Implementación en FPGA del algoritmo ICA para Cancelación de ruido en dispositivos móviles

Luz Noé Oliva Moreno

Instituto Politécnico Nacional - Escuela Superior de Computo, Av. Juan de Dios Bátiz s/n, La Escalera, Gustavo A. Madero, 07738 Ciudad de México, Distrito Federal, 57296000 ext. 52062

loliva@ipn.mx

Margarita Elizabeth Gómez Mayorga

Instituto Politécnico Nacional - Escuela Superior de Computo, Av. Juan de Dios Bátiz s/n, La Escalera, Gustavo A. Madero, 07738 Ciudad de México, Distrito Federal, 57296000 ext. 52039

mgomezm@ipn.mx

Claudia Alejandra López Rodríguez

Instituto Politécnico Nacional - Escuela Superior de Computo, Av. Juan de Dios Bátiz s/n, La Escalera, Gustavo A. Madero, 07738 Ciudad de México, Distrito Federal, 57296000 ext. 52062

calopezr@ipn.mx

Miguel Ángel Mentado Contreras

Instituto Politécnico Nacional - Escuela Superior de Computo, Av. Juan de Dios Bátiz s/n, La Escalera, Gustavo A. Madero, 07738 Ciudad de México, Distrito Federal, 57296000 ext. 52062

mentado07@hotmail.com

Resumen

Actualmente se ha incrementado el uso de dispositivos móviles como celulares y tabletas electrónicas, las cuales manipulan muchas aplicaciones, y requieren de procesadores con mayor potencial de procesamiento. En este artículo se propone el uso de una arquitectura paralela en un FPGA (del inglés Field Programmable Gate Array), que realice la cancelación del ruido en tiempo real e independiente de las tareas del procesador de un equipo. La técnica utilizada es el Análisis de Componentes Independientes (ICA del inglés Independent Component Analysis), que se basa en las propiedades estadísticas

de las señales en espacios multidimensionales, calcula los pesos sinápticos dato a dato y originalmente está inspirada en la arquitectura de las Redes Neuronales Artificiales. En el desarrollo se realizó la arquitectura con Simulink mediante el software de Matlab y posteriormente con un lenguaje de Descripción de Hardware HDL-Verilog (HDL del inglés hardware description language), se realizó la implementación en un FPGA; La conversión de las señales analógicas a digitales y digitales a analógicas, se realizó mediante un ADC y un DAC de 12 bits a una frecuencia de muestreo de 44Khz; y se obtuvieron resultados de la simulación y de los experimentales.

Palabras Claves: Análisis de Componentes Independientes, Cancelación de ruido adaptativo, FPGA, HDL-Verilog, Redes Neuronales.

1. Introducción

El Análisis de Componentes Independientes [1,2] es una técnica utilizada para encontrar componentes independientes dentro de espacios multidimensionales. Al aplicar una trasformación de los datos, se pueden obtener las señales independientes que intervienen. Al problema de separar señales sin conocer ninguna de sus fuentes originales, se le conoce como Separación Ciega de Señales (BSS del inglés Blind Source Separation). El BSS es un problema clásico de procesamiento de señales con aplicaciones potenciales en sistemas de reconocimiento de voz, procesamiento de señales médicas, telecomunicaciones, etc.

En la actualidad, estos algoritmos han tenido su auge, con herramientas de software, para resolver problemas fuera de línea debido a su gran complejidad en el manejo de datos y su tiempo de cálculo [3], existiendo el problema de procesamiento en tiempo real. Los dispositivos digitales programables como el FPGA ofrecen una solución al procesamiento de las señales en tiempo real, debido al desarrollo de arquitecturas paralelas dedicadas a estas tareas, permitiendo depurar arquitecturas y optimizando los recursos y tiempos de procesamiento. Para la implementación del algoritmo ICA en un FPGA, se utiliza la herramienta Simulink del software de Matlab, la programación se

realiza mediante la unión de bloques para formar una arquitectura de las ecuaciones que despeñan el algoritmo para la separación de las fuentes, posteriormente, el diseño se implementa en el FPGA con la programación en lenguaje HDL-Verilog.

2. Desarrollo

El problema consiste en separar una señal de voz inmersa en una señal de ruido, sin el conocimiento previo de dichas señales. La técnica de ICA consiste en recibir por medio de un arreglo de dos micrófonos la mezcla de ambas señales, que por sus características y la posición de cada micrófono, reciben una combinación lineal diferente de las señales originales, como se muestra en la Fig.1.

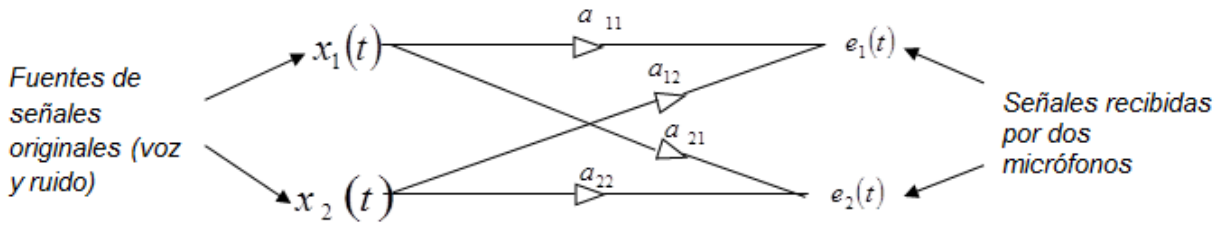

Fig. 1. Arreglo de transductores y fuentes originales de tamaño 2x2.

