ISSN 2347 - 3983
Volume 8. No. 2, February 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter21822020.pdf
https://doi.org/10.30534/ijeter/2020/21822020

Real Time Hardware Co-Simulation for Blind Image Separation Algorithm Using ZYNG 7000 & Xilinx System Generator

Mohcin Mekhfioui[1], Rachid Elgouri[2], Amal Satif[1], Laamari Hlou[1]
[1]Laboratory of Electrical Engineering and Energy Systems, Faculty of Science, Ibn Tofail University, Morocco, mohcin.mekhfioui@uit.ac.ma
[2]Laboratory of Electrical Engineering and Telecommunications Systems, ENSA, Ibn Tofail University, Morocco

ABSTRACT

The implementation of blind image processing required a solid background knowledge of the design and description languages of the material. This paper describes an effective method for implementing real-time blind image separation algorithms on hardware, without needing a comprehensive knowledge of hardware design and description languages(VHDL).This study concerns the separation of a mixture of images, with no prior information on the images or on the way they have been mixed together. It describes the methodology for implementing real-time DSP applications on FPGAs and the concept of hardware/software co-simulation of the blind image separation algorithm using MATLAB Simulink and Xilinx System Generator (XSG). The performance of efficient architectures is implemented on the ZYBO Z7 Zynq-7020 FPGA.

Key words : Blind Image Separation, Image Processing, Field Programmable Get Array (FPGA), VHDL, Xilinx SystemGenerator (XSG).

1. INTRODUCTION

The blind image separation consists in estimating a set of unknown images called sources from known images called observations (or mixtures). Observations are mixtures of source images received by sensors (cameras, radars...). Source separation is considered complete when each source image is returned at a precise scalar factor, it is not possible to predict on which output of the separator the unknown image is returned. Image separation is carried out by exploiting the information relating mainly to the images themselves [1].
This paper presents the hardware co-simulation and implementation of the Blind image separation algorithm on FPGA.

Currently, the use of FPGAs in research and development of digital systems applied to specific tasks is increasing. This is due to the advantage that FPGAs have over other programmable devices. These advantages include: high clock rate, high number of operations per second, low cost, parallel processing, security, ability to interact with high or low interfaces and intellectual property (IP) preservation [2]. The Xilinx System Generator for Digital Signal Processing (DSP) is a tool that operates at a higher degree of abstraction, it is useful for those with familiar of MATLAB and Simulink.

Therefore, this paper presents an efficient approach for implementing a real-time blind image separation algorithm on FPGAs without requiring a detailed knowledge of hardware design and description languages. The concept consists of using the Xilinx System Generator for DSP, which integrates with the MATLAB-based Graphics environment and eliminates the need for HDL text programming.

This workis structured as follows: Section II presents the principle of the blind image separation and its mathematical model, the Xilinx System Generator and the ZYBO board, Section III presents proposed work and study case. Section III describes the hardware co-simulation, Implementation, and results. Finally the Section IV concludes this paper.

2. MATERIAL AND METHODOLOGY

2.1 Blind Image Separation Problem

The main objective of the blind image separation is to estimate a set of unknown source (images) s(t) using only the mixed images obtained by a series of sensors x(t), also called "observations" Fig. 1[3].

The term Blind denotes the fact that there is no a priori information about the sources or how they have been mixed.

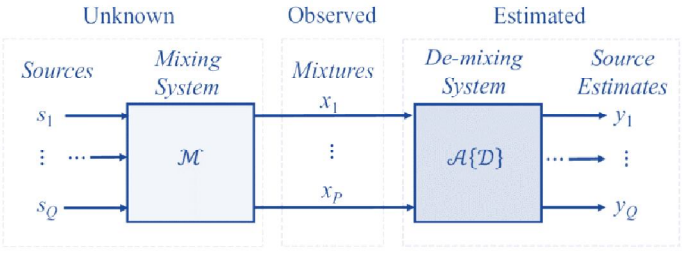

Figure 1: Principle of the Blind Image Separation

2.2 Mathematical Model

We suppose that N images $s_1, \ldots, s_i, \ldots\ldots, s_N$ of size (m, n) are combined and those M linear mixtures of these original

images are observed. These last mixtures can be modelized by the following linear system[4]:

$$x(m,n) = \mathrm{A}s(m,n) + v(m,n) \quad (1)$$

Where

$x(m,n) = [x_1(m,n),\dots,x_N(m,n)]^T$ is an M×1 vector of image pixel mixing;

