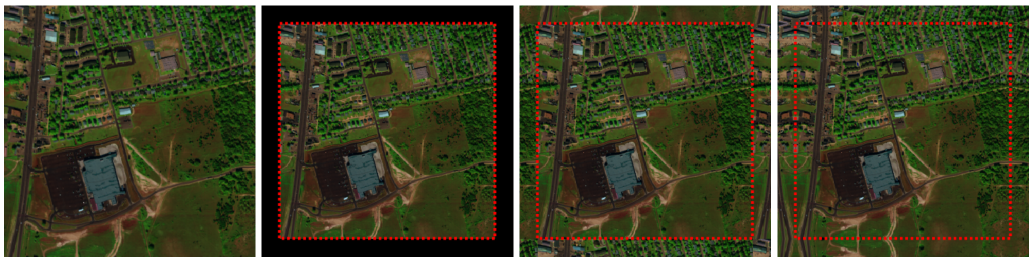


Figure 4. The convolution process by moving the kernel.



(**a**) An input image. (**b**) Constant-BCs. (**c**) Periodic-BCs. (**d**) Reflective-BCs.

Figure 5. Three types of boundary conditions (BCs) applied on the input image.

2.4. Image Quality Assessment

The relative reconstruction error (RRE) is used to evaluate similarities between the reconstructed image $\hat{\mathbf{X}}_k^p$ and the reference image $\mathbf{X}^p$:

$$RRE = \frac{||\hat{\mathbf{X}}_k^p - \mathbf{X}^p||}{||\mathbf{X}^p||} \tag{6}$$

where $||.||$ denotes the Euclidean distance. The peak signal-to-noise ratio (PSNR) computed in a band-by-band manner is mathematically expressed as follows:

$$PSNR = 10\log_{10}\frac{(M \times N)(\mathbf{X}_{max}^p)^2}{||\mathbf{X}^p - \hat{\mathbf{X}}_k^p||^2} \tag{7}$$

where $\mathbf{X}_{max}^p$ is the maximum value of the image $\mathbf{X}^p$ and $M \times N$ is the number of samples in one band [23]. The denominator refers to the mean square error (MSE) between the restored image and the ideal image. When there are no differences between the images, MSE equals zero and the $PSNR$ value is ideally equal to infinity.

The structural similarity index (SSIM) takes into account parameters such as luminance, contrast, or structure in the images and measures similarity between a restored image and the reference image via a method more similar to the human eye:

$$SSIM(\hat{\mathbf{X}}_k^p, \mathbf{X}^p) = [l(\hat{\mathbf{X}}_k^p, \mathbf{X}^p)^\alpha \cdot [c(\hat{\mathbf{X}}_k^p, \mathbf{X}^p)]^\beta \cdot [s(\hat{\mathbf{X}}_k^p, \mathbf{X}^p)]^\gamma \tag{8}$$

where $l(\hat{\mathbf{X}}_k^p, \mathbf{X}^p)$ is related to the luminance difference, $c(\hat{\mathbf{X}}_k^p, \mathbf{X}^p)$ is related to the contrast differences, and $s(\hat{\mathbf{X}}_k^p, \mathbf{X}^p)$ is related to the structure variations. The parameters α, β, and γ define the relative importance of each component. $SSIM$ ranges between 0 and 1, where 1 refers to the complete similarity [23].

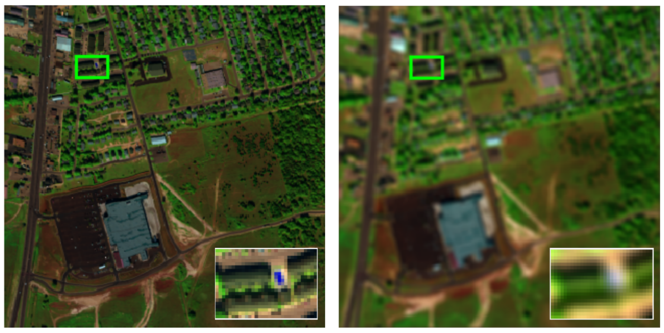
In order to evaluate the quality of the hyperspectral cube, a mean-PSNR (M-PSNR) and a mean-SSIM (M-SSIM) are computed as averages of the $PSNR$ and $SSIM$ values in each band.

3. RL Deconvolution Algorithm Analysis

The RL deconvolution algorithm was analysed in order to define the ranges of parameters such as sizes and types of kernels, boundary condition support, and algorithm variations that are of interest in the design process. The trade-off analysis for deciding upon parameter ranges affecting the architectural choices is based on three criteria: the level of added complexity, the quality of the deconvolution process, and the performance increase.

3.1. Hyperspectral Data Set

The analysis was performed on the *Urban* HSI data set collected by the HYDICE sensor with a total of P = 162 spectral bands of size 307 × 307 [24]. The three-band composite (100, 55, and 30) of the degradation-free image, used as a reference, is shown in Figure 6a. The image bands are synthetically degraded by Gaussian blur with $\mu = 0$ and σ_D = 2.3, shown in Figure 6b. The M-PSNR and the M-SSIM measures between the blurred image and the reference image are equal to 41.15 dB and 0.7859 %, respectively.



(**a**) (**b**)

Figure 6. (**a**) Original three-color composite of three spectral bands (100, 55, and 30) from the Urban data set (**b**) degraded with Gaussian blur, with σ_D = 2.3 and size 7 × 7. The portion of the image marked with a green rectangle is enlarged in the right corner.

3.2. Boundary Conditions

Three types of boundary conditions, zero (Z)-BCs, variable-constant (VC)-BCs and modified periodic (MP)-BCs, are used to assess the algorithm behavior in terms of different boundary conditions. For VC-BCs, a constant value for the padded border pixels is set to the value of the first image sample and the value varies for each iteration. For MP-BC, only the upper and lower image borders are extended by using VC-BCs whereas the border samples from the neighboring rows are assigned to the added border pixels on the sides of the input image. Restoration is performed by using both standard and accelerated RL deconvolution. A restoration Gaussian kernel of size 7 × 7 and σ_R = 2.3 was used. The resulting M-RRE values as a function of the number of iterations are presented in Figure 7, where the solid and dashed lines correspond to the standard RL deconvolution and the accelerated RL deconvolution, respectively. The M-PSNR and M-SSIM for the standard RL deconvolution at the minimum M-RRE are shown in Table 1. The relative improvement in M-PSNR and M-SSIM was computed by comparing M-PSNR and M-SSIM values resulting from the image before and after restoration. The resulting images after performing RL deconvolution on the input image extended with three BC types (Z-BC, VC-BC, and MP-BC) are presented in Figure 8. It can be observed that Z-BCs produces visible ringing artifacts on the image borders whereas visually similar resulting images are produced for VC-BCs and MP-BCs. The values of objective metrics M-PSNR and M-SSIM are in accordance with the visual assessment, as shown in Table 1. However, the process of adding VC-BCs to the input image requires the assignment of new samples for every image border sample. In this manner, the streaming of pixels is interrupted and the position of image data is constantly checked, which causes a higher complexity of the control path and increased latency.