El sistema descrito anteriormente, se representa en la siguiente ecuación:

$$e_i(t) = \sum_{i,j=1}^{n} a_{ij} s_j(t) \qquad (1)$$

$$(i, j = 1,2,.....,n)$$

En donde: $e_i(t)$ Es la señal de salida continua en el tiempo, del i-ésimo micrófono.

$s_j(t)$ Es la señal desconocida, emitida por la j-ésima fuente.

a_{ij} Es un número escalar de la combinación lineal.

En forma matricial, la ecuación (1) se expresa como:

$$\mathbf{e}(t) = \mathbf{A}\mathbf{s}(t) \tag{2}$$

En donde ***A*** representa la matriz de mezcla de tamaño $n \times n$. En el modelo descrito en la ecuación (2) se desprecian los retardos de tiempo que ocurren durante la mezcla, por lo tanto se dice que es un modelo de mezclado instantáneo.

La arquitectura de la red neuronal propuesta por Jutten y Hérault [4], para dar una solución al problema de separación, utiliza un filtro adaptable lineal recursivo como el que se muestra en la Fig. 2. Cada círculo negro representa una operación de suma y cada triángulo representa la función de activación de la neurona. Esta arquitectura es similar a la red neuronal de Hopfield [5].

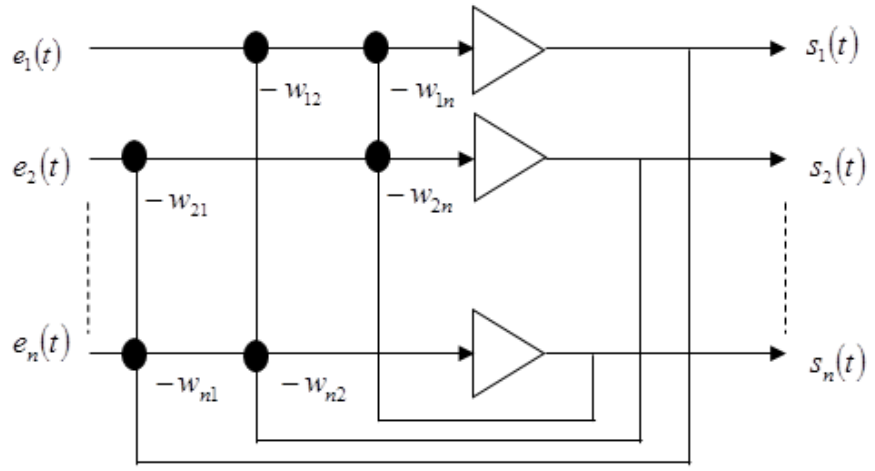

Fig. 2 Arquitectura neuronal de un filtro adaptable lineal recursivo.

El sistema descrito en la Fig. (2) se representa por la siguiente ecuación:

$$s_i(t) = e_i(t) - \sum_{i=1}^{n} w_{ij} s_j(t) \qquad (3)$$

$$(i, j = 1,2,.....,n)$$

$$i \neq j$$

En donde: $e_i(t)$ Es la señal continua en el tiempo, de la salida del i-ésimo micrófono.

$s_i(t)$ Es la señal continua en el tiempo, de la i-ésima salida de la red neuronal.

w_{ij} Es el valor del peso sináptico de la red.

La salida i-ésima $s_i(t)$ es el resultado de una suma de la señal de entrada $e_i(t)$ y del producto parcial $-w_{ij}s_j(t)$, en dónde $j \neq i$.

Expresando la ecuación (3) en notación matricial se tiene:

$$S(t) = E(t) - WS(t) \qquad (4)$$

De esta ecuación se obtiene:

$$S(t) = (I + W)^{-1} E(t) \qquad (5)$$

Siendo *W* la matriz de los coeficientes de pesos inhibitorios de la red e I la matriz identidad.

Para este problema se tomó el caso de *n = 2* y en la Fig. 3 se muestra el arreglo neuronal 2x2.

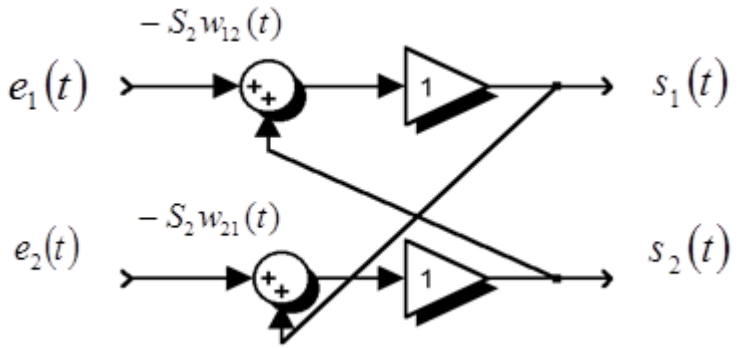

Fig. 3 Arreglo neuronal para n =2.

