2015 12th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico, City. Mexico

FPGA Implementation of the ICA Algorithm using Multiplexing

G.M. Tornez-Xavier[1], L.M. Flores-Nava[1], F. Gómez-Castañeda[1], J.A. Moreno-Cadenas[1],
[1]Department of Electrical Engineering, CINVESTAV-IPN, Mexico D.F., Mexico
Phone (52) 55 5747 3800 Ext. 6261
E-mail: {gtornez, lmflores, fgomez, jmoreno} @cinvestav.mx

***Abstract* — This work presents an optimized version in FPGA technology of a digital system, which solves in real time the Blind Source Separation problem using the Independent Component Analysis, ICA algorithm and following the Maximum Information technique, INFOMAX. To demonstrate the FPGA realization, we use a mix of three sinusoidal signals, which represents three independent sources, with 1000Hz, 800Hz and 600Hz values in frequency. The mixed signal is treated by the ICA system. The digital system in FPGA was analyzed first in Simulink of Matlab, evaluating its performance. Then, the FPGA architecture, which was optimized observing a multiplexing scheme, is proposed where the number of used DSP resources is minimal. This leads to extend this multiplexing scheme to cover future designs with more signals.**

Keywords — Independent Component Analysis, Blind Source Separation, Maximum Information principle, Simulink, VHDL, FPGA.

I. INTRODUCCION.

The independent Component Analysis or ICA algorithm is a statistical technique to find the original sources from a mix, inside a multidimensional space and, where no information is given about the sources namely, this is a Blind Source Separation problem or BSS problem [1, 2]. The ICA algorithm is also used in finding feature vectors and cancelling noise, adaptively. The BSS problem deals with determining through a set of transductors, the original signals or sources that were mixed, where the proportion of their mix is unknown. The BSS concepts recognize at present many applications in the fields of telecommunications, bioelectronics, smart sensors and instrumentation.

In this work, the INFOMAX algorithm, which is an extension of the original ICA algorithm, is designed in FPGA to solve the BSS problem. To get this objective, there were two stages namely, 1) using software to analyze the parallel architecture of processing blocks; we refer to Simulink and 2) using a digital platform which describes the temporal response of the FPGA implementation.

II. ICA ALGORITHM.

The objective of the ICA algorithm is also used to recover the independent signals **S**, which are unknown and pass through a mixing matrix **A** and, observing **X**, which are collected by the transducers. Under this statement we get the model below.

$$\boldsymbol{X} = \boldsymbol{AS} \tag{1}$$

The ICA algorithm finds the separation matrix **W**, which is the inverse matrix of **A**, doing in this manner the components of the matrix **S** independent as far as possible.

$$\boldsymbol{S} = \boldsymbol{W}\mathbf{X} \tag{2}$$

One way that the ICA algorithm approaches the sources S is using the principle of maximum information established by the INFOMAX algorithm [3, 4]; this algorithm uses the entropy as a measure of independence. Then, the set of signals that shows the maximum entropy is selected from the mix, due to they are taken as the independent sources for solving the BSS problem.
The INFOMAX algorithm which is expressed by (3), follows the model for the change on **W**, where α means "proportional to".

$$\Delta \boldsymbol{W} \alpha [\boldsymbol{I} - \varphi(\boldsymbol{u})\boldsymbol{u}^T]\boldsymbol{W} \tag{3}$$

Where $\varphi(.)$ is the nonlinear transfer function of the neural network model, represented by (4).

$\boldsymbol{u}$ is the output vector of the neural network.

$$\varphi(u_i) = \frac{e^{u_i} - e^{-u_i}}{e^{u_i} + e^{-u_i}} \tag{4}$$

III. SOFTWARE IMPLEMENTATION.

Fig. 1 shows the block diagram of the ICA algorithm using Simulink in Matlab. We observe three main processing blocks namely, Mixer, Neural Network and Learning. The Learning block generates the matrix **W** of the INFOMAX algorithm which is returned to the Neural Network, for updating its weights.

The way for testing the ICA algorithm in Simulink is presenting three sinusoidal signals that act as independent sources and whose frequencies are 1000Hz, 800Hz and 600Hz. They were mixed according to the matrix **A** in (4).

