FPGA HARDWARE DESIGN, SIMULATION AND SYNTHESIS FOR A INDEPENDENT COMPONENT ANALYSIS ALGORITHM USING SYSTEM-LEVEL DESIGN SOFTWARE

Alan Paulo Oliveira da Silva*, Ana Maria Guimarães Guerreiro*, Adrião Duarte Dória Neto*

*Federal University of Rio Grande do Norte,
UFRN - CT - PPgEEC,
Campus Universitário - Centro de Tecnologia - 59072-970,
Natal, RN. Brazil

Emails: alan@dca.ufrn.br, anamaria@dca.ufrn.br, adriao@dca.ufrn.br

Abstract— In this work we proposed the design, simulation and synthesis of a hardware that performs the Independent Component Analysis (ICA) in a reconfigurable hardware platform, more specifically a FPGA. The simulation of the hardware was done by models implemented in Simulink environment and the synthesis was possible through the Altera system-level design software DSP Builder, that contains specific FPGA blocks that can be synthesized in hardware. In order to validate the hardware, manually generated data and real electroencephalogram signals were used in the experiments.

Keywords— ICA, FPGA.

1 Introduction

Independent Component Analysis is a technique main applied in Blind Source Separation (BSS) problems that estimates source signals from its observed mixtures. This technique can be used in many signal process applications such as biomedical applications, noise extraction and telecommunications.

This work proposes the design, simulation and the synthesis of a reconfigurable hardware that performs the ICA technique by implementing the fixed-point algorithm, FastICA, in a Altera Cyclone II FPGA board.

Signals generated manually in Simulink and real electroencephalogram signals were used to perform the simulations while the hardware synthesis was possible through the system-level design software called DSP Builder, from Altera Corporation[1], which contains specific Altera FPGA blocks for the use in Simulink models that can be synthesized in a Altera FPGA board. The electroencephalogram signals were extracted from experiments done by the International Institute of Neuroscience of Natal Edmond and Lily Safra - IINN-ELS[2].

Some efforts have been done by researchers to work with ICA and implement in hardware attacking several problems. Some of these works are cited below. In 2001, Nordin, Hsu and Szu proposed a FPGA design of the algorithm FastICA applied in hiper-spectral (HIS) image processing. These images are generally used in problems of remote recognition of geographical areas. They translated part of the code available in the FastICA Package[3] in C language modules and made simulations (Anis Nordin and Szu, 2001).

Du and Qi proposed a new parallel implementation of ICA technique in a FPGA platform (Du and Qi, 2004) to perform dimension reduction in hyper-spectral images (HIS) with the main goal of reducing the computational time necessary to process the great amount of data present in HIS processing.

In 2005, Charoensak and Sattar proposed the generation of a Simulink model that performs the BSS problem using an ICA algorithm based in a modified Torkkola network and the synthesizing of the model on a Vixtex-E FPGA board through the Xilinx software System Generator (Charoensak and Sattar, 2005).

In 2006, Shyu and Li proposed the implementation of the FastICA algorithm embedded on a FPGA board in Hardware Description Language (HDL), including the implementation of an architecture that performs floating-point arithmetic in order to perform accurate calculations (Shyu and Li, 2006). Though the HDL design may be difficult, there is a greater control in hardware use.

In (Kuo-Kai Shyu and Lee, 2008), the work proposed in (Shyu and Li, 2006) was implmented in a Pipeline architecture, with the purpose of reaching higher sample rate and clock frequency, keeping the good results obtained in the previous work.

This work implements the FastICA in hardware using System-level software DSP Builder. Our goal is to develop the hardware design, preprocess the real data of electroencephalogram. The preprocessed data will be processed in a FPGA wich contains the systhesized hardware. This work is part of a higher system for disease

[1] http://www.altera.com
[2] http://natalneuroscience.com
[3] http://www.cis.hut.fi/projects/ica/fastica/

detection. This system is not the topic of this work.