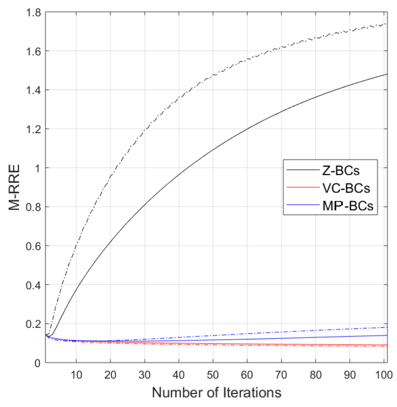
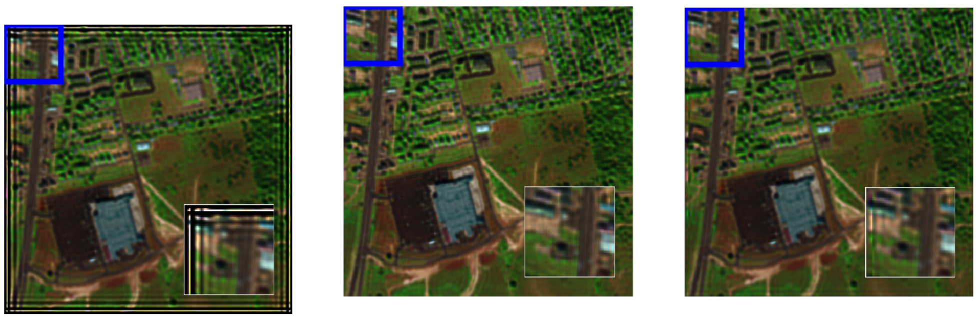


Figure 7. Mean reconstruction error as a function of the number of iterations for deconvolution by Gaussian kernel with σ_R = 2.3 and BCs, Z-BCs, VC-BCs, and MP-BCs. The solid line corresponds to standard RL deconvolution ($\beta = 1$), whereas the dashed line is for accelerated RL deconvolution, with $\beta = 2$.

Table 1. Mean peak signal-to-noise ratio (M-PSNR) and mean structural similarity index (M-SSIM) for images deblurred using a 7 × 7 Gaussian kernel with σ_R = 2.3 and three different BCs.

BCs	**M-PSNR(dB)**		**M-SSIM**	
	M-PSNR	**Relative Improvement (dB)**	**M-SSIM**	**Relative Improvement (%)**
Z-BC	20.75	−20.37	0.7805	−0.69
VC-BC	45.00	3.85	0.8920	13.50
MP-BC	41.37	0.23	0.8859	12.72



(**a**) Zero (Z)-BCs (**b**) Variable constant (VC)-BCs (**c**) Modified periodic (MP)-BCs

Figure 8. The resulting images after RL deconvolution performed on an Urban image with different BCs.

3.3. *Effects of Kernel Standard Deviation on Image Reconstruction*

The effects of choice of reconstruction kernel with respect to a fixed degradation kernel were analyzed when designing the optimized RL deconvolution algorithm implementation. The Urban image was initially degraded by 7 × 7 Gaussian blur with σ_D = 2.3, and then it was restored by the Gaussian kernels with standard deviations, $\sigma_R = [1.0, 2.0, 2.3, 2.6, 5.0]$. In this manner, the reconstruction kernels both smaller, equal, and larger than the degradation kernel were provided for analysis. Each kernel size was set to $3\sigma_R$. The quality of the image reconstruction performed by both the standard and accelerated RL deconvolution algorithms for different number of iterations was evaluated by the use of M-RRE metrics. Figure 9 shows that the M-RRE curve for the restoration kernel equal to the degradation kernel $\mathbf{H}_R(2.3)$ is characterized by the fast and stable convergence. This is also observed in M-PSNR and M-SSIM values for image reconstruction with different kernels presented in Table 2 for the standard RL deconvolution and in Table 3 for the accelerated RL deconvolution. For reconstruction kernel $\mathbf{H}_R(2.3)$, the RL deconvolution algorithm converges rapidly for the first ∼500 iterations. For 5000 iterations of the accelerated RL deconvolution, the achieved M-PSNR and M-SSIM values are 48.84 dB and 0.9708 %, whereas the improvement in M-PSNR and in M-SSIM relative to the degraded image are 7.69 dB and 23.53 %, respectively. For the accelerated RL deconvolution algorithm, the corresponding reconstructed images are shown in Figure 10b,f after 300 iterations and 5000 iterations, respectively. Although the M-RRE decreases for each iteration, the images are without significant visible differences for both k = 300 and k = 5000.

When the reconstruction kernel and the degradation kernel are not equal, the deconvolution result depends on the similarity between coefficients of the restoration kernel and the ideal kernel. The M-RRE curves for reconstruction kernels with standard deviations larger than the degradation kernel standard deviation, $\sigma_R > \sigma_D$, start to diverge after a number of iterations. For a kernel with standard deviation $\sigma_R = 2.6$, divergence starts after 22 iterations, whereas for reconstruction by a kernel with standard deviation $\sigma_R = 5$, divergence is observed from the first iteration. For the restoration kernel $\mathbf{H}_R(2.6)$, the relative improvements in M-SSIM compared to the degraded image for accelerated RL deconvolution after 7 and 300 iterations are equal to 5.52 % and −2.38 %, respectively, as presented in Table 3 and Figure 10g. This means that restoration using this kernel might improve the spatial image resolution if RL deconvolution is stopped early. On the other hand, for restoration kernel $\mathbf{H}_R(5.0)$, the relative improvements in M-SSIM after one iteration are equal to 0.37 % for the standard RL deconvolution and −0.78 % for the accelerated RL deconvolution. The positive result for standard RL deconvolution and the negative one for accelerated RL deconvolution suggest a need for both algorithms. The restoration results indicate that restoration kernels smaller than the degradation kernel are used in the restoration process. An improvement in M-SSIM for both $\mathbf{H}_R(1.0)$ and $\mathbf{H}_R(2.0)$ compared to the M-SSIM of the degraded image is observed, and the M-RRE values do not vary significantly with respect to the number of iterations, as shown in Figure 9. Figure 10a–d show the resulting images after performing the accelerated RL deconvolution algorithm at the minimum M-RRE for kernels with σ_R = [2.0, 2.3, 2.6, 5.0], respectively. In the case in which the algorithm is not stopped at the minimum M-RRE, the images shown in Figure 10e–h are the results of RL deconvolution using the reconstruction kernels $\mathbf{H}_R(\sigma_R)$ with σ_R = [2.0, 2.3, 2.6, 5.0], respectively.