A is the M×N mixing matrix of the row of the complete column (i.e., M≥N);

$s(m,n) = [s_1(m,n),\dots,s_N(m,n)]^T$ is an N×1 image source vector consisting of the array of associated pixels of the source images, and the superscript T denotes the transpose operator;

$v(m,n) = [v_1(m,n),\dots,v_N(m,n)]^T$ is the M×1 noise vector that affects any pixel in the image mixture.

The matrix form of equation (1) is written as follows:

$$X(t) = HS(t) \quad (2)$$

Where H is a mixture matrix of size $N \times M$

It isabout finding in the ideal case the matrix W of size $N \times M$ which inverts the mixture and provides the output vector:

$$y(m,n) = W\,x(m,n) = W\,H\,s(m,n) \approx s(m,n) \quad (3)$$

The estimated images are given by the vector $s(m,n)$ and their corresponding projections for the different cameras are given by the estimated matrix: $H = W^{-1}$[5].

2.3 Xilinx System Generator

Xilinx System Generator is a DSP design tool for linking ISE and Matlab that enables the use of MathWorks Model-Based Simulink for FPGA design. Xilinx System Generator, which simplifies FPGA hardware design, was the first to launch the idea of compiling an FPGA program from MATLAB and the Simulink model. It enables system modeling and automatic code generation from MATLAB and Simulink. When designing the Simulink model, the system generator blocks can be used just like the other Simulink blocks. In addition, the designer can also use the two native Simulink blocks and the Xilinx System Generator blocks at the same time[6],[16]. This tool also allows hardware co-simulation used to test user-created cores on the target hardware simultaneously with the model present in the Simulink environment Fig.2.

The Xilinx System Generator (XSG) offers the advantages:

- Less time in the design and development
- Automatic generation of the target synthesizable code
- Easy to modify the design architecture and requirements
- Reusability and flexibility