Las ecuaciones (1) y (3) son escritas a continuación en forma de ecuaciones lineales para el arreglo neuronal de tamaño 2x2.

$$e_1(t) = a_{11}x_1(t) + a_{12}x_2(t) \tag{6}$$

$$e_2(t) = a_{21}x_1(t) + a_{22}x_2(t) \tag{7}$$

$$s_1(t) = e_1(t) - w_{12}s_2(t) \tag{8}$$

$$s_2(t) = e_2(t) - w_{21}s_1(t) \tag{9}$$

Sustituyendo $e_1(t)$ y $e_2(t)$ en la ecuación (3) se obtiene:

$$s_1(t) = \frac{(a_{11} - w_{12}a_{21})x_1 + (a_{12} - w_{12}a_{22})x_2}{1 - w_{12}w_{21}} \tag{10}$$

$$s_2(t) = \frac{(a_{21} - w_{21}a_{11})x_1 + (a_{22} - w_{21}a_{12})x_2}{1 - w_{12}w_{21}} \tag{11}$$

Dos pares particulares de coeficientes w_{ij} pueden proponerse, dando las soluciones correspondientes:

i) si $w_{12} = a_{12}/a_{22}$ y $w_{21} = a_{21}/a_{11}$

Las señales de salida son $s_1(t) = a_{11}x_1(t)$ y

$$s_2(t) = a_{22}x_2(t) \tag{12}$$

ii) si $w_{12} = a_{11}/a_{21}$ y $w_{21} = a_{22}/a_{12}$

Las señales de salida son:

$$s_1(t) = a_{12}x_2(t) \quad \text{y} \quad s_2(t) = a_{21}x_1(t) \tag{13}$$

Una condición de estabilidad en el sistema es la siguiente:

$$w_{12}w_{21} < 1 \tag{14}$$

Esta condición se puede verificar en las ecuaciones (10) y (11).
Teóricamente, la arquitectura de la red permite separar fuentes desconocidas. No obstante, el valor de w_{ij} es un coeficiente desconocido.

A continuación, se tratará la regla de aprendizaje propuesto por Jutten y Hérault [4] para estimar la matriz de pesos sinápticos *W*. Jutten y Hérault desarrollaron un algoritmo matemático de la regla de aprendizaje basado en el método de *descenso del gradiente.* El algoritmo matemático ajusta los pesos sinápticos de la red neuronal en forma continua, de acuerdo a la siguiente expresión conocida como la regla de Hebb o regla de adaptación:

$$\frac{dw_{12}(t)}{dt} = \eta \cdot f[s_1(t)] \cdot g[s_2(t)] \tag{15}$$

Donde η es la razón de aprendizaje (de valor positivo).
Dicho algoritmo se puede expresar de forma recursiva como:

$$w_{12}(t+1) = w_{12}(t) + \eta \cdot f\left[s_1(t)\right] \cdot g\left[s_2(t)\right] \tag{16}$$

Para lograr una independencia estadística, Jutten y Hérault propusieron el uso de dos funciones no lineales e impares, preservando la condición de media cero [6].

Si se expresan las funciones *f* y *g* de la ecuación (16) en series de Taylor de los términos impares se tiene que:

$$f(x) = \sum_{j=1}^{\infty} f_{2j+1} x^{2j+1} \quad g(x) = \sum_{m=1}^{\infty} g_{2m+1} x^{2m+1} \tag{17}$$

Y sustituyendo (17) en (15) se obtiene:

$$\frac{dw_{12}(t)}{dt} = \eta \cdot \sum_{j=1}^{\infty} \sum_{m=1}^{\infty} f_{2j+1} g_{2m+1} s_1^{2j+1}(t) s_2^{2m+1}(t) \tag{18}$$

Después, definiendo el n-ésimo momento de una señal estacionaria como:

$$\overline{s^n} = \frac{1}{T} \int_0^T s^n(t) dt \tag{19}$$

Y expresando la ecuación (19) en un intervalo de tiempo:

$$\left| \left\langle \overline{s^n} \right\rangle = \frac{1}{T_2 - T_1} \int_{T1}^{T2} s^n(t) dt \right. \tag{20}$$

En donde $\left\langle \overline{s^n} \right\rangle$ es el promedio en el tiempo $T_2 - T_1$. La convergencia del algoritmo corresponde ahora a la siguiente condición:

$$\left\langle \frac{dw_{12}}{dt} \right\rangle = \eta \cdot \sum_{j=1}^{\infty} \sum_{m=1}^{\infty} f_{2j+1} g_{2m+1} \left\langle s_1^{2j+1}(t) s_2^{2m+1}(t) \right\rangle = 0 \tag{21}$$

La convergencia se alcanza cuando los momentos de orden superior $\left\langle s_i^{2j+1}(t) s_j^{2i+1}(t) \right\rangle$ son cero [7], esto implica que $\left\langle s_i^{2j+1}(t)\, s_j^{2i+1}(t) \right\rangle = \left\langle s_i^{2j+1}(t) \right\rangle \left\langle s_j^{2i+1}(t) \right\rangle = 0$, logrando independencia estadística entre $s_i(t)$ y $s_j(t)$. Un gran número de funciones impares pueden ser utilizadas para la separación. Experimentalmente fueron determinadas en *f* y *g* las siguientes restricciones, asegurando una convergencia rápida y estable de los pesos [7].