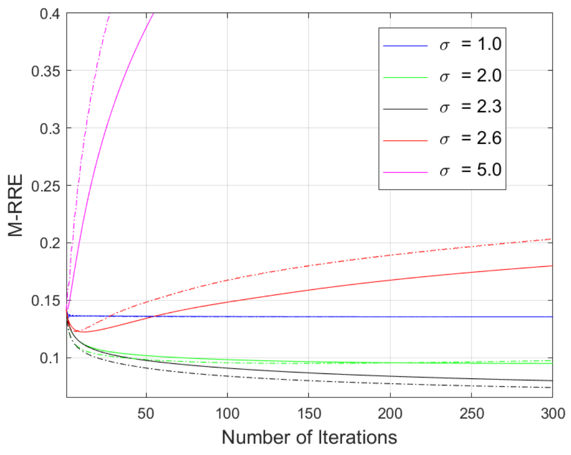
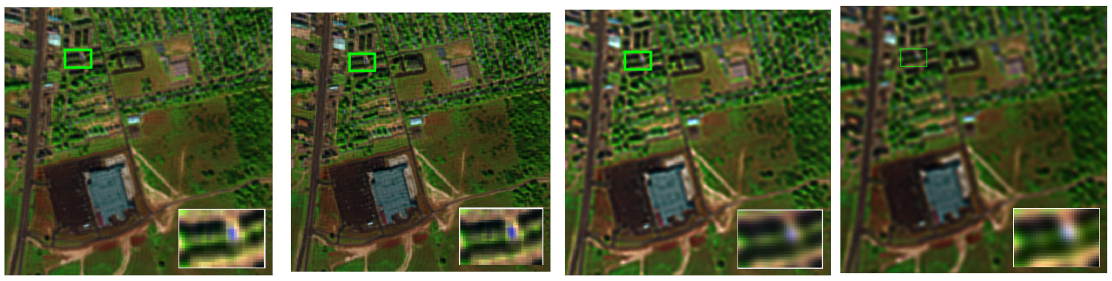
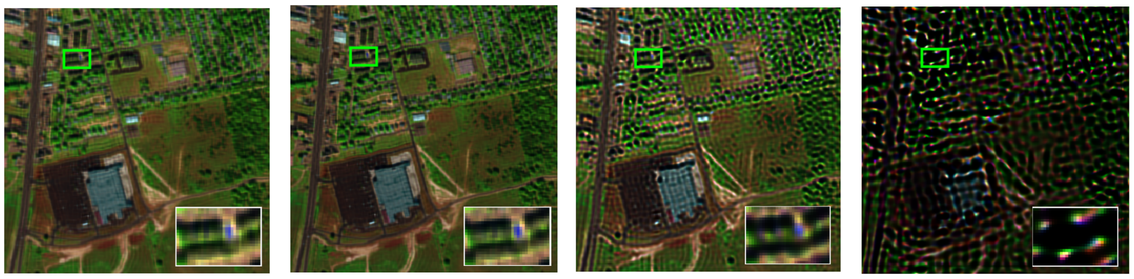


Figure 9. Mean relative reconstruction error (M-RRE) as a function of the number of iterations for different reconstruction kernels using standard (solid lines) and accelerated RL deconvolution (dashed lines).



(**a**) $\mathbf{H}_R(\sigma)$ = 2.0, k = 147. (**b**) $\mathbf{H}_R(\sigma)$ = 2.3, k = 300. (**c**) $\mathbf{H}_R(\sigma)$= 2.6, k = 7. (**d**) $\mathbf{H}_R(\sigma)$ = 5.0, k = 1



(**e**) $\mathbf{H}_R(\sigma)$ = 2.0, k = 300. M-PSNR=44.38, M-SSIM=0.9092. (**f**) $\mathbf{H}_R(\sigma)$ = 2.3, k = 5000. M-PSNR=48.84, M-SSIM=0.9708. (**g**) $\mathbf{H}_R(\sigma)$ = 2.6, k = 300. M-PSNR=38.02, M-SSIM=0.7672 (**h**) $\mathbf{H}_R(\sigma)$ = 5.0, k = 300. M-PSNR=25.39, M-SSIM=0.2195

Figure 10. Visual accelerated RL deconvolution results with varying kernel coefficients. k indicates the number of RL deconvolution iterations.

Table 2. M-PSNR and M-SSIM at the minimum M-RRE for different kernels running the standard RL deconvolution.

σ_R	iter.	M-PSNR(dB)		M-SSIM	
		M-PSNR	**Relative Improvement (dB)**	**M-SSIM**	**Relative Improvement (%)**
1.0	300	41.57	0.42	0.8025	2.12
2.0	294	44.61	3.46	0.8985	14.33
2.3	300	46.11	5.97	0.9207	17.16
2.6	22	42.42	1.27	0.8287	5.45
5.0	1	41.14	0.0	0.7888	0.37

Table 3. M-PSNR and M-SSIM at the minimum M-RRE for different kernels running the accelerated RL deconvolution.

σ_R	iter.	M-PSNR(dB)		M-SSIM	
		M-PSNR	**Relative Improvement (dB)**	**M-SSIM**	**Relative Improvement (%)**
1.0	165	41.57	0.42	0.8027	2.14
2.0	147	44.61	3.64	0.8986	14.34
2.3	300	46.79	5.64	0.9361	19.11
2.6	7	42.40	1.26	0.8293	5.52
5.0	1	39.85	-1.29	0.7807	-0.78

4. Proposed Implementation

4.1. Overall Architecture

The proposed architecture is designed to perform both standard RL deconvolution with $\beta = 1$ and accelerated RL deconvolution with $\beta = 2$. The RL deconvolution core is designed to provide image dimensions as the customization parameters prior to synthesis, whereas the core's parameters such as algorithm type, kernel size, and number of iterations can be configured directly by the dedicated registers through the CPU's memory map. A generalized block diagram, consisting of a *Core IP* for a complete RL deconvolution processing pipeline and a *Configuration* block with an AXI4-Lite Slave interface, is shown in Figure 11. The end-user chooses the kernel size by writing the value, together with the number of iterations and the algorithm variant, into the register stored in the configuration block. The degraded input image $\mathbf{Y}^p$ is reused in each iteration for residual computation, and it is required to be either stored in the internal memory of the FPGA or streamed from an external memory for each new iteration. The former minimizes the communication time between the core on FPGA and the external memory, whereas the latter minimizes FPGA resource utilization. The original image size is preserved by using the MP-BCs due to lower latency and decreased complexity of the control path compared to VC-BC. A feature summary for the proposed architecture is presented in Table 4.

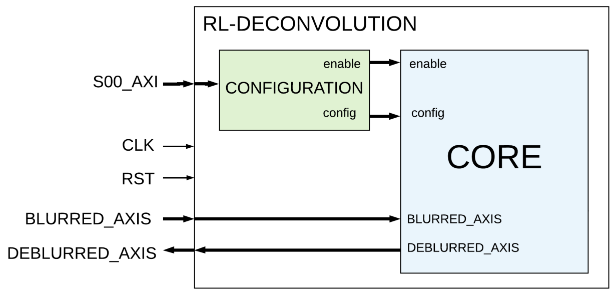


Figure 11. A generalized block diagram for the proposed architecture.

Table 4. Feature summary for the proposed architecture.

Feature	Proposed Architecture
Standard RL deconvolution	✓
Accelerated RL deconvolution	✓
Initial value, $\hat{\mathbf{X}}^p_{(0)}$	$\mathbf{Y}^p$
BCs	MP-BCs
Generic image size	✓
Run-time conf. kernel size	✓
Run-time conf. number of iterations	✓
Intermediate image data stored internally	✓

Figure 12 shows the submodules and their interconnections in the data processing pipeline of the proposed Core IP to perform RL deconvolution.