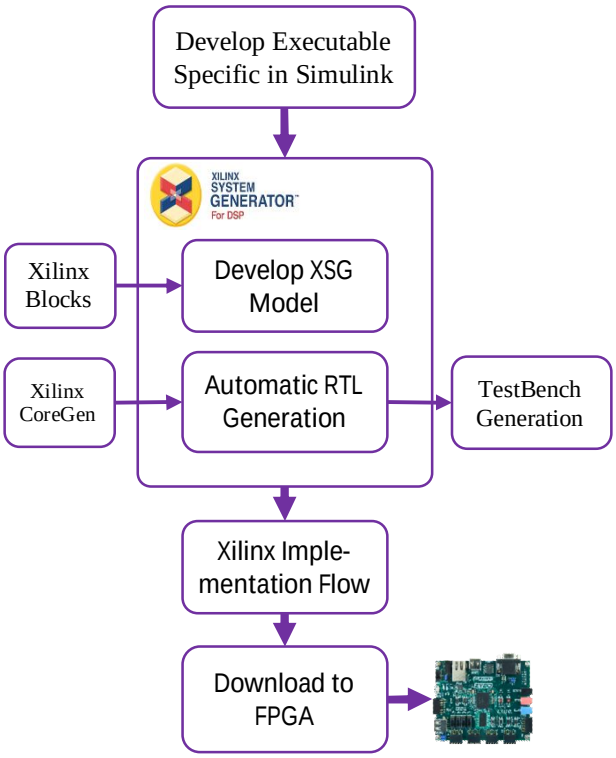

Figure 2: Design flow for Xilinx system generator

2.4 Overview of the ZYNQ family

The Zynq is particularly interesting in this context because it incorporates both a dual-core ARM cortex A9 processor, interrupt handler, converter Analog-to-Digital and other devices, in an area called "Processing System" (PS). This component has a second area called "Programmable". Logic" (PL), consisting of programmable logic cells, memories and blocks for digital signal processing, Fig. 3.
The embedded system, made by our students, requires the use of the processor ARM associated with two asynchronous serial (UART) devices to which is added a "homemade" device made by our students. It is a frequency meter, created at based on the VHDL hardware description language, and encapsulated as an IP block ("Intellectual Property"). This is connected to the processor via an AXI-Lite bus. This bus allows the communication between the "PS" processor system and the realized IP block, and integrated in the "PL" programmable logic area.[7],[17].

3. Proposed Work

The objective of this work is to present a comparison between the different algorithms of the Blind image separation based

on the calculation of the Performance Index (PI), the optimal algorithm obtained will be implemented in the ZYBO Z7 Zynq-7020 board to testits performance in real time. By using the fewest possible System Generator Blocks in FPGA use Xilinx System Generator for image processing as shown in proposed block diagram, fig 4.

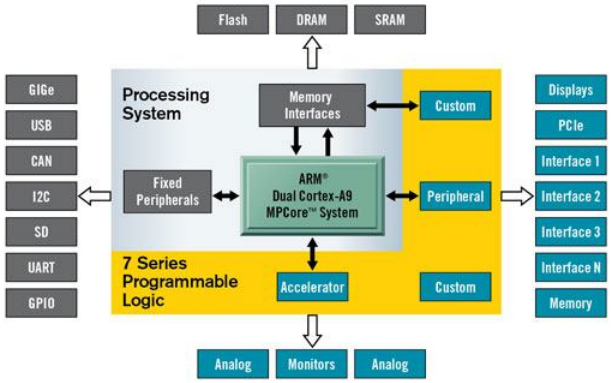

Figure 3: ZYBO family architecture

The separation method will be implemented in the FPGA hardware to meet real-time applications. The implementation will be carried out in a prototyping environment, Simulink and Xilinx System Generator through three main phases:

- Image pre-processing blocks;
- Image mixing blocks;
- Image processing using Blind Image Separation Algorithm;
- Image post-processing blocksto recreating image;

Image source and image viewer are matlab/Simulink blocks, using these blocks, the image can be given as input and the output image can be depicted on the video viewer block.

The pre-processing and post-processing units transmit the input image in the appropriate blind image processing standard for the next unit. They are designed by the XSG blocks. The Blind Image Separation algorithm is designed by Xilinx blocks.

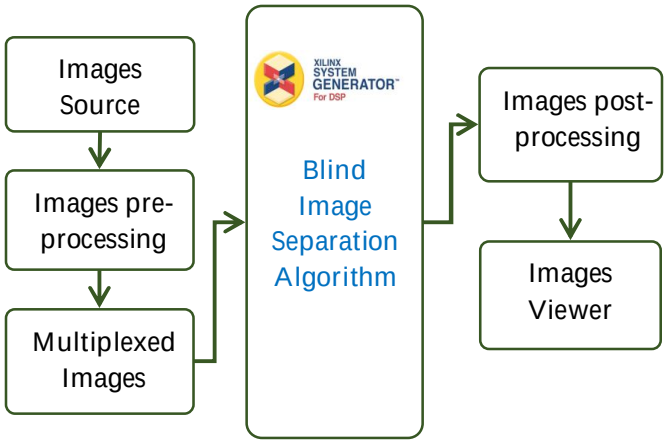

Figure 4: Design flow of hardware implementation of Blind Image Separation algorithm