- Para una convergencia estable, *f* debe tener una curvatura positiva y *g* una curvatura negativa.
- Para una convergencia rápida, *f* y *g* deben ser ortogonales en el origen.

Las funciones *f* y *g* propuestas por Jutten y Hérault son:

$$\text{Par 1: } f(x) = arctg(x) \text{ y } g(x) = sign(x) \tag{22}$$

$$\text{Par 2: } f(x) = \tanh x \qquad y \qquad g(x) = x^3 \tag{23}$$

A pesar de que: $f(x) = x$ es una función lineal, la combinación de ambas funciones consiguen la separación de fuentes independientes. Aunque ya existen otras implementaciones [7,8], la solución propuesta en este artículo reduce el número de compuertas lógicas, por lo que es de baja complejidad.

Para la implementación de la arquitectura se utilizó: una Spartan-3E FPGA 500 Nexys2; un módulo PmodDA2(DAC121S101) que convierte señales con valores digitales a voltajes analógicos en dos canales simultáneamente, de 12 bits, a una frecuencia de muestreo de 44100Khz; y un módulo PmodAD2 (ADCS7476AIM) convertidor de analógico a digital de dos canales, de 12 bits. Ambos módulos operan con el protocolo de comunicación SPI.

Para el diseño esquemático de las ecuaciones (1), (3) y (16), se utilizó simulink como se muestra en la Fig. 4. La arquitectura está constituida por la ecuación (1) y representada en el bloque 1, la arquitectura de la red neuronal está constituida por la ecuación (3) y

representada en el bloque 2, y la arquitectura de la regla de aprendizaje está constituida por la ecuación (16) y representada en el bloque 3.

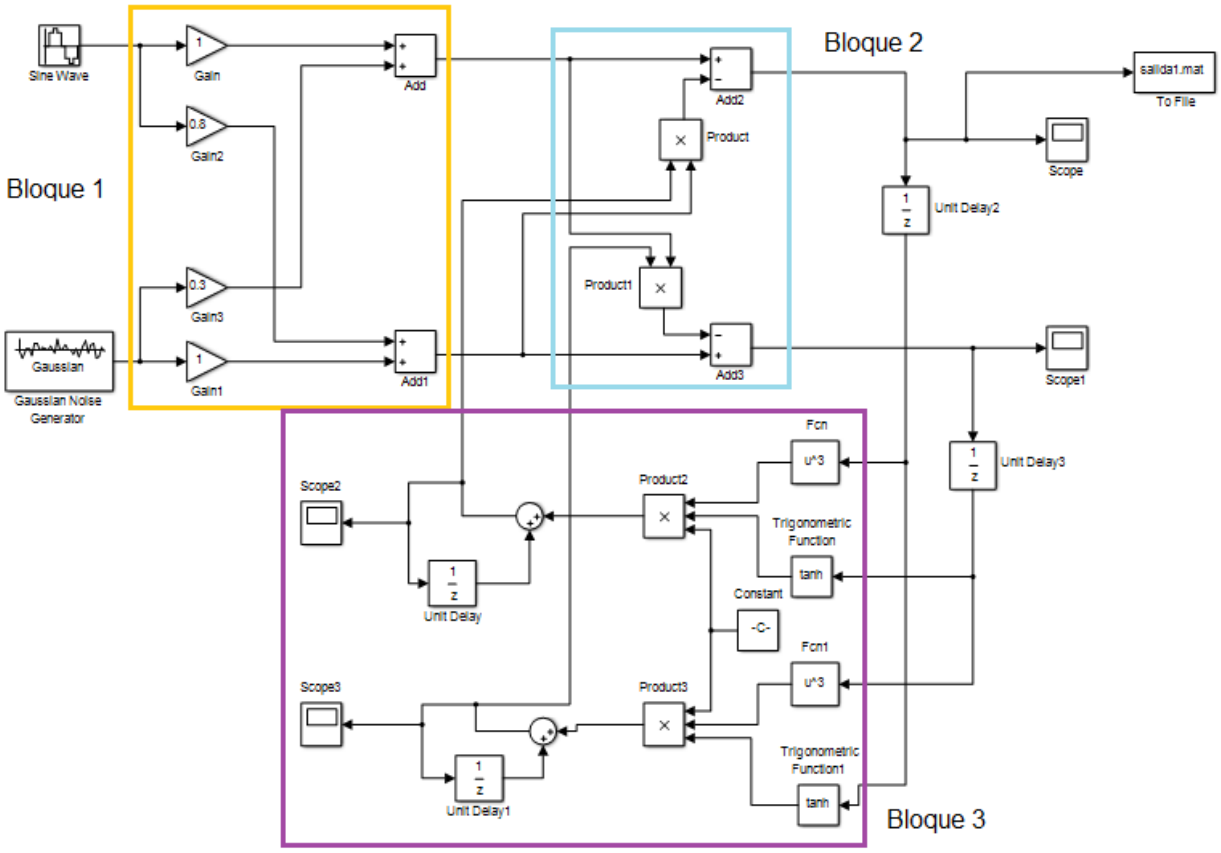

Fig. 4. Diagrama a bloques del algoritmo ICA mediante Simulink.