This paper is organized as follows: in section 2 we detail the ICA technique by maximization of nongaussianity. In section 3 we detail the hardware and system-level design. In section 4 we present the obtained results and in section 5 the conclusions and future works.

2 Background on ICA Algorithm

Independent Component Analysis is a techinique mainly applied in Blind Source Separation (BSS) problem and it consists in recovering source signals from it's observed mixtures. The relationship between source signals s and observed mixtures x is given, in matrix notation, as follows:

$$x = As \tag{1}$$

where A is a full rank matrix wich is called mixing matrix.

Under some assumptions, ICA performs the BSS problem by finding an inverse linear transformation such that maximizes the statistical independence between the observed mixtures, which would necessarily correspond to the original sources (Aapo Hyvärinen, 2001). In practice, ICA finds an unmixing matrix W^* so that $y = W^*x$ and $\parallel y - s \parallel = min$.

2.1 Preprocessing in ICA

It is highly recommended that some preprocessing are made before applying the ICA algorithm in order to simplify the estimation of the sources. The first step in preprocessing is called *Centering* and consists in subtracting from each observed mixture it's mean.

The second step is called *Whitening* and consists in linearly transform of the centered observed mixtures, so that we obtain new vectors which are *white*. The components of a whitened vector are uncorrelated and their variances equals to unity. This means that the covariance matrix of whitened data is equal to identity matrix. One way to perform whitening is using Eigenvalue Decomposition (EVD) method. The whitening matrix is now given by

$$V = ED^{-\frac{1}{2}}E^T \tag{2}$$

where E is the orthogonal matrix of eigenvector calculated from the covariance matrix $E\{xx^T\}$ and D is the diagonal matrix of the eigenvalues associated with each eigenvector.

2.2 The FastICA Algorithm

A good and fast algorithm for ICA estimation is the FastICA [(Aapo Hyvärinen, 2001), (Aapo Hyvärinen, 2000) e (Hyvarinen, 1999)]. It's a fixed-point iteration algorithm that, in each iteration, finds a vector w that maximizes the statistical independence of w^Tz (Aapo Hyvärinen, 2000). Based on the Central Limit Theorem, the algorithm maximizes the independence by measuring the nongaussianity and iteratively maximizing it until a stop criterion.

A good measure for nongaussianity is called *Negentropy* and is defined using the concept of the diferential entropy (H) of a random variable y with density $p_y(\eta)$, $H(y)$ (Aapo Hyvärinen, 2001), so that the Negentropy (J), of a random variable y, is defined as

$$J(y) = H(y_{gauss}) - H(y) \tag{3}$$

where y_{gauss} is a random variable with the same covariance matrix as y but with gaussian distribution, and $H(y)$ is defined as

$$H(y) = \int f(p_y(\eta))d\eta \tag{4}$$

Due to it's computational complexity and the need to know the probability density function, negentropy is usually aproximated to more simple functions by using high-order cumulants. This gives a good aproximation of negentropy based on expectations of any nonquadratic function G and suggests the FastICA iteration as $w \leftarrow E\{zg(w^Tz) - E\{g^{'}(w^Tz)\}w\}$ [see (Aapo Hyvärinen, 2001)], where $g(y) = y^3$. So, the FastICA main iteration is given by:

$$w^* \leftarrow E\{z(w^tz)^3\} - 3w \tag{5}$$

Considering that the data was preprocessed, the FastICA algorithm for the calculation of one source is given by

1. Choose an initial weight vector w.

2. $w \leftarrow E\{zg(w^Tz)\} - E\{g'(w^Tz)\}w$.

3. $w \leftarrow \dfrac{w}{\parallel w \parallel}$.

4. If not converged, back to step 2.

The algorithm needs to be executed once for each source. In order to prevent that the algorithm estimates the same component more than one time, the following orthogonalization is made:

$$w = w - W^TWw \tag{6}$$

$$w = w/ \parallel w \parallel \tag{7}$$

where W is the matrix containing the w vectors previously calculated.