4. SIMULATION RESULTS

All simulations are performed on “Lena.png” and “fruits.png” images in 512×512. The number of mixtures observed is M=2. The algorithms are developed on the MATLAB environment.In our study we used the TOOLBOX ICALAB for image processing (version 2.0) [8], [18] grouping several algorithms of the blind image separation that we tested (15 algorithms tested) to determine the algorithm that gives the higher performance. However, tests have shown that only 7 algorithms are stable:

- ✓ SIMBEC: SIMultaneous Blind Extraction using Cumulants [9];
- ✓ SOBI-RO: Second Order Blind Identification with Robust Orthogonalization[10];
- ✓ SANG: Self Adaptive Natural Gradient with nonholonomic constraints [11];
- ✓ FPICA: Fixed Point ICA [12]
- ✓ ERICA : Extended Robust ICA – based on Cumulants [13];
- ✓ BSE-K: Blind Source Extraction optimization of Kurtosis [14]
- ✓ UNICA: Unbiased quasi Newton algorithm for ICA [15];

The results obtained by calculating the Signal-to-interference (SIR) and the performance index of separability (PI) of these algorithms are presented in Table 1.

Table 1: The performance index obtained for the algorithms tested

Algorithm	PI
SIMBEC	0.007
FPICA	0.011
SANG	0.012
ERICA	0.044
BSE-K	0.096
SOBI-RO	0.129
UNICA	0.135

These results show that the SIMBEC algorithm (SIMultaneous Blind Extraction using Cumulants) present a separation technique that suits more our case.

The Fig. 5 shows a design of blind image separation by the SIMBEC algorithm using the Matlab / Simulink and Xilinx System Generator (XSG).

Fig. 6 shows the origin images (s_1 and s_2), the multiplexed images (x_1 and x_2) and the output images (y_1 and y_2) for the SIMBEC algorithm.