El bloque 1 no se programó, solo representa el modelo de la mezcla que se efectúa en el medio ambiente. Los bloques 2 y 3 son programados en lenguaje Verilog.

El código que a continuación se muestra, describe la arquitectura de la red neuronal y su regla de aprendizaje.

```
`timescale 1 ns / 1 ns
module Red_HJ(In1,In2,Out1,Out2,clk,reset,clk_enable, ce_out);
  input   clk;  // señal de reloj
  input   reset; // señal de reset
```

```
  input   clk_enable;  // señal de habilitación
  input   [11:0] In1;  // entrada 1 de 12 bits del ADC
  input   [11:0] In2;  // entrada 2 de 12 bits del ADC
  output  [11:0] Out1;  // salida 1 de 12 bits del DAC
  output  [11:0] Out2;  // salida 2 de 12 bits del DAC
  output  ce_out;      // Señal de disparo de salida
  parameter Constant_out1 = 4.0000000000000002E-004;
  wire enb;
  wire enb_1_1_1;
  real In1_1;
  real Add2_out1;
  real Unit_Delay2_out1;
  real Fcn_out1;
  real Constant_out1;
  real Trigonometric_Function1_out1;
  real Unit_Delay3_out1;
  real Fcn1_out1;
  real Product3_out1;
  real Product5_out1;
  real Sum1_out1;
  real Unit_Delay1_out1;
  real Product1_out1;
  real In2_1;
  real Add3_out1;
  real Trigonometric_Function_out1;
  real Product2_out1;
  real Product4_out1;
  real Sum_out1;
  real Unit_Delay_out1;
  real Product_out1;
////////////// ARQUITECTURA DE LA RED NEURONAL ////////////////
```

```
always @* Add2_out1 <= In1_1 - Product_out1; // red neuronal-2
always @* Product1_out1 <= Sum1_out1 * In1_1;
always @* Add3_out1 <= In2_1 - Product1_out1; // red neuronal 1
always @* Product_out1 <= Sum_out1 * In2_1;

////////// ARQUITECTURA DE LA REGLA DE APRENDIZAJE ///////////////
always @* Product3_out1 <= Trigonometric_Function1_out1 * Fcn1_out1;
always @* Product5_out1 <= Constant_out1 * Product3_out1;
always @* Sum1_out1 <= Product5_out1 + Unit_Delay1_out1;
always @* Product2_out1 <= Trigonometric_Function_out1 * Fcn_out1;
always @* Product4_out1 <= Constant_out1 * Product2_out1;
always @* Sum_out1 <= Product4_out1 + Unit_Delay_out1;

////////UNIDADES DE RETARDO CON RESET Y HABILITACIÓN ////////
always @ (posedge clk or posedge reset)
     begin: Unit_Delay2_process ///////// Unit_Delay2
     if (reset == 1'b1) begin
       Unit_Delay2_out1 <= 0;
     end
     else begin
       if (enb == 1'b1) begin
         Unit_Delay2_out1 <= Add2_out1;
       end
     end
   end // Unit_Delay2_process

always @ (posedge clk or posedge reset)
   begin: Unit_Delay3_process /////// Unit_Delay3
     if (reset == 1'b1) begin
       Unit_Delay3_out1 <= 0;
     end
```

```
      else begin
        if (enb == 1'b1) begin
          Unit_Delay3_out1 <= Add3_out1;
        end
      end
    end // Unit_Delay3_process
  always @ (posedge clk or posedge reset)
    begin: Unit_Delay1_process ///// Unit_Delay1
      if (reset == 1'b1) begin
        Unit_Delay1_out1 <= 0;
      end
      else begin
        if (enb == 1'b1) begin
          Unit_Delay1_out1 <= Sum1_out1;
        end
      end
    end // Unit_Delay1_process
  always @ (posedge clk or posedge reset)
    begin: Unit_Delay_process  ///// Unit_Delay
      if (reset == 1'b1) begin
        Unit_Delay_out1 <= 0;
      end  else begin
        if (enb == 1'b1) begin
          Unit_Delay_out1 <= Sum_out1;
        end
      end
    end // Unit_Delay_process
/////////////// VAR. AUX. //////////
     always @* In2_1 <= $bitstoreal(In2);
  always @* In1_1 <= $bitstoreal(In1);
  assign Out1 = $realtobits(Add2_out1);
```

```
  assign Out2 = $realtobits(Add3_out1);
  assign enb = clk_enable;
  assign enb_1_1_1 = clk_enable;
  assign ce_out = enb_1_1_1;
endmodule  // Red_HJ
```

Donde los puertos de entrada de la red neuronal son los registros de 12bits In1 e In2 y los puertos de salida son Out1 y Out2 de 12bits, además se requiere de señales de control como: reset(reset), reloj (clk) y habilitación (clk_enable). Para realizar la comunicación SPI, se utilizaron registros de conversión serial a paralelo y de paralelo a serial, también se utilizó una memoria ROM de 1KB para el almacenamiento de la función trigonométrica (tanh).