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} \text{sine}\,(2\pi 1000t) \\ \text{sine}\,(2\pi 800t) \\ \text{sine}\,(2\pi 600t) \end{bmatrix} \tag{4}$$

978-1-4673-7839-0/15/$31.00 ©2015 IEEE

Therefore, the resulting mixed signals are x_1, x_2 and x_3. Their graphical representation is in Fig. 2

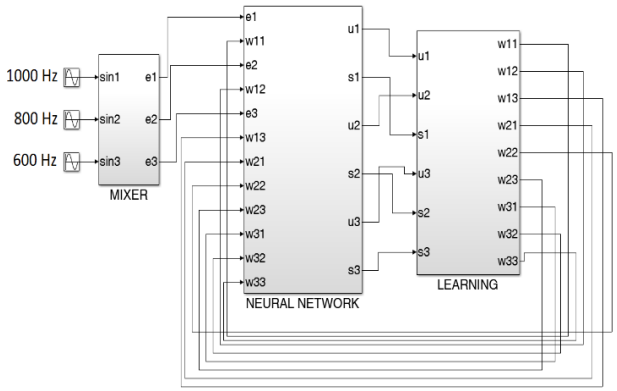

Fig 1. ICA algorithm in Simulink.

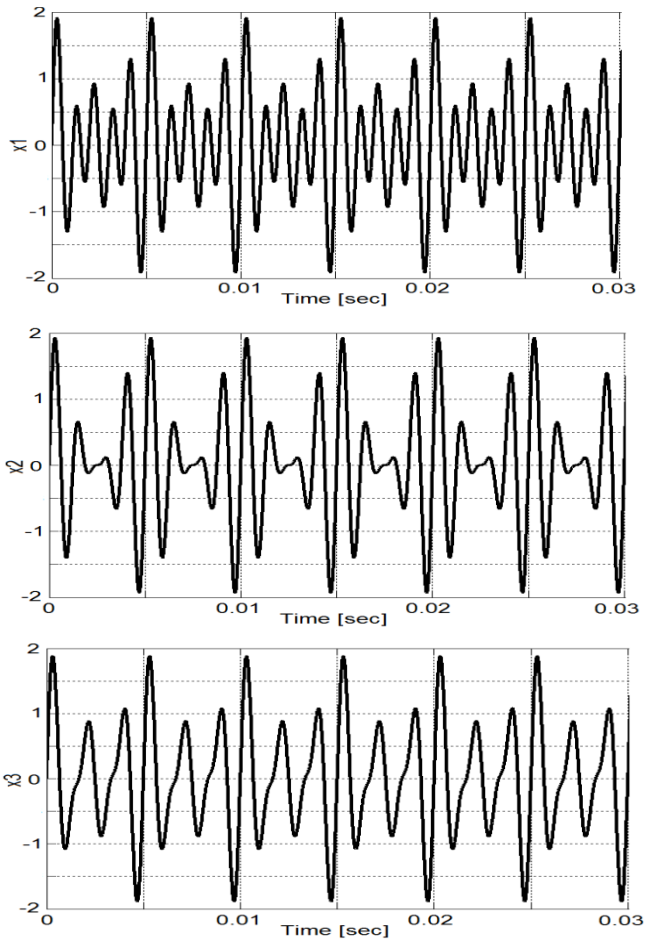

Fig 2. Mixed signals.

The signals in matrix **A** are taken by a forward neural network with three neurons, whose output activation function is a saturated linear function. So, there are nine synaptic weights to adapt.

IV. HARDWARE IMPLEMENTATION

The ICA algorithm was described in VHDL code for its hardware realization using the ISE prototype development system, Version 14.2 by Xilinx. The final FPGA digital part was the Spartan-6 XC6SL45. The simulation software in this design cycle was ModelSim SE 10.0c.
The hardware architecture is designed for multiplexing the processing resources namely, Mixer, Neural Network and Learning.

For getting the greatest precision in the numeric representation of the digital words, they were tailored to the largest word in the multipliers with fixed point [5], signed numbers and 2's complement representation. The length of the digital words was 18-bit and divided as shown in Fig. 3.