3 Hardware Design

In this section we present the hardware designed to perform the FastICA algorithm. Basically, the hardware was designed based on the

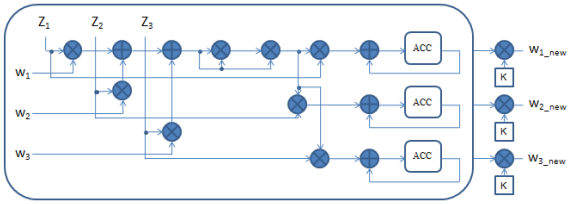

Figure 1: Hardware for the FastICA main iteration for $n = 3$.

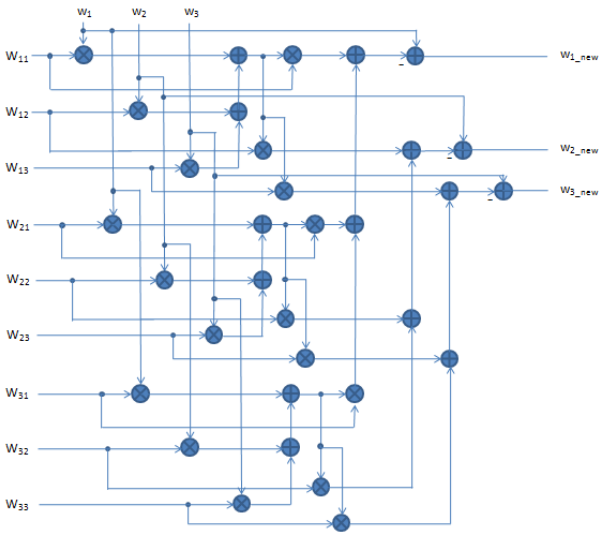

Figure 2: Hardware of the orthogonalization for $n = 3$.

main iteraion and orthogonalization expressions. The preprocessing steps are performed via software, so that the preprocessed data are inserted in the generated models for the simulations and therefore the models are synthesized throuhg the DSP Builder software.

The FastICA main iteration is given in equation (5). In this work we take $n = 3$, so the following expression was used in the design of the hardware:

$$\begin{aligned} w_{i\_new} &= (Z_{i1}(Z_{11}w_1 + Z_{21}w_2 + Z_{31}w_3)^3 + \\ &\quad Z_{i2}(Z_{12}w_1 + Z_{22}w_2 + Z_{32}w_3)^3 + ... + \\ &\quad Z_{ik}(Z_{1k}w_1 + Z_{2k}w_2 + Z_{3k}w_3)^3)K \\ &\quad -3w_i \end{aligned} \quad (8)$$

where $i = 1, ..., n$, $k =$ is the vector length, Z_{ik} represents the preprocessed data and $K = 1/k$.

In figure 1 we show the hardware designed for the main iteration of the FastICA algorithm. A similar hardware is found in (Shyu and Li, 2006). In order to complete the main iteration computation, it is needed to add a simple hardware that performs the $-3w_i$ part of the expression 8.

Whitening guarantees that the independent components are all in orthogonal spaces, so we need to perform an orthogonalization step in order to estimate all of the components. This process may be achieved through the Gram-Schmidt method, in expression 7. The hardware for this step follows the equation 9 and is shown in figure 2.

$$\begin{aligned} w_{i\_new} &= w_i - (W_{1i}(W_{11}w_1 + W_{12}w_2 + W_{13}w_3) \\ &\quad +W_{2i}(W_{21}w_1 + W_{22}w_2 + W_{23}w_3) + \\ &\quad W_{3i}(W_{31}w_1 + W_{32}w_2 + W_{33}w_3)) \end{aligned} \quad (9)$$

After the orthogonalization step, we take the norm of the new projetion w_i_new and the stop criterion calculation is made. This calculation consists in observe if the difference between the new and previous projection is within a tolerance previously specified (η), ie: $w_{i\_new} - w_i \leq \eta$. As consequence of one of the ambiguities of the basic ICA model, the sources can be reversed, so, the convergence of the algorithm must also take into account the criterion $w_{i\_new} + w_i$.