3. Resultados

Para realizar las pruebas se utilizó una señal de ruido y una señal de voz simuladas con MATLAB. En la Fig. 5, se muestra la señal de voz y una señal de ruido, ambas con una duración de cuatro segundos y con una frecuencia de muestreo de 44100Hz, y son utilizadas como señales originales.

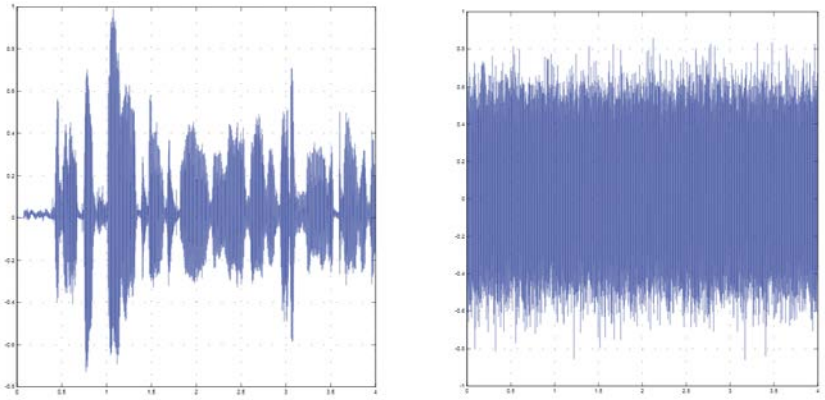

Fig. 5. Señales originales de simulación de voz y de ruido con respecto del tiempo.

Las señales de la Fig. 5 se mezclan utilizando bloques de ganancia y sumadores, dando como resultado las señales que se muestran en la Fig. 6.

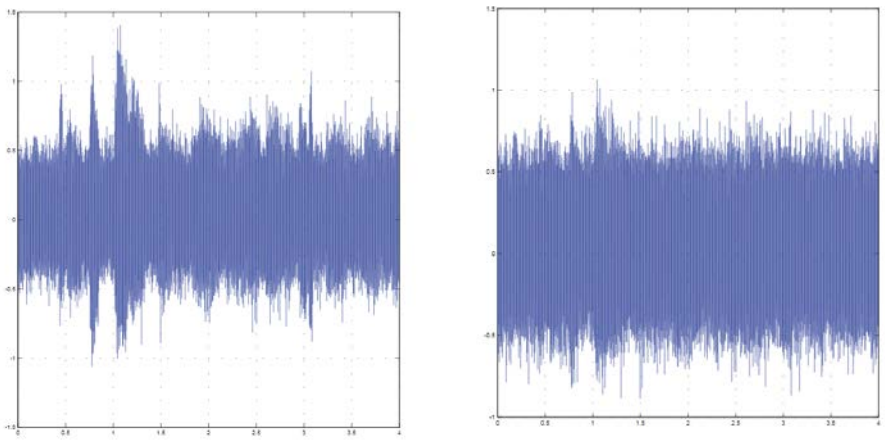

Fig. 6. Señales mezcladas de simulación e_1 y e_2 con respecto del tiempo.

En la Fig. 7 se muestra la adaptación de los pesos sinápticos w_{12} y w_{21} con respecto al tiempo, inicializados con cero y alcanzando su valor óptimo para la separación a 0.3 segundos aproximadamente.

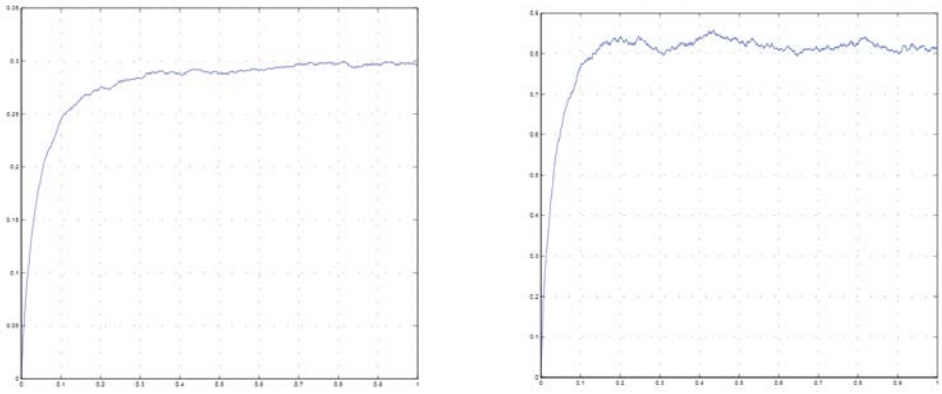

Fig. 7. Señales de simulación de adaptación de los pesos w_{12} y w_{21} con respecto del tiempo.