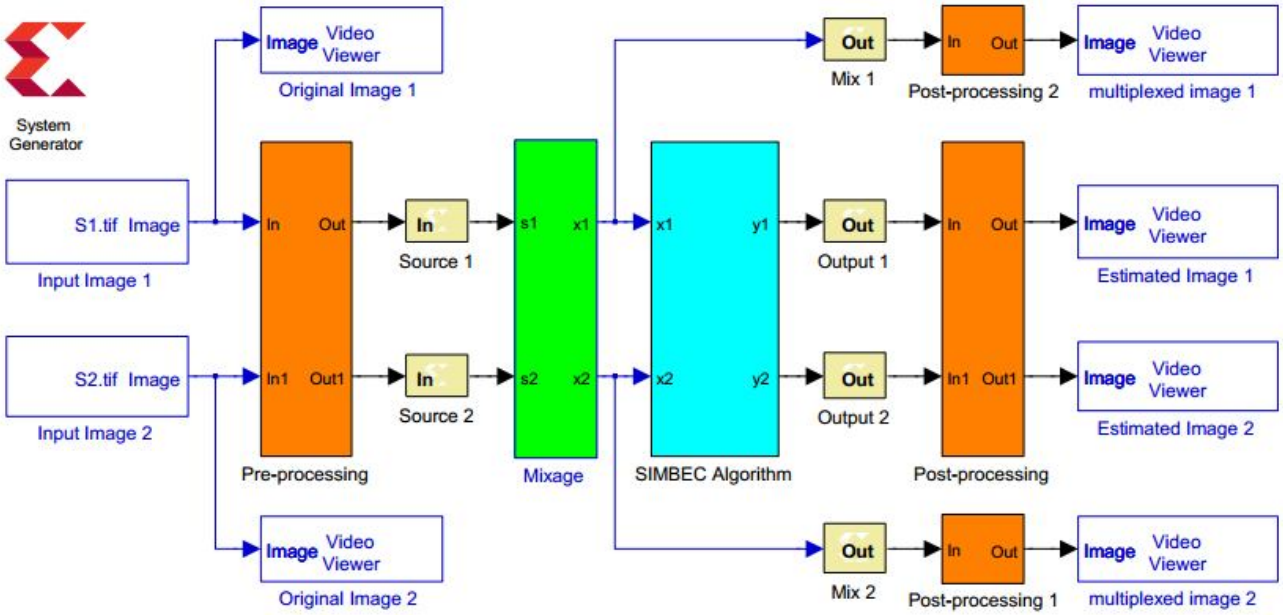

Figure 5: Model simulation for Separation of tow image

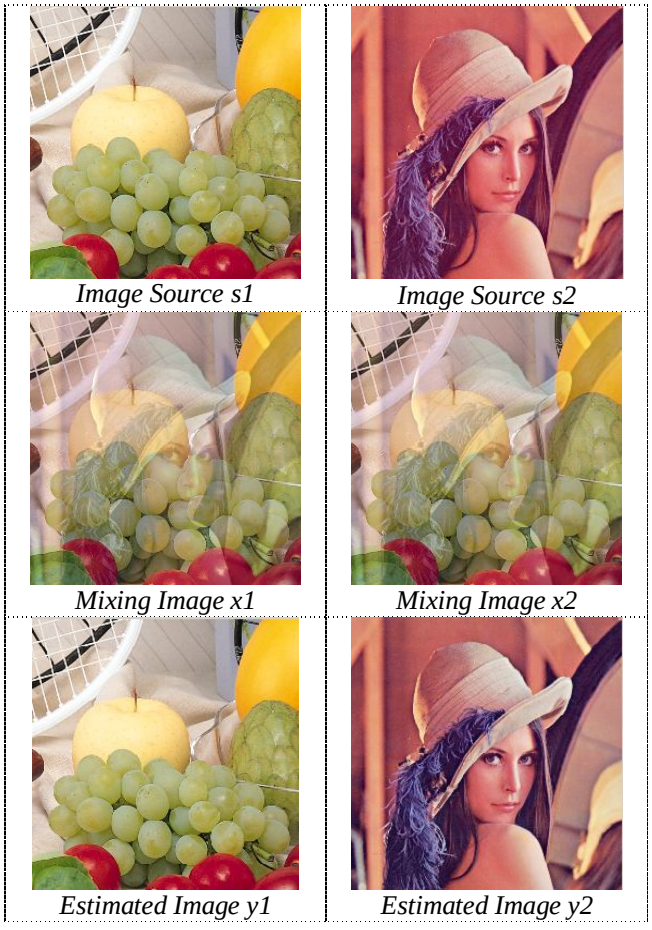

Figure 6: Simulation results of the SIMBEC algorithm

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 Hardware Co-Simulation

The blind image separation algorithm is designed in MATLAB/Simulink and implemented on the Zynq-7020 board. The separation module in Figure 7 is behaviorally equivalent to thousands of lines of the VHDL program code. With hardware co-simulation, it is possible to select a subsystem from a system generator model to execute in hardware, while the part of the model is simulated on a computer. The hardware co-simulation block has been generated easily and the processing speed and hardware resources were achieved by using the ISEDesign Suite implementation tool. All implementation steps start by generating the Simulink model for the system using Xilinx System Generator in MATLAB/Simulink until the file download '*.bit' on the FPGA card, illustrated in the Fig. 2.

5.2 Experimental results

In real application, the SIMBEC algorithm has been implemented and tested with a computer to send two basic images (s1 and s2) of the same size. These two images will be mixed with a matrix A in the Zynq-7020 board to produce the two mixing images x1 and x2. Fig. 8