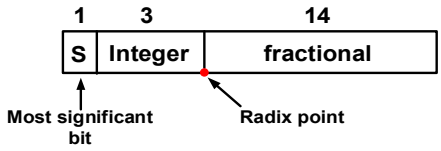

Fig 3. Binary representation.

The processing blocks that perform the ICA algorithm are in Fig. 4; there, the Learning block is multiplexed, while the rest are not. This scheme reduces the usage of hardware in the case of three input signals and can be extended for more; keeping mainly the DSP resource demand low.

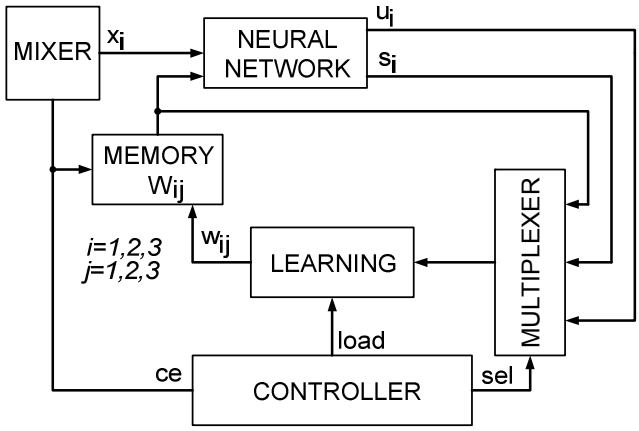

Fig 4. ICA digital system.

Mixer.

This subsystem, which is presented in Fig. 5, has three random access memories (RAM), each one stores the sampled sinusoidal signals, which were numerically created before in Simulink. The total number of stored samples is 500, as 18-bit digital words. The samples presented to the FPGA followed a 2 us period and the same samples were presented to the ICA system in Simulink each 10 us. Therefore, the time-scale in Fig. 6 is 5 times faster than the shown in Fig. 2.

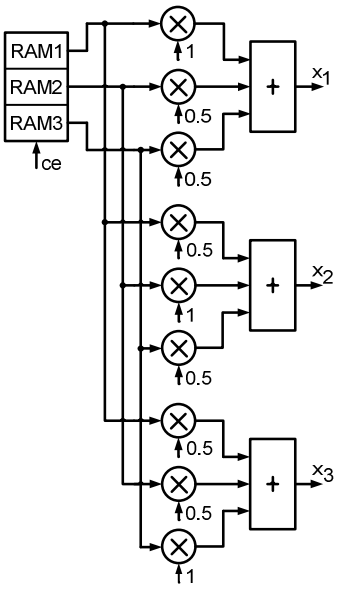

Fig 5. Mixer.

Fig. 6 is the set of the three sinusoidal mixed signals in ModelSim to test the digital design.

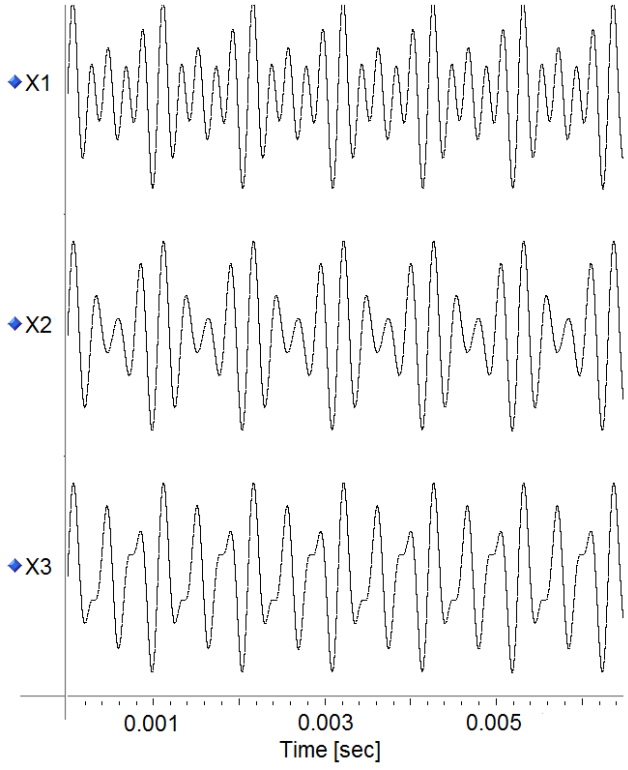

Fig 6. Mixed signals in ModelSim.