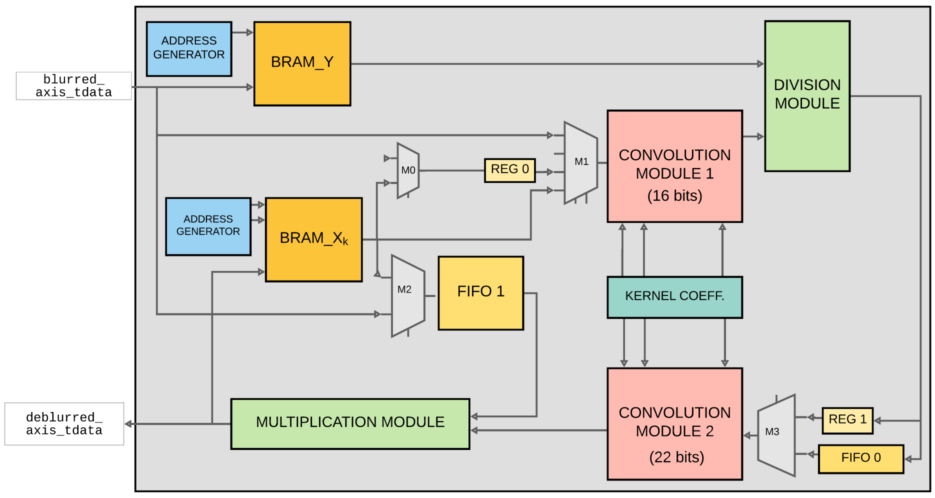


Figure 12. Complete block diagram for the RL deconvolution datapath.

The convolution module 1 with 16-bit input signals $\mathbf{H}_R^p$ and $\hat{\mathbf{X}}_{(k)}^p$ computes the divisor in the residual computation step in Equation (5). For the first iteration, the initial value of $\hat{\mathbf{X}}_{(k)}^p$ is equal to the input signal `blurred_axis`, whereas for the consequent iterations, $\hat{\mathbf{X}}_{(k)}^p$ is fetched from $BRAM\_X_k$. Additionally, to preserve the original size of the input image, the convolution is firstly performed on the data stored in REG_0, which is equal to a predefined constant for the first iteration of the algorithm and the first sample stored in $BRAM\_X_k$ for the consequent iterations. The number of estimated samples is equal to $(N \times 4) \times 2$. The degraded input image $\mathbf{Y}^p$ remains constant during the deconvolution process. The image is initially streamed from the external memory into the Core IP through the AXI-Stream `blurred_axis` input and is stored in $BRAM\_Y$. The address generators ensure correct address reading and writing for the Block RAMs (BRAMs). The valid samples from the first convolution module are streamed into the divisor of the Xilinx Division IP [25], whereas the data stored in $BRAM\_Y$ is used as the dividend. In order to preserve the output data precision, the quotient width is set to 22 bits with 20-bit fractional part. The second convolution module computes the correction factor $\phi_{(k)}$ in Equation (5). To preserve the original image size, the convolution is performed initially on the data in REG_1, which stores the first valid quotient value, and then on the delayed data from the division module. The division output samples are stored in a FIFO [26] during processing of the data from REG_1. The depth of $FIFO\_0$ depends on the image width and the kernel size as follows:

$$Depth(\text{FIFO\_0}) = N \times (W-1)/2 + 3, \tag{9}$$

where N is the image width and W is the kernel size. For $W = 9$, the depth of $FIFO\_0$ is $N \times 4 + 3$. In the first iteration, an initial estimate $\hat{\mathbf{X}}_{(0)}^p$ is equal to the degraded input image. As a valid output sample is produced by the second convolution module, its multiplication with $\hat{\mathbf{X}}_{(0)}^p$ starts. The proposed design uses a true dual-port BRAM with one read and write port, which can be used simultaneously. However, since one write and two read ports with different addresses are required, a $FIFO\_1$ is added. The input data, `blurred_axis`, at the first iteration and the data from $BRAM\_X_k$ at the consequent iterations enter simultaneously into $FIFO\_1$ and convolution module 1. The depth of $FIFO\_1$ is given as follows:

$$Depth(\text{FIFO\_1}) = N \times (W-1) \times 4 + 52. \tag{10}$$

Finally, the 22-bit convolution module output and the 16-bit $FIFO\_1$ output are inputs to the multiplication module corresponding to `INPUT_1` and `INPUT_2` in the block diagram shown in Figure 13. The choice between the standard and the accelerated RL deconvolution is performed in this module by the multiplexer MUX_1, which is controlled by the end-user through the configuration block. If the targeted number of iterations is performed, the multiplication output is streamed to the external FPGA memory via `deblurred_axis`; otherwise, the output is stored in $BRAM\_X_k$.

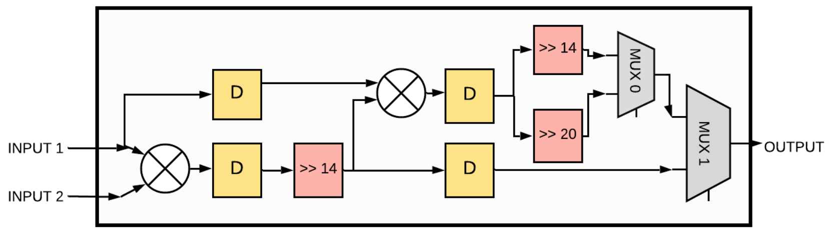


Figure 13. Multiplication module. D represents one clock cycle delay.

4.2. Convolution Module

A convolution of a 2-D image of size $N \times M$ with a kernel of size $W_x \times W_y$ is performed by a convolution module proposed in [27], where $W_x = W_y = W$. The 2-D convolution is accelerated by utilizing kernel separability, where a convolution by a 1-D vertical kernel $\mathbf{H}_y$ and the input data column vector of size W are followed by a convolution of W intermediate results and a 1-D horizontal kernel $\mathbf{H}_x$. The central samples are then required as neighbors in the convolution operations, and in order to avoid sample reloading into the convolution module, line buffers are used. The block diagram, consisting of line buffers and two 1-D convolutions, is shown in Figure 14. In each clock cycle, an input sample is stored in a register (Figure 15a). In the subsequent clock cycle, the sample is forwarded to the first convolution with a vertical kernel, and at the same time, it is stored in the first line buffer, as shown in Figure 15b. The size of the line buffer depends on the width of the input image and is equal to $N - 1$. After each line buffer, a register is concatenated to fetch the data from the buffer for convolution. When the first line buffer and the corresponding register are filled, as illustrated in Figure 15c, the data from the register is moved to the next buffer line. The complete image row is moved between the buffers in a sample-by-sample manner in N clock cycles. The total number of line buffers depends on the kernel size and is equal to $W - 1$. After $W - 1$ rows are streamed into the convolution module, the input image row i is stored in the buffer line $(W - i)$.

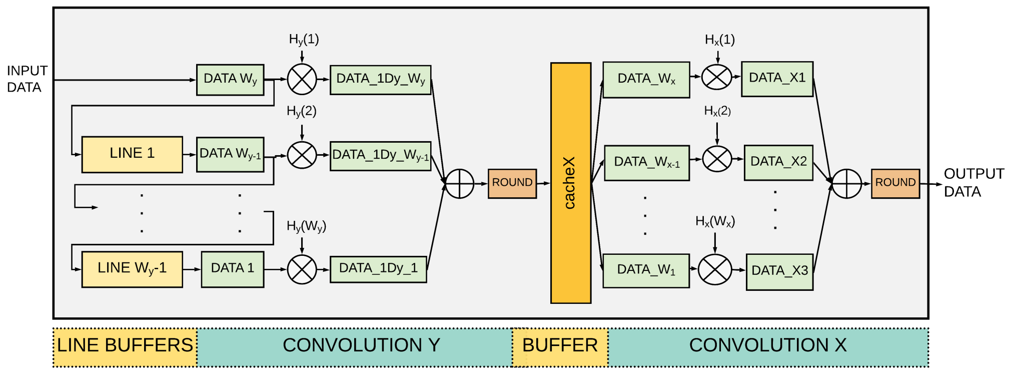
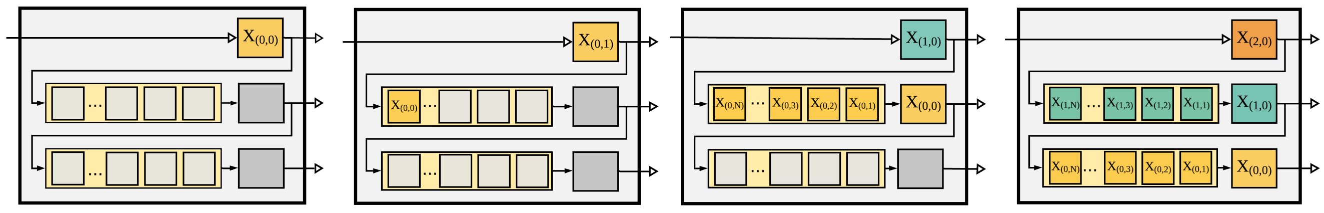


Figure 14. A block diagram for the 2-D convolution with a separable $W_y \times W_x$ kernel.