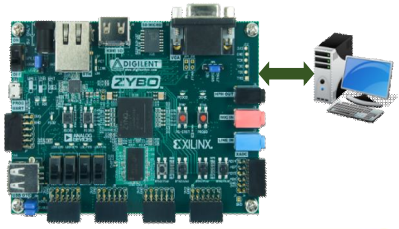

Figure 8: Hardware setup for system separation

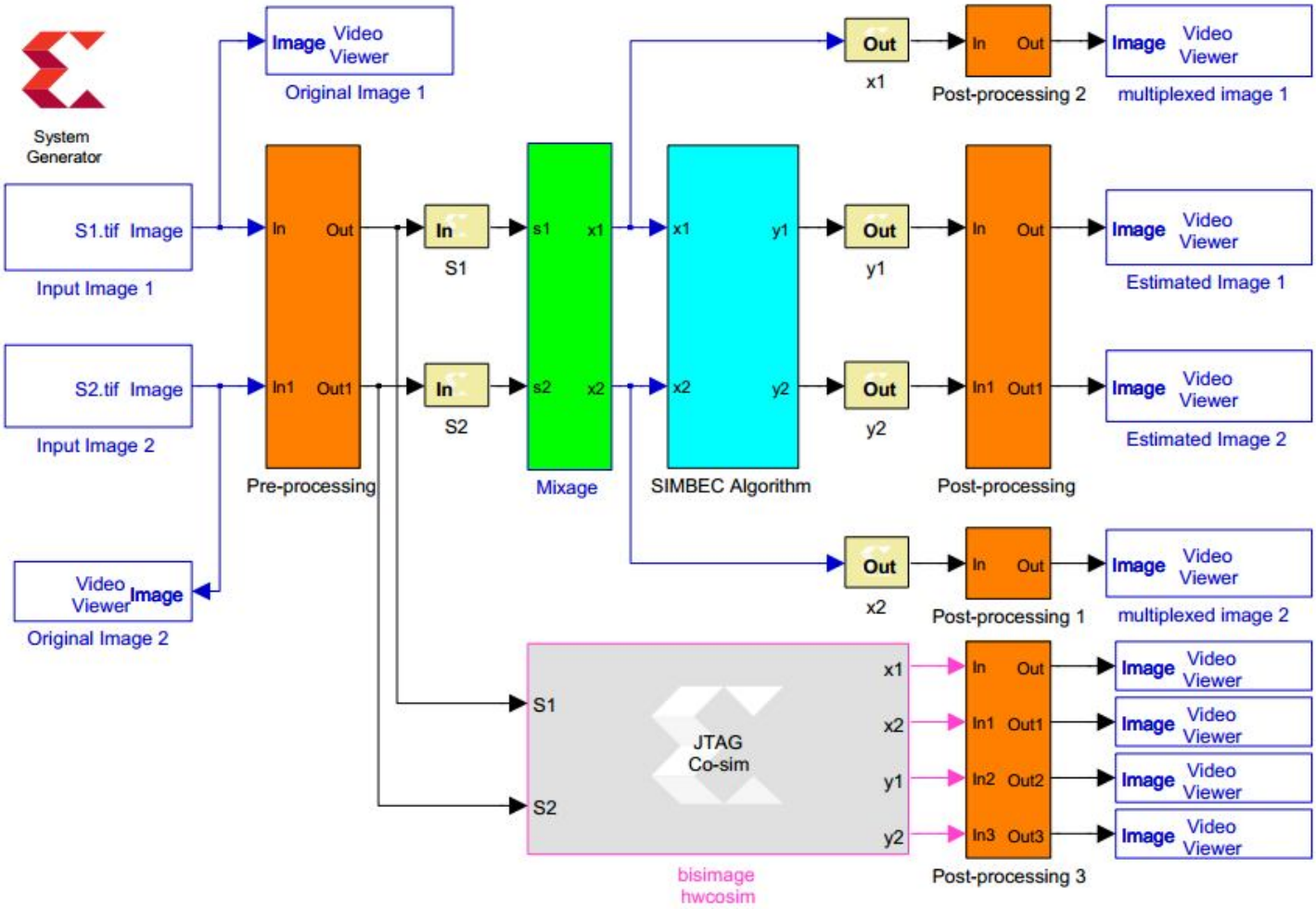

Figure 7: Model Hardware co-simulation for Separation of tow images

The different input and output images of the SIMBEC algorithm implemented in the Zynq-7020 board are shown in Fig. 9. To observe the different images (source images, blended images and estimated images), we used the Matlab Simulink. The results of the implementation verify that the behavior and graphics of our implemented model are similar to the results obtained by simulation on Simulink.