When the stop criterion is reached, the return of the algorithm is calculated and the process is repeated for the remaining independent components.

3.1 System-level Design

Altera DSP Builder software offers a set of specific Altera FPGA blocks that can be used in MATLAB Simulink models for functional simulation of the FPGA hardware design. After the simulation, it's possible to automatically generate a HDL (Hardware Description Language) and synthesize the design on a FPGA platform through the DSP Builder Signal Compiler block.

In figure 3 we show the DSP Builder design for the main iteration of the FastICA algorithm and how to import data from MAT files, which usually are floating-point signals, into DSP Builder blocks, casting these signals to fixed-point format. In this work, the signals are represented using 12 bits for the integer and fractional parts, for a total of 24 bits.

In figure 4 we show the DSP Builder design for the orthogonalization process.

Although we show only the DSP Builder designs for the main iteration and orthogonalization, the whole design consists in more subsystems. These subsystems were developed to control all the process. It contains a set of multiplexer, bus builder, divider, magnitude and other blocks.

4 Results

In this section we show the obtained results from the experiments made to validate the hardware designed. The experiments were done by generating MATLAB Simulink models using Altera DSP Builder blocks for the FPGA functional simulation and synthesis, some signals to test the hardware and electroencephalogram biological signals.

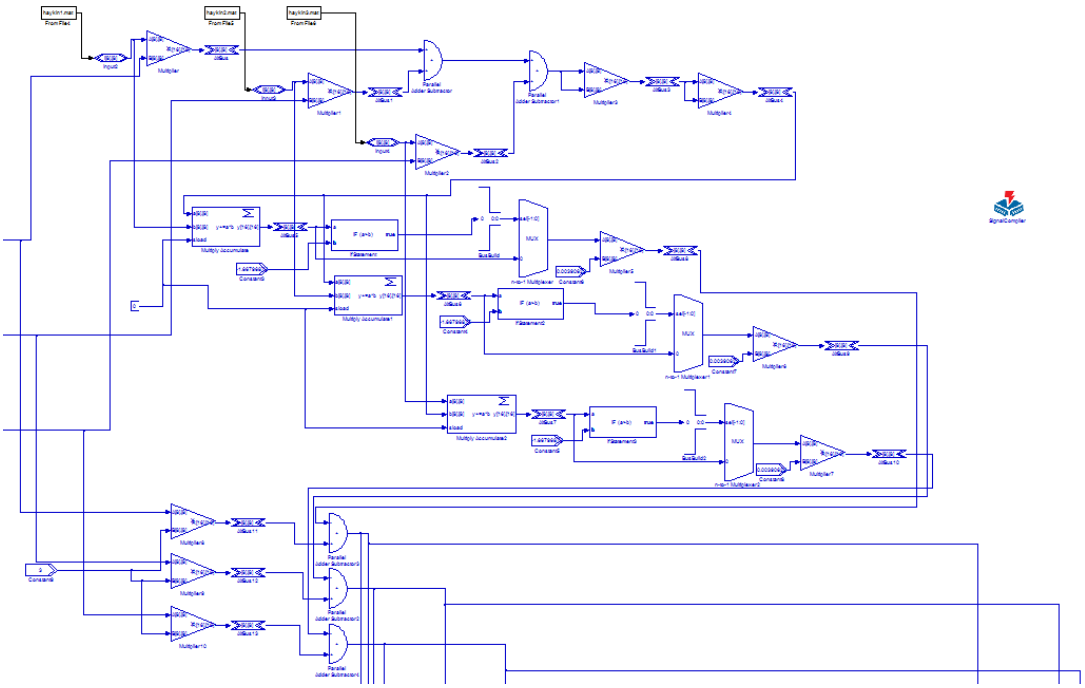

Figure 3: System-level design for the FastICA main iteration.