After initialization, an image sample is output from each line buffer into a register. The data elements are then multiplied with kernel elements at the corresponding kernel positions. The sum of W products is forwarded to the intermediate buffer, which stores the resulting data from the first convolution into an array of the kernel size. The array elements from the intermediate buffer are sent to the corresponding registers for multiplication with the horizontal kernel. The resulting output is computed as a rounded summation of the products of horizontal kernel elements and the results of the first convolution.



X(0,0)
X(0,1)
X(0,0)
X(1,0)
X(0,N) X(0,3) X(0,2) X(0,1)
X(0,0)
X(2,0)
X(1,N) X(1,3) X(1,2) X(1,1)
X(1,0)
X(0,N) X(0,3) X(0,2) X(0,1)
X(0,0)
(a)
(b)
(c)
(d)

Figure 15. Initialization of the line buffers.

The proposed convolution module has a fixed maximum kernel size, which is chosen to be 9×9. Additionally, the module supports kernel sizes equal to 7×7. The sizes of the array in the intermediate buffer and the corresponding registers are fixed and equal to the maximum kernel size. For kernel size 7×7, data samples from the registers $DATA\_W_y$ and $DATA\_W_{y-1}$ are multiplied with zeros. For the second 1-D convolution, two registers ($DATA\_W_x$ and $DATA\_W_{x-1}$) in the CacheX buffer are initialized with zeros.

4.3. Data Processing Pipeline

The data flow and the latency information between different stages in two iterations of the RL deconvolution are presented in Figure 16. As shown, the data processing starts three clock cycles after the input signal `enable` is received from the configuration block. The convolution module is characterized by the latency equal to $(N \times 4) \times 2 + 13$, whereas the division module has a latency of 38 clock cycles. The latency of the multiplication module is equal to either three clock cycles for the standard RL deconvolution or four clock cycles for the accelerated version. Thereafter, one valid output sample is produced per clock cycle.

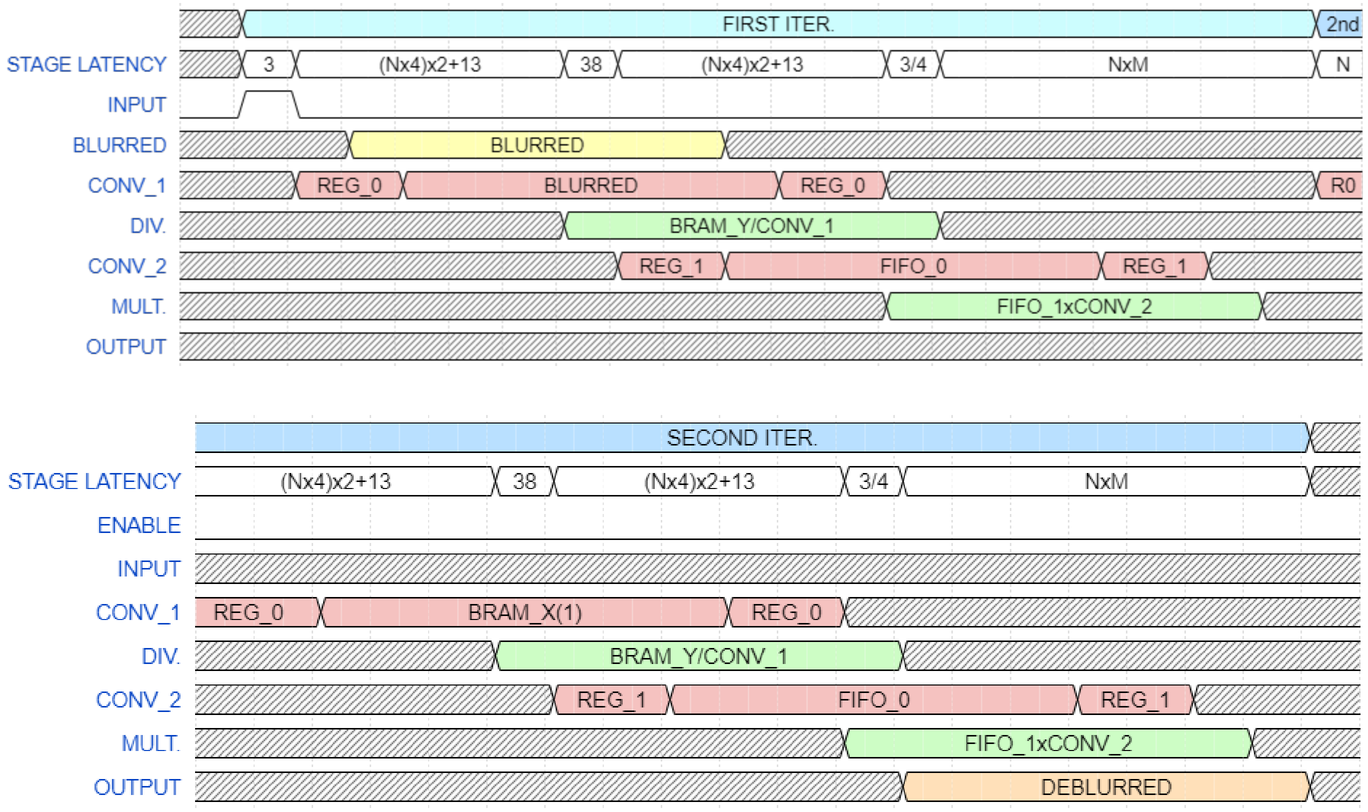


Figure 16. A timing diagram representing two iterations of the RL deconvolution algorithm.

5. Results

The proposed architecture was implemented in the Xilinx Vivado development environment [28] and described in VHSIC Hardware Description Language (VHDL). Synthesis, implementation, testing, and verification were performed on a Zedboard platform [29] with a built-in Zynq-7020 FPGA. The initial unit tests include functional verification of both the standard and the accelerated versions of RL deconvolution against the software implementation written in C programming language, which uses fixed-point representation and MP-BC boundary conditions. A verification model is presented in Figure 17. The C-code block produces both the degraded image and the restored image, which is used as a reference in the verification. Before the RL deconvolution is enabled, the user-defined parameters, number of iterations, and restoration kernel size are sent to the the *Design Under Test* (DUT) through an AXI4-Lite interface. Afterwards, the degraded image is streamed to the DUT and the restored image from the DUT is compared to the reference image in the monitor block. The test passes if the error count is equal to zero.