Neural Network.

This subsystem has three neurons, where the output transfer function in VHDL is s, a hyperbolic Tangent is needed, but due to its complexity to implement in FPGA a linear function was proposed; it follows the relation (5) below.

$$s = \begin{cases} -1 & u \le -1 \\ u & -1 < u < +1 \\ +1 & u \ge +1 \end{cases} \qquad (5)$$

This neural network uses 9 DSP's and is constructed according to the diagram in Fig. 7.

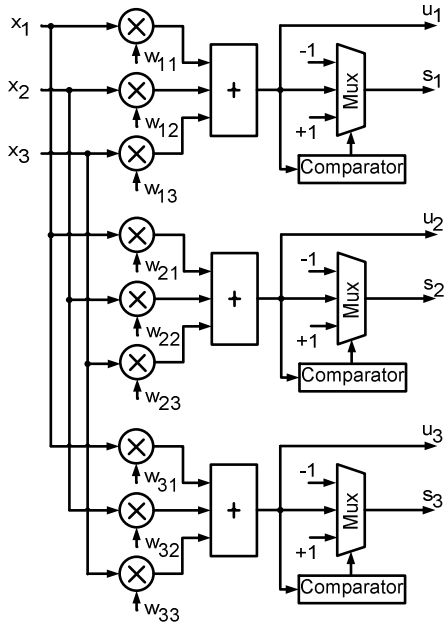

Fig 7. Neural Network.

Learning.

This subsystem would require 45 DSP's if it is not multiplexed. The total available for the selected FPGA is 58. However, adopting a multiplexing scheme according to the diagram shown in Fig. 8, the number of DSP's needed reduces to five. Where C is the time differential with a value of 1E-04. Table 1 contains the parameters that are sorted in time by the Learning block.

Table 1. Multiplexers and their inputs

sel(3:0)	Mux 1	Mux 2	Mux 3	Mux 4	Mux 5	Mux 6	Mux 7
0000	w_{11}	u_1	w_{12}	u_2	w_{13}	u_3	s_1
0001	w_{12}	u_2	w_{11}	u_1	w_{13}	u_3	s_2
0010	w_{13}	u_3	w_{11}	u_1	w_{12}	u_2	s_3
0011	w_{21}	u_1	w_{22}	u_2	w_{23}	u_3	s_1
0100	w_{22}	u_2	w_{21}	u_1	w_{23}	u_3	s_2
0101	w_{23}	u_3	w_{21}	u_1	w_{22}	u_2	s_3
0110	w_{31}	u_1	w_{32}	u_2	w_{33}	u_3	s_1
0111	w_{32}	u_2	w_{31}	u_1	w_{33}	u_3	s_2
1000	w_{33}	u_3	w_{31}	u_1	w_{32}	u_2	s_3

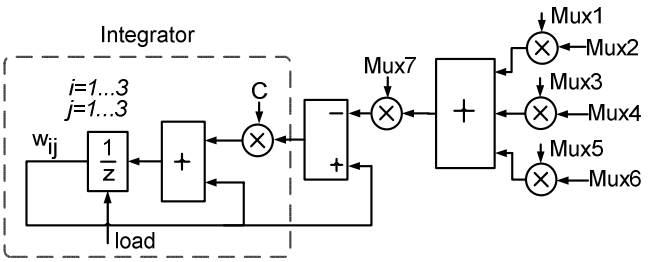

Fig 8. Learning subsystem for multiplexing.

Controller.

This subsystem is in charge of generating three signals namely, **sel**, **load**, and **ce**. They coordinate the flow of information properly to get concurrently the partial results from the Mixer, Neural Network and Learning subsystems until a steady state is reached in the values in the matrix **W**. In particular, **sel** selects nine input signals to the multiplexers. **load** enables the weights into the neural network, and **ce** indicates the action for accessing the RAM. The whole set of control signals in ModelSim is in Fig. 9.

