UNIVERSIDAD NACIONAL DE COSTA RICA
DOCTORADO EN CIENCIAS NATURALES PARA EL DESARROLLO
ÉNFASIS EN TECNOLOGÍAS ELECTRÓNICAS APLICADAS

ARQUITECTURAS MICROELECTRÓNICAS PARA APLICACIONES DE PROCESAMIENTO NUMÉRICO LOCAL

Tesis para optar por el grado académico de Doctora en Ciencias Naturales para el Desarrollo, Énfasis en Tecnologías Electrónicas Aplicadas

Ana Clevis Lozano Rivera

Diciembre 2019

MIEMBROS DEL TRIBUNAL EXAMINADOR

FRANCISCO SAN LEE CAMPOS (FIRMA)
PERSONA FISICA, CPF-01-0750-0349.
Fecha declarada: 20/03/2024 02:33:50 PM
Esta es una representación gráfica únicamente,
verifique la validez de la firma.

Dr. Francisco San Lee Campos

Representante del Consejo Central de Posgrado

Dr. Giovanni Sáenz-Arce

Coordinador del posgrado

Dr.-Ing. Alfonso Chacón Rodríguez

Tutor de tesis

Dr. Carlos Meza

Miembro del Comité Asesor

Dr. Roberto Pereira

Miembro del Comité Asesor

Ana Clevis Lozano Rivera

Sustentante

Dedicatoria

a Asdrúal y Francy

Agradecimientos

Al finalizar una etapa de crecimiento profesional llena de tantos altibajos, no encuentro palabras para expresar el sentimiento de gratitud hacia todas aquellas personas, que de una u otra forma, me brindaron un espacio de su tiempo para una consulta, una palabra de aliento, un consejo, un café o una broma. A todos los colegas, administrativos, personal técnico y los jóvenes del DCILab de la Escuela de Ingeniería Electrónica del Tecnológico de Costa Rica les agradezco infinitamente.

Un agradecimiento especial a mi director de tesis Dr-Ing. Alfonso Chacón Rodríguez por su confianza, consejo oportuno en los momentos difíciles de éste proceso, por su gran paciencia y por no dejarme desfallecer durante el tiempo de desarrollo de ésta investigación.

Una compañera que se convirtió en amiga, en este casi interminable camino, Cindy Calderón, quien siempre dispone un minuto para escuchar mis angustias, brindarme un hombro donde apoyarme o simplemente compartir nuestras experiencias disfrutando de un café, muchas gracias amiga.

Por último, pero no menos importante, a mi esposo y a mi hija, quienes siempre me han apoyado incondicionalmente, brindándome no solo su amor, sino su compañía en esta aventura que enmarca una dura etapa de convivencia con grandes retos que afianzaron los lazos que unen ésta pequeña familia. A mis dos ángeles, mi gratitud eterna.

Índice general

Índice de figuras

Índice de tablas

Lista de Abreviaturas y Acrónimos

- DSP: Procesadores digitales de señales.
- ASIC: Circuito Integrado de aplicaciones especificas.
- FPGA: Arreglos Lógicos Programables en Campo.
- HDL: Lenguaje de Descripción de Hardware.
- RTL: Nivel de transferencia de registros
- VLSI: Integración a gran escala.
- SoC: Sistema en un chip.
- CMOS: Semiconductor complementario metal-oxido.
- EDA: Automatización de diseño electrónico.
- ICA: Análisis de componentes independientes
- EMA: Promedio de movimiento exponencial.
- STC: Condiciones de pruebas estándar.
- PVS: Sistema fotovoltaico.

Capítulo 1

INTRODUCCIÓN

1.1. Enfoque general y justificación de esta tesis

Los resultados de la investigación científica en ingeniería debe medirse en términos de sus efectos sobre el entorno, pues promueve el desarrollo no solo sustentable, sino que además, generan mejoras en las condiciones sociales y económicas de la población actual y futura. En el caso particular de la investigación en ingeniería, los resultados conducen, muchas veces, al desarrollo de nuevas tecnologías que inducen cambios sustanciales en la historia de la humanidad. Un ejemplo es la invención de los minúsculos circuitos integrados (ICs) de silicio, que trajo consigo la integración de miles de millones de transistores en un solo componente y, consecuentemente, el vertiginoso avance de los microprocesadores. El avance tecnológico ha impulsado la creación de complejos sistemas más compactos, robustos y eficientes que mejoran la vida cotidiana. Estos demuestran que la generación de conocimiento científico a través del método ingenieril es un mecanismo crucial de estructuración y dinámica social y, por ende, un factor que transforma la productividad mundial.

El aumento en la capacidad de computación numérica derivada de la integración de

circuitos integrados junto con el estudio de los fenómenos naturales cotidianos, ha llevado a la formulación científica de los procesos mediante modelos no lineales, de difícil sino imposible resolución simbólica. Un ejemplo clásico de esta integración es la dinámica de fluidos, donde en los años sesenta, Edward Lorenz estudió el problema de convección, derivando el sistema de tres ecuaciones diferenciales ordinarias no lineales [8]. Dicho estudio demostró que no eran necesarios múltiples formas para obtener estados complejos en fluidos, y que a pesar de conocer las condiciones iniciales, los sistemas deterministas son en principio predecibles. En la práctica esta predictibilidad puede ser difícil de obtener, debido a la sensibilidad frente a las condiciones de reposo del sistema, lo que vuelve los cálculos extremadamente pesados [9].

Para la solución de las ecuaciones diferenciales determinísticas que describen el comportamiento de un sistema natural con características especificas, se ha creado gran variedad de robustas herramientas computacionales. Dichas herramientas permiten implementar complejos algoritmos y definir criterios para evaluar su rendimiento y comportamiento. Estos criterios se centran principalmente en el uso eficiente de los recursos computacionales, que suelen medirse en función de dos parámetros: el espacio, es decir, memoria que utiliza, y el tiempo, lo que tarda en ejecutarse [10]. Ambos criterios representan los costos que supone encontrar una solución al problema planteado mediante un algoritmo. No obstante, el uso de la computación digital tradicional basada en microprocesadores de proceso general para el estudio de estos fenómenos —construidos siguiendo las arquitecturas inspiradas por Von Neumann [11, 12]—, ha conllevado la limitante de que muchos de los mismos deban llevarse a cabo en laboratorios o instalaciones dotadas del costoso equipo computacional especializado, y no necesariamente en el campo mismo, donde pueden ser de más utilidad.

Una potencial solución alternativa para implementar los modelos matemáticos que describen un sistema no lineal son los procesadores digitales de señales (DSPs, del inglés

Digital Signal Processing), como apoyo a sistemas embebidos con procesadores de propósito general. Pero a medida que aumenta la complejidad en los algoritmos de procesamiento se requieren sistemas con mejor desempeño y de menor tiempo de ejecución. Los DSPs son altamente flexibles debido a su programabilidad, pero por su esquema secuencial, se ven limitados en aplicaciones de muy alta frecuencia (aquellas donde las tasas de muestreo superan el millón de muestras por segundo) [4].

De ahí el creciente interés en arquitecturas de procesamiento numérico local dedicadas y a la medida, que permitan una aplicación directa y en línea sobre el estudio del problema, sin limitantes por portabilidad y flexibilidad. Estas soluciones pueden implementarse ya sea sobre circuitos integrados de aplicación específica (ASICs, del inglés Aplication Specific Integrated Circuit) o en Arreglos Lógicos Programables en Campo (FPGAs, del inglés Field-programmable Gate Array), incluso dotados de su propias unidades de DSP y de procesamiento general en un SoC (del inglés System on Chip). No obstante, en tamaño y velocidad los FPGAs son equiparables, en ciertos casos, a los ASICs, pero los FPGAs por ser más flexibles, tener un ciclo de diseño más corto y con herramientas de diseño más asequibles, se han convertido en la primera opción para los ingenieros [4].

Actualmente, los fabricantes de FPGAs compiten para optimizar al máximo la implementación de ciertas operaciones, como sumas y productos, para conseguir un rendimiento elevado con un mínimo costo de recursos. Los complejos algoritmos implementados en estas unidades dedicadas, pueden hacer uso intensivo de estas operaciones aritméticas sin perjudicar su desempeño [10]. Existen ejemplos de uso en aceleración de simulaciones numéricas tan complejas, como el modelado de redes neuronales biológicamente realistas, tal como los presentados en [13] (desarrollado en el laboratorio de diseño de circuitos integrados DCILab, de la Escuela de Ingeniería Electrónica del ITCR, en el que se llevó a cabo la misma tesis), que muestra precisamente las ventajas de este tipo de tecnologías para obtener rápidos resultados de manera eficiente.

En el caso particular de la presente tesis, se ha considerado la necesidad de aplicar el método iterativo (por iteraciones) de diseño en ingeniería para resolver dos problemas complejos de procesamiento numérico, aplicados en sistemas robustos que deben trabajar en ambientes industriales y, de ser posible, estar implementados en una FPGA verificando la factibilidad de integrarse en un ASIC, de bajo consumo de potencia. Con esta premisa en mente la tesis se ha centrado en el análisis de dos problemas reales particulares: por un lado, a la problemática de la localización de un objeto móvil en un ambiente industrial altamente ruidoso, tal como el esperado en una instalación portuaria o industrial, con decenas de operarios y de máquinas automatizadas que deben mover o procesar gran cantidad de material con el mínimo riesgo para las personas involucradas. Cabe aquí anotar que este problema fue el que dio inicio a esta tesis, enmarcada dentro del proyecto internacional denominado "3D Gigascale Integrated Circuit for Nonlinear Computation, Filter and Fusion with applications in Industrial Field Robotics" [2], financiado por la Agencia Nacional de Promoción Científica y Tecnológica de Argentina, en el cual participaban destacados investigadores de Argentina, Estados Unidos, Australia y Costa Rica, el cuál incluía los fondos necesarios para la fabricación de los prototipos ASIC. Este proyecto se nutría de la experiencia del DCILab no solo en desarrollo en FPGAs, tal como ya se vio en [13], así como en múltiples aplicaciones ya integradas en silicio, como las descritas en [14, 15, 16, 17, 18, 19, 20, 21], algunas de las cuales se retomarán más adelante como base para las soluciones que se propondrán en esta tesis. Sin embargo, dicho proyecto se vio afectado por la crisis económica Argentina, lo que impidió al final contar con los fondos esperados, y obligó a reorientar parcialmente la tesis. Por otro lado, a consecuencia de los buenos resultados obtenidos en la pasantía de investigación, realizada como parte de los requisitos del programa de doctorado, se determinó la importancia de desarrollar un sistema local para la estimación de parámetros eléctricos de un sistema fotovoltaico, que permite el monitoreo en tiempo real de la potencia suministrada, con miras a la administración automática de una red inteligente compuesta por decenas o centenas de paneles

solares.

El reto al que se enfrentó ésta tesis fue por tanto establecer una metodología para implementar los dos sistemas, que por estar descritos mediante ecuaciones diferenciales no lineales presentan un alto grado de complejidad algorítmica. Además, aprovechar las capacidades de procesamiento numérico de las arquitecturas a gran escala VLSI (del inglés, Very Large Scale Integration), ya sea implementadas en una FPGA o en un ASIC, para ofrecer soluciones que sirvan de base para la construcción de sistemas de producción y generación más eficientes, seguros y confiables –incluso en términos energéticos– brindando así un mejoramiento en las condiciones existenciales de la humanidad.

1.2. Objetivos y metodología

A partir de lo expuesto en la sección anterior, se definió como objetivo general diseñar arquitecturas microelectrónicas eficientes enérgicamente, para procesamiento de señales en aplicaciones de sistemas dinámicos no lineales, usando metodologías integrales de diseño de hardware que permiten llevar a cabo la comprobación funcional y física del circuito integrado de aplicaciones específicas.

Para alcanzar el objetivo general de la tesis, fue necesario cumplir con una serie de objetivos parciales, cada uno por sí fundamentado en el proceso metodológico iterativo de diseño en ingeniería.

- Seleccionar una solución eficiente al problema inicial planteado de separación de fuentes sonoras, lo que requeriría primero el estudio de los algoritmos más prometedores en dicha área, y cuantificar de una manera sólida sus posibilidades de implementación en hardware.

- Proponer una solución para la estimación de parámetros eléctricos de un sistema fotovoltaico, que permitiese controlar en tiempo real la potencia suministrada para el aprovechamiento de recursos renovables, y cuantificar sus posibilidades de implementación en hardware.

- Modelar las estructuras aritméticas derivadas de las soluciones propuestas en los objetivos anteriores, siguiendo la metodología de diseño sincrónico RTL (del inglés, Register Transfer Level), que no presente restricciones asociadas a la elección de una determinada tecnología.

- Implementar las soluciones RTL en un dispositivo FPGA, para su validación numérica con datos de campo.

- Trasladar la solución a una estructura física comprobada y correcta para su fabricación en un potencial ASIC, siguiendo los pasos estipulados en el flujo de diseño digital de circuitos integrados.

Para lograr los objetivos específicos antes mencionados, fue necesario un extenso proceso de validación y verificación funcional de los algoritmos matemáticos después de ser implementados en hardware los dos casos elegidos para este trabajo. La verificación incluyó la caracterización de circuitos a través de mediciones reales y el contraste con los resultados de modelos de referencia creados en software de alto nivel.

En todo caso, un extenso proceso de diseño digital fue implementado, partiendo de la sub división del sistema, la definición de las condiciones iniciales de operación y los criterios de convergencia, entre otros aspectos que requieren largos procesos de verificación antes de llevar el diseño a un nivel de abstracción mayor, RTL. No llevar a cabo este proceso completo conduce a la necesidad de rediseños, y a la pérdida del control de lo que está ocurriendo con la síntesis. Estos procesos de verificación en alto nivel, basados en simulación, requieren la creación de entornos de prueba complejos, laboriosos y propensos

a errores con la posibilidad de no cubrir todos los casos de prueba necesarios.

Durante años se ha tratado de crear técnicas que ayuden a solventar estos problemas a los diseñadores tratando de evitar que la verificación se convierta en el cuello de botella del diseño de un sistema digital. No obstante, cabe aclarar que el uso de las técnicas avanzadas de verificación existentes en la industria escapa a la meta general de esta tesis, centrada en ofrecer pruebas de concepto sobre la solución de sistemas dinámicos no lineales de manera eficiente. Es por ello que la única técnica de validación realizada en esta tesis es funcional.

1.3. Contribuciones de la tesis

La rápida evolución de los componentes electrónicos, su elevada velocidad y su reducido costo [17, 22], cada día hace posible el desarrollo de aplicaciones reales que anteriormente solo eran abordables directamente a través de software sobre sistemas computacionales no portátiles. Esto aumenta las posibilidades de crear nuevos productos portátiles (tales como los llamados "wearables"), para ello se hace necesaria la aplicación tanto de los fundamentos teóricos como la aplicación de una metodología de implementación de sistemas descritos mediante ecuaciones no lineales y llevarlas al diseño de circuitos integrados, que garanticen el correcto funcionamiento de cada modelo en particular.

Esta tesis ha demostrado con resultados satisfactorios la aplicación de un método para el desarrollo de dos problemas reales específicos, que contienen un alto grado de procesamiento numérico de datos de sistemas no lineales. Este proceso puede extenderse a otros problemas en los que sea necesaria la computación localizada. Por ello, se presenta todo el trasfondo experimental a partir del cual se fundamentó el diseño de los dos circuitos integrados ASICs.

Por otro lado, se ha comprobado funcionalmente dos prototipos en ASIC usando tecnología CMOS, los cuales pueden conducir a la creación de nuevos productos que contribuyan a mejoras en la calidad de operación de cada uno de los sistemas estudiados.

1.4. Organización de la tesis

El Capítulo 2 resume los fundamentos y antecedentes teóricos generales de esta tesis, y describen la metodología de diseño digital microelectrónico, desde la implementación de un modelo dorado hasta la síntesis física en una tecnología CMOS (del inglés, complementary metal-oxide-semiconductor). Se da especial énfasis a las técnicas de diseño CMOS de bajo consumo en tecnologías de bajo costo.

El Capítulo 3 establece las bases del primer problema que se abordó en la tesis: el desarrollo de un sistema de separación ciega de fuentes sonoras y de bajo consumo, para aplicaciones en seguridad en ambientes industriales. Para ello, se realizó el estudio de tres algoritmos de separación de fuentes que utilizan técnicas de análisis de componentes independientes (ICA), con el fin de determinar su potencial portabilidad a un circuito integrado. Los algoritmos estudiados fueron: FastICA, Adaptativo basado en Gradiente Natural y Adaptativo Basado en Gradiente Relativo. El análisis se centró en el estudio de su costo computacional, el potencial costo de traslado a hardware, la validación de las características de operación de cada uno, el tiempo de convergencia y en la calidad de la separación de las fuentes de cada uno. El resultado final de este capítulo es el definir qué algoritmo es más viable para su implementación física.

El Capítulo 4 exhibe los resultados de la implementación del algoritmo adaptativo basado en gradiente natural, escogido en el capítulo anterior. El modelo validado en MATLAB

sirvió como marco de referencia o modelo dorado para la correcta operación. El algoritmo se trasladó a su implementación en hardware mediante un lenguaje de descripción de hardware (HDL) validado sobre una FPGA. Posteriormente, se realizó su implementaci ón en una tecnología CMOS comercial de 180 nm mediante el flujo EDA (del inglés, Electronic Design Automation) de circuitos integrados.

El Capítulo 5 presenta el análisis y la evaluación de la implementación en FPGA de un modelo lineal para la estimación de parámetros eléctricos de un sistema fotovoltaico PVS (del inglés Photovoltaic System), teniendo en cuenta la portabilidad en hardware de estructuras aritméticas de aplicaciones reales, ya sea en un FPGA o un ASIC de bajo costo.

Finalmente, en el Capítulo 6 se recopilan las principales conclusiones y aportes generados, además de las futuras líneas de investigación que se desprenden de ésta tesis.

Capítulo 2

ANTECEDENTES Y FUNDAMENTOS TEÓRICO

2.1. Introducción

El procesamiento digital de señales (DSP, por sus siglas en inglés Digital Signal Processing) siendo un tema inmerso en un extenso conjunto de disciplinas posee una amplia historia. El DSP trata de la representación, transformación y manipulación de señales y de la importancia que el proceso representa en cada aplicación [23, 24, 25]. En la mayoría de ellas es deseable que los sistemas funcionen en tiempo real, es decir, que el sistema en tiempo discreto se implemente de forma que las muestras de salida se calculen con la misma velocidad a la que se muestrea la señal en tiempo continuo. Algunas aplicaciones requieren análisis previo de los datos, para lo cual se utilizan modernas herramientas de software se reducen en varios órdenes de magnitud el tiempo de cómputo [1, 24].

Uno de los desarrollos importantes en la historia del DSP ocurrió en el terreno de la electrónica. Aunque los primeros microprocesadores eran demasiado lentos (60 - 200 MHz)

[1] para implementar en tiempo real, en la actualidad la tecnología de los circuitos integrados ha avanzado hasta el nivel de permitir las operaciones en coma fijo y en coma flotante, con arquitecturas especialmente diseñadas para ejecutar algoritmos de procesamiento en tiempo discreto [1, 4].

Por otro lado, los sistemas no lineales se producen naturalmente en muchas áreas de la ingeniería, como el análisis estructural, la dinámica y los circuitos eléctricos, entre otras [24, 26]. Los microproccesadores han permitido resolver de forma rápida y precisa sistemas de ecuaciones cada vez más grandes. Esto no solo permitió a los ingenieros manejar problemas cada vez más complejos donde los sistemas lineales ocurren de manera natural, sino que también abrió las puertas para resolver problemas donde no ocurren de manera natural. Es así como se ha convertido en una práctica estándar, analizar un problema transformándolo en un sistema de ecuaciones lineales y luego resolverlas por medio de software especializado [23, 1].

En línea con los objetivos de la presente investigación, se abordan en este capítulo los fundamentos teóricos básicos para la transformación de señales no lineales a estructuras lineales, para dos casos de estudio: la estimación de parámetros eléctricos de sistemas fotovoltaico y la separación ciega de fuentes. Se resumen los antecedentes de cada aplicación especifica y la metodología de diseño digital para la implementación física en tecnología CMOS.

2.2. Sistemas dinámicos

El modelado de los sistemas dinámicos se encuentra en el centro de atención de los científicos desde hace muchas décadas. Dicha atención se dirige a obtener información necesaria para el control automático de los sistemas dinámicos [24]. Estos modelos necesitan

simular el comportamiento real en los casos en que existe un conocimiento previo limitado de la estructura del sistema. Los sistemas dinámicos pueden ser técnicos, como los circuitos eléctricos o mecánicos, puede ser biológicos como el control de la presión arterial o el control de insulina, pero también están presentes en muchas otras ramas como la economía, la sociedad, las ciencias, entre otras. En todos los sistemas es necesario modelar dinámicamente para conocer las condiciones iniciales de funcionamiento y su evolución en tiempo real [24]. En la actualidad existen gran variedad de herramientas CAD (del inglés, Computer Assitance Design) para el diseño preliminar de sistemas en tiempo real, estas herramientas llevan a cabo complejas operaciones aritméticas que permiten simular y definir los parámetros más óptimos de procesamiento de la señales de los sistemas implementados [23, 24, 27].