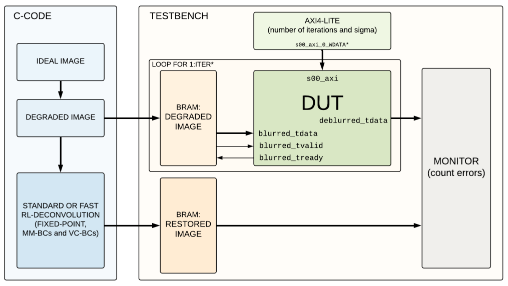


Figure 17. Verification of the RL deconvolution *Design Under Test* (DUT).

The proposed RL deconvolution module is fully verified on the Zedboard platform. The system consists of a processing system (PS) with two ARM Cortex-A9 CPU cores, a DDR3 external memory, a direct memory access core, a concatenation IP core, and a *RL deconvolution* IP, as shown in Figure 18. Communication between the external memory and the implemented core is tested by using both AXI DMA [30] or CubeDMA IP [31]. The links between the custom RL deconvolution core and DMA core for streaming the input image and the resulting image are 64-bit AXI4-Stream, whereas the communication between the DMA module and external DDR3 memory is established through an AXI4-memory mapped interface. The deconvolution core and DMA core are initialized and controlled by the PS through the 32-bit AXI4-Lite interface connected to the same interconnect. For AXI DMA, the Concat IP core connects the interrupt channels (DMA memory map to stream and stream to memory map), `s2mm_introut` to the PS interrupt request (IRQ) generator. An interrupt is sent when either the last element of the input image is streamed to the DMA by setting the `m_axis_mm2s_wlast` signal high or when the last element of the output image is sent to the PS by setting the `m_axi_s2mm_rlast` signal to one.

Data processing is performed band by band for a 3-D hyperspectral image, where 3-D HSI is stored in band-by-band order (BSQ order) in the external memory. Initialization starts by writing the desired number of iterations k into the configuration register and by deciding which version of the algorithm to run. The degraded band, $\mathbf{Y}^p$, is sent to the hardware module, and after k iterations, the restored image $\hat{\mathbf{X}}^p$ is received by the PS and is saved in the SD card. The process repeats for all the bands in the hyperspectral data set.

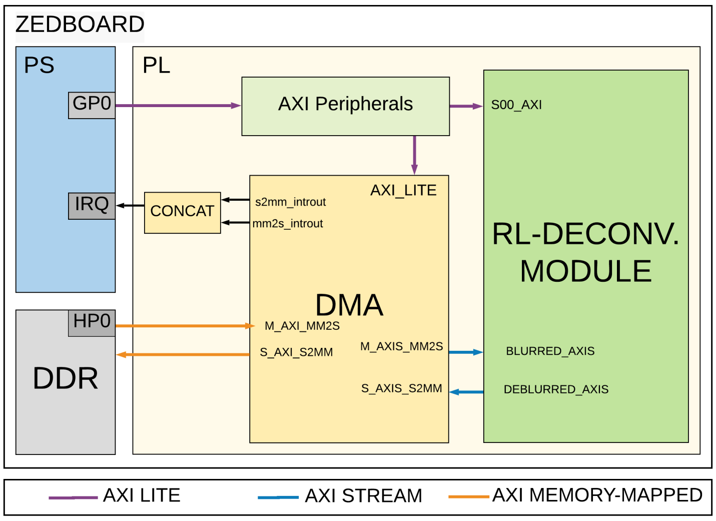


Figure 18. The overall SoC with RL deconvolution module.

5.1. Resource Utilization

The estimated resource utilization results for the proposed architecture on the Zynq-7020 FPGA were presented. The used resources in terms of Look Up Tables (LUTs), registers, and block RAMs were elaborated in more details for two fixed image widths, namely $N = 150$ and $N = 640$. The height M was chosen to be in the ranges of 43–640 samples for $N = 150$ and of 10–150 samples for $N = 640$, as presented in Figure 19. In the proposed architecture, there is a need to internally store the degraded image and the reconstructed image from each iteration, both of size $N \times M$ and 16-bit width. In addition, there are two FIFOs with the depths dependant on image width and kernel size. Block RAMs [32] are used for storing both images and FIFOs. However, LUT elements are used as distributed RAMs for storing image rows in the line buffers of the convolution modules. Initially, it was noticed that the synthesis tools inferred the block RAMs ineffectively for different image dimensions due to the array depth extension to the closest power of 2. This is solved by splitting block RAMs into several parts, which resulted in linear dependence with respect to image size. The LUT utilization increases also linearly with the image dimensions. The architecture uses 38 DSPs independent of the image size, where 2×18 DSPs are used in convolution modules and 2 DSPs are used in the multiplication module.

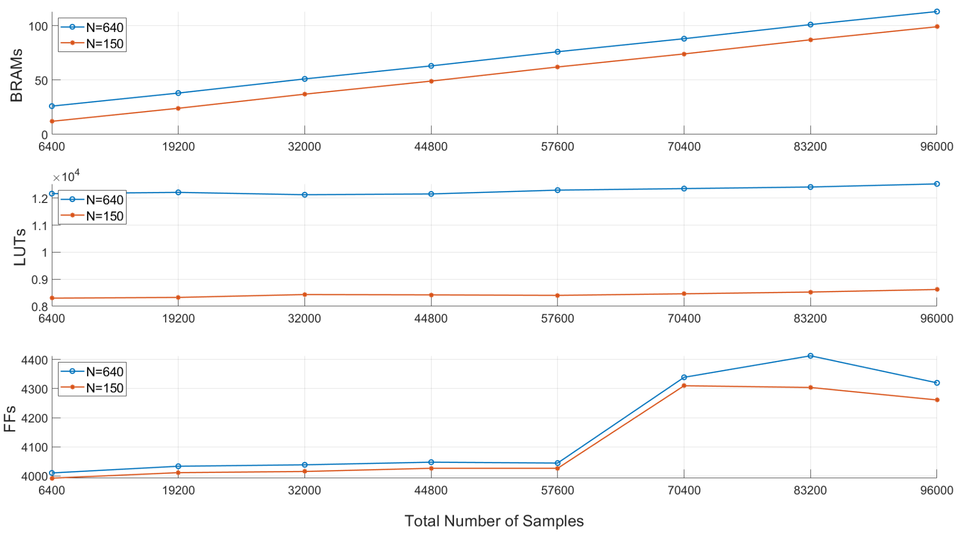


Figure 19. Resource utilization as a function of the image width equal to 640 samples (blue) and 150 samples (red).

5.2. Power Estimation

Power estimation is performed on the post-implementation design using the power estimation tools provided by Xilinx Vivado. Figure 20 shows estimated total on-chip power estimation for an image of size equal to either $640 \times M$, marked blue, or $150 \times M$, marked red. The power is estimated for the same ranges of image height M used in resource utilization.For resource utilization, the power consumption grows as image dimensions increase.