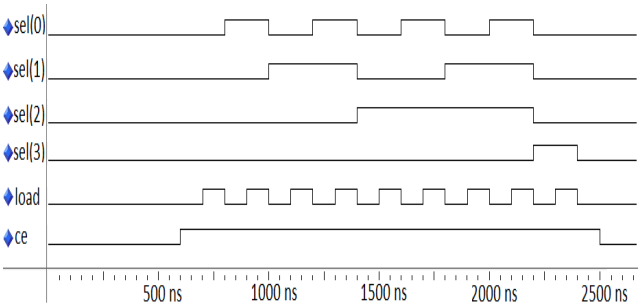

Fig 9. Control signals in ModelSim.

Logic Resources Used.

The total number of logic resources that were implemented for the ICA algorithm in the Spartan-6 XC6SL45 FPGA is in Table 2. We realize that this multiplexing scheme reduces this number and it allows considering another ICA systems with more mixed input signals and low complexity in hardware, still.

Table 2. Logic resources used for the FPGA

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	669	27288	2%
Number of Slices	330	6822	4%
Number of RAM B16BWERs	3	116	2%
Number of RAM B8BWERs	1	232	1%
Number of DSP48A1s	16	58	27%

V. RESULTS.

Running the final VHDL code of the whole ICA Algorithm and using the INFOMAX method we get the expected independent sources, or signals in matrix **S**, according to those in Fig. 10.

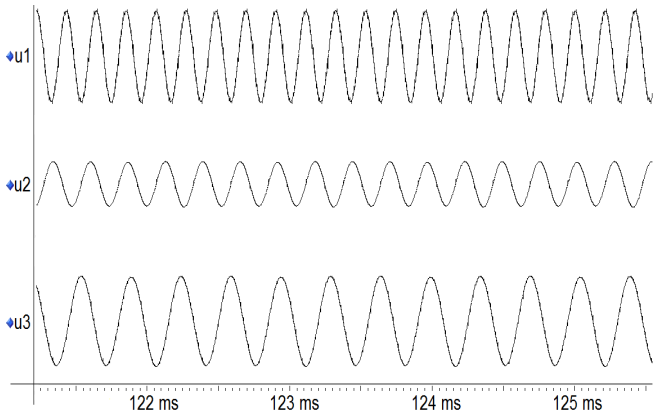

Fig 10. Independent sources obtained by the ICA algorithm in ModelSim.

As a matter of completeness, Table 3 is a comparison of the number of resources that the parallel and the multiplexed methods would use for the ICA algorithm in FPGA. We conclude that the multiplexing method saves hardware resources, getting the original goal of this project.

Table 3. DSP's employees for each technique

Inputs n	Multiplexing	Parallel
2	4	16
3	5	45
4	6	96
5	7	175
	$n+2$	$n^2(n+2)$

VI. CONCLUSIONS.

There is always the tradeoff between reduction in processing time and cost of hardware when special purpose or dedicated digital core is designed. This balance is reached fairly in FPGA technology if the multiplexing of hardware is chosen. This complex digital system, in the case of the ICA algorithm, demonstrates this concept.

REFERENCES

[1] Te-Won Lee, M. Girolami and T.J. Sejnowski "Indepent Component Analysis Using an Extended Infomax Algorithm for Mixed Subgaussian and Supergaussian Sources*" Neural Computation, 11, pp. 417-441, 1999.*

[2] Te-Won Lee "Indepent Component Analysis*" Kluwer Academic Publisher, California, USA, 2000.*

[3] L.N Oliva, J. De laCruz, J.A. Moreno "Optimized Infomax-ICA algorithm on FPGA Architecture for Blind Source Separation". *In 7th International on Electrical Engineering, Computing Science and Automatic Control, CCE 2010* .

[4] L.N Oliva, Miguel.A. A. Alemán, J.G Lamont "Implementation of Infomax ICA Algorithm for Blind Source Separation". *In Electronics, Robotics and Autotive Mechanics Conference 2008*.

[5] A. Savich, M. Moussa, S. Areibi "The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study".*IEEE Transactions on Neural Networks, Vol.18, No.1, January 2007, pp. 240-252*.