La Fig. 8 muestra las señales estimadas de salida de la red neuronal durante su adaptación y se observa que una vez que alcanzada dicha adaptación, la señal es recuperada nuevamente del ruido. La atenuación de la relación señal a ruido es aproximadamente de 30dB.

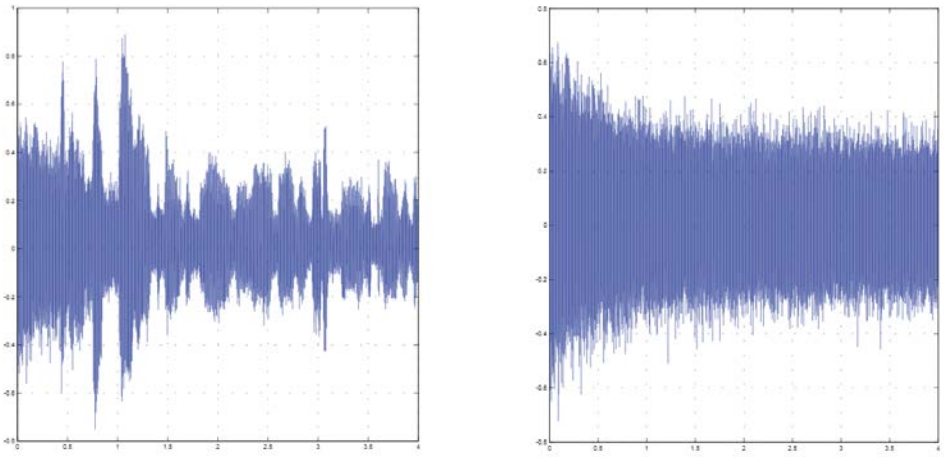

Fig. 8. Señales estimadas de simulación de salida con respecto del tiempo.

Para los resultados experimentales, las señales se generaron de forma independiente como se muestran en la Fig. 9.

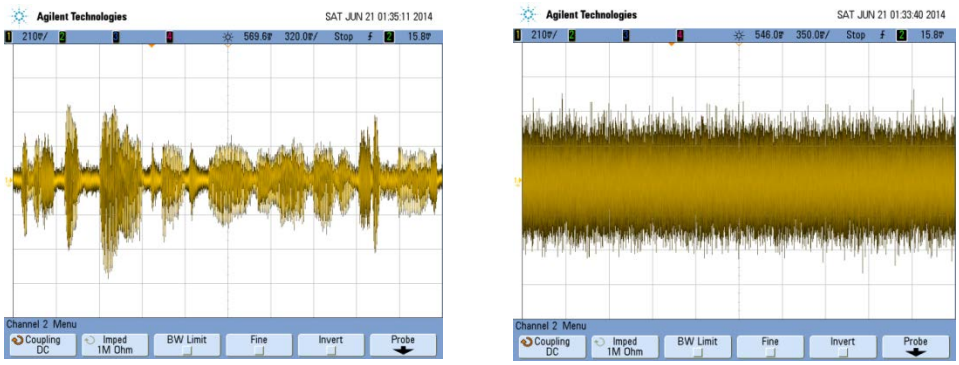

Fig. 9. Señales originales de voz y de ruido con respecto del tiempo.

Después se realizó la combinación lineal de las señales anteriores, a través de módulos de ganancia, para ser ingresadas a los ADC. En la Fig. 10 se muestran los resultados de la combinación.

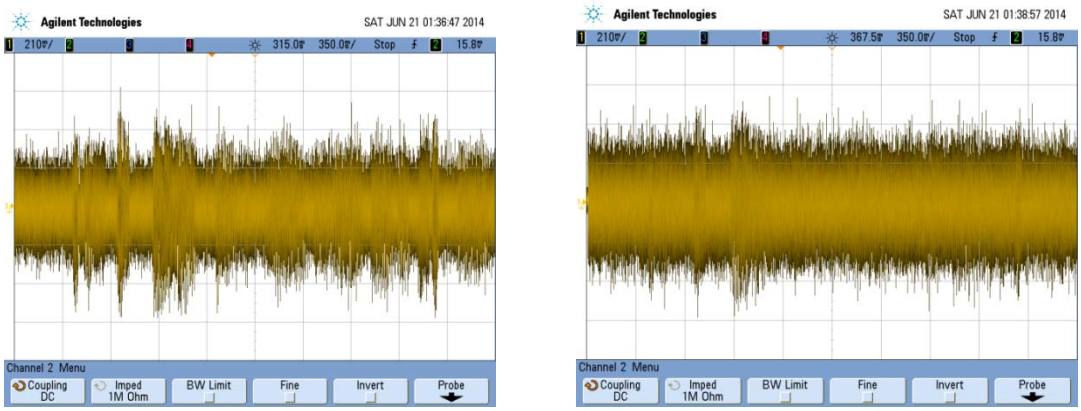

Fig. 10. Señales mezcladas e_1 y e_2 con respecto del tiempo.