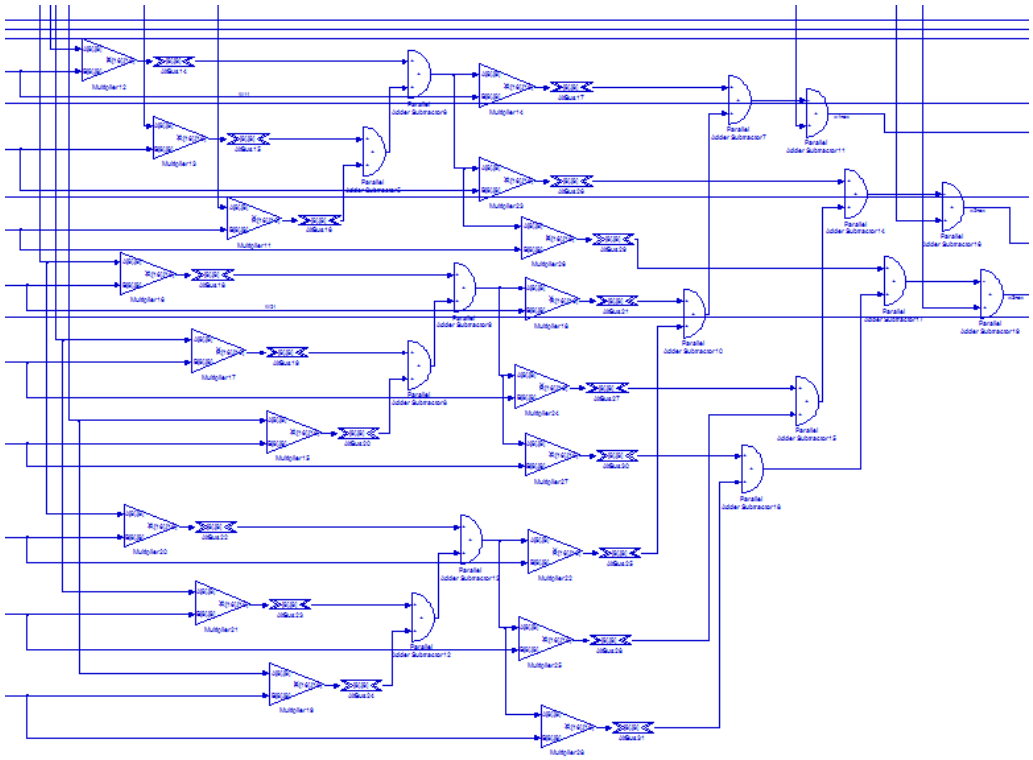

Figure 4: System-level design for the orthogonalization step.

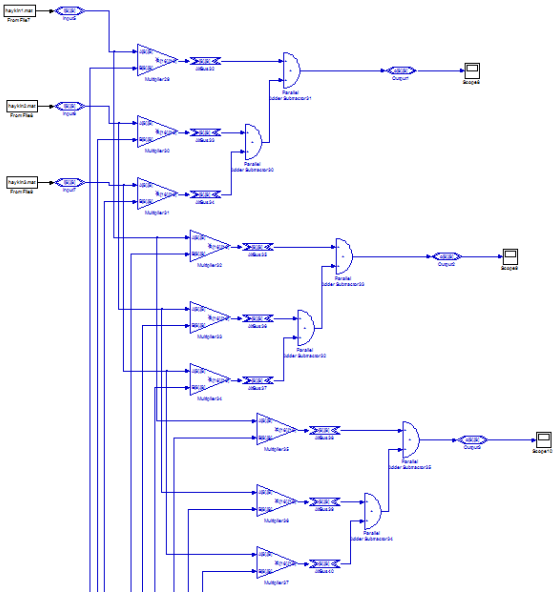

Figure 5: System-level design of the process output using Simulink scope blocks.