Image Source s1 *Image Source s2*

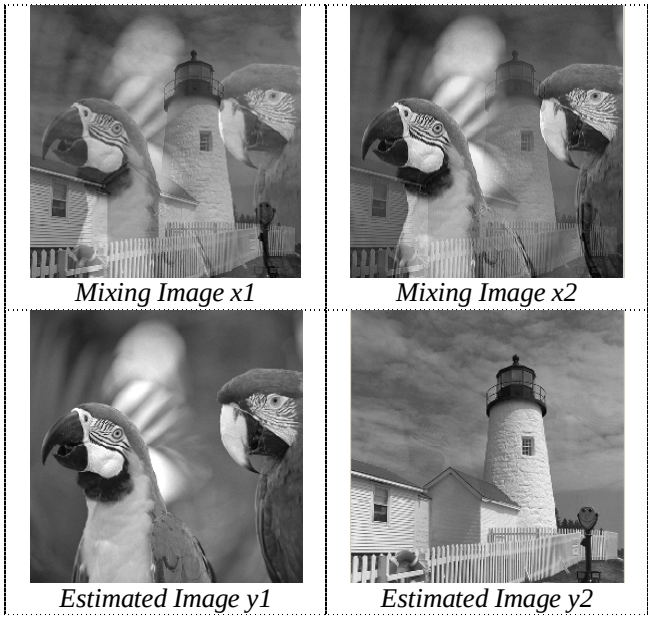

Figure 9: Result of Real time implementation using the SIMBEC algorithm

This study aims to reproduce the performance of the implementation of the SIMBEC algorithm (SIMultaneous Blind Extraction using Cumulants) as expected in Matlab Xilinx System Generator. The experiment carried out in this paper is still in the onset and provide a background to be developed in future works concerning the performance of the SIMBEC algorithm in real time, which is a promising area of research.

6. CONCLUSION

The objective of this paper was to demonstrate the use of System Generator for implementing the Blind image Separation algorithms on a FPGA. Since it does not require aprevious knowledge of hardware design methodologies, also has the advantage of using large memory, and embedded multipliers. Advances in FPGA technology with the development of intelligent and robust tools for simulation, synthesis and modeling have made the FPGA a most practical platform. Flexible and reasonable use of DSP building blocks provided in theXSG can easily turn the flowchart of algorithm into a corresponding implement on FPGA.

The results indicate the Xilinx System Generator tool offers an easy and efficient method for implementing Simultaneous Blind Extraction using Cumulants algorithm into FPGA board. The design was implemented on ZYBO Z7 Zynq-7020 devices and their utilization summaries are showed.

REFERENCES

[1] C. J. Ortiz-Echeverri, S. Salazar-Colores, J. Rodríguez-Reséndiz, et R. A. Gómez-Loenzo, « **A New Approach for Motor Imagery Classification Based on Sorted Blind Source Separation, Continuous Wavelet Transform, and Convolutional Neural Network** », *Sensors*, vol. 19, n° 20, oct. 2019, doi: 10.3390/s19204541.

[2] M. Alareqi, R. Elgouri, K. Mateur, A. Zemmouri, A. Mezouari, et L. Hlou, « **Optimization of high-level design edge detect filter for video processing system on FPGA** », *Intelligent Systems and Computer Vision (ISCV)*, 2017, p. 1-8, doi: 10.1109/ISACV.2017.8054900.

[3] M. Y. Abbass et H. Kim, « **Blind image separation using pyramid technique** », *EURASIP J. Image Video Process.*, vol. 2018, n° 1, p. 38, mai 2018, doi: 10.1186/s13640-018-0276-8.