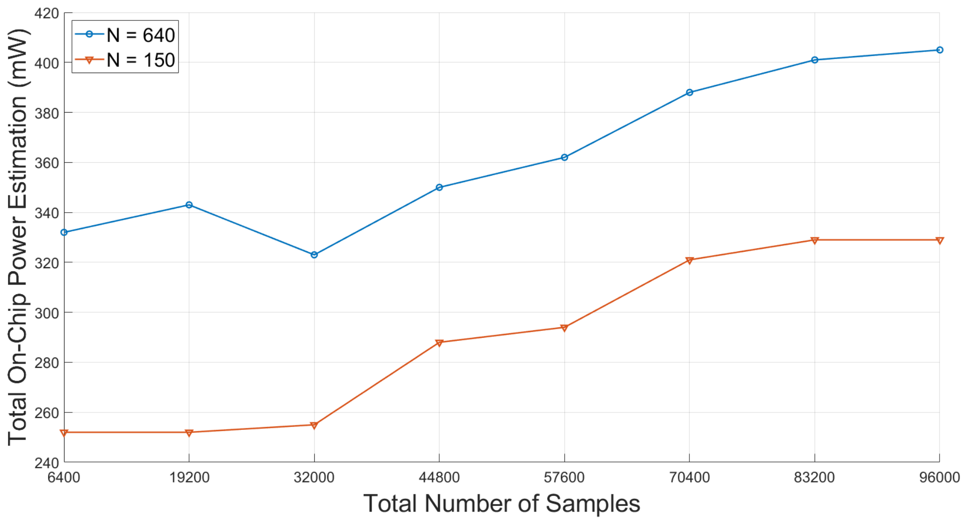
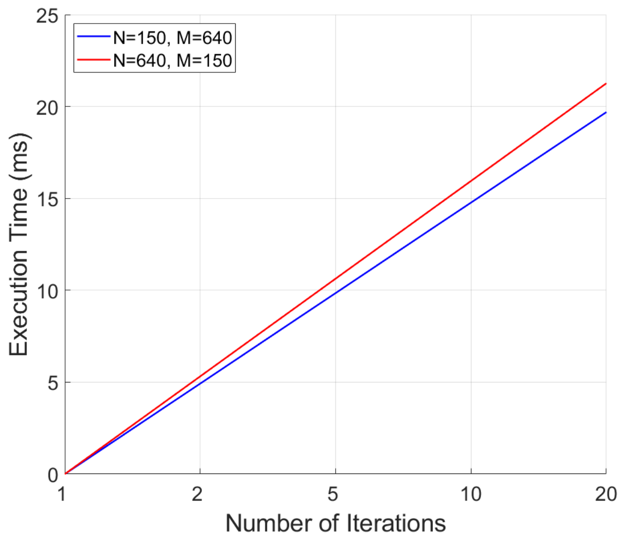


Figure 20. Power estimation for the proposed architecture using the same amount of samples for image with width equal to 640 and 150, shown in blue and red lines, respectively.

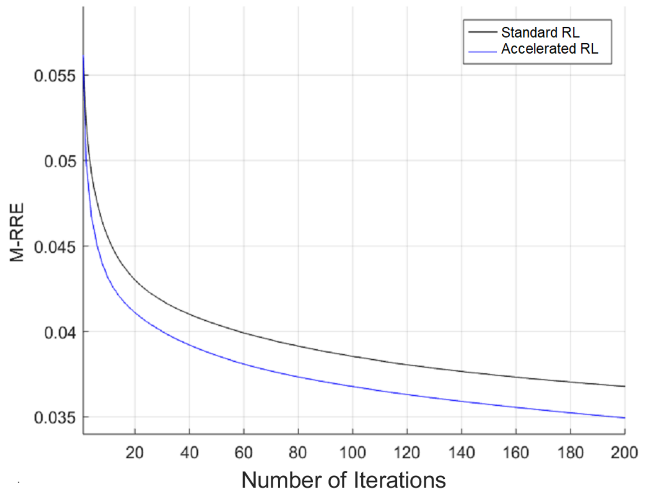
5.3. Execution Time

In the implementation of the standard RL deconvolution algorithm, one iteration of the RL deconvolution algorithm takes a total of $32 + N \times (16 + M)$ clock cycles. For the accelerated algorithm with β=2, one additional clock cycle is required. The maximum operating frequency of the proposed implementation is 110 MHz for image of size 150×640 and 103 MHz for image of size 640×150. Execution time plots for both image sizes as a function of number of iterations are shown in Figure 21a. The difference in the number of iterations required for standard and accelerated RL deconvolution is presented in Figure 21b. The same M-RRE values are achieved twice as fast for the accelerated RL deconvolution compared to the standard RL deconvolution.

With respect to execution time, the proposed architecture is compared with state-of-the-art implementations, our previous work on the HW/SW codesign implementation of RL deconvolution algorithm [17], and a software-only implementation of RL deconvolution tested on the target SoC. Table 5 shows a comparison between the different implementations. The proposed architecture outperforms the previously implemented SW-only and HW/SW codesign solutions. For image size 150×640, a speed-up by a factor of 61.3 and 26.2 is achieved compared to the SW-only implementation and HW/SW codesign implementation, respectively. The proposed architecture compares well with the state-of-the-art solutions. The architecture in [14] is the most similar to the proposed architecture in terms of algorithm implementation, i.e., the kernel is assumed to be space-invariant and the images are extended before performing the convolution. The computed throughput in [14] based on the reported computation time, image size, and number of iterations is 52.5 samples/s, whereas the throughput of the proposed architecture is 97.96 samples/s for image size 150×640, resulting in a speed-up factor of 1.9 compared to the [14]. In addition, the proposed architecture implements the accelerated RL deconvolution, and consequently, the total number of iterations can be reduced.



(**a**)



(**b**)

Figure 21. Estimated execution time as a function of number of iterations (**a**) and M-RRE as a function of number of iterations (**b**).

Table 5. Comparison between several RL deconvolution implementations.

	Iter.	**Image Size**	**PSF**	**Time (ms)**	**Freq. (MHz)**	**Throughput (Samples/s)**
[16]	15	640×480	<10	40	-	115.2
[14]	10	800×525	9×9	80	61	52.5
[13]	60	64×64	13×13	78	100	3.1
SW-Only	1	150×480	9×9	60.1	100	1.6
HW/SW codesign [17]	1	150×480	9×9	25.7	100	3.7
Proposed work, *SoC	1	150×640	9×9	0.98	100	97.96
Proposed work, *SoC	1	640×150	9×9	1.06	100	90.6
Proposed work, max freq.	1	150×640	9×9	0.87	110	110.3

6. Conclusions

A real-time FPGA implementation of the RL deconvolution algorithm used for reducing HSI degradation is presented in this paper. The HSI degradation is modeled as a convolution of the 2-D cross-channel-independent images and a point spread function. Both the standard and accelerated RL deconvolution versions are implemented, and the choice of the algorithm version is available at run-time. The proposed architecture is optimized with respect to communication with the external memory by using extensive internal storage for the intermediate data. The flexibility of the proposed deconvolution core is introduced through generic image size parameters prior to synthesis and the available update of the configuration registers controlled by the CPU with the number of iterations, algorithm type, and the kernel size parameters. This results in a more general solution compared to some of the state-of-the-art implementations, which support only a limited number of iterations. The flexibility of the design in terms of the kernel size and number of iterations, and the speed-up in the execution time are introduced compared to the state-of-the-art works, targeting accelerated implementations of the RL deconvolution algorithm.

Author Contributions: Conceptualization, K.A. and M.O.; methodology, K.A.; software, K.A.; validation, K.A.; investigation, K.A. and M.O.; writing–original draft preparation, K.A.; writing—review and editing, M.O.; visualization, K.A. and M.O.; supervision, M.O.; funding acquisition, M.O. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Norwegian Research Council through the MASSIVE project (grant no. 291270959).