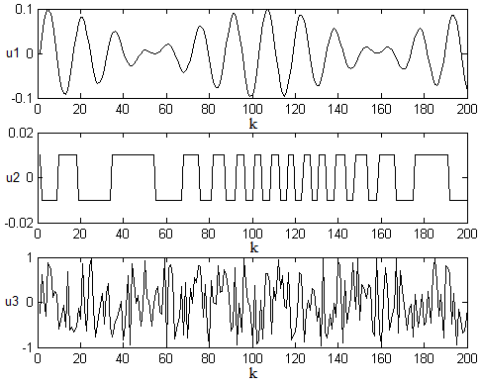

Figure 6: Source signals from first experiment.

Figure 5 shows how to use the Simulink scope block to display the signals generated by the FPGA platform during the process.

The first application is a computational experiment proposed in (Haykin, 2001). We use their experiment to test our developed system. The experiment proposes the estimation of the independent components using the following source signals and mixing matrix.

1. $u_1(k) = 0,1sen(400k)cos(30k)$

2. $u_2(k) = 0,01sign(sen((500k) + 9cos(40k)))$

3. $u_3(k) =$ Uniform distributed noise in the interval [-1, 1]

$$A = \begin{pmatrix} 0,56 & 0,79 & -0,37 \\ -0,75 & 0,65 & 0,86 \\ 0,17 & 0,32 & -0,48 \end{pmatrix} \quad (10)$$

The source signals and mixtures manually generated are shown in figures 6 and 7. Figure 8 shows the independent components obtained

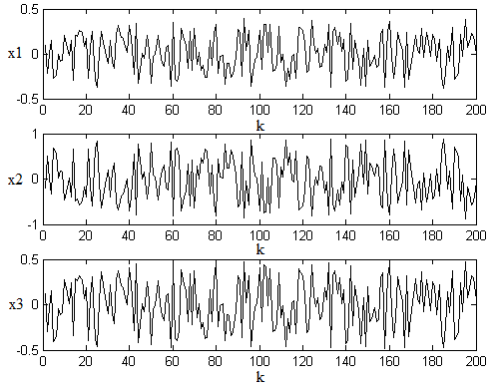

Figure 7: Observed mixtures from the frist experiment.

through our developed system. The system was able to separete the signals.

In the second experiment, biological electroencephalogram signals given by the International Institute of Neuroscience of Natal Edmond and Lily Safra - IINN-ELS were used. Those signals were extracted in experiments where 22 sensors were put on the subject's scalp to measure EEG signals, and other 4 sensors measure muscular activity form ocular movement, chin muscle activity and heart beat, in a total of 26 signals. It would be necessary a much more bigger hardware than the one we present here in this work to estimate all of the independent components. To overcome this, as in (Ricardo Vigário and Oja, 2000), we made a dimension reduction through the PCA process. In this experiment, the PCA was performed to decrease the dimension to $n = 3$.

In figure 9 we have the estimated independent components from the second experiment. The first and third components represents EEG signals and the second represents muscular activity.

5 Conclusions and Future Works

In this work we propose the design, simulation and synthesis of a hardware that performs the FastICA algorithm, by generating MATLAB Simulink models and using Altera DSP Builder software, which offers a set of blocks that can be synthesized on a FPGA platform, to perform the FPGA functional simulation and synthesis on a Altera Cyclone II board of the design. Through the DSP Builder software we may also use Simulink scope blocks to display the signals generated by the FPGA on the process.

While all the signals from the first experiment were successfully estimated, this is far from being a real time application. In the second experiment, though we used real electroencephalogram signals and three independent components were also successfully estimated, it is also an experiment that can't be said to be real time.