El modelo de un sistema dinámico representa una aproximación de un proceso real, de alguna máquina o mecanismo. Normalmente, su representación se realiza mediante un modelo finito que describe la dinámica del sistema, específicamente su funcionamiento. Dicho modelo se describe por medio de ecuaciones diferenciales ordinarias de primer orden, principalmente no lineales, las cuales representan las variables del sistema y su orden depende de la cantidad de variables o detalles del proceso que se tomen en cuenta [23]. Los coeficientes de las ecuaciones diferenciales representan los parámetros propios de cada sistema, que deben ser estimados con un alto grado de precisión, para lograr el punto óptimo entre las predicciones del modelo y los datos observados en el campo [23, 24].

En la predicción del comportamiento dinámico de un sistema (fase de análisis), ya sea en su evolución temporal o frecuencial (fase de diseño), se requiere conocer y entender el modelo matemático. En las últimas décadas se ha desarrollado una gran variedad de algoritmos aritméticos que permiten la estimación de parámetros de salida de un sistema dinámico, obteniendo así una aproximación lineal de un sistema no lineal [23, 24].

2.2.1. Proceso de modelado de sistemas dinámicos

El proceso de modelado analítico se divide en tres grandes etapas. La primera de ellas consiste en la delimitación del modelo en función de los fenómenos que resultan relevantes de acuerdo al problema que se quiere resolver. Esta es una etapa que no puede sistematizarse fácilmente y que requiere una cierta dosis de intuición y por sobre todo de una vasta experiencia en relación con el sistema a modelar [24]. Una vez delimitados los parámetros que se consideraron relevantes para la construcción del modelo, se pasa a la siguiente etapa en la que se deben formalizar las relaciones constitutivas y estructurales asociadas a los fenómenos considerados y a la forma en que estos se ejecutan dentro del sistema. En los sistemas físicos, dichas relaciones constitutivas y estructurales encuentran su expresión formal (matemática) en las leyes físicas fundamentales de los dominios asociados a los fenómenos mencionados. Por último, se realiza la etapa de validación del modelo propuesto [24, 28].

Cuando nos encontramos con un sistema real que se quiere modelar, se plantean esquemas ideales, que son producto de simplificaciones que se realizan acorde al problema a estudiar. Es muy importante no perder de vista que los modelos obtenidos resultarán adecuados sólo para resolver determinados problemas y dentro de un rango de operaciones dado [29]. Es decir, el sistema físico idealizado dependerá no sólo del sistema real en sí, sino también del problema a resolver y del intervalo de validez que se pretenda tener para el modelo resultante. Lamentablemente, no hay una metodología que nos permita realizar estas simplificaciones de forma sistemática. Esta etapa del modelado (que es quizás la más importante en virtud de que todo el resto dependerá de lo que se haga aquí) se resuelve en gran medida a partir de consideraciones sujetas a la experiencia y al conocimiento del proceso real. Sin embargo, los sistemas complejos pueden habitualmente dividirse en subsistemas más simples de los cuales se encuentran modelos en términos de simplificaciones ya probadas en problemas similares. Después del modelado tendremos un número de

relaciones matemáticas que vincularán las variables descriptivas del modelo y que serán consecuencia tanto de los fenómenos físicos considerados (relaciones constitutivas) como de la interacción entre los mismos en función de su disposición en el sistema (relaciones estructurales) [24, 28].

De esta forma, una vez que se tiene el sistema físico idealizado el siguiente paso hacia la obtención de un modelo matemático será el de reunir las relaciones constitutivas y estructurales involucradas. En muchos casos encontraremos conjuntos de relaciones matemáticamente equivalentes, por lo que no serán necesarias todas las relaciones para la construcción del modelo. Cuando se obtienen todas las relaciones matemáticas necesarias, aún el modelo matemático no es muy útil, lo que se ha logrado es una mezcla de ecuaciones diferenciales y algebraicas. Dado que se trata con sistemas físicos bajo hipótesis de parámetros concentrados, se busca llegar a los modelos existentes en la teoría de ecuaciones diferenciales. En general, se utilizan dos tipos de expresiones: Las ecuaciones Diferenciales Ordinarias (EDO) y los sistemas de ecuaciones de estado y de salida [24].

Un sistema dinámico descrito mediante las ecuaciones (2.1 y 2.2), donde la ecuación (2.2) es la salida del sistema, se compone de un conjunto de EDO, donde $f(t)$ y $h(t)$ son funciones que representan la dinámica del sistema. Dichas funciones están en términos de la variable de estado $x(t)$ y la entrada $u(t)$ [25].

$$\dot{x}(t) = f(x(t), u(t)); \qquad x(t_0) = x(0) \tag{2.1}$$

$$y(t) = h(x(t)) \tag{2.2}$$

Por otra parte, un modelo de entrada-salida describe un sistema dinámico basándose en los datos de entrada y de salida. Este tipo de modelos supone que la salida del sistema puede ser predicha a partir de las entradas y salidas pasadas del mismo. Si el sistema se

supone, además determinista, invariante en el tiempo, de una entrada - una salida (SISO), el modelo de entrada-salida es representado en la ecuación (2.3) [25]:

$$y(k) = f(y(k-1), y(k-2), ..., y(k-n), u(k-1), u(k-2), ..., u(k-m)) \tag{2.3}$$

donde u(k), y(k) representa el par de entrada-salida en el tiempo k. Los enteros positivos n y m son el número de salidas pasadas (también llamado el orden del sistema) y el número de entradas pasadas. En la práctica m es, normalmente, menor o igual que n. $f(y)$ puede ser cualquier función no-lineal definida desde el espacio de entradas y salidas pasadas hasta el espacio de salidas futuras [24, 28].

Después de la estimación, la pregunta es, si el modelo que se ha derivado es adecuado para el uso que se le pretende dar. Éste es el difícil y subjetivo problema de la validación. La mejor manera de atacar el problema es confrontar al modelo con todas las fuentes de información disponibles, incluyendo el conocimiento previo, los datos experimentales y la experiencia usando el modelo. Para tener más confianza en el modelo el consejo general es emplear tantas herramientas de validación como sea posible [24].

2.2.2. Sistemas dinámicos no lineales

Los avances tecnológicos actuales han generado una enorme variedad de nuevos problemas y aplicaciones que son no lineales en esencia. Los sistemas dinámicos no lineales constituyen la mayoría de los sistemas que encontramos en nuestro alrededor. Los sistemas dinámicos no lineales son aperiódicos pero determinísticos con un patrón asociado a su comportamiento, además, son dependientes de condiciones iniciales, que permiten llevarlos a una solución, una pequeña variación de dichas condiciones resultará en grandes diferen-

cias en el comportamiento a lo largo del tiempo, efecto conocido como efecto mariposa [25, 26].

El estudio de los sistemas dinámicos no lineales comprende dos partes: una más elemental de carácter local, que se relaciona con el estudio de sistemas lineales cuando el punto de equilibrio entorno al que se estudia el comportamiento del sistema es hiperbólico; y una parte más específica en la que se ponen en evidencia comportamientos completamente nuevos con relación a los que presentan los sistemas lineales [25, 26].

Un sistema dinámico no lineal puede representarse por una EDO de la forma:

$$\frac{dy(t)}{dt} = F(y(t)) \qquad y(t) = [y_1(t), y_2(t), ..., y_n(t)] \tag{2.4}$$

donde F es un campo vectorial no lineal y n es la dimensión del sistema. asumiendo que bajo condiciones normales de trabajo el sistema opera a lo largo de una trayectoria $y(t)$.

Para llevar a cabo la estimación de los parámetros de un modelo no lineal, éste puede expresarse como un modelo matemático lineal. La linealización del sistema se realiza tomando cada término no lineal de la EDO y analizando su comportamiento alrededor del punto de trabajo o de la trayectoria de operación. A modo de ilustración la ecuación 2.5 representa un modelo no lineal, donde el sistema alcanza su régimen permanente estacionario caracterizado por $x(t) = \bar{x}$, $\dot{x}(t) = 0$ y $\ddot{x}(t) = 0$ [24].

$$\frac{1}{m}F(t) = \ddot{x}(t) + \frac{b}{m}(\dot{x}(t))^2 + \frac{k}{m}x(t) \tag{2.5}$$

Donde m, b y k son los coeficientes que describen el estado del sistema en el punto de estudio, dadas las condiciones iniciales.

El término cuadrático de la EDO anterior se debe linealizar. Asumiendo que el movimiento del sistema no lineal se encuentra en los alrededores de la trayectoria nominal del mismo, y que la distancia desde la trayectoria es tan pequeña como $\Delta x(t)$, dando como resultado la ecuación 2.6 [24].

$$\frac{\Delta F(t)}{m} = \Delta\ddot{x}(t) + k_1\Delta\dot{x}(t) + \frac{k}{m}\Delta x(t) \tag{2.6}$$

Es natural asumir que el sistema se mueve en estrecha proximidad con la trayectoria nominal y de igual manera será sustentado por las nuevas entradas escaladas a una pequeña cantidad $\Delta F(t)$. En general, el sistema obtenido es variante en el tiempo. Es importante definir las condiciones iniciales para el sistema linealizado, de manera tal que permanezca cercano a la trayectoria nominal [24, 28].

2.2.3. Discretización de sistemas no lineales

La noción de sistemas en tiempo discreto cobra auge a raíz del empleo de las computadoras digitales, las cuales almacenan información, como señales de sensores, sonidos e imágenes, a intervalos discretos de tiempo, la capacidad del almacenamiento se encuentra limitada por la resolución y en última instancia, dicha resolución se traduce en dígitos significativos. Resulta entonces imposible, además de innecesario, almacenar las señales para todos los instantes de tiempo en un intervalo dado, por tal motivo es necesario seleccionar el intervalo apropiado para efectuar esta labor [24, 25].

Los sistemas en tiempo discreto se representan por medio de ecuaciones en diferencias, usualmente, aunque no de manera exclusiva, se trata de expresiones obtenidas como aproximación de las ecuaciones diferenciales. Aún cuando sea el resultado de la medición de una cantidad en un sistema físico en tiempo real, el hecho de considerar al procesador digital como una parte del sistema hace de todo el conjunto un sistema discreto, ya que el almacenamiento y procesamiento de información se realizan a instantes definidos por un periodo de muestreo, determinado por el ciclo del reloj interno del microprocesador [24, 25].

En la literatura de los sistemas dinámicos lineales se encuentran varios métodos para la discretización de los modelos de sistemas en tiempo continuo a tiempo discreto. Comúnmente se usan, la aproximación integral y el método de Euler [24]. El método de aproximación definido por la ecuación 2.7, se basa en la suposición que las entradas al sistema son constantes durante el periodo de muestreo por su forma escalonada.

$$f(t) = f(kT), \quad kT \leq t < (k+1)T, \qquad k = 0, 1, 2, ... \tag{2.7}$$

El método de Euler, es un método sencillo y menos preciso que el de aproximación integral. Es basado en el cálculo de la primera derivada en el instante de tiempo $t = kT$, como lo indica la ecuación 2.8. En la aproximación de Euler es necesario tener cuidado con la elección del intervalo de muestreo T, el cual debe ser suficientemente pequeño para obtener resultados satisfactorios [24].

$$\dot{x}(t) = \frac{dx(t)}{dt} \approx \frac{1}{T}(x((k+1)T) - x(kT)) \tag{2.8}$$

2.3. Métodos de Separación ciega de fuentes

La separación ciega de fuentes (en inglés Blind Source Separation, en adelante BSS) fue inicialmente propuesto por Herault y Jutten en 1991 [27], como una solución matemática a un problema biológico que ocurre en el sistema nervioso central cuando se transmite información mezclada por distintas fibras nerviosas [30, 31]. A partir de la formulación inicial de la BSS, se ha desarrollado un sinnúmero de métodos para la separación de fuentes en diversas áreas, principalmente en aplicaciones de procesamiento de señales acústicas.

Uno de los métodos matemáticos más utilizado para resolver el problema BSS, es el método de análisis de componentes independientes (en inglés, Independent Component Analysis, en adelante ICA), el cual ha sido el punto de partida de muchos algoritmos de aplicación específica [32, 33]. En todos los algoritmos desarrollados, el objetivo es separar una o varias fuentes linealmente superpuestas, o mezcladas. Dado que, los datos tienen una estructura lineal, la dificultad reside en estimar la **matriz de mezcla**, es decir, el conjunto de coeficientes de la superposición lineal, que es desconocida y que refleja la geometría del problema [34, 35, 36].

El problema de BSS parte de la suposición que las mezclas, captadas por varios sensores, son el resultado de la superposición lineal de las fuentes originales, de las cuales se desconocen características tales como: el retardo en el tiempo, la localización espacial, condiciones ambientales, entre otras. Además, el único criterio que se asume es su independencia estadística como base para la aplicación de los métodos ICA [34, 37].

2.3.1. BSS para la separación de fuentes acústicas

En recintos cerrados donde existen reflexiones acústicas, unas provenientes directamente de la fuente que emite el sonido y otras que llegan después de reflejarse, una o más veces, en las paredes del recinto o en los objetos que contiene el mismo, convirtiendo la BSS en un problema de alta complejidad. Estas señales llegan al micrófono con cierto retardo de tiempo y desde diferentes direcciones. Como el camino más corto entre dos puntos es la linea recta, entonces, el sonido que llega primero es el directo, que es el utilizado para localizar la fuente sonora por medio de la escucha binaural, tema expuesto en [38, 39]. Posteriormente llegan las reflexiones, que han perdido parte de su potencia en los choques con las superficies, por lo que contienen menos energía que el sonido directo, fenómeno denominado reverberación.

En las señales acústicas, el efecto "cocktail party", se produce cuando una persona focaliza su atención en una sola conversación, en una habitación en la que hay más de una conversación al mismo tiempo. El británico Colin Cherry en 1953, fue el primero que hizo referencia al problema "Cocktail party", con el fin de describir la capacidad del cerebro humano para filtrar una conversación de entre muchas otras simultáneas [40]. Para que ocurra éste fenómeno son necesarios ambos oídos (sensores). Así, el cerebro humano es capaz de filtrar esa conversación de entre todas las que hay a la vez. Investigaciones en procesamiento digital de señales, han adoptado interesantes desarrollos para reproducir dicho proceso, sin embargo, sigue siendo un tema de alta complejidad para los algoritmos de BSS [40].

Como muestra la figura 2.1, las interfaces que capturan señales de audio, por lo general presentan el efecto "cocktail party", cuando las fuentes acústicas y otros sonidos ambientales están presentes en el medio. Las mezclas de fuentes recolectadas por los sensores

acústicos pueden ser de diversos tipos, y están en función del tipo de mezcla recibida en los sensores, por tal motivo, se utiliza diferentes modelos para la resolución del problema de BSS. Las mezclas pueden ser lineales y no lineales. En las mezclas lineales, las observaciones son combinaciones lineales de las fuentes de interés. A su vez, las mezclas lineales pueden ser de dos tipos, instantáneas y convolutivas. En los sistemas de mezclas lineales e instantáneas, la separación se realiza gracias a que las fuentes se combinan en cada sensor de forma diferente y aprovechando la independencia de las mismas, esto es lo que se denomina diversidad espacial [37, 41].

Por lo ya expuesto, el procesamiento de las señales acústica se presenta como uno de los problemas más complejo para las técnicas BSS [35].

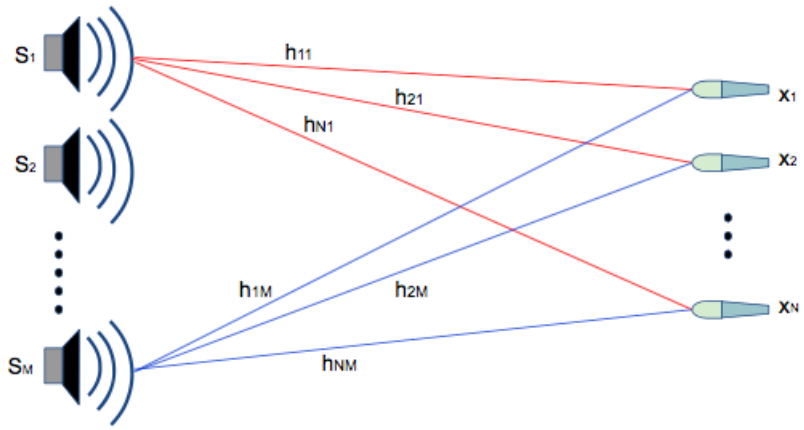


Figura 2.1: Efecto "cocktail party" con M fuentes y N micrófonos.

2.3.2. Modelo clásico de BSS

Al realizar la separación, las fuentes individuales se convierten en "componentes independientes", de ahí nace el término análisis en componentes independientes, el cual está íntimamente ligado al tema de separación ciega de fuentes. Se puede decir que BSS es el

planteamiento del problema e ICA una forma más extendida de solucionarlo [31, 32].

Uno de los modelos clásicos de BSS se describe considerando un conjunto de *fuentes* desconocidas capturadas al mismo tiempo t:

$$s(t) = (s_1(t), s_2(t), \ldots, s_J(t))^T \tag{2.9}$$

donde $s(t) \in \Re^J$ y J es la cantidad de fuentes. Considerando también un conjunto de *mezclas* medibles en el mismo tiempo t:

$$x(t) = (x_1(t), x_2(t), \ldots, x_P(t))^T \tag{2.10}$$

donde $x(t) \in \Re^P$ y P es la cantidad de mezclas. Existe un mapeo desconocido $\boldsymbol{A}$ de $\Re^J$ a $\Re^P$, tal que:

$$x(t) = \mathbf{A}s(t) + \mathbf{n}(t) \tag{2.11}$$

donde $\mathbf{n}(t)$ simboliza cualquier tipo de ruido que pueda estar presente en el proceso de mezcla y/o en la propia red de sensores.

Representando el modelo clásico en forma matricial, las mismas señales en N tiempos consecutivos diferentes, se tiene:

$$S = (s_1, s_2, \ldots, s_J)^T = \begin{bmatrix} s_1(1) & s_1(2) & \cdots & s_1(N) \\ s_2(1) & s_2(2) & \cdots & s_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ s_J(1) & s_J(2) & \cdots & s_J(N) \end{bmatrix} \tag{2.12}$$

$$X = (x_1, x_2, \ldots, x_P)^T = \begin{bmatrix} x_1(1) & x_1(2) & \cdots & x_1(N) \\ x_2(1) & x_2(2) & \cdots & x_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ fx_P(1) & x_P(2) & \cdots & x_P(N) \end{bmatrix} \tag{2.13}$$

$$X = As + N \tag{2.14}$$

donde **A** representa una matriz de mezcla de tamaño PxJ.

Como describe el modelo clásico previo, el problema de BSS no tiene solución determinística o algebraica, al tener más incógnitas que ecuaciones disponibles, adicional, no existe información previa sobre la matriz de mezcla **A**. Aún, con una matriz de mezcla conocida, el problema tendría infinitas soluciones [27, 31, 33].

Para obtener aproximaciones a la solución del problema, se deben considerar ciertas suposiciones con respecto a las características estadísticas de las señales, y en algunos casos, es necesario omitir cualquier retardo temporal que pueda darse durante el proceso de mezclado de las fuentes originales. Además, para facilitar el análisis de las señales mezcladas, se plantea matemáticamente como un **modelo de mezcla instantánea**. También se asume que éste modelo básico es estático, en cuanto a que las variables que intervienen, son variables aleatorias [35].

2.3.3. Características del modelo lineal instantáneo

En la recuperación de las señales acústicas usando los métodos ICA, se deben realizar suposiciones sobre ciertas características de las muestras obtenidas en los sensores, que permiten el correcto funcionamiento del modelo lineal instantáneo [33, 34]. Las restricciones

más importantes son:

- Se asume que las fuentes son estadísticamente independientes entre sí. Las fuentes tienen media cero, no están correlacionadas entre ellas y tienen varianza unitaria.
- Todas las mezclas deben tener distribuciones no gaussianas, con la posible excepción de la existencia de solo un componente gaussiano.
- Se asume que el número de fuentes independientes debe ser igual a la cantidad de mezclas observables. Además, la matriz de mezcla debe ser una matriz cuadrada.

Las anteriores posiciones se suelen considerar principalmente en aplicaciones de audio [34]. Existen múltiples algoritmo implementados cuando se trabaja con mezclas tanto instantáneas como convolutivas. Igualmente, para el procesamiento en el dominio del tiempo o el de la frecuencia [41].