[4] W. Souidene, A. Aissa-El-Bey, K. Abed-Meraim, et A. Beghdadi, « **Blind Image Separation using Sparse Representation** », in *2007 IEEE International Conference on Image Processing*, 2007, vol. 3, p. III-125-III-128, doi: 10.1109/ICIP.2007.4379262.

[5] M. Mekhfioui, R. Elgouri, A. Satif, A. Benahmed, E. Hamzaoui, H. Abdelkader and L. Hlou, « **A Comparative Study and Implementation of Blind Source Separation Algorithm using MATLAB and TMS320c6713 DSK** », *Journal of Engineering and Applied Sciences*, vol. 15, n° 5, p. 1074-1081, dec. 2019, doi: 10.36478/jeasci.2020.1074.1081.

[6] Xilinx, « **Xilinx System Generator User Guide** », *Xilinx System Generator User Guide*, 30-10-2019. [Online]. Available: www.xilinx.com. [Accessed: 20-01-2020].

[7] Digilent Inc, « **ZYBO Reference Manual** », *ZYNQ family*, 14-02-2014. [Online]. Available: www.digilentinc.com. [Accessed: 22-01-2020].

[8] A. Cichocki, S. Amari, K. Siwek, et T. Tanaka, « **ICALAB toolbox for Image Processing** », Japan, 2004.

[9] S. Cruces, A. Cichocki, et S. Amari, « **Criteria for the simultaneous blind extraction of arbitrary groups of sources** », in *Proc. Int. Conf. on ICA and BSS*, 2001, p. 740-745.

[10] A. Belouchrani et A. Cichocki, « **Robust whitening procedure in blind source separation context** », *Electron. Lett.*, vol. 36, n° 24, p. 2050-2051, nov. 2000, doi: 10.1049/el:20001436.

[11] S. Amari, T. P. Chen, et A. Cichocki, « **Nonholonomic orthogonal learning algorithms for blind source separation** », *Neural Comput.*, vol. 12, n° 6, p. 1463-1484, juin 2000, doi: 10.1162/089976600300015466.

[12] A. Hyvärinen et E. Oja, « **A Fast Fixed-Point Algorithm for Independent Component Analysis** », *Neural Comput.*, vol. 9, n° 7, p. 1483-1492, juill. 1997, doi: 10.1162/neco.1997.9.7.1483.

[13] S. Cruces, L. Castedo, et A. Cichocki, « **Novel blind source separation algorithms using cumulants** », présenté à 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100), 2000, vol. 5, p. 3152-3155.

[14] A. Cichocki et S. Amari, « ***Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*** », 2002. https://doi.org/10.1002/0470845899

[15] S. Cruces, A. Cichocki, et L. Castedo, « **Blind source extraction in gaussian noise** », in *2nd International Workshop on Independent Component Analysis and Blind Signal Separation (ICA'2000)*, Finland, 2000, p. 63-68.

[16] Basa, Janus & Cu, Patrick & Malabag, Nathaniel & Naag, Luigi & Abacco, Dan & Siquihod, Mar & Madrigal, Gilfred & Tolentino, Lean Karlo. (2019). **Smart Inventory Management System for Photovoltaic-Powered Freezer Using Wireless Sensor Network**. *International Journal of Emerging Trends in Engineering Research.* 7. 393-397. 10.30534/ijeter/2019/057102019.

[17] Srivastava, Varun. (2019). **An efficient Software Source Code Metrics for Implementing for Software quality analysis.** *International Journal of Emerging Trends in Engineering Research.* 216-222. 10.30534/ijeter/2019/01792019.

[18] K, Sreelatha. (2019). **A Comprehensive review of Security Challenges for Data Deduplication and Integrity Auditing.** *International Journal of Emerging Trends in Engineering Research.* 7. 725-732. 10.30534/ijeter/2019/527112019.