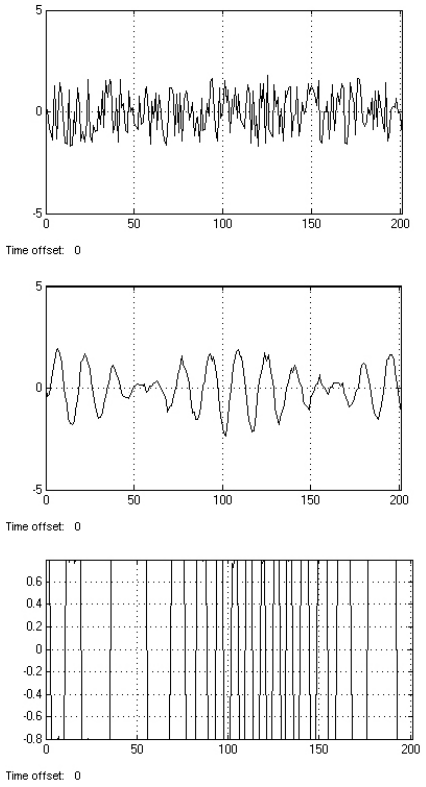

Figure 8: Independent components estimated through the hardware design.

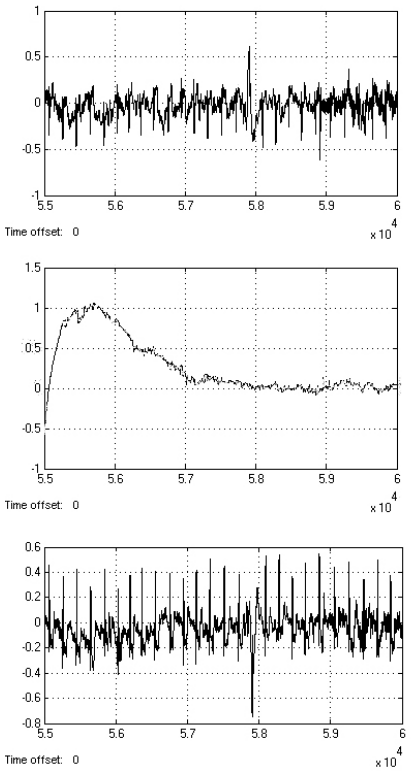

Figure 9: Independent components estimated in the experiment using biological electroencephalogram signals.

As future work, we pursuit the improving of the design so that it can be used in real time applications, such as Blind Source Separation in biological environments for the purpose of healthy care. Also, this work will integrate a bigger system implemented in the same FPGA, for the purpose of disease detection.

References

Aapo Hyvärinen, E. O. (2000). *Independent Component Analysis: Algorithms and Applications*, Neural Networks.

Aapo Hyvärinen, Juha Karhunen, E. O. (2001). *Independent Component Analysis*, John Willey & Sons Inc.

Anis Nordin, C. H. and Szu, H. (2001). *Desing of FPGA ICA for hyperspectral imaging processing*, Proc. SPIE Signal Image Process, vol. 4391, pp. 444-454.

Charoensak, C. and Sattar, F. (2005). *Design of Low-Cost FPGA Hardware for Real-time ICA-Based Blind Source Separation Algorithm*, EURASIP Journal on Applied Signal Processing 18:3076-3086.

Du, H. and Qi, H. (2004). *An FPGA implementation of parallel ICA for dimensionality reduction in hyperspectral images*, IEEE Proceedings, vol. 5, pp. 3257-3260.

Haykin, S. (2001). *Redes Neurais: Princípios e Prática*, Bookman.

Hyvarinen, A. (1999). *Fast and robust fixed-point algorithm for independent component analysis*, IEEE Transactions on Neural Networks, 10.

Kuo-Kai Shyu, Ming-Huan Lee, Y.-T. W. and Lee, P.-L. (2008). *Implementation of Pipelined FastICA on FPGA for Real-Time Blind Source Separation*, IEEE Trans., vol. 19, no. 6.

Ricardo Vigário, Jaakko Särelä, V. J. M. H. and Oja, E. (2000). *Independent Component Approach to the Analysis of EEG and MEG Recordings*, IEEE Trans., vol. 47, no. 5.

Shyu, K.-K. and Li, M.-H. (2006). *FPGA Implementation of FastICA based on Floating-Point Arithmetic Design for Real-Time Blind Source Separation*, Int. Joint Conf. on Neural Networks.