En la Fig. 11 se muestra el resultado experimental durante la adaptación de las señales, a través del FPGA con el uso de un DAC.

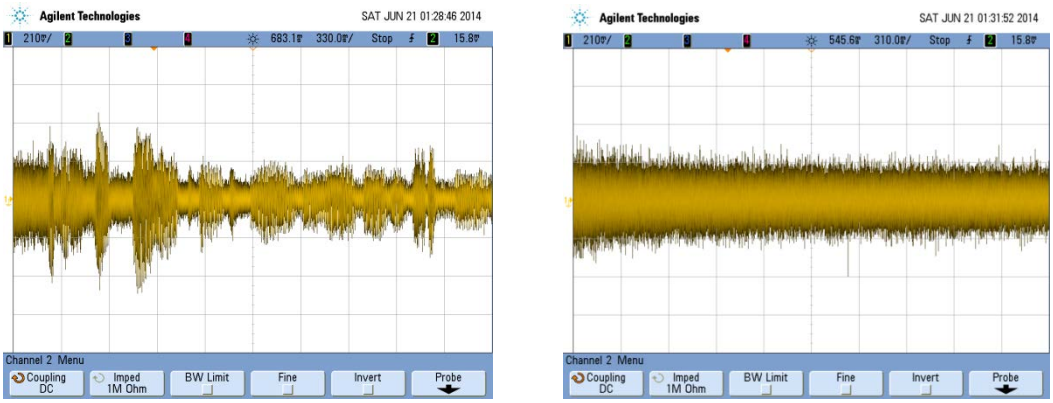

Fig. 11. Señales estimadas de salida con respecto del tiempo.

Una vez que la red neuronal ha alcanzado su adaptación, de las señales estimadas de salida se observa la recuperación de la señal original.

La atenuación de la relación señal a ruido es similar a los resultados de simulación realizados.

4. Conclusiones

La atenuación de la señal de ruido con respecto a la señal de voz obtenida con la arquitectura propuesta fue de 30dB aproximadamente, con lo que se demuestra que a una señal se le puede reducir el nivel de ruido utilizando un método estadístico (ICA) básico.

El diseño de la arquitectura digital dentro del FPGA funciona de manera paralela y dedicada, realizando exclusivamente su función, la de cancelar el ruido del ambiente en un dispositivo móvil, por ejemplo cuando se está realizando una llamada y se requiere atenuar o cancelar otro tipo de fuentes de señales no deseadas, entre otras aplicaciones.

El procesamiento de los datos en tiempo real dentro de un FPGA libera de múltiples tareas al procesador del equipo móvil.

Se planea como trabajo a futuro la comparación y optimización de la arquitectura para reducir el consumo de potencia sin perder la calidad de desempeño en el sistema.

Además de rediseñar su arquitectura sin ocupar más de los recursos necesarios para realizar esta tarea.

5. Referencias

[1] Comon, P. (1994) Independent component analysis, a new concept? Elsevier. Vol. 36. No. 3. pp. 287-314.

[2] Hyvärinen, A., Karhunen, J., & Oja, E. (2004) Independent component analysis. John Wiley & Sons Vol. 46.

[3] Oja, E. & Hyvarinen, A. (1997) A fast fixed-point algorithm for independent component analysis. Neural computation. pp. 1483-1492.

[4] Cohen, M. & Andreou, A. (1992) Current-mode subthreshold MOS implementation of the Hérault-Jutten auto adaptive network. IEEE journal of solid-state circuits, Vol. 27. No. 5. May.

[5] Haykin, S. (1999) Neural Networks: a comprehensive foundation. Prentice Hall, Inc. pp. 680-690.

[6] Cooper, G. & McGillem, C. (1986) Probabilistic Methods of Signal and System Analysis, 2da Ed. New York: Holt, Rinehart and Winston.

[7] Handong, B., Zhiguo, Z., Jing, H. & Zhiwen, L. (2011) FPGA Implementation of ICA Algorithm for Adaptive Noise Canceling. International Conference on Awareness Science and Technology (iCAST) 3rd Ed. pp. 452-456.

[8] Ounas, M., Chitroub, S., Touhami, R. & Yagoub, M. (2008) Digital circuit design of ICA based implementation of FPGA for real time Blind Signal Separation. MLSP. IEEE Workshop on Machine Learning for Signal Processing. pp. 181-186.

6. Autores

Dr. en C. Luz Noé Oliva Moreno con especialidad en Ingeniería Eléctrica de la Sección de Electrónica de Estado Solido del CINVESTAV-IPN en el Área de Diseño VLSI

M. en C. Margarita Elizabeth Gómez Mayorga obtuvo su título de Maestría en Tecnología Avanzada con especialidad en Procesamiento de Imágenes en la Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas del IPN

M. en C. Claudia Alejandra López Rodríguez con especialidad en Ingeniería Eléctrica de la Sección de Electrónica de Estado Solido del CINVESTAV-IPN.