Por otro lado, en la estimación de los componentes independientes aparecen algunas ambigüedades [27, 31]:

- No queda determinada la varianza (energía) de los componentes independientes. Los componentes estimados son proporcionales a los originales, sin embargo, existe un factor de escala α que no se puede determinar. Por ello se suele asumir que los componentes a estimar tienen varianza unitaria.
- El orden de los componentes independientes no puede ser determinado con exactitud. Esto es debido a que se modela la mezcla mediante una matriz, y ésta no refleja el orden espacial de las fuentes originales.

Antes de aplicar alguno de los métodos de ICA para encontrar una solución única y válida del problema de BSS, es importante considerar los siguientes factores [39]:

- Las señales de las fuentes pueden ser reales o complejas, y estadísticamente, pueden ser mutuamente independientes, con idénticas distribuciones en el tiempo o ciclo estacionarias.

- El modelo de la mezcla es un factor que depende de algunos parámetros constantes desconocidos, sin embargo, existen diferentes formas de modelar la matriz de mezcla **A** [39, 35], (omitiendo el componente de ruido):

  - La matriz **A** representa una mezcla lineal instantánea. Se puede modelar mediante la ecuación 2.15 [41]:

$$x_i(n) = \sum_{j=1}^{J} A_{ij} s_j(n) \tag{2.15}$$

    En éste modelo no se contemplan los retrasos temporales entre las señales, es decir, cada fuente llega de manera simultánea a todos los sensores. Cada sensor aporta diferentes versiones de la señal original al escalarla por valores A_{ij}, adicionando la diversidad requerida.

  - La matriz **A** representa una mezcla lineal convolutiva. Se puede modelar como [39]:

$$x_i(n) = \sum_{j=1}^{J} \sum_{k=0}^{L-1} A_{ij}(k) s_j(n-k) \tag{2.16}$$

    Donde cada fuentes aparece filtrada en las mezclas (filtro de orden L), es decir, que la señal llega con diferentes retrasos a cada mezcla, y es posible que posteriormente se obtengan nuevas replicas de la misma (este es el caso del sonido con la reverberación).

  - La matriz **A** representa cualquier otra mezcla no lineal.

Al trabajar con mezclas instantáneas, a la salida del proceso de separación se obtiene una matriz $\mathbf{Y}$, que contiene la estimación de J componentes independientes tal que:

$$Y = BX \tag{2.17}$$

donde $\mathbf{B}$ es la matriz de separación de tamaño JxP.

La matriz de separación también puede ser descrita por $\mathbf{G = BA}$, y determinada por el producto de una matriz diagonal no singular $\mathbf{D}$ y una matriz de permutación $\mathbf{P}$ [36]:

$$G = PD \tag{2.18}$$

Lo anterior representa que las fuentes aparecen escaladas por diferentes valores a la salida, y un cambio en el orden de las mismas se puede presentar, como se indicó anteriormente.

2.3.4. Pre-procesamiento de señales para métodos ICA

Antes de aplicar el algoritmo para separar las fuentes, es conveniente realizar algunas operaciones sobre los datos obtenidos de los sensores. Dichas operaciones no alteran la forma de las señales, sin embargo, simplifican en gran medida la ejecución del algoritmo [42, 43]. Como se muestra en la figura 2.2, el proceso de acondicionamiento de la señal consiste en el centrado y en algunos casos, es necesario un posterior blanqueo (incorrelación) de los datos [43].

Los algoritmos utilizados para BSS aprovechan las medidas de tendencia central, para obtener información respecto a condiciones estadísticas de un conjunto de valores. Es así como, los algoritmos se desempeñan mejor teniendo media cero. Con esta consideración,

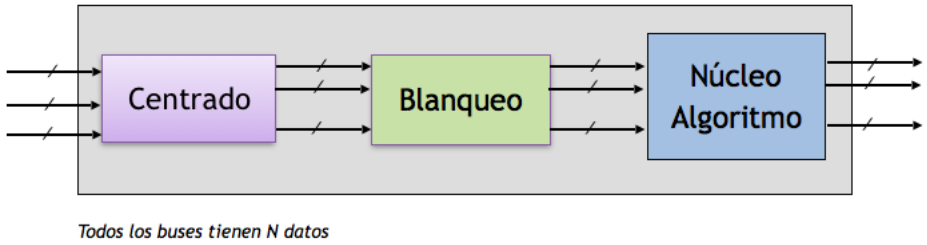


Figura 2.2: Etapas de pre-procesamiento para métodos ICA.

la etapa de centrado consiste en extraer la media de las muestras de la siguiente forma:

$$m = E\{X\} \tag{2.19}$$

Para completar la estimación de las señales, el proceso de centrado debe ser revertido después de estimar la matriz de separación $\mathbf{B}$, por lo tanto, es importante sumar la media a las salidas estimadas. Esta nueva media para $\mathbf{Y}$ debe calcularse como $m_{out} = Bm$ [44].

En los casos que sea necesaria la etapa de blanqueo, la matriz observada $\mathbf{X}$ se transforma linealmente, de tal manera que se obtenga una nueva matriz $\hat{\mathbf{X}}$ [36]. Matemáticamente, se busca que la matriz de covarianza sea igual a la matriz identidad [44]:

$$C_x = E\left\{\hat{X}\hat{X}^T\right\} = I \tag{2.20}$$

Posteriormente, la matriz de covarianza $\mathbf{C}_x$ se descompone en sus valores y vectores propios [43]. Dado que la matriz de covarianza es real y simétrica, los vectores propios son reales y ortogonales [45].

La etapa de preprocesamiento se aplica considerando las características de operación de cada uno de los algoritmo de BSS, por tal motivo, en el capítulo 3, se presenta el análisis

de ésta para los algoritmo elegidos.

2.3.5. Criterios de contraste utilizados por ICA

Por último, pero no menos importante, existe un criterio de contraste, llamado también, funciones de contraste. Dichas funciones representan un criterio de optimización, donde su máximo global corresponde al parámetro que efectúa la separación de todas las fuentes. Estas funciones se utilizan, principalmente, cuando no se tiene información de las propiedades estadísticas de las fuentes ni del ruido presente; cuando se conozcan dichas características, es preferible aplicar la maximización de algún criterio de probabilidad específico [35, 41].

Entre las funciones de contraste utilizadas por los métodos ICA están los momentos y cumulantes, los cuales son ampliamente utilizados en muchas aplicaciones para cuantificar propiedades estadísticas de las señales. Los momentos se pueden definir como [42]:

$$\mu_n = E[x^n] \tag{2.21}$$

Por su parte, los cumulantes κ_n representan los coeficientes en la expansión de Taylor de los momentos sobre el origen. Por ejemplo, $\kappa_1 = \mu_1$, $\kappa_2 = \mu_2 - \mu_1^2$, $\kappa_3 = \mu_3 - 3\mu_2\mu_1 + 2\mu_1^3$. En particular, $\kappa_1 = \mu_1$ representa la media y κ_2 la varianza.

Cuando se trabaja con variables aleatorias con media cero, se define el cumulante de cuarto orden o curtosis de la siguiente forma [44]:

$$kurt(v) = E\left[v^4\right] - 3(E\left[v^2\right])^2 \tag{2.22}$$

La curtosis es una medida estadística que describe el apuntamiento o achatamiento de una cierta distribución con respecto a una distribución normal. La curtosis positiva indica una distribución relativamente apuntada, y la negativa indica una distribución relativamente achatada. El caso de curtosis excesiva (mayor de tres) indica que hay una mayor probabilidad de que los retornos observados estén más alejados de la media que en una distribución normal [46].

Para una variable puramente gaussiana, la curtosis es cero. Para variables con densidades centradas en un pico alto y puntiagudo, y con colas más anchas a los costados (conocidas como súper gaussianas) la curtosis es positiva. Si la densidad tiene un pico bajo y más ancho, con colas delgadas a los costados (sub gaussianas) la curtosis es negativa [46]. Otras propiedades de la curtosis son:

$$kurt(v_1 + v_2) = kurt(v_1) + kurt(v_2) \tag{2.23}$$

$$kurt(\alpha v_1) = \alpha^4 kurt(v_1) \tag{2.24}$$

Puesto que la curtosis es una medida de la condición de gaussianidad para variables aleatorias, puede plantearse como una función de contraste a optimizar, por tal motivo, es utilizada por varios algoritmos ICA, principalmente por su simplicidad y facilidad de cálculo. También existen algoritmos que utilizan momentos o cumulantes de ordenes mayores o menores. Sin embargo, la curtosis presenta ciertos problemas; por ejemplo, resulta ser muy sensitiva ante la presencia de valores atípicos, lo que en ocasiones puede generar observaciones erróneas [44].

2.3.6. Medida de desempeño de los algoritmos ICA para BSS

Con el objetivo de evaluar la efectividad de un algoritmo ICA con diferentes parámetros, se plantean ciertas medidas objetivas que determinan la calidad del proceso de separación de las fuentes. Uno de los criterios más empleado para estimar la calidad de separación de los algoritmos es la relación de señal a interferencia, SIR, (*Signal to Interferences Ratio* por sus siglas en inglés). La calidad de separación para la j_{esima} fuente separada se puede calcular como:

$$SIR_j = 10 \log \left(\frac{E\left\{(y_{j,s_j})^2\right\}}{E\left\{(\sum_{i \neq j} y_{j,s_i})^2\right\}} \right) \tag{2.25}$$

En la ecuación anterior y_{j,s_i} representa la j_{esima} fuente separada que se obtiene cuando sólo la fuente s_i está activa. Para aplicar la evaluación por medio de la SIR se requiere tener por separado la respuesta original entre cada fuente y micrófono, para así, compararse con la señal separada cuando todas las fuentes estén presentes al mismo tiempo [47].

En general, la ecuación 2.25 representa una medida de la razón de potencias en dB entre la señal objetivo y las interferencias externas presentes en cada señal separada [47].

2.4. Diseño de Circuitos Integrados (ICs)

En todo desarrollo práctico de un sistema microelectrónico, existen diferentes alternativas de implementación que permiten aumentar las prestaciones del dispositivo sin incrementar el precio del mismo y conservando en un 100 % los requerimientos del diseño. Las tecnología moderna avanza en función de aplicaciones con algoritmos de procesamiento

de señales que requieren una alta carga computacional y al mismo tiempo mayor potencia en los cálculos aritméticos. Existen numerosas alternativas hardware/software para la implementación física de los algoritmos; entre ellas se destacan los dispositivos DSP, ASIC y FPGA. Estas alternativas ofrecen diferentes grados de eficiencia altamente ligados a la forma en que están llevados a cabo los algoritmos: específicamente si estos son secuenciales, recursivos o concurrentes y que deben ser equilibrados con respecto a factores como costos, consumo de potencia, área y recursos de diseño.

El diseño e implementación de circuitos integrados, tanto analógicos como digitales, ha dejado de ser una técnica reservada exclusivamente a un grupo de especialistas que se encargaban del desarrollo de los mismos, ya sea para realizar funciones estándar o de propósito general (procesadores, memorias, etc.). Las técnicas modernas de fabricación, permiten a los ingenieros e investigadores tener a disponibilidad un entorno de trabajo más completo, donde cuenta con herramientas para síntesis y construcción de sus propios ICs que realicen funciones específicas de acuerdo a las necesidades de las aplicaciones en tiempo real, dando lugar a los ASICs.

El proceso de miniaturización esta enmarcado con la tecnología en función del tamaño de los transistores, que cada vez son más pequeños. El primer IC, fue desarrollado en 1958 por el ingeniero Jack Kilby [1]. Se trataba de un dispositivo de germanio que integraba seis transistores en una misma base semiconductora para formar un oscilador de rotación de fase. En 1961 se produjo la comercialización de los primeros ICs, que contenían menos de 10 componentes [1, 2].

El procesador 4004 de Intel creado en 1971 contaba con 2.300 transistores, el procesador Core I5-661 del año 2010 cuenta con 559 millones de transistores, aproximadamente 243.000 veces más. Como consecuencias de éste crecimiento (la integración del número de

transistores por área), ha aumentado la complejidad en el diseño de los ICs, generando el desarrollo de diferentes metodologías, niveles de abstracción y herramientas de diseño asistido por computador CAD (en inglés, Computer Aided Design) que minimizan y facilitan el ciclo de diseño [1, 22].

En los últimos años, el proceso tecnológico CMOS (del inglés, Complementary Metal Oxide Semiconductor) ha sido dominante por su alta funcionalidad y su relativa efectividad en costos de producción de circuitos integrados VLSI (sigla en inglés de Very large scale integrated). Las ventajas relativas de esta tecnología estriban en su bajo nivel de disipación de potencia, tiempos de propagación medios y bajo costo por transistor [1, 2].

En los dispositivos de muy alta velocidad, se están explorando nuevas tecnologías, se llevan a cabo trabajos de investigación en el campo de la nanotecnología, que reciben el nombre de ERD, Emerging Research Devices [48, 49]. Entre ellos hay que destacar los dispositivos SET, (por sus siglas en inglés, Single Electron Transistors), los transistores realizados mediante combinación de nanotubos de carbono, los TD (por sus siglas en inglés, Tunnelling diodes) y finalmente, los dispositivos basados en componentes de química orgánica, que dan lugar a la nueva área de la electrónica molecular o molecular electronics [48, 49].

Por otro lado, mientras se reducía el tamaño del transistor, a finales de los 80 surgen los lenguajes de descripción de hardware (HDLs) como Verilog (1986) y VHDL (1987), que también servirían como entrada para las herramientas de síntesis lógica. En los 90, surgieron las primeras herramientas comerciales de síntesis de alto nivel, como Synopsys Behavioral Compiler y en los 2000 surge el concepto de nivel de sistema [1].

2.4.1. Metodologías de diseño de circuitos integrados (IC)

Con la reducción del tamaño de los ICs, la metodología de diseño también ha ido evolucionando. En un principio, los circuitos digitales y las primeras computadoras se diseñaron con componentes discretos y la metodología de diseño se denominaba diseño aleatorio o no estructurado. La introducción de los ICs, que incorporaban bloques funcionales, dio lugar a una nueva forma de diseñar, el diseño se convirtió en un método estructurado [1, 2].

Con el desarrollo de la familia CMOS, una de las familias lógicas más empleadas en la fabricación de ICs, el diseño digital causó un fuerte impacto en la metodología de diseño electrónico. La principal característica de los CMOS es el uso conjunto de transistores de tipo pMOS y tipo nMOS configurados de tal forma que, en estado de reposo, el consumo de energía es únicamente el debido a las corrientes parásitas [2].

En la figura 2.3 se muestra carta Y (creada por Gajski y Kuhn en 1983), donde se presenta la división del proceso de diseño en sus tres dominios. El dominio de comportamiento, el estructural y el físico. En cada dominio se puede especificar una variedad de niveles de abstracción, indicados por los círculos concéntricos en la carta Y. A partir de la carta 'Y' se pueden definir una serie de procedimientos usuales en el entorno de diseño, mediante los cuales es posible, ya sea manualmente, o mediante el uso de herramientas informáticas de ayuda al diseño, la traducción de una representación a otra o el paso de un nivel de abstracción a otro [1].

El problema del diseño de un sistema integrado, como los que hoy se desarrollan, es muy complejo, tanto así, que la primera fase consiste en la estructuración y jerarquización del sistema total, de forma que su diseño pueda ser abordado por partes y se planteen varias alternativas de solución antes de tomar una decisión final. Además, debido a la gran

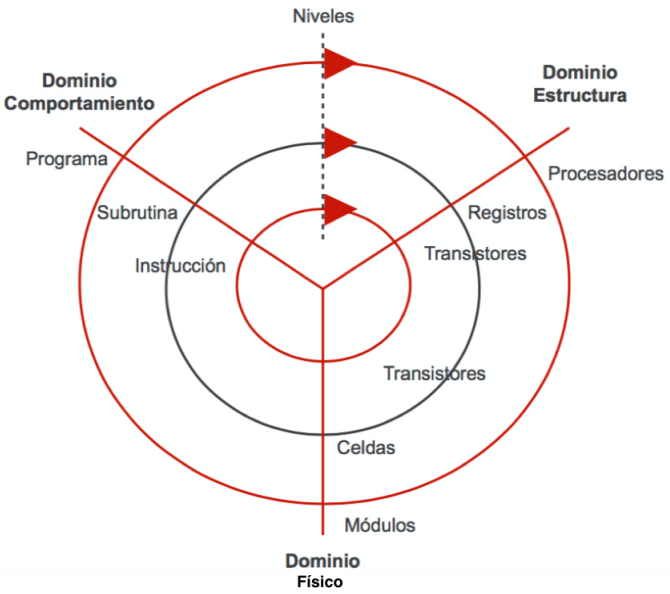


Figura 2.3: Carta Y, con los tres dominios de diseño VLSI, donde se pueden definir los procedimientos usuales en el entorno del diseño [1].

cantidad de variables que se deben controlar, se requiere conocer los diferentes niveles de abstracción, lo que permite reducir la cantidad de información que es necesario manejar en cada momento [1].

2.4.2. Tipos de diseño de un IC

Un ASIC es un dispositivo que ha sido diseñado para un propósito o aplicación específica que hace parte de un producto electrónico que ejecuta una función determinada [1, 22].

La figura 2.4 muestra las tres distintas maneras de abordar el diseño de un ASIC [3]. La primera, el diseño totalmente personalizado o "$full-custom$", es una de los más costosas. En ésta técnica el diseñador puede definir el tamaño de cada uno de los transistores y otros dispositivos del circuito, generar todas las celdas de acuerdo con las necesidades,

esto contempla desde la concepción funcional, lógica y fisica de cada celda. También, se desarrollan las máscaras necesarias para la fabricación del chip sobre la oblea de silicio. Los fabricantes invierten tiempo en maximizar cada micrómetro cuadrado disponible, incluyendo circuitos analógicos, optimizando celdas de memoria y en lo posible mejorar la eficiencia de las estructuras de conexión y ubicación de los componentes del chip. La principal desventaja es el alto riesgo de re-diseño, factor que puede aumentar el costo total del ASIC [1].

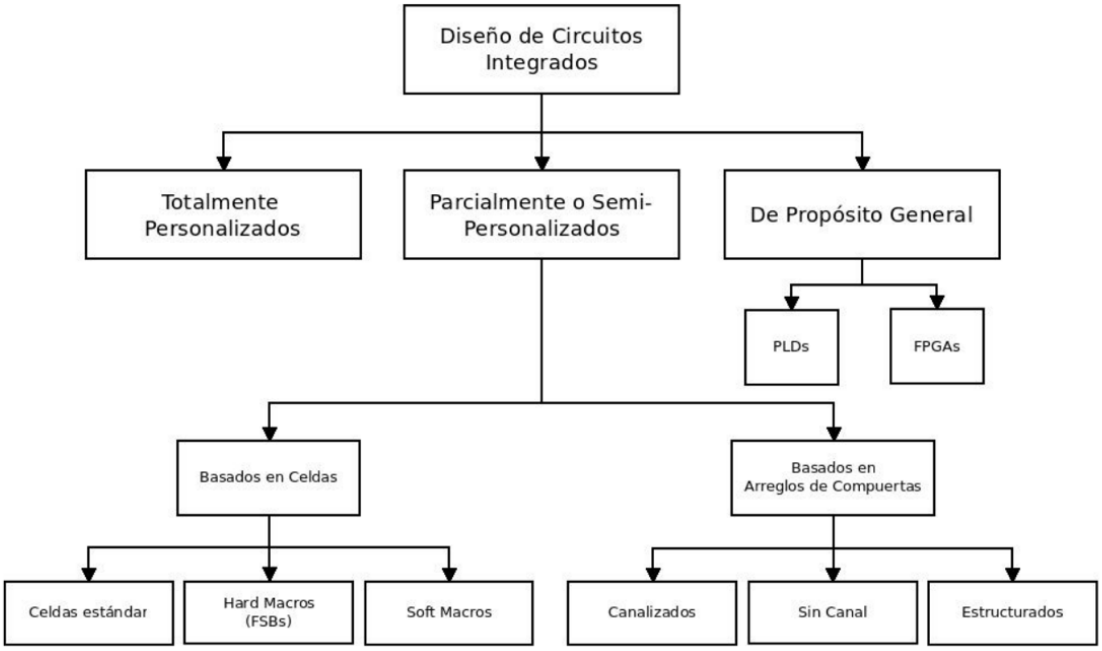


Figura 2.4: Tipos de Diseño de un ASIC y sus derivaciones. Cada tendencia de diseño presenta diferentes criterios y aplicaciones específicas [2]

En general se puede decir que el diseño totalmente personalizado, aunque costoso, es más adecuado cuando se está diseñando un circuito de muy altas prestaciones o bien cuando el volumen de producción es muy alto, por cuanto en este caso cualquier pequeña reducción del área total del circuito va a tener un reflejo importante en el costo total del IC [1].

El diseño semi personalizado es aquel que utiliza una biblioteca de celdas básicas o estándar. El contenido de ésta biblioteca es muy extenso, va desde puertas básicas, elementos de memoria hasta módulos de alta complejidad, como puede ser un microprocesador, y posiblemente todas las máscaras de diseño están disponibles. Además, contiene los modelos de simulación y todos los parámetros necesarios para el uso de las herramientas CAD, lo que hace el diseño más fácil [1, 2]. A su vez, éste tipo de diseño se divide en dos partes:

- **Basado en celdas estándar** ("standard cells"): el diseño de los ASICs basado en celdas estándar es uno de los métodos frecuentemente utilizado, hace uso de bloques funcionales para alcanzar densidades de puertas muy altas, y un buen desempeño eléctrico. Cada fabricante de ASIC crea los bloques funcionales con características eléctricas conocidas, tales como los tiempos de propagación, capacitancias e inductancias, que son representadas en las herramientas desarrolladas por terceros. El uso de ésta técnica reduce los tiempos de diseño y la probabilidad de error debido a que los layouts de las celdas han sido comprobados cuidadosamente y están siendo utilizados por otros diseñadores, y a que la ubicación y conexionado se hacen automáticamente. El riesgo de re-diseño es medio/bajo [2, 4].

- **Basado en arreglo de compuertas** ("Gate arrays"): en este tipo de diseño los transistores se encuentran pre-definidos en un patrón dado o arreglo base. En esta técnica únicamente se define la interconexión y el enrutado dentro de las celdas y con las adyacentes, usando las máscaras personalizadas. La gran ventaja de los "$Gate-arrays$" es que se comparte el costo de todas las máscaras del circuito, salvo las de personalización (de 3 a 5), lo que reduce en gran medida el costo final. Sin embargo, el hecho de necesitar sólo unas pocas máscaras específicas para el circuito presenta algunas restricciones, entre las cuales podemos citar que los ASICs basados

en "$gate-arrays$" suelen conseguir densidades de integración efectivas pobres y que no es posible introducir módulos complejos (por ejemplo, memorias). Sin embargo, elegir ésta opción es adecuada cuando se trata del diseño de circuitos de bajas prestaciones, de volúmenes de producción bajos, totalmente digitales y que no requieran módulos programables en su interior [2].

Por último, se encuentra el diseño de ICs de propósito general, en éste método se utilizan dos herramientas las FPGAs y los PLDs.

- **Diseño con FPGA** : éstas no requieren máscaras específicas para su personalización, son circuitos integrados que contienen un número elevado de puertas o dispositivos básicos y que llegan al diseñador tras haber pasado todas las etapas del proceso tecnológico y ya encapsulados. Dispone de una biblioteca de celdas y un conjunto de herramientas CAD que facilitan el diseño, la simulación del circuito y la proyección de éste a la FPGA (technology mapping o "mapeado" sobre la pastilla de silicio). Como muestra la figura 2.5, la estructura interna de las FPGAs es transparente al usuario [4, 50].

  El diseño de un ASIC utilizando una FPGA se lleva a cabo especificando la función lógica a desarrollar, bien mediante un sistema CAD por medio de esquemático o usando un lenguaje de descripción de hardware HDL (por sus iniciales en inglés, Hardware Description Language). HDL es utilizado para describir circuitos digitales y para la automatización de diseño electrónico [4, 50].

  Una vez definidas las características del ASIC y las operaciones a realizar, el diseño se traslada a la FPGA. Este proceso programa los bloques lógicos configurables (CLBs) para realizar una función específica (existen miles de bloques lógicos configurables en la FPGA). La configuración de estos bloques y la flexibilidad de sus interconexiones

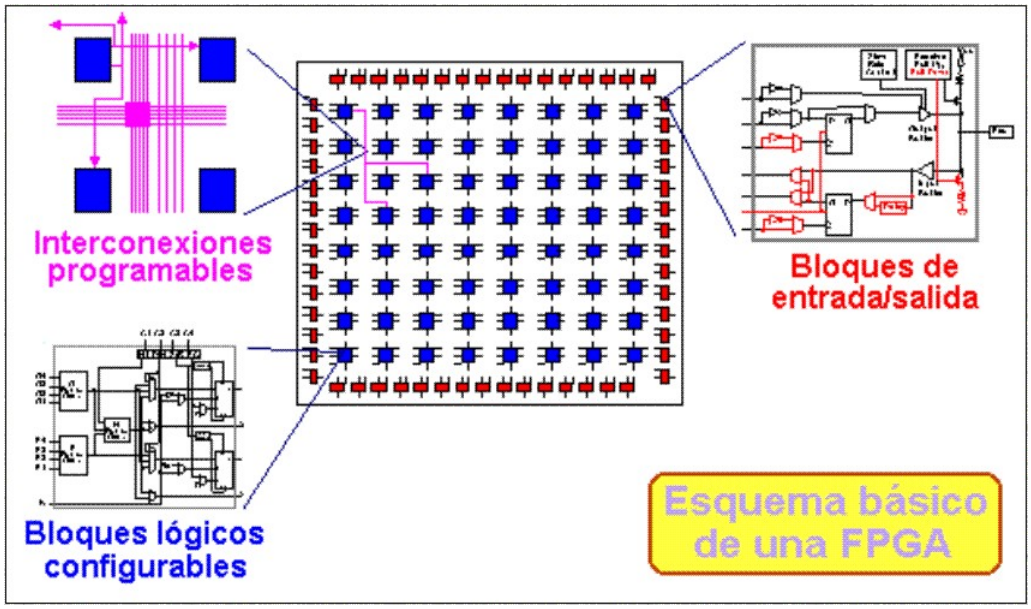


Figura 2.5: Estructura interna de una FPGA [3].

son las razones por las que se pueden conseguir diseños de gran complejidad usando las FPGAs. Las interconexiones permiten conectar los bloques lógicos (CLBs) entre sí. Finalmente, dispone de células de memoria de configuración (CMC, Configuration Memory Cell) distribuidas a lo largo de todo el chip, las cuales almacenan toda la información necesaria para programar los elementos programables. Estas células de configuración suelen consistir en memoria RAM y son inicializadas en el proceso de carga del programa de configuración [4, 50].

Además de los bloques programables también es importante la tecnología utilizada para crear las conexiones entre los canales. Las más importantes son las siguientes [4, 3, 50]:

- Antifusible (Antifuse): una FPGA que utiliza este tipo de tecnología sólo se puede programar una sola vez, es decir, después de programado ya no se puede recuperar. La diferencia entre un fusible y un antifusible es que el primero se desactiva deshabilitando la conexión, en cambio, para el segundo se produce

una conexión cuando son programados, por lo que normalmente se encuentran abiertos. La ventaja es que disminuye considerablemente el tamaño y costo de los dispositivos.

- SRAM (StaticRAM): son utilizadas como generadores de funciones, sirven para controlar los multiplexores (que están incluidos en los FPGAs) y la interconexión entre bloques. En éstas el contenido se almacena mediante un proceso de configuración en el momento de encendido del circuito que contiene la FPGA. Ya que al ser SRAM, el contenido de la memoria se pierde cuando se deja de suministrar energía; la información binaria de las celdas SRAM generalmente se almacena en memorias seriales EEPROM conocidas como memorias de configuración o celdas de configuración.

- Flash: las FPGAs basadas en celdas Flash recogen las principales ventajas de las dos técnicas anteriores. Su tamaño es más reducido que el de una celda de SRAM, aunque sin llegar al tamaño de un antifusible; son reprogramables, aunque la velocidad de programación es menor que en el caso de una SRAM; y son no volátiles, por lo que no necesitan un dispositivos auxiliar para guardar la configuración interna, como en el caso de la SRAM.

- **Diseño con PLDs**: un dispositivo lógico programable (PLD) es un chip LSI que contiene una estructura de circuito "regular", pero que permite al diseñador adecuarlo para una aplicación específica. Los PLDs se utilizan para realizar prototipos que posteriormente se llevarán a un ASIC más complejo. Entre las múltiples arquitecturas de los PLDs se encuentran los PAL (Programmable Array Logic), también llamados PLAs, son el tipo de PLDs mas popular, en los que se pueden programar las uniones en la matriz de puertas AND, siendo fijas las uniones en la matriz de puertas OR. Otro tipo es el FPLA (Field Programmable Logic Array) que se uti-

lizan fundamentalmente para construir máquinas de estados por su baja velocidad de procesamiento y PROM (Programmable Read Only Memory) que es un sistema combinacional completo que permite realizar cualquier función lógica con las n variables de entrada [4, 3].

2.4.3. Flujo de diseño típico de un ASIC

La incursión de los sistemas digitales en aplicaciones reales, exige que éstos sistemas sean tolerantes a fallos y presenten un alto desempeño. Es por ello que en la práctica, el diseño de un circuito integrado conlleva una serie de etapas, las que siguen una secuencia lógica según el nivel de abstracción del algoritmo o función [17, 51].

La figura 2.6 muestra la secuencia básica o flujo de diseño de un ASIC. Los pasos 1 al 4 se denomina diseño lógico y analógico, y los pasos 5 al 9 son parte del diseño físico. A continuación una breve descripción de las etapas [4].

- Un equipo de diseño comienza con la compresión no formal de las funciones requeridas por el ASIC a diseñar, usualmente derivada del análisis de requerimientos. Se construye una descripción del ASIC para alcanzar los objetivos, utilizando un lenguaje de alto nivel (VHDL o Verilog). Este procedimiento usualmente es llamado el diseño RTL.
- La validez del diseño es verificada a través de simulaciones. El objetivo principal de la simulación previa al diseño es desarrollar restricciones del mismo, mientras que la simulación posterior es verificar el cumplimiento de dichas restricciones. Las restricciones de diseño sirven para garantizar que el producto final cumple con las estándares establecidos en las tecnologías de fabricación y con las exigencias de desempeño y funcionamiento.

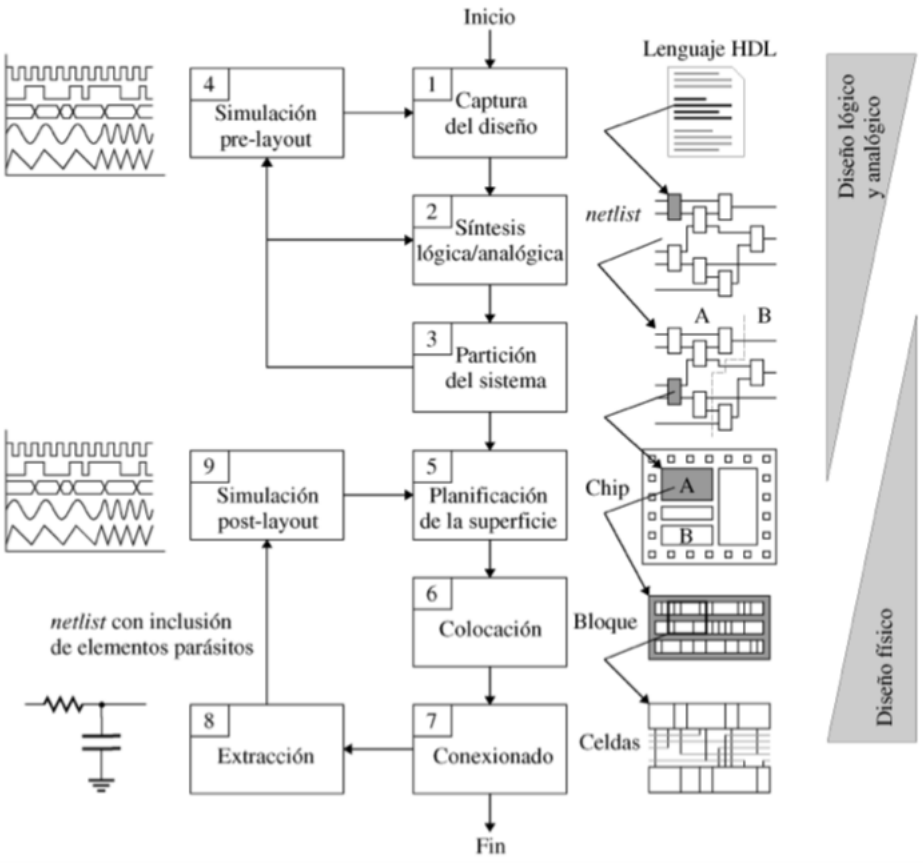


Figura 2.6: Flujo de diseño de un ASIC, donde se observan los pasos de diseño lógico y físico [4].

- Una herramienta de síntesis lógica convierte el diseño RTL en un gran conjunto de elementos de bajo nivel, llamados celdas estándar. Estos elementos son tomados desde una biblioteca, que consiste en una colección de compuertas pre-caracterizadas. Las celdas estándar usualmente son especificas para el fabricante del ASIC. El conjunto resultante de celdas estándar, junto a la interconexión de ellas, es llamado la lista de nodos a nivel de compuertas.

  La síntesis se ha convertido en una herramienta fundamental en el diseño automático de ICs, siendo la más utilizada por las herramientas CAD que trabajan con HDL's. Por medio del proceso de síntesis se genera una descripción estructural a partir de una descripción conductual. La síntesis de alto nivel (HLS) produce bloques específicos o macros, por medio de los cuales se pueden crear implementaciones tecnológicamente más eficientes [1, 2]. De acuerdo con los diferentes niveles de abstracción con los que se trabaje, se pueden definir los siguientes niveles de síntesis:

  - Síntesis de arquitectura
  - Síntesis RTL
  - Síntesis lógica
  - Síntesis a nivel de circuito

- La lista de nodos es luego procesada por una herramienta de posicionamiento (placement), la cual ubica las celdas estándar en una región que representa el ASIC final. Esta ubicación está sujeta a un conjunto de restricciones. En ocasiones se utilizan técnicas avanzadas para optimizar el posicionamiento.

- La herramienta de enrutamiento (routing) toma la ubicación física de las celdas, y utiliza el listado de nodos para crear las conexiones eléctricas entre ellas. La salida de esta etapa es un conjunto de foto máscaras, con las que el fabricante producirá los ICs.

- Se puede hacer una estimación bastante precisa de los retardos finales, las resistencias y capacitancias parásitas y del consumo de energía. Estas estimaciones son usadas en la etapa final de pruebas. Estas pruebas demostrarán que el dispositivo funcionará en los rangos de temperatura y voltaje extremos esperados. Cuando estas pruebas finalizan, la información de las foto máscaras es entregada para la fabricación del chip.

2.5. Tecnologías de fabricación de un IC

Un proceso de fabricación tiene siempre limitaciones de orden tecnológico que hacen que existan diferencias entre la estructura que se espera obtener a partir de un determinado layout y la que realmente resulta tras el proceso de fabricación. Estas limitaciones son debidas básicamente a dos causas: a) no idealidad de las etapas de fabricación, b) posibles desajustes en el uso de las máscaras durante el proceso [2].

Un objetivo habitual de los diseñadores de ICs es que el tamaño de estos sea el menor posible, es decir, que tenga una menor superficie de silicio, lo cual repercute de forma directa en el precio y en muchas de las prestaciones eléctricas del IC. La solución a este problema es considerar la existencia de tolerancias en el proceso de diseño del layout. Estas tolerancias se traducen en una serie de reglas de diseño que fijan distancias y grosores mínimos, de forma que si, éstas se respetan, hay una alta probabilidad de que el circuito diseñado sea correcto. Las reglas de diseño están ligadas a la calidad del proceso tecnológico y por tanto dependen del fabricante [2, 51].

Para el diseño en CMOS existen dos maneras distintas de especificar las reglas: como distancias absolutas (en μm), o como múltiplos enteros de un parámetro único, denominado λ, relacionado con la calidad del proceso [2].

La automatización de diseño electrónico cumple un rol fundamental al momento de crear complejos sistemas, ya que son herramientas de hardware y software que cumplen varias funciones específicas y poseen características especiales al momento de diseñar, verificar y ejecutar. Antes de EDA, los circuitos integrados se diseñaban a mano, y se desarrollaban manualmente. A mediados de los 70, desarrolladores empezaron a automatizar el diseño junto con los croquis. Las primeras herramientas EDA fueron desarrolladas académicamente y fueron evolucionando a la medida que pasó el tiempo hasta llegar a unas EDA más sofisticadas y compuestas de características y funciones únicas [2, 51].

El diseño del hardware también es una rama que engloba dichas herramientas, pero el elevado costo (la oblea de 300 mm para poder realizar grabaciones a 28 nm cuesta más o menos 4000 dólares [19]) es un impedimento al momento de hacer el prototipo. Para resolver este tipo de inconvenientes existen alternativas, tal como crear simulaciones y verificaciones de los mismos antes de implementar el prototipo. Algunas herramientas EDA poseen características esenciales cuando se diseñan circuitos y son:

- Soportan lenguajes de descripción de circuitos complejos.
- Permiten generar diagramas funcionales
- Simulación funcional, digital y eléctrica.
- Simulación de eventos, etc.

Además, las herramientas EDA son muy utilizadas para resolver problemas científicos de ingeniería, matemática, vectores, electrónica, electricidad y algoritmos entre otros.

2.6. Implementación de estructuras aritméticas en hardware

El diseño en FPGA exige una metodología propia, en la mayoría de los casos diferente a la utilizada para el diseño de ASICs, que está condicionada por la estructura y los recursos disponibles en la misma FPGA. Muchos principios del diseño de ASICs no se cumplen en el diseño en FPGAs debido, a los grandes retardos provocados por el enrutado, la división en LUTs (del inglés Luck Up Table) de los circuitos y a los recursos fijos del dispositivo. Por todo esto, la viabilidad de algunos circuitos, métodos de diseño y axiomas clásicos deben reconsiderarse en función de las características de la FPGA elegida [24].

Al implementar en hardware un algoritmo de procesamiento de señales, es común encontrarse con complejas funciones diferenciales, integrales o no lineales, las cuales representan una gran carga computacional y su implementación en un dispositivo electrónico es compleja. En algunos casos, es incluso necesario utilizar diferentes métodos para obtener aproximaciones del modelo en tiempo continuo previo a su programación. Con dichas aproximaciones se crea el modelo dorado o modelo de referencia en lenguajes de alto nivel, que permiten comprobar la funcionalidad de cada ecuación que compone el sistema descrito. Existen, por ejemplo, unidades completas que pueden integrarse de manera genérica al sistema de procesamiento central, que se conocen típicamente como unidades de coprocesamiento aritmético, las cuales muchas veces se desarrollan siguiendo un estándar de representación numérica en coma flotante. Un ejemplo particular se puede ver en [19], donde una unidad completa de coma flotante (FPU por sus siglas en inglés), basada en el estándar de IEEE 754, se ha adaptado a un núcleo RISC-V (arquitectura de hardware "Open Source") para una aplicación de reconocimiento de patrones acústicos por medio de modelos ocultos de Markov [20]. Esta FPU no solo implementa operaciones tradiciones aritméticas como suma, resta o multiplicación, sino que incorpora una sub-unidad

que puede resolver algunas operaciones trigonométricas y exponenciales, aplicando el algoritmo CORDIC. No obstante, es necesario recordar que las soluciones propuestas tienen como principal objetivo conseguir el mejor equilibrio entre la velocidad de procesamiento y el consumo de recursos hardware de la FPGA, así como evitar el uso de dispositivos de memoria externa que reducen la velocidad de procesamiento del sistema e incrementan su tamaño del IC final [4, 3]. Es por eso que usar unidades genéricas como la propuesta en [19] son quizás demasiado onerosas en consumo de recursos, si la funcionalidad buscada está acotada.

La complejidad de un algoritmo numérico puede ser medida de diferentes maneras. Una primera métrica de la complejidad puede ser la cantidad de instrucciones u operaciones realizadas. Dado que cada operación consume una cierta cantidad de potencia, es un indicador útil para medir o comparar las características de bajo consumo de un algoritmo respecto de otro, para una arquitectura determinada [21]. En realidad, no todas las operaciones consumen lo mismo, claramente una multiplicación o división han de consumir mucho más recursos que una adición o substracción. Para una correcta comparación no solo se debe medir el total de operaciones, sino también ponderar con el tipo de operaciones realizadas. Otra métrica para evaluar la complejidad de los algoritmos, es la cantidad de datos en memoria. Los accesos a registros o memorias (principalmente memorias externas) suelen ser costosos, en término de consumo. Por ello, los algoritmos más apropiados para implementaciones de bajo consumo en FPGA o en ASIC son aquellos que no solo minimizan la cantidad de operaciones, sino que además la cantidad de accesos a memoria [4]. Ejemplos de estos circuitos, usados para la estimación del retardo entre señales acústicas, son los descritos en [14, 15, 16], que realizan la estimación del retardo por medio de una simple cadena de retardos de tiempo en lazo cerrado, que evitan la necesidad de una unidad de cálculo compleja, aunque con un tiempo de convergencia lento. Una variación de los mismos, reportada en [17] usa más bien un filtro lineal de Kalman simplificado, que acelera el cálculo minimizando tiempo y gasto energético, mientras que para aplicaciones

que buscan ya disminuir efectos como la reverberación (que pueden alterar la efectividad de una medición acústica, debido a la interferencia multicamino), se ha mostrado en [18].

Aprovechando precisamente esta gran flexibilidad de implementación de diseños digitales basados en arquitecturas aritméticas que ofrecen los dispositivos FPGA, y considerando que la optimización de uno o varios parámetros de un algoritmo numérico se puede atacar en diferentes los niveles durante el ciclo de diseño, es posible llevar a cabo los ajustes durante la evaluación de los resultados obtenidos. Además, para la implementación de complejas operaciones como por ejemplo una división, que requiere el uso de algoritmos adicionales (non-restoring, restoring, SRT en diferentes bases, Neivion-Raphson, Goldschmith, entre otros), es necesario tomar en cuenta el hardware requerido. Tal como se ha visto, cada una de éstas alternativas tienen sus propios valores de área, velocidad y consumo en cada tecnología existente en el mercado [4], y compete entonces al diseñador encontrar aquella estrategia que mejor se ajuste a las restricciones finales de consumo de potencia, velocidad requerida de cálculo, y área o recursos disponibles en la FPGA.

2.6.1. Segmentación o pipeline

Una vez seleccionada la topología y creado el modelo discreto, se deben considerar las técnicas arquitectónicas como paralelismo, segmentación o secuenciación. La segmentación o pipeline es una de las técnicas de diseño comúnmente usada en sistemas digitales para mejorar el rendimiento de la propia estructura algorítmica al dividir largo retardos combinacionales y de ruteo, ejecutados en un solo ciclo de reloj, en varios retardos que son ejecutados en ciclos de reloj más cortos o en el mismo ciclo original pero ahora con mayor slack (definido como el Tiempo Requerido o periodo del reloj menos Tiempo de Arribo de la señal). En otras palabras, en lugar de duplicar hardware se procede a particionar la operación A en N sub operaciones colocando registros entre ellas y logrando un pipeline de N etapas [4, 3]. Los diferentes fabricantes de FPGAs incluyen herramienta para el análisis

de todos los caminos sincrónicos del sistema y así verificar que el algoritmo funcione a la frecuencia solicitada por el diseñador. Si el sistema puede funcionar a la frecuencia requerida el valor del slack es positivo; si el slack es negativo indica que el sistema no puede funcionar correctamente es la frecuencia requerida. Una técnica para solucionar el slack negativo o para mejorar un slack positivo muy pequeño es el reducir el retardo (lógico + ruteo) en el o los caminos críticos [4, 3].

La segmentación, en la mayoría de los casos, se asocia al aumento de velocidad y tiene un fuerte impacto en el área y el consumo de circuitos integrados, debido al aumento de la cantidad de elementos procesados por unidad de tiempo, y que el procesamiento se realiza concurrentemente gracias a la separación que establecen los elementos de memoria. En realidad, la segmentación conlleva un aumento del área debido a los elementos de memoria añadidos y la lógica de control extra, además, al aumentar los niveles de pipeline el uso de registros de sincronización puede aumentar el consumo de potencia. Desde el punto de vista del rendimiento del circuito se debe tener en cuenta la existencia de la latencia o retardo necesario para obtener el primer resultado, la cual se incrementa en forma proporcional a la cantidad de registros sumados al camino crítico. Una de las ventajas mas importantes de la segmentación es la disminución de glitches o fallas que generan los registros de sincronización en la lógica combinacional [52].

2.6.2. Representación de los datos aritméticos

Otro aspecto relevante en la implementación de estructuras aritméticas en FPGA, es la elección de la representación de los datos. El diseñador dispone de diferentes alternativas, por ejemplo: punto fijo versus punto flotante, signo magnitud (signo valor absoluto - SVA) versus complemento a dos (C2), datos codificados versus datos sin codificar. Cada decisión

involucra ponderar diferentes aspectos como: exactitud, facilidad de diseño, prestaciones, área, consumo, etc. Aquí, lo primero es decidir si utilizar punto fijo o punto flotante. El punto fijo, como es de esperar, necesita menos hardware y por consiguiente menos consumo dependiente de la cantidad de bits o representación de los datos. Desafortunadamente, las dificultades en cuanto al rango de representación se evidencian en el escalamiento de los datos que puede ser por software, pero requiere naturalmente de decodificación extra [19]. El punto flotante, por el contrario, modera las dificultades del rango de representación a expensas de utilizar mucho más hardware, consecuentemente más capacidad, y más consumo de potencia [52].

Al margen de los problemas de exactitud y longitud de palabra, se debe seleccionar también, una aritmética de representación para los datos. Por ejemplo, complemento a dos (C2), signo magnitud (también llamado signo valor absoluto: SVA) ó cero desplazado (CD) son posibles representaciones aritméticas de datos. La representación en C2 es la más utilizada. En ella los bits menos significativos (LSBs) son bits de datos, en tanto los bits más significativos (MSBs) son bits de signo. Como resultado los bits MSBs contienen información redundante que conlleva a una mayor actividad (más consumo) cuando hay una cantidad de transiciones de signo importante. Existen múltiples alternativas a la hora de elegir una representación de datos para aplicaciones de bajo consumo, pero no hay una ideal. Por el contrario, el diseñador debe realizar un análisis del sistema en términos de precisión requerida, prestaciones y operaciones a realizar sobre los datos para determinar el sistema de representación idóneo para cada aplicación [52].

Como es de suponer, arquitecturas con mayores anchos de palabras consumen más potencia. Por tanto una arquitectura debe utilizar solo el ancho de palabra necesario para cumplir con los requerimientos de precisión del algoritmo. Existen diferentes algoritmos que computan la misma función con requerimientos de anchos de palabra internos totalmente diferentes.

Capítulo 3

ANÁLISIS PRELIMINAR DE ALGORITMOS DE SEPARACIÓN DE FUENTES (BSS)

3.1. Introducción

En función del objetivo de diseñar arquitecturas microelectrónicas eficientes enérgicamente, para procesamiento de señales en aplicaciones de sistemas dinámicos no lineales, se presenta la evaluación de tres algoritmos de separación de fuentes que utilizan las técnicas ICA, expuestos en el capítulo 2. Dichos algoritmos fueron elegidos considerando la simplicidad y el alto potencial de implementación en un circuito integrado de bajo consumo de energía y de bajo costo [7, 53].

El análisis se centró en la validación de las características de operación, el tiempo de convergencia y en la calidad de la separación de las fuentes mediante simulaciones numéricas, de los algoritmos: FastICA, Adaptativo basado en Gradiente Natural y Adaptativo

Basado en Gradiente Relativo. También se cuantificó un aproximado del costo computacional de cada uno, en función de las operaciones aritméticas llevadas a cabo.

3.2. Ajuste de parámetros operativos de los algoritmos ICA para mezclas instantáneas

Con el fin de analizar la cantidad de recursos computacionales requeridos por los algoritmos, el primer paso fue evaluar la forma en que cada uno procesa los datos de entrada. Esto permite determinar si es necesario utilizar memoria para almacenar los datos antes de la ejecución del núcleo del algoritmo. Posteriormente, se evalúa la calidad de la separación de las fuentes estimadas para conocer su eficiencia y del tipo de operaciones aritméticas llevadas a cabo en cada etapa.

De forma general, se utiliza la siguiente convención para las variables involucradas en el análisis:

- P: Número de micrófonos (sensores).
- N: Tamaño de muestra de datos.
- J: Número de fuentes a estimar.
- **S**: Matriz de fuentes de sonido originales, JxN.
- **A**: Matriz de proceso de mezclado, PxJ.
- **V**: Matriz de señales observadas en los micrófonos, PxN.
- **X**: Matriz de mezclas posterior al pre-procesamiento, PxN.
- **B**: Matriz de separación final, JxP.

- **Y**: Matriz de estimaciones de las fuentes originales, JxN.

M. Stanacevic y G. Cauwenberghs [54] demostraron que es posible utilizar algoritmos BSS en su versión instantánea en ambientes reales, incorporando un arreglos de micrófonos miniatura, donde los retrasos temporales entre las señales se hacen muy pequeños debido a la estratégica ubicación entre ellos (distancia menor a 1m). De tal modo, para el análisis presentado se utiliza el modelo para mezclas instantáneas.

3.2.1. Algoritmo FastICA

Tal como se introdujo en el capitulo 2, FastICA es uno de los algoritmo más conocidos para BSS, y es utilizado en un gran número de aplicaciones debido a su robustez y la velocidad de convergencia [44, 45, 47, 55]. Este algoritmo opera con datos en bloque, lo que hace necesario disponer de memoria suficiente para almacenar los bloques de N datos en cada etapa del procesamiento. La figura 3.1 muestra las tres etapas que comprenden el algoritmo FastICA. Las dos primeras, el centrado y el blanqueo, son operaciones de preprocesamiento que optimizan la ejecución del núcleo ICA.

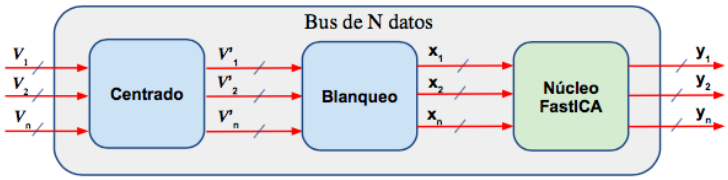


Figura 3.1: Etapas del algoritmo FastICA. Las dos primeras, centrado y blanqueo, son operaciones de preprocesamiento que optimiza el proceso subsecuente del núcleo ICA [5]

Considerando la matriz de datos de entrada V, donde cada fila representa las mezclas observadas $v_1, v_2, \ldots, v_n$, el proceso de centrado consiste en aplicar un filtro de media móvil representado mediante la ecuación 3.1. Este es un filtro digital ampliamente utilizado por

ser intuitivo, fácil de implementar y rápido de calcular [56].

$$\bar{\mathbf{v}}_i = \mathbf{v}_i - \frac{1}{N}\sum_{j=1}^{N} v_{ij} \tag{3.1}$$

Suponiendo que no hay desbordamiento en la sumatoria, para centrar un vector fila se requieren N sumas para la sumatoria, una operación de división entre N para cada fila. Es posible reducir considerablemente el uso de recursos, precalculando $1/N$, y a nivel de hardware se utiliza un multiplicador y N restas para extraer la media a cada fila. Debido a que la cantidad de datos, N, está en el orden de los miles, es posible crear un bloque secuencial con un acumulador que centre una sola fila. A nivel de hardware este bloque puede replicarse P veces para calcular la media en paralelo de todas las filas, o insertar una lógica de secuenciación para utilizar los mismos recursos en P tiempos diferentes. Con esta consideración la cantidad de operaciones aritméticas para el proceso de centrado se resumen en la tabla 3.1.

Tabla 3.1: Resumen de las operaciones aritméticas básicas para el proceso de centrado de datos, usando el filtro media móvil (ecu. 3.1).

Operación	**Cantidad por mezcla**	**Cantidad total**
Suma	N	PN
Multiplicación	1	P
Resta	N	PN
Total	$2N+1$	$2PN+P$

De manera semejante, es posible utilizar otro tipo de filtro digital para el proceso de centrado, por ejemplo, un filtro exponencial EMA (del inglés, Exponential Moving Average). El filtrado con EMA es uno de los más empleados en electrónica digital por los buenos resultados, unido a una eficiencia computacional y sencilla implementación [56].

El cálculo requiere sencillas operaciones aritméticas, como se muestra en la ecuación 3.2, donde α es la constante de ajuste del filtro [27, 30, 56]. El valor de α adecuado dependerá de las características del sistema, de la señal muestreada y del ruido que se desee eliminar. El ajuste de la constante condiciona el comportamiento del filtro, ya que un valor inadecuado, por ejemplo, en el rango de las milésimas (valores muy bajos) puede eliminar componentes frecuenciales de interés, clasificando como ruido lo que realmente era una variación normal de la señal [56]. En cuanto a requisitos de memoria, es necesario almacenar el valor filtrado anterior ($m_i(t-1)$), lo que supone una ventaja computacional frente a otros filtros que requieren guardar N valores y ejecutar cálculos sobre todos ellos.

$$m_i(t) = \alpha v_i(t) + (1-\alpha) m_i(t-1) \tag{3.2}$$

Sin embargo, considerando que la teoría central del algoritmo FastICA se basa en datos con media cero y disponiendo del bloque completo de muestras de entrada, la ejecución del filtro media móvil es comúnmente más utilizado.

El proceso de blanqueamiento viene dado por los siguientes pasos:

- Calcular la matriz de covarianza $\mathbf{C} = E[\bar{\mathbf{V}}\bar{\mathbf{V}}^T]$.
- Calcular la matriz de vectores propios $\mathbf{E}$ y la matriz de valores propios $\mathbf{D}$ de la matriz de covarianza.
- Calcular la matriz de blanqueamiento $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{E}$.
- Por último, se calculan los datos de salida de la etapa $\mathbf{X} = \mathbf{PV}$.

El cálculo de la matriz de covarianza consiste en la multiplicación de una matriz por su transpuesta a la vez que se promedian los resultados. De esta operación resulta una matriz

simétrica de tamaño PxP, que de forma simplificada, se consigue mediante la ecuación 3.3 [57].

$$\mathbf{C} = \frac{\bar{\mathbf{V}}\bar{\mathbf{V}}^T}{N} \tag{3.3}$$

Este paso requiere las siguientes operaciones:

- N multiplicaciones.
- $(N-1)$ sumas (para sumar los N resultados de las multiplicaciones).
- Adicional, una multiplicación por $1/N$, para obtener la media de los datos.

Aprovechando la simetría de la matriz covarianza, sólo se calculan los términos superiores de la diagonal principal. La diagonal contiene P elementos, mientras que los términos simétricos tendrá $\frac{P^2-P}{2}$ elementos. El total de elementos se cuantifica como indica la ecuación 3.4.

$$\frac{P^2-P}{2} + P = \frac{P^2+P}{2} \tag{3.4}$$

El bloque de multiplicación de los dos vectores puede ser replicado $\frac{P^2+P}{2}$ veces para calcular en paralelo todos los elementos de la matriz. Como normalmente N tiende a ser un número grande, y P un número más pequeño, podría ser viable implementar P multiplicadores (vector por vector) que trabajen en paralelo, que trabajen en N secuencias de tiempo diferente.

En la tabla 3.2 se resume la cantidad de operaciones requeridas para el proceso completo de blanqueo, donde k representa el número de iteraciones, parámetro que determina la convergencia del algoritmo [55].

Tabla 3.2: Resumen de las operaciones aritméticas básicas para la ejecución del proceso de blanqueamiento, resaltando la gran cantidad de operación de sumas, restas y multiplicaciones, y adicional requiere de división y raíz cuadrada

Operación	**Cantidad total**
Suma	$\frac{1}{2}(3N-1)P^2 - \frac{1}{2}(N+1)P + k((2P-1)\sum_{j=1}^{P-1} j + P^3 - P)$
Multiplicación	$\frac{3}{2}(N+1)P^2 + \frac{1}{2}(N+3)P + k(P^3 + 3P^2 + 3P\sum_{j=1}^{P-1} j)$
Resta	$kP\sum_{j=1}^{P-1} j$
División	$kP + 1$
Raíz	$kP + P$
Total	$(3N+1)P^2 + 2P + 1 + k(2P^3 + 3P^2 + ((6P-1)\sum_{j=1}^{P-1} j) + P)$

Por último, el núcleo del algoritmo FastICA tiene dos versiones: una basada en curtosis y otra que toma como base la negentropía de los datos. En este caso, fue seleccionada la versión que utiliza curtosis, ya que ha demostrado tener mayor velocidad de convergencia comparada con otros algoritmo similares [44]. Considerando los esquemas de la figura 3.2, presentado en [6], se analiza el consumo de recursos computacionales del núcleo FastICA. Esta etapa requiere un elemento de memoria $\mathbf{W}$ para almacenar los J vectores $\mathbf{w}$ de tamaño P cada uno. Se inicia la secuencia con la estimación de los J separadores $\mathbf{w}$ y de las J fuentes $\mathbf{y}$.

Inicialmente la matriz $\mathbf{W}\mathbf{W}^T$ se separa en sus filas $\mathbf{u}_1$, $\mathbf{u}_2$, ..., $\mathbf{u}_P$ y se efectúa un producto punto entre cada una de ellas y el vector $\mathbf{w}_i$. Un producto punto puede ser implementado con una estructura secuencial o de manera paralela si P es lo suficientemente pequeño. En cualquier caso se requerirán: (P^2) multiplicaciones y $(P)(P-1)$ sumas. El resultado es un vector de tamaño P que se resta del vector original, que adiciona P restas.

El normalizado de los datos se realiza mediante la sumatoria de todos los elementos al

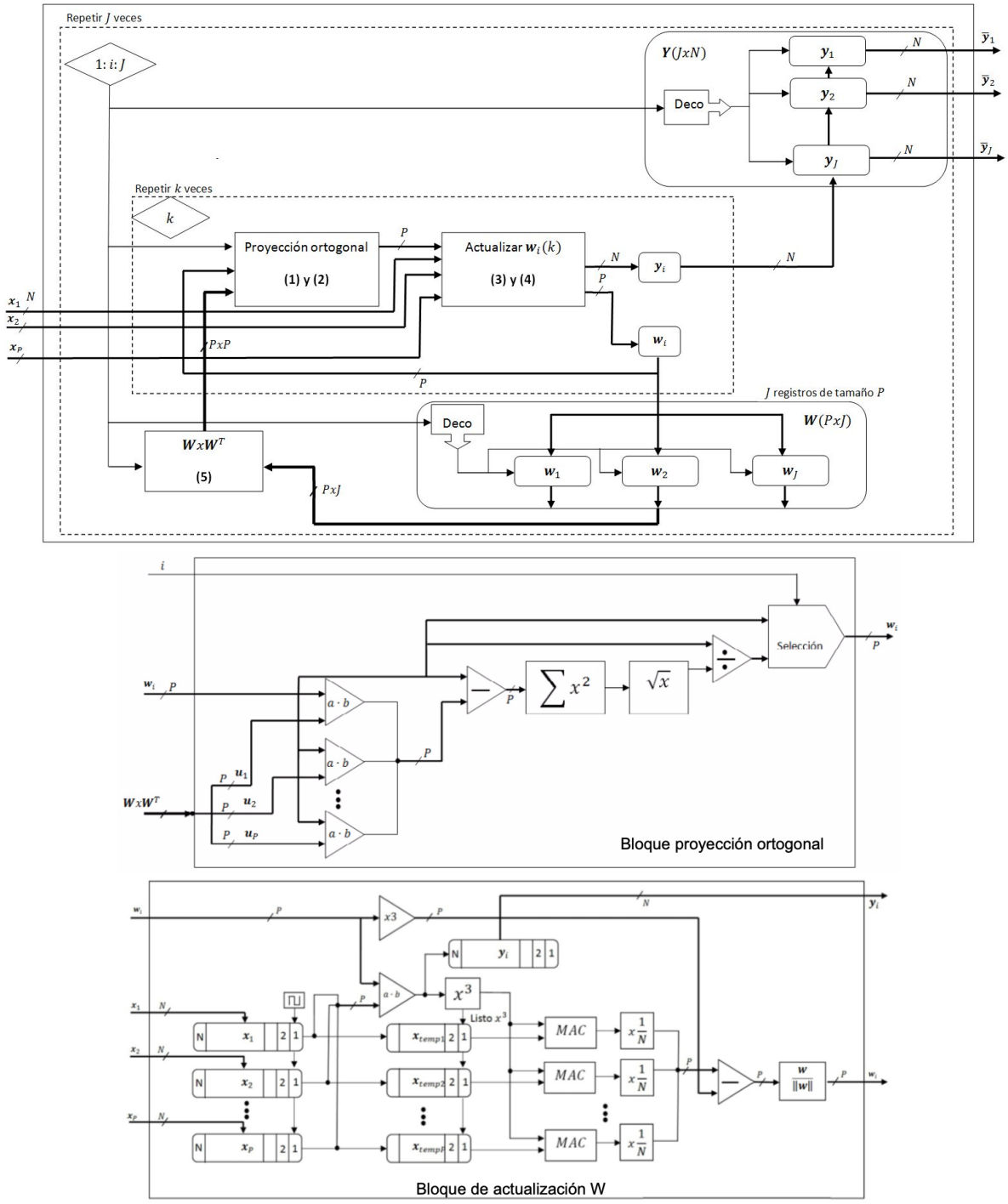


Figura 3.2: Diagrama de flujo de datos del núcleo de algoritmo FastICA basado en Curtosis. La implementación permite definir la cantidad de operaciones aritméticas realizadas [6].

cuadrado, y aplicando la raíz cuadrada, lo que requiere: P multiplicaciones, $(P-1)$ sumas y una operación de raíz. Para finalizar, es necesario dividir los elementos de $\mathbf{w}$ entre la norma. Debido a que la división suele requerir más recursos, conviene efectuar una sola división $1/||\mathbf{w}||$ y multiplicar este valor por los P valores en $\mathbf{w}$. Siendo así, se requerirá una división y P multiplicaciones.

Por otro lado, en cada ciclo se actualiza el vector $\mathbf{w}_i$, que toma el valor de $\mathbf{w}_i$ obtenido de la proyección y los datos en la matriz $\mathbf{X}$. La multiplicación de la matriz $\mathbf{X}^T(NxP)$ por el vector $\mathbf{w}(Px1)$ se efectúa de manera secuencial mediante registros de corrimiento. Se realiza el producto punto entre una columna de $\mathbf{X}$ y el vector $\mathbf{w}_i$. Como resultado de este producto punto se obtiene un valor de $\mathbf{y}_i$ que se almacena en otro registro de corrimiento. Cada elemento de $\mathbf{y}$ se va elevando a la tres para la siguiente operación que consiste en calcular $\mathbf{X}(\mathbf{y}_i^T)^3$ para obtener otros nuevos P elementos. Cada uno de estos P elementos se puede obtener mediante la multiplicación y acumulación de cada valor obtenido de $\mathbf{y}$ con su respectivo valor de $\mathbf{x}$ (N acumulaciones). Finalmente se multiplica cada uno de los P valores por $1/N$, se les resta $3\mathbf{w}_i$, luego se normaliza el vector. Este bloque completo requerirá en general: $4P + 2N(P+1)$ multiplicaciones, $N(2P-1)-1$ sumas, P restas, una operación de raíz y una división.

Después de k iteraciones se ha estimado una de las fuentes, que es almacenada mientras se realiza de nuevo el proceso para obtener las demás fuentes a estimar. El proceso de multiplicación $\mathbf{WW}^T$ para la estimación de cada fuente, brinda una matriz que por definición es simétrica y de tamaño PxP. Como la matriz es simétrica, no es necesario calcular todos los elementos. La diagonal contiene P elementos, mientras que los términos simétricos tendrán $\dfrac{P^2-P}{2}$ elementos. El total de elementos se calcula usando la ecuación (3.4). Este proceso solo se requiere cuando falten fuentes por estimar, por tanto, se ejecutará $(J-1)$ veces. En la tabla 3.3 se resumen las operaciones aritméticas básicas del núcleo del algoritmo FastICA basado en curtosis.

Tabla 3.3: Resumen de operaciones aritméticas básicas para ejecutar el núcleo del algoritmo FastICA basado en curtosis.

Operación	**Cantidad total**
Suma	$k\left[(J-1)P^2+2NJP-J(N+2)+1\right]+(J-1)^2\frac{P^2+P}{2}$
Multiplicación	$k\left[(J-1)P^2+(6J+2NJ-2)P+2NJ\right]+J(J-1)(\frac{P^2+P}{2})$
Resta	$kP(2J-1)$
División	$k(2J-1)$
Raíz	$k(2J-1)$
Total	$k\left[2(J-1)P^2+(4NJ+8J-3)P+NJ+2J-1\right]+(2J-1)(J-1)(\frac{P^2+P}{2})$

3.2.2. Modelos adaptativos para BSS

Los algoritmos adaptativos fueron propuesto por S. Amari, A. Cichocki y H. H. Yang en 1996 [58]. Son algoritmos de aprendizaje adaptativo que se ejecutan en función de la minimización de la dependencia estadística entre las fuentes estimadas. Para lograr dicha minimización el algoritmo toma como base el parámetro de información mutua. Los modelos de aprendizaje realizan la optimización de un parámetro en un espacio Riemanniano (es decir, no Euclidiano). En un espacio Euclidiano la distancia más corta entre dos puntos es una línea recta. Sin embargo, en un espacio Riemanniano, la distancia más corta entre dos puntos es una curva [59]. El objetivo de un algoritmo de gradiente es seguir la curvatura de la variedad para encontrar la mejor solución de un parámetro, midiendo la dirección del descenso más pronunciado de acuerdo con la geometría Riemanniana del espacio. Esta dirección de descenso es natural en el sentido de que el algoritmo sigue el flujo natural de la geometría [59].

Los algoritmos adaptativos, como su nombre lo indica, dependen de una regla adaptativa de actualización basada en un gradiente descendiente, propuesta en [58]. La regla de adaptación presentada en la ecuación (3.5) se aplica para estimar fuentes súper gaussianas (curtosis positiva), que representa el comportamiento de la mayoría de las fuentes de audio reales. Para el caso de fuentes sub gaussianas (curtosis negativa), la regla de adaptación se modifica como se muestra en la ecuación 3.6 [38].

$$\frac{dB}{dt} = \mu(t) \left\{ I - f(y) y^T \right\} B \tag{3.5}$$

$$\frac{dB}{dt} = \mu(t) \left\{ I - y^T f(y) \right\} B \tag{3.6}$$

donde $\mathbf{f}(\mathbf{y})$ es la función de contraste, la cual introduce no linealidad al algoritmo, y μ representa el tamaño del paso que es un parámetro constante, el cual controla la covarianza característica del algoritmo, o sea, cuán rápido esta técnica estima el vector de ponderación óptimo, como se mencionó en el capítulo previo. Esta función es derivada de una función de densidad de probabilidad. De esta manera, el algoritmo sólo puede operar sobre señales que tengan la misma función de densidad de probabilidad. Para el caso de señales acústicas, las cuales son súper gaussianas en su mayoría, el algoritmo necesita una función de activación con características estadísticas súper gaussiana [35].

Las funciones de contraste como: $\mathbf{f}(\mathbf{y}) = arctan(y)$, $\mathbf{f}(\mathbf{y}) = tanh(y)$, $\mathbf{f}(\mathbf{y}) = y$ y $\mathbf{f}(\mathbf{y}) = signo(y)$, han demostrado contribuir positivamente en los resultados, tomando en cuenta las características de las señales observadas [45, 58]. Estas funciones son parte de la distribución de las salidas del separador, pero con un cambio en la notación, que se pueden escribir también en términos del sistema de separación mismo [60]. El problema es entonces la maximización de la función de valor real $\Upsilon(\mathbf{B})$ con el vector $\mathbf{b} = (b_1, b_2, \vdots, b_P)^T$ que representa un filtro espacial para la extracción de una de las fuentes y que representa

un vector columna de $\mathbf{B}^T$ [60]. Bajo estas condiciones, el vector $\mathbf{b}$ puede actualizarse en cada iteración mediante diversas estrategias numéricas como la búsqueda por gradiente o por el método de Newton [61].

En general, los algoritmos iterativos pueden operar en dos formas de procesamiento [60]:

- Métodos de bloque (*Batch Methods*): operan sobre un bloque de N muestras observadas $X_N = [\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(N)]$, y se obtiene también una salida en bloque.
- Optimización estocástica: conocidos también como adaptativos, recursivos, estocásticos, neuronales, o en línea. Se ejecutan en una base de análisis de muestra por muestra. Están basados en una regla de actualización con la forma general [60]:

$$\mathbf{b}(t+1) = \mathbf{b}(t) + \mu\Phi(\mathbf{b}(t), \mathbf{x}(t+1)) \tag{3.7}$$

  donde la función Φ cuantifica la variación inducida en el sistema de separación por el arribo de una nueva muestra en el instante $(t+1)$. El término μ, conocido como *tasa de aprendizaje*, determina la relación entre velocidad de convergencia y precisión de la solución. Un μ más grande implica mayor velocidad de convergencia a costa de menor precisión y viceversa [60].

Como se introdujo en el capítulo 2, los momentos y cumulantes son ampliamente utilizados en muchas aplicaciones de separación de fuentes para cuantificar propiedades estadísticas de las señales. Las señales que se asumen aleatorias para BSS son evaluadas mediante el cumulante de cuarto orden o curtosis. De acuerdo a la ecuación 2.22, para una variable puramente gaussiana, la curtosis es cero (sección 2.3.5.).

Por otro lado, el parámetro de información mutua que representa una medida de independencia entre las señales y es cero sólo si los vectores aleatorios son mutuamente

independientes, puede plantearse como un parámetro a minimizar. Considerando mezclas instantáneas lineales $\mathbf{y} = \mathbf{Bx}$ se puede plantear la optimización del separador $\mathbf{B}$ de la siguiente forma [60]:

$$H(\mathbf{y}) = H(\mathbf{x}) + \log|\det(\mathbf{B})| \tag{3.8}$$

Tomando en cuenta que $y_1, \cdots, y_p$ son P vectores aleatorios con densidades conjuntas $p_{y_1,\cdots,y_p}$ y densidad marginal $p_{y_1}, \cdots, p_{y_p}$, se define el parámetro de información mutua entre la conjunta y marginales como [60]:

$$I[y_1, \cdots, y_p] = -\int p_{y_1,\cdots,y_p}(y_1, \cdots, y_p) \log \frac{p_{y_1}(y_1) \cdots p_{y_p}(y_p)}{p_{y_1,\cdots,y_p}(y_1, \cdots, y_p)} \tag{3.9}$$

Que también puede escribirse en la forma [44]:

$$I[y_1, \cdots, y_p] = \sum_{i=1}^{P} H(y_i) - H(\mathbf{y}) \tag{3.10}$$

donde $H(\mathbf{y}) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}$ y $H(y_i) = -\int p_i(y_i) \log p_i(y_i) dy_i$.

A. Hyvarinen y E. Oja [44] demostraron que minimizar la información mutua implica mayor independencia entre las señales, lo que es equivalente a maximizar el parámetro de negentropía, que aumenta a su vez la no gaussianidad de las señales. En general, todos los algoritmos de aprendizaje automático tienen que lidiar con la incertidumbre. Independientemente de la fuente, dichos algoritmos calculan estadísticas sobre los datos de entrenamiento y utilizan esa información para determinar los parámetros que mas se ajusten a las características del problema dado [59].

3.2.3. Algoritmo Adaptativo basado en Gradiente Natural

A diferencia del algoritmo FastICA que opera en bloque, los algoritmos adaptativos pueden operar dato a dato [58]. La ventaja es que no requiere grandes cantidades de memoria para almacenar grupos de datos, pero tiene la desventaja de que en condiciones iniciales se requerirá un tiempo importante para que el algoritmo empiece a converger a valores estables [47]. Este algoritmo se ejecuta en dos etapas, como se muestra en la figura 3.3.

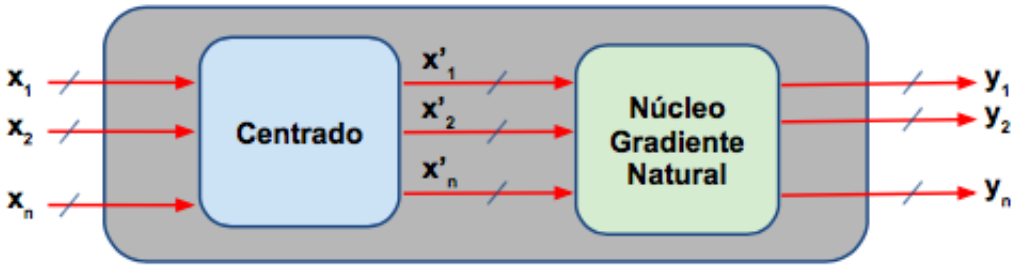


Figura 3.3: Etapas del algoritmo Adaptativo Gradiente Natural. La primera, centrado, es la operación de preprocesamiento que optimiza el proceso subsecuente del núcleo mismo

Para fines prácticos, el proceso de centrado de los datos se aproxima mediante el modelo del filtro paso bajo representado con la ecuación 3.11, donde α es la constante de ajuste del filtro [27, 30]. La media estimada m_i, es calculada a cada dato de las mezclas x_i. La evaluación del valor de α más conveniente para este caso de estudio, se realizó mediante simulaciones en la plataforma de pruebas diseñada en LabView. Se utilizaron las dos funciones de activación que han demostrado tener efecto positivo en la calidad de la separación, $f(y) = tanh(y)$ y $f(y) = sign(y)$.

$$m_i(t) = \alpha x_i(t) + (1 - \alpha) m_i(t - 1) \tag{3.11}$$

La tabla 3.4 resume los resultados obtenidos en función de la media y la desviación estándar de la calidad de separación de las fuentes estimadas SRI (dB), con la misma matriz de mezclas, 15 iteraciones y 10000 muestras. Además, considerando las gráficas de la figura 3.4 del intervalo de confianza de la evaluación del parámetro α evaluado en función de la calidad de separación SIR(dB), permite determinar los valores donde el algoritmo presenta la mayor eficiencia. Dado que dicho intervalo determina la precisión con que se desee estimar el parámetro, tomando el intervalo más estrecho, para el caso en estudio con la función $f(y) = sign(y)$ que presenta resultados mas constantes en la calidad de separación, el valor de α aplicado es 0.01. Se utilizan estas medidas de dispersión ya que los resultados de las simulaciones muestran diferentes valores para las mismas condiciones iniciales mencionadas.

Tabla 3.4: Evaluación empírica de la constante α (ajuste del filtro paso bajo para el centrado) en función de la calidad de la separación obtenida para las dos diferentes funciones $f(y)$. NC = no converge.

α	$f(y) = sign(y)$		$f(y) = tanh(y)$	
	SIR (dB)		**SIR (dB)**	
	σ	**Media**	σ	**Media**
0,010	3.19	9.80	3.11	8.30
0,030	3.51	8.50	2.12	3.00
0,050	NC	NC	NC	NC
0,001	5.71	9.00	7.91	10.00
0,003	7.40	8.40	NC	NC
0,005	6.58	8.60	5.50	8.40
0,007	5.59	7.60	4.15	7.20
0,009	5.50	7.80	7.30	10.40
0,0001	6.34	8.20	6.61	8.80

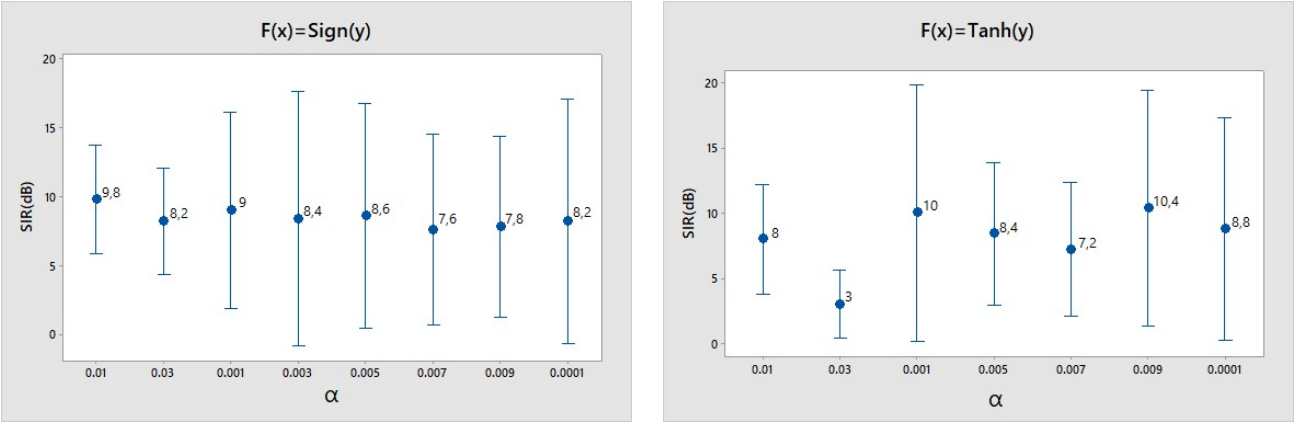


Figura 3.4: Intervalo de confianza de la evaluación del parámetro α para el filtro en función de la calidad de separación SIR(dB), que permite determinar los valores donde el algoritmo presenta la mayor eficiencia.

Es importante resaltar que al usar algunos valores de α, por ejemplo, 0.05, el algoritmo no logra estabilizarse. Esta condición se ilustra en la figura 3.5, donde los resultados obtenidos con $\alpha = 0.003$ indican el algoritmo adaptativo EASI no logra converger a un valor estable en ninguna de las tres señales de prueba, mientras que el algoritmo de gradiente natural estima las tres fuentes aproximadamente a las 6 iteraciones.

También se evidencia que para valores muy bajos de α, el filtro se hace inestable generando mayor dispersión en la calidad de separación, lo indicado por la desviación estándar de los resultados. En función de los datos obtenidos, se utiliza $\alpha = 0{,}01$ como el valor de ajuste para este caso de estudio, para otro tipo de fuentes de audio es recomendable ajustar nuevamente el criterio.

Finalmente, la implementación del modelo de la ecuación 3.11 del bloque de centrado, requiere dos multiplicaciones y una suma. Extraer el dato centrado requerirá finalmente una resta. Considerando que ingresan P datos este proceso requiere de $2P$ multiplicaciones, P sumas y P restas.

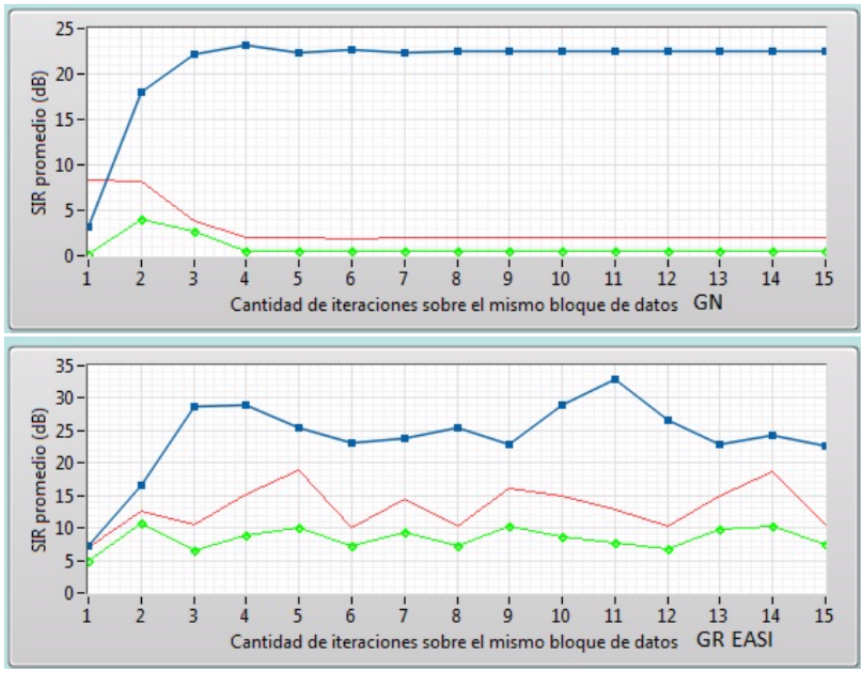


Figura 3.5: Ilustración de la condición de no convergencia (NC) de las pruebas realizadas a los algoritmos adaptativos para $\alpha = 0.003$, donde se resalta la no estabilización a un valor fijo del gradiente EASI.

Posterior al centrado se ejecuta el núcleo principal del algoritmo, representado por las ecuaciones: 3.12 y 3.13. El factor clave en la convergencia es la tasa de aprendizaje μ. El análisis se realiza en función de calidad de la separación (SIR(dB)), con respecto a las funciones, $f(y) = sign(y)$ y $f(y) = tanh(y)$.

$$y(t) \quad = B(t-1)x(t) \tag{3.12}$$

$$B(t) \quad = B(t-1) - \mu[f(y)y^T - I]B(t-1) \tag{3.13}$$

Los resultados de la evaluación empírica del parámetro μ se presentan en la tabla 3.5 de donde se observa que el efecto de μ en el algoritmo de gradiente marca el paso de búsqueda del mínimo local donde el algoritmo converge. Como es de esperarse, para este tipo de algoritmos adaptativos los valores grandes de μ generan alto riesgo de no encontrar

el valor óptimo de la función convexa, el comportamiento se ilustra en la figura 3.7. Por el contrario, un valor muy pequeño en este parámetro (0,001 o menor), hace que tarde demasiado en converger (encontrar un mínimo) y por lo tanto el algoritmo se vuelve demasiado lento, tomando gran cantidad de iteraciones antes de llegar al punto estable, a pesar que tiende a mejorar la calidad de separación.

Tabla 3.5: Evaluación empírica de la tasa de aprendizaje μ, para el núcleo central del algoritmo de separación de fuentes por gradiente natural. Se resalta que los mejores resultados se obtienen con valores muy bajos.

μ	$f(y) = sign(y)$		$f(y) = tanh(y)$		Iteraciones
	SIR (dB)		**SIR (dB)**		
	σ	**Media**	σ	**Media**	
0,010	NC	NC	NC	NC	15
0.050	NC	NC	NC	NC	15
0,090	NC	NC	NC	NC	15
0,001	6.23	9.40	6.51	9.00	15
0,003	3.21	8.50	2.70	10.40	15
0,005	3.85	8.20	6.48	7.80	15
0,007	2.35	6.00	5.71	8.80	15
0,0001	5.32	5.00	6.43	6.00	15
0,0001	4.12	15.00	4.12	15.00	30
0,0001	4.12	15.00	4.12	15.00	50

En función de los resultados obtenidos en la figura 3.6, para este caso de estudio se descarta $\mu = 0{,}007$ por presentar muy baja la calidad de separación 6 dB en promedio con $f(y) = sign(y)$. De igual forma, una razón de aprendizaje con valor tan baja, $\mu = 0{,}0001$, no es considerada ya que requiere de mayor cantidad de iteraciones para conseguir

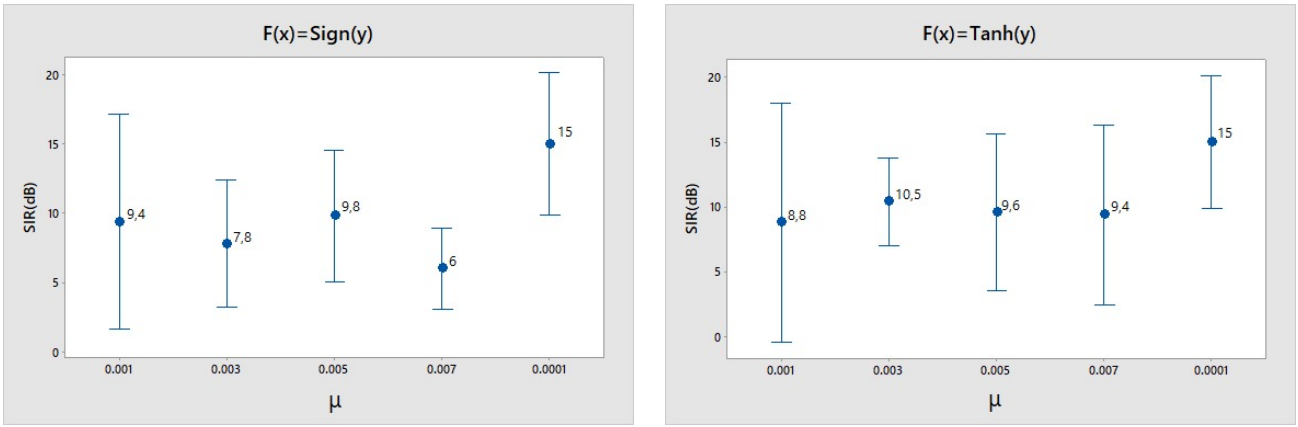


Figura 3.6: Intervalo de confianza de la evaluación del parámetro μ para el núcleo central del algoritmo de gradiente natural en función de la calidad de separación SIR(dB), que permite determinar los valores donde el algoritmo presenta la mayor eficiencia.

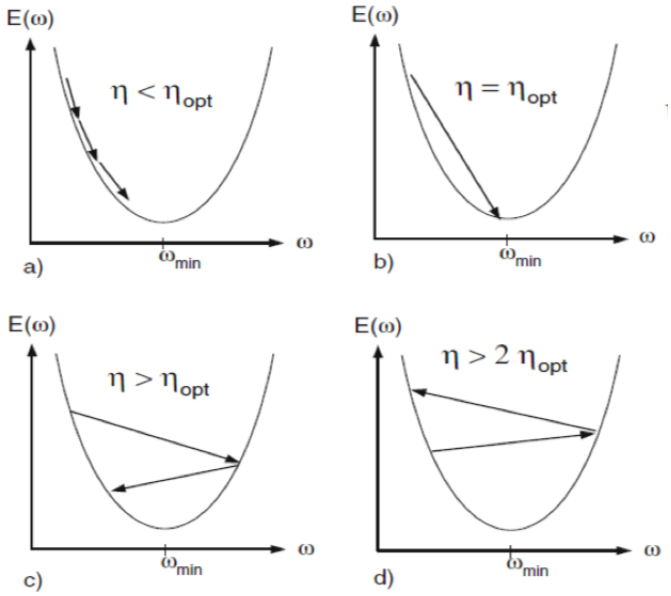


Figura 3.7: Ilustración del comportamiento de los algoritmos basados en gradiente descendiente, donde se resalta la importancia del ajuste del parámetro μ en una búsqueda de mínimo local de una función cóncava.

mejores resultados, lo que hace que tome más tiempo de ejecución del núcleo de algoritmo.

Del análisis realizado se desprende que la existencia de una red adaptativa en el núcleo del algoritmo hace que el ajuste de los parámetros sea altamente dependiente de la matriz aleatoria inicial. En consecuencia, las pruebas realizadas con los mismos parámetros de configuración arrojaron diferentes valores de calidad de separación en pruebas consecutivas; sin embargo, se puede determinar un aproximado de la calidad de la separación del algoritmo fijando los valores de matriz inicial.

Después de obtener los valores ideales de las condiciones de los parámetros críticos del centrado y el núcleo mismo del algoritmo, se estima la salida $\mathbf{y}$ en función del valor de $\mathbf{B}$ de la iteración anterior o la condición inicial según corresponda. Se efectúan J productos punto entre el vector $\mathbf{x}$ y cada una de las P filas de $\mathbf{B}$ para obtener los J valores de $\mathbf{y}$, lo cual requiere de (PJ) multiplicaciones y $J(P-1)$ sumas. Luego, para la actualización de $\mathbf{B}$ en el segundo paso, se calcula $\mathbf{f}(\mathbf{y})$ para los J valores de $\mathbf{y}$. Cada uno de estos J valores obtenidos se multiplica por el vector original $\mathbf{y}$ mediante el bloque de operación escalar, para obtener J nuevos vectores de tamaño J. Sin considerar el costo de calcular $\mathbf{f}(\mathbf{y})$, se han requerido J^2 multiplicaciones para esta parte. La evaluación de la función de activación o contraste $\mathbf{f}(\mathbf{y})$ se presenta más adelante, debido a que la misma puede variar dependiendo del tipo de dato de entrada y de la aplicación especifica que se considere.

Adicional, por las características propias de la matriz identidad solo requiere un total J restas en este paso. El resultado se ha de multiplicar por $\mathbf{B}(t-1)$, efectuando P productos puntos entre la respectiva fila del resultado anterior y las P columnas de $\mathbf{B}$ para obtener una nueva fila de datos. Cada bloque requerirá efectuar JP multiplicaciones y $P(J-1)$ sumas. Puesto que se requieren J bloques esta etapa tiene en total J^2P multiplicaciones y $PJ(J-1)$ sumas. Finalmente se multiplican las J filas por el valor de ajuste o tasa de

aprendizaje μ y el resultado se resta de la matriz original $\mathbf{B}$. Lo que requiere JP multiplicaciones y JP restas.

En general, toda la etapa de actualización de $\mathbf{B}$, sin considerar el cálculo de $\mathbf{f}(\mathbf{y})$ ejecuta las siguientes operaciones simples:$(1+P)J^2+PJ$ multiplicaciones, PJ^2-PJ sumas y $(1+P)J$ restas. La tabla 3.6 resume la cantidad de operaciones aritméticas básicas del núcleo del algoritmo gradiente natural. Comparando la cantidad de operaciones aritméticas básicas de FastICA y Adaptativo Gradiente Natural, se resalta que éste último es una alternativa más fácil de implementar.

Tabla 3.6: Resumen de las operaciones aritméticas básicas para el núcleo del algoritmo Adaptativo basado en Gradiente Natural

Operación	**Cantidad**
Suma	PJ^2-J
Multiplicación	$(1+P)J^2+(2P)J$
Resta	$(1+P)J$
Total	$(2P+1)J^2+3PJ$

3.2.4. Algoritmo Adaptativo EASI (Equivariant Adaptive Separation via Independence) basado en Gradiente Relativo

El algoritmo adaptativo EASI, al igual que el gradiente natural, opera dato a dato y requiere de una etapa de centrado previa a la ejecución del núcleo del mismo. Para centrar los datos, utiliza el mismo bloque presentado en la figura 2.2. El núcleo principal del algoritmo se plantea en las siguientes ecuaciones.

$$\mathbf{y}(t) = \mathbf{B}(t-1)\mathbf{x}(t) \tag{3.14}$$

$$\mathbf{B}(t) = \mathbf{B}(t-1) - \mu[\mathbf{y}\mathbf{y}^T - \mathbf{I} + \mathbf{g}(\mathbf{y})\mathbf{y}^T - (\mathbf{g}(\mathbf{y})\mathbf{y}^T)^T]\mathbf{B}(t-1) \tag{3.15}$$

Como se observa el paso 1 (ec. 3.14) es idéntico al del gradiente natural, por lo tanto, se considera el mismo costo computacional. La diferencia radica en el bloque de actualización de $\mathbf{B}$ (ec. 3.15), donde para obtener $\mathbf{y}\mathbf{y}^T$ cada uno de los J valores y_i se multiplica por el vector completo $\mathbf{y}$ que requiere J^2 multiplicaciones. La resta de la matriz identidad sólo requerirá J restas, como se indicó anteriormente.

Luego se calcula $\mathbf{g}(\mathbf{y})$ para los J valores de $\mathbf{y}$. Cada uno de estos J valores obtenidos se multiplica por el vector original $\mathbf{y}$, y para obtener $\mathbf{g}(\mathbf{y})\mathbf{y}^T$ que representa J nuevos vectores de tamaño J, se ejecutan J^2 multiplicaciones. Posteriormente, a este resultado se le resta su traspuesta $(\mathbf{g}(\mathbf{y})\mathbf{y}^T)^T$ lo que requerirá J^2 restas. Para finalizar el análisis del consumo del paso 2, se considera J^2 sumas para adicionar los bloques ya procesados. Totalizando la etapa de actualización del vector $\mathbf{B}$ del algoritmo EASI, se tienen: $(2+P)J^2 + PJ$ multiplicaciones, $(1+P)J^2 - PJ$ sumas y $J^2 + (1+P)J$ restas. Es importante resaltar que el análisis fue realizado sin considerar el cálculo de la función $\mathbf{g}(\mathbf{y})$, que representa un criterio de optimización definida en términos de la convergencia de cada algoritmo, como se indicó en la sección anterior.

En la tabla 3.7 se resumen la cantidad de operaciones aritméticas requeridas por el núcleo del algoritmo EASI, para obtener una sola muestra de las J fuentes a estimar. Los dos algoritmos adaptativos son altamente dependientes del número de fuentes (crecen con complejidad J^2), mientras que FastICA es prácticamente lineal.

Tabla 3.7: Resumen de operaciones aritméticas básicas para el núcleo del algoritmo EASI

Operación	**Cantidad**
Suma	$(1+P)J^2 - J$
Multiplicación	$(2+P)J^2 + (2P)J$
Resta	$J^2 + (1+P)J$
Total	$2(P+2)J^2 + (3P)J$

3.2.5. Costo computacional de los algoritmos estudiados

Considerando los resultados mostrados en las tablas 3.3, 3.6 y 3.7, es posible estimar la cantidad de operaciones aritméticas simples que requiere cada algoritmo. Tomando para éste análisis $P = 3$ mezclas, $J = 3$ fuentes originales y un bloque de datos de tamaño $N = 10000$. Los resultados se muestran en la tabla 3.8. A primera vista se resalta que el algoritmo FastICA necesita menos operaciones aritméticas básicas, pero requiere llevar a cabo varias operaciones complejas diferentes (división y raíz cuadrada), que son independientes entre sí.

Tabla 3.8: Cantidad total de operaciones aritméticas básicas requeridas por cada algoritmo para tres mezclas (J = 3 y P = 3). Los algoritmos adaptativos utilizan $\mu = 0{,}003$. FastICA* indica la sustitución de las divisiones y raíz cuadrada por sumas, restas y multiplicaciones.

Operaciones	**FastICA**	**Grad. Natural**	**EASI**	**FastICA***
Suma y resta	2056107	2896897	3532500	3165107
Multiplicación	2896898	4230000	4185000	4476897
División	68	0	0	0
Raíz Cuadrada	85	0	0	0
Total	4953158	7126897	7717500	7642004

Considerando que la división se puede desarrollar utilizando el algoritmo SRT-Radix4,

así mismo, la raíz cuadrada puede ser implementada mediante el método de Newton-Raphson o método de Newton-Fourier descrito por la ecuación 3.16, que es comúnmente utilizado para encontrar aproximaciones de los ceros o raíces de una función real, lo que reduce el número de operaciones complejas del algoritmo FastICA pero a costa de aumentar la cantidad de operaciones básicas. Estos métodos de aproximaciones brindan más posibilidades de paralelizar las operaciones del algoritmo, por lo tanto, involucra más recursos de hardware.

$$x_{i+1} = \frac{1}{2} x_i (3 - b x_i^2) \tag{3.16}$$

Al aplicar los métodos mencionados y sustituir las sumas, restas y multiplicaciones requeridas para la división y la raíz cuadrada (FastICA* en la tabla), se observa que el total de operaciones aritméticas para FastICA aumenta el 7.22 % con respecto al algoritmo de gradiente natural pero es mas bajo que el EASI.

3.3. Evaluación de la calidad de separación de los algoritmos elegidos

A partir del análisis previo que permite identificar la complejidad de cada uno de los algoritmos estudiados y con el fin de evaluar los diferentes procesos de estimación de las fuentes, se implementó una plataforma de pruebas. Dicha plataforma ejecuta la versión instantánea de los algoritmos, descrita en la sección 3.2. Los datos de entrada se cargan mediante archivos de audio *.wav*, y se incluye la posibilidad de adicionar varios tipos de ruido para la generación de las mezclas artificiales o sintética.

Para verificar el correcto funcionamiento de los algoritmos en diferentes condiciones de operación, se utilizaron los siguientes conjuntos de datos: el primer conjunto se compone de tres sonidos presentes en un ambiente industrial, la bocina de un camión, el motor de un tractor en estado estable y una motosierra que acelera y desacelera. Estos sonidos se des-

cargaron de la Internet considerando la frecuencia de muestreo de $f_s = 44{,}2$ KHz, usados principalmente para confirmar la correcta operación de los algoritmos implementados, así como de la plataforma de pruebas desarrollada en LabView. El segundo conjunto de datos está compuesto por sonidos reales de tres máquinas de un taller de maderas industrial. Las señales fueron obtenidas una por una con un micrófono convencional de alta calidad LifeChat LX-6000, localizado a 1m de distancia de las tres máquinas. Las máquinas grabadas fueron: una cepilladora, una canteadora (sierra de bordes) y una de corte recto.

Después de tener definidas las señales de prueba, se evaluó la gaussianidad de cada una de ellas mediante el parámetro de curtosis. Tomando el primer conjunto de fuentes mostrados en la figura 3.8. En la tabla 3.9 se resume el cálculo de la curtosis para cada fuente y su clasificación utilizando la ecuación 2.22. Las fuentes originales de audio *BocinaCamión*, *MotorTractor* y *Motosierra* fueron mezcladas con la matriz aleatoria **A** indicada en 3.17.

Tabla 3.9: Clasificación de las fuentes originales mediante la evaluación del parámetro de curtosis de *BocinaCamión*, *MotorTractor* y *Motosierra* con $N = 10000$, donde se puede determinar que todas son súper Gausianas.

Fuente	**Curtosis**	**Clasificación**
BocinaCamión	8.935	Súper-Gaussiana
MotorTractor	2.834	Súper-Gaussiana
Motosierra	0.986	Súper-Gaussiana

$$\mathbf{A} = \begin{bmatrix} -0{,}807 & -0{,}300 & 0{,}831 \\ 0{,}437 & -0{,}146 & -0{,}744 \\ 0{,}656 & 0{,}203 & -0{,}953 \\ 0{,}334 & 0{,}745 & -0{,}567 \end{bmatrix} \tag{3.17}$$

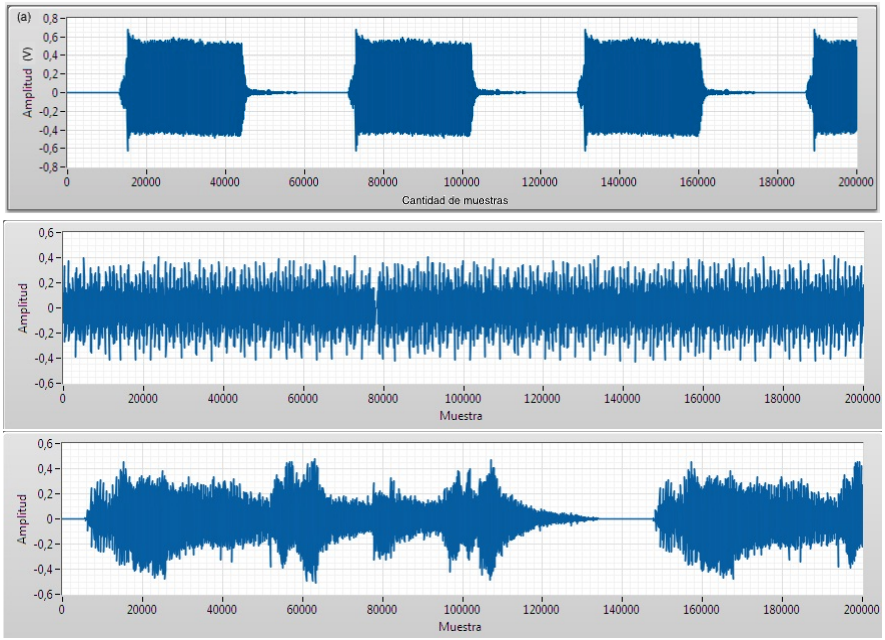


Figura 3.8: Forma de onda de los sonidos (a) Bocina Camión, (b) Motor Tractor y (c) Motosierra, utilizados para verificar el correcto funcionamiento de los algoritmos en la plataforma de pruebas.

3.3.1. Verificación funcional de FastICA

Para la evaluación funcional del algoritmo FastICA se dividió cada fuente original en bloques de $N = 10000$ datos, antes de ser mezcladas. En la figura 3.9 se presentan los resultados obtenidos de la ejecución de un bloque. Se aprecia que la cantidad de iteraciones en todo el proceso se ha mantenido baja y que la calidad de separación es alta, en promedio 40 dB. Este criterio de desempeño fue expuesto en el capítulo 2 (ecu. 2.25). Se deduce también que en la cuarta iteración todas las fuentes han superado cerca del 90 % de su valor final de estimación. Incluso la fuente MotorTractor demostró converger prácticamente en la primera iteración y la Motosierra presentó la mejor calidad de separación. Cabe resaltar que la fuente BocinaCamión es la que presenta la menor curtosis (ver tabla 3.9), situación que puede producir una baja calidad de separación en algunos casos [44, 57]. Esto se debe a que la curtosis es un parámetro estadístico que maximiza el algoritmo FastICA [31].

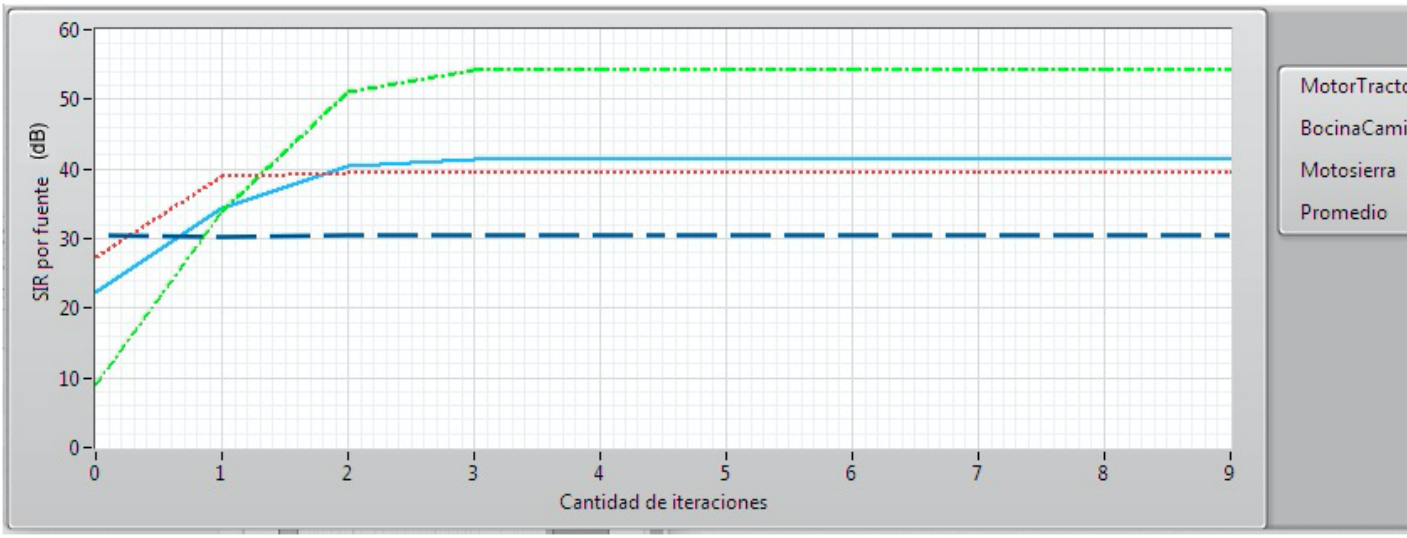


Figura 3.9: Convergencia del algoritmo FastICA en función de la cantidad de iteraciones y la calidad de la separación. MotorTractor demostró converger prácticamente en la primera iteración a los 30 dB y la Motosierra presentó la mejor calidad de separación (mayor curtosis).

Con el propósito de evaluar la influencia del tamaño del bloque de muestras en el comportamiento del algoritmo se dividieron las señales originales en diferentes tamaños de bloques, N cantidad muestras diferentes. Los resultados se muestran en la tabla 3.10. En todos los casos se utilizaron 15 iteraciones en el proceso de blanqueamiento. A partir de los resultados obtenidos se observa que éste un indicador indirecto que demuestra que el algoritmo FastICA requiere una capacidad de almacenamiento grande para obtener tasas muy eficientes, es decir, a mayor cantidad de muestras mejor es la calidad de separación, mientras que para bloques pequeños, la calidad es baja. Además, se deduce que la cantidad de iteraciones es menor a medida que la cantidad de muestras aumenta, lo que se traduce en menor tiempo de operación.

Tabla 3.10: Influencia del tamaño N del bloque de datos sobre la calidad de separación y velocidad de convergencia del algoritmo FastICA

N	**BocinaCamión**		**MotorTractor**		**Motosierra**	
	Iteraciones	SRI (dB)	Iteraciones	SRI (dB)	Iteraciones	SRI (dB)
512	175	-5.34	45	9.37	75	-10.37
1024	8	14.81	8	27.81	33	14.45
4096	7	16.24	12	19.70	6	15.51
8192	46	17.24	5	21.87	9	18.07
16384	30	24.32	5	27.35	12	16.04
32768	16	30.63	4	38.58	7	13.73
65536	4	40.77	3	32.23	2	28.72
131072	5	47.39	3	41.21	2	36.37

3.3.2. Verificación funcional del algoritmo basado en gradiente natural

Se utiliza la mezcla recortada en 10000 datos y la función de contraste $f(y) = tanh(y)$. Los resultados para diferentes valores de la tasa de aprendizaje μ se muestran en la figura 3.10. El primer efecto notable en la variación de la tasa de aprendizaje es la aceleración del proceso de convergencia, como es de esperarse en este tipo de algoritmo adaptativos, con $\mu = 0.0001$ se requieren cerca de 47 iteraciones, con $\mu = 0.0005$ toma aproximadamente 20 iteraciones y con $\mu = 0.001$ se estabiliza con apenas 10 iteraciones, lo que indica que valores muy pequeños de μ el algoritmo requiere mayor tiempo de convergencia. Adicional, el algoritmo mostró signos de no estabilizarse cuando se utilizan valores mayores de 0.001. En todos los casos el SRI promedio ronda los 25 dB, valor muy cercano al obtenido con el algoritmo FastICA en el bloque de 16000 muestras. Por su parte, el gradiente natural con $\mu = 0{,}001$ converge cerca de 15 iteraciones mientras que FastICA requiere de 30 iteraciones para obtener los mismos resultados debido a que debe procesar el bloque completo de datos.

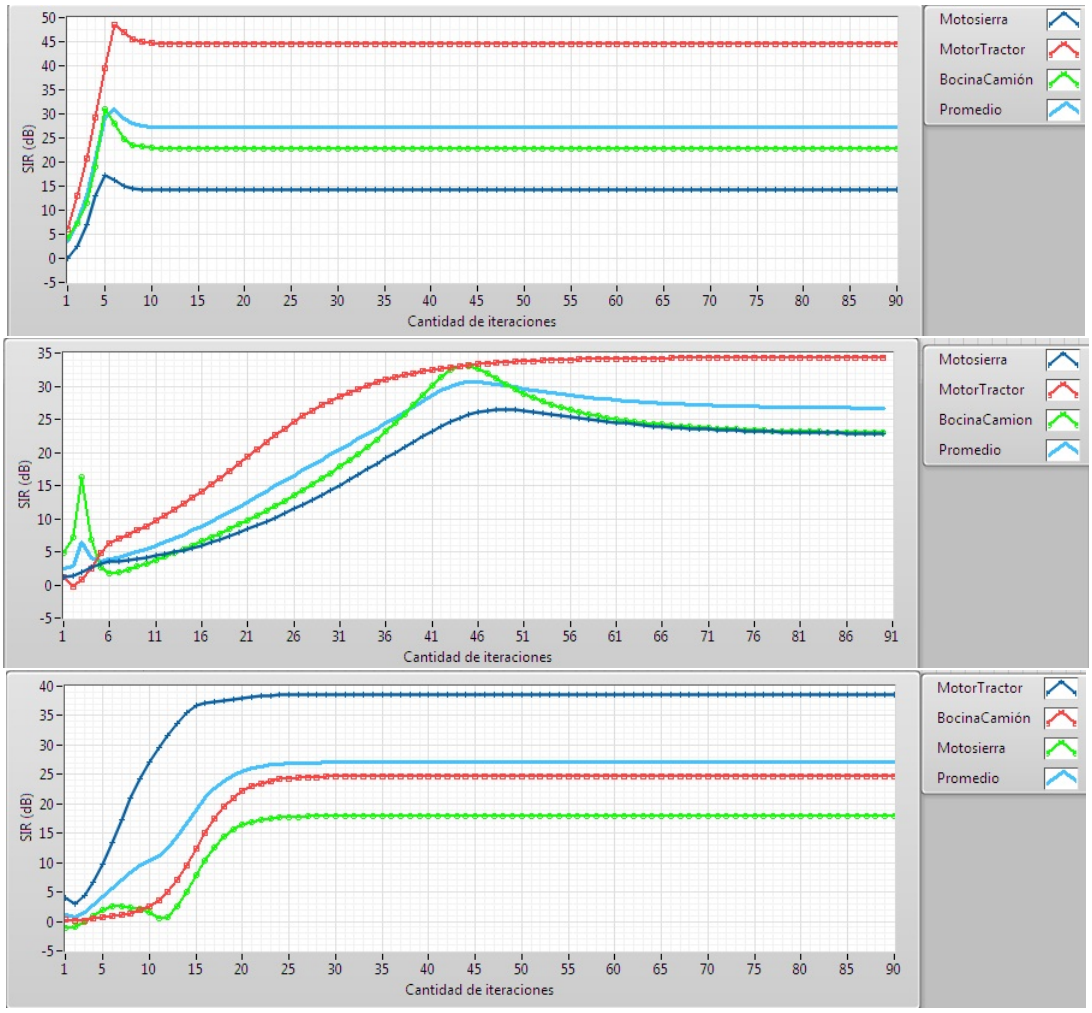


Figura 3.10: Influencia de la cantidad de iteraciones en función de la calidad de la separación para el algoritmo de gradiente natural con: $\mu = 0{,}001$(superior), $\mu = 0{,}0005$(centro) y $\mu = 0{,}0001$(abajo). En todos los casos el SRI promedio es de 25 dB. Con éstos parámetros de operación la mejor estimación se obtuvo para MotorTractor y la peor para Motosierra.

Debido a que las pruebas de los algoritmos parte de condiciones iniciales desconocidas, durante el proceso de convergencia aparecen valores atípicos (ver fig. 3.10) los cuales varían en amplitud de acuerdo con la tasa de aprendizaje μ.

Por último, se realizaron pruebas con diferentes funciones de contraste $f(y)$. Los mejores resultados obtenidos se muestran en la tabla 3.11, para cada caso se ha iterado manualmente hasta encontrar una tasa de aprendizaje μ que lleve el algoritmo a la convergencia, dicho valor se indica en la tabla junto con la función evaluada. Es así como se define que la función $signo(y)$ es una de las opciones más atractiva para trabajar con éste tipo de algoritmos. Primero por su eficiencia en la calidad de separación y, además, la simplicidad de la misma función permite ser implementada en hardware con un costo computacional relativamente bajo. Considerando la calidad de separación, se resalta que la señal con mayor curtosis (Motosierra) presenta los resultados más favorables aplicando $f(y) = signo(y)$.

Tabla 3.11: Efecto de la función de contraste $\mathbf{f(y)}$ sobre la calidad de la separación SRI(dB) en el algoritmo de gradiente natural. Se puede definir que la función $signo(y)$ en combinación con μ ajustado en el rango de las milésimas, es una de las mejores opciones para trabajar con éste tipo de algoritmos.

$\mathbf{f(y)}$	μ	**Motosierra SRI (dB)**	**BocinaCamion SRI (dB)**	**MotorTractor SRI (dB)**	**Promedio SRI (dB)**
$tanh(y)$	0.0001	37.937	35.490	38.021	37.149
$(1 - \frac{2}{1+e^{-y}})$	0.00005	36.215	33.469	35.779	35.154
$signo(y)$	0.0001	42.967	33.713	34.253	36.978

3.3.3. Verificación funcional del algoritmo EASI

Considerando la similitud entre éste algoritmo y el gradiente natural, se aplicaron los mismos experimentos. Primero se evaluó diferentes valores de la tasa de aprendizaje μ, en términos de la convergencia mediante la cantidad de iteraciones. A partir de los resultados expuestos en la figura 3.11, se puede decir que la variación de la tasa de aprendizaje acelera

del proceso de convergencia de la misma forma que ocurre con el gradiente natural. Es así como, con $\mu = 0.0001$ se requieren aproximadamente 39 iteraciones, con $\mu = 0.0005$ toma 12 iteraciones y con $\mu = 0.001$ parece estabilizarse con apenas 7. En todos los casos el SRI promedio oscila entre 24 y 30 dB. Dando la mejor estimación en el MotorTractor y la peor para Motosierra.

Al igual que el gradiente natural, la segunda evaluación fue la influencia de la función de contraste. Los resultados obtenidos se registran en la tabla 3.12, los cuales muestran ser similares a los ya obtenidos en el comportamiento del algoritmo frente a diferentes $f(y)$. La única diferencia radica en que el algoritmo EASI muestra una reducción en la calidad de la separación en promedio del 20 %. Manteniendo la mejor calidad con la función $signo(y)$ y una razón de aprendizaje μ de 0.0001. Otra observación interesante es que el algoritmo EASI no parece ofrecer ninguna ventaja respecto al gradiente natural, puesto que es claramente más complejo y tiene baja calidad de la separación, como ya se mencionó.

Tabla 3.12: Influencia de la no linealidad elegida $\mathbf{f(y)}$ sobre la calidad de la separación SRI en dB en algoritmo EASI

$\mathbf{f(y)}$	μ	**Motosierra SRI (dB)**	**BocinaCamion SRI (dB)**	**MotorTractor SRI (dB)**	**Promedio SRI (dB)**
$tanh(y)$	0.0001	31.871	29.888	26.427	29.395
$(1 - \frac{2}{1+e^{-y}})$	0.00015	22.620	24.890	22.801	23.440
$signo(y)$	0.0001	46.210	29.113	28,153	34.49

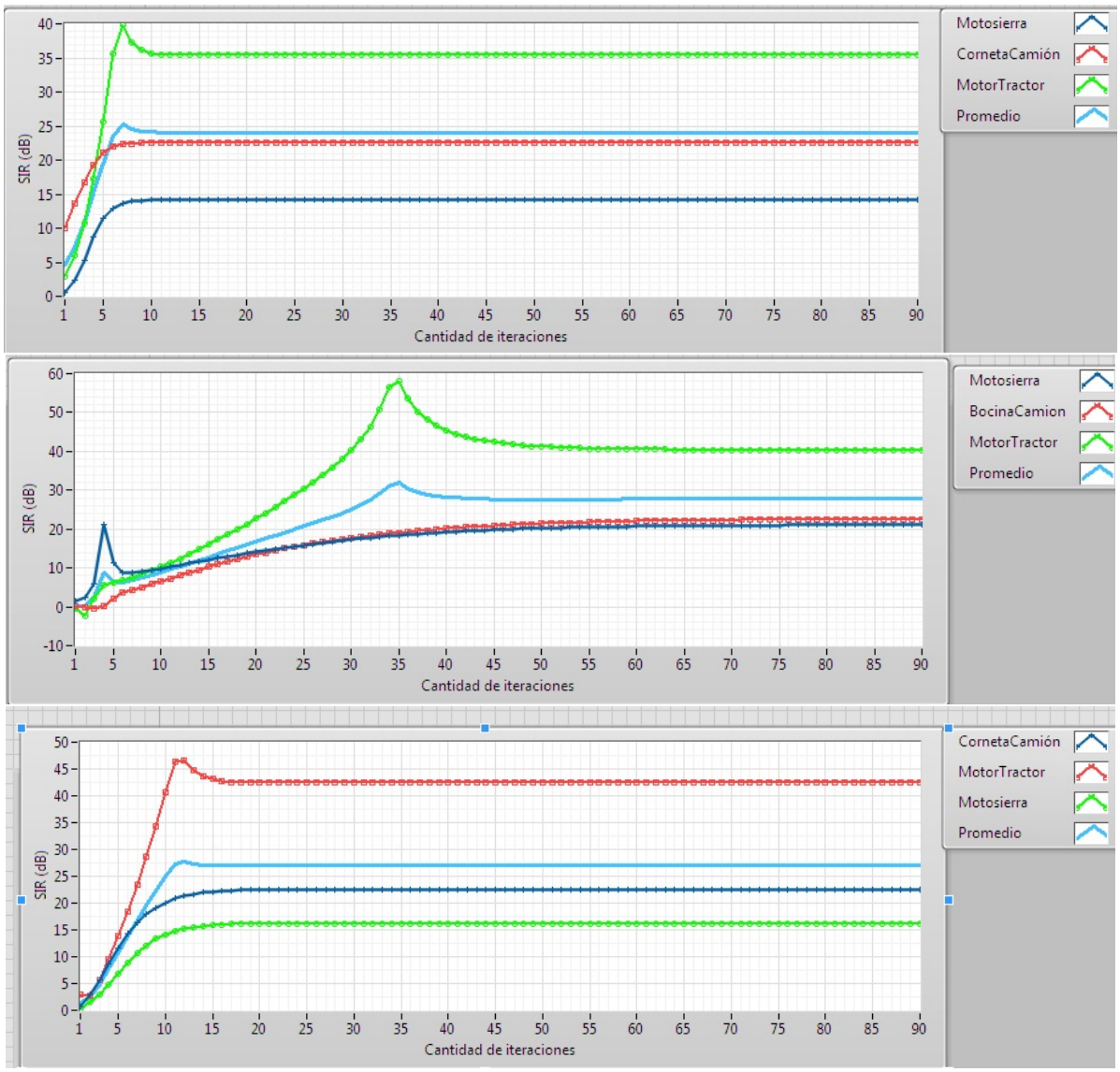


Figura 3.11: Influencia de la cantidad de iteraciones sobre la calidad de la separación para algoritmo EASI con: μ=0.001(superior), μ=0.0001(centro) y μ=0.0005(abajo).En todos los casos el SRI promedio oscila entre 24 y 30 dB. Dando la mejor estimación en el MotorTractor y la peor para Motosierra

3.4. Conclusiones del análisis preliminar

A partir de los resultados presentados en este capítulo, se concluye lo siguiente:

1. El algoritmo EASI no parece ofrecer ninguna ventaja respecto al gradiente natural, puesto que es claramente más complejo y no refleja una mejor calidad de la separación.

2. En la evaluación realizada en términos de la función de activación, se considera utilizar $f(y) = sign(y)$, porque la implementación en un modelo digital donde requiere muy pocos recursos de hardware es menos compleja comparada con las otras funciones evaluadas, adicional, ha demostrado mejores resultados en los algoritmos adaptativos.

3. El total de operaciones aritméticas para FastICA aumenta el 7.22 % con respecto al algoritmo de gradiente natural, al considerar los métodos de aproximaciones para resolver las operaciones de división y raíz cuadrada. Además, FastICA requiere de memoria adicional para almacenar el bloque de datos a procesar, aumentando el costo en hardware.

4. Se podría reducir el costo computacional del algoritmo de gradiente natural optando por filtrar los datos analógicamente, antes que se ejecute el núcleo central del mismo. Esta opción implica realizar una análisis previo de las mezclas, lo cual conlleva a un diseño especifico del filtro, dependiendo de la aplicación que se esté trabajando. Tema que se sale del contexto de esta tesis.

5. En problemas de aplicación real en entornos industriales donde no se podrıa saber con exactitud la presencia de posibles fuentes sub-gaussianas (curtosis negativa) en el ambiente, es hace necesario incluir un estimador de curtosis en los algoritmos adaptativos, de manera que pueda ajustar los parámetros del algoritmo en función de la detección del tipo de fuente que se esta separando, tema que se propone como trabajo futuro de la

investigación.

6. Cuando se utiliza ICA para trabajar con aplicaciones reales es importante tener en cuenta las limitaciones que tienen los algoritmos. Además, se resalta que existen múltiples fuentes de audio en ambientes reales con distintos comportamientos estadísticos, características que deben considerarse en el momento de elegir un algoritmo basado en las técnicas ICA.

Por último, considerando la evaluación realizada a los algoritmos en función a la calidad de separación, el comportamiento frente a las condiciones especificas de funcionamiento, el tipo de fuentes, la convergencia de cada uno y el costo computacional, se determinó implementar el algoritmo de gradiente natural en una estructura digital, resultados presentados en el siguiente capítulo.

Capítulo 4

IMPLEMENTACIÓN DIGITAL DEL ALGORITMO ADAPTATIVO BASADO EN GRADIENTE NATURAL

4.1. Introducción

En el capítulo anterior se presentó la evaluación preliminar de tres algoritmos para BSS: adaptativo basado en gradiente natural, adaptativo basado en gradiente relativo EASI y FastICA. Dicho análisis fue publicado en [5]. El análisis se enfocó en evaluar las capacidades y eficiencia en términos de desempeño y costo de hardware de cada algoritmo, con el fin de seleccionar el que pueda ser utilizado con pocas fuentes y con un costo equivalente o menor al más efectivo, en este caso FastICA, que es complejo para llevar a hardware, por sus necesidades de memoria para almacenar bloques de datos y de complejas operaciones (división y raíz cuadrada), además, que éste algoritmo ha sido llevado a hardware en muchas otras aplicaciones e investigaciones.

En este capítulo se presentan los resultados de la validación del algoritmo adaptativo basado en gradiente natural, portado y verificado primero sobre una FPGA. Además, se expone el proceso de traslado de la implementación a una tecnología CMOS de 180 nm, para mostrar la viabilidad de su implementación en un ASIC. Esta implementación es verificada mediante simulaciones.

4.2. Implementación digital del algoritmo de separación de fuentes por gradiente natural.

Para realizar la validación funcional de la arquitectura aritmética desarrollada, se generó un modelo en Labview que permite comparar las salidas estimadas a partir de un modelo dorado con los resultados obtenido del modelo implementado en Verilog. Con el fin de realizar pruebas de desempeño inicialmente se tomaron dos entradas, es decir, dos mezclas de dos sonidos y luego con tres señales de audio, todas mezcladas artificialmente. Todas las señales están muestreadas a 44 kHz. Como vectores de prueba, se eligieron 10000 muestras de datos de los audios indicados en el análisis preliminar presentado en el capítulo 3.

Para las pruebas funcionales el algoritmo fue integrado a los módulos que se muestran en la figura 4.1. El diagrama de bloques muestra el flujo general de datos del algoritmo implementado en lenguaje de hardware Verilog. Los módulos de memoria ROM contienen los datos de las mezclas de entrada (señales provenientes de los micrófonos), que se cargan con cada ciclo de reloj por medio del contador de direcciones. La transmisión de los datos desde la FPGA a la computadora se lleva a cabo mediante un convertidor serie a paralelo en conjunto con el Transmisor-Receptor Asíncrono Universal (UART, del inglés

de Universal Asynchronous Receiver-Transmitter). Todo el sistema es controlado por una máquina de estados general, FSM (del ingles, Finite State Machine) e internamente el núcleo central tiene otra FSM dedicada exclusivamente al control de las operaciones del módulo. El sistema completo se implementó en una placa de desarrollo del fabricante DIGILENT Inc., que consta de un chip de lógica programable Spartan 6, producido por la empresa Xilinx.