Institutional Review Board Statement: not applicable.

Informed Consent Statement: not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Sample Availability: Samples of the compounds are available from the authors.

References

1. Smith, R.B. *Introduction to Hyperspectral Imaging*; MicroImages: Lincoln, NE, USA, 2006.
2. Shippert, P. Introduction to Hyperspectral Image Analysis. *Online J. Space Commun.* **2003**, *3*, 13.
3. Richardson, W.H. Bayesian-Based Iterative Method of Image Restoration. *J. Opt. Soc. Am.* **1972**, *62*, 55–59. doi:10.1364/JOSA.62.000055.
4. Lucy, L.B. An iterative technique for the rectification of observed distributions. *Astron. J.* **1974**, *79*, 745–754. doi:10.1086/111605.
5. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA 2002.
6. Dines, K.; Kak, A. Constrained least squares filtering. *IEEE Trans. Acoust. Speech Signal Process.* **1977**, *25*, 346–350. doi:10.1109/TASSP.1977.1162965.
7. Hunt, B.; Kubler, O. Karhunen-Loeve multispectral image restoration, part I: Theory. *IEEE Trans. Acoust. Speech Signal Process.* **1984**, *32*, 592–600. doi:10.1109/TASSP.1984.1164363.
8. Galatsanos, N.; Chin, R. Digital restoration of multi-channel images. *IEEE Int. Conf. Acoust. Speech Signal Process.* **1987**, *12*, 1244–1247. doi:10.1109/ICASSP.1987.1169796.
9. Galatsanos, N.P.; Katsaggelos, A.K.; Chin, R.T.; Hillery, A.D. Least squares restoration of multichannel images. *IEEE Trans. Signal Process.* **1991**, *39*, 2222–2236. doi:10.1109/78.91180.
10. Henrot, S.; Soussen, C.; Brie, D. Fast Positive Deconvolution of Hyperspectral Images. *IEEE Trans. Image Process. Publ. IEEE Signal Process. Soc.* **2012**, *22*. doi:10.1109/TIP.2012.2216280.
11. R. Hunt, B. Super-resolution of images: Algorithms, principles, performance. *Int. J. Imaging Syst. Technol.* **2005**, *6*, 297–304. doi:10.1002/ima.1850060403.
12. Jemec, J.; Pernuš, F.; Likar, B.; Bürmen, M. Deconvolution-based restoration of SWIR pushbroom imaging spectrometer images. *Opt. Express* **2016**, *24*, 24704–24718. doi:10.1364/OE.24.024704.
13. Wang, Z.; Weng, K.; Cheng, Z.; Yan, L.; Guan, J. A co-design method for parallel image processing accelerator based on DSP and FPGA. In *MIPPR 2011: Parallel Processing of Images and Optimization and Medical Imaging Processing*; International Society for Optics and Photonics: Guilin, China, 2011; Volume 8005, p. 800506.
14. Anacona-Mosquera, O.; Arias-García, J.; Muñoz, D.M.; Llanos, C.H. Efficient hardware implementation of the Richardson-Lucy Algorithm for restoring motion-blurred image on reconfigurable digital system. In Proceedings of the 2016 29th Symposium on Integrated Circuits and Systems Design (SBCCI), Belo Horizonte, Brazil, 29 August–3 September 2016; pp. 1–6. doi:10.1109/SBCCI.2016.7724056.
15. Sims, O. Efficient Implementation of Video Processing Algorithms on FPGA. Ph.D. Thesis, University of Glasgow, Glasgow, UK, 2007.
16. Carrato, S.; Ramponi, G.; Marsi, S.; Jerian, M.; Tenze, L. FPGA implementation of the Lucy-Richardson algorithm for fast space-variant image deconvolution. In Proceedings of the 2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA), Zagreb, Croatia, 7–9 September 2015; pp. 137–142. doi:10.1109/ISPA.2015.7306047.
17. Avagian, K.; Orlandić, M.; Johansen, T.A. An FPGA-oriented HW/SW Codesign of Lucy-Richardson Deconvolution Algorithm for Hyperspectral Images. In Proceedings of the 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 10–14 June 2019; pp. 1–6.
18. Bertero, M.; Boccacci, P.; Desidera, G.; Vicidomini, G. Image deblurring with Poisson data: From cells to galaxies. *Inverse Probl.* **2009**, *25*, 123006. doi:10.1088/0266-5611/25/12/123006.
19. Biggs, D.S.; Andrews, M. Acceleration of iterative image restoration algorithms. *Appl. Opt.* **1997**, *36*, 1766–1775. doi:10.1364/AO.36.001766.
20. Meinel, E.S. Origins of linear and nonlinear recursive restoration algorithms. *J. Opt. Soc. Am. A* **1986**, *3*, 787–799. doi:10.1364/JOSAA.3.000787.
21. Lanteri, H.; Roche, M.; Cuevas, O.; Aime, C. A general method to devise maximum-likelihood signal restoration multiplicative algorithms with non-negativity constraints. *Signal Process.* **2001**, *81*, 945–974. doi:10.1016/S0165-1684(00)00275-9.
22. Almeida, M.; Figueiredo, M. Deconvolving Images With Unknown Boundaries Using the Alternating Direction Method of Multipliers. *IEEE Trans. Image Process.* **2013**, *22*, 3074–3086. doi:10.1109/TIP.2013.2258354.
23. Fang, H.; Luo, C.; Zhou, G.; Wang, X. Hyperspectral Image Deconvolution with a Spectral-Spatial Total Variation Regularization. *Can. J. Remote Sens.* **2017**, *43*, 384–395. doi:10.1080/07038992.2017.1356221.

24. Zhu, F.; Wang, Y.; Fan, B.; Meng, G.; Pan, C. Effective Spectral Unmixing via Robust Representation and Learning-based Sparsity. *arXiv* **2014**, arXiv:1409.0685.
25. X. Inc. Divider Generator (v5.1). 2016. Available online: https://www.xilinx.com/support/documentation/ip_documentation/div_gen/v5_1/pg151-div-gen.pdf (accessed on 27 May 2019).
26. X. Inc. FIFO Generator (v13.1). 2017. Available online: https://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v13_1/pg057-fifo-generator.pdf (accessed on 27 May 2019).
27. Orlandić, M.; Svarstad, K. An adaptive high-throughput edge detection filtering system using dynamic partial reconfiguration. *J. -Real-Time Image Process.* **2018**, *16*, 2409–2424. doi:10.1007/s11554-018-0753-4.
28. X. Inc. Vivado Design Suite-HLx Editions. 2015. Available online: https://www.xilinx.com/products/design-tools/vivado.html (accessed on 27 May 2019).
29. Avnet. ZedBoard, Hardware User's Guide. 2014. Available online: http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf (accessed on 16 December 2018).
30. ARM, A. AXI DMA v7.1 LogicCore IP Product Guide. 2018. Available online: https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf (accessed on 10 December 2018).
31. Fjeldtvedt, J.; Orlandić, M. CubeDMA–Optimizing three-dimensional DMA transfers for hyperspectral imaging applications. *Microprocess. Microsyst.* **2019**, *65*, 23–36.
32. X. Inc. 7 Series FPGAs Memory Resources. 2017. Available online: https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf (accessed on 20 June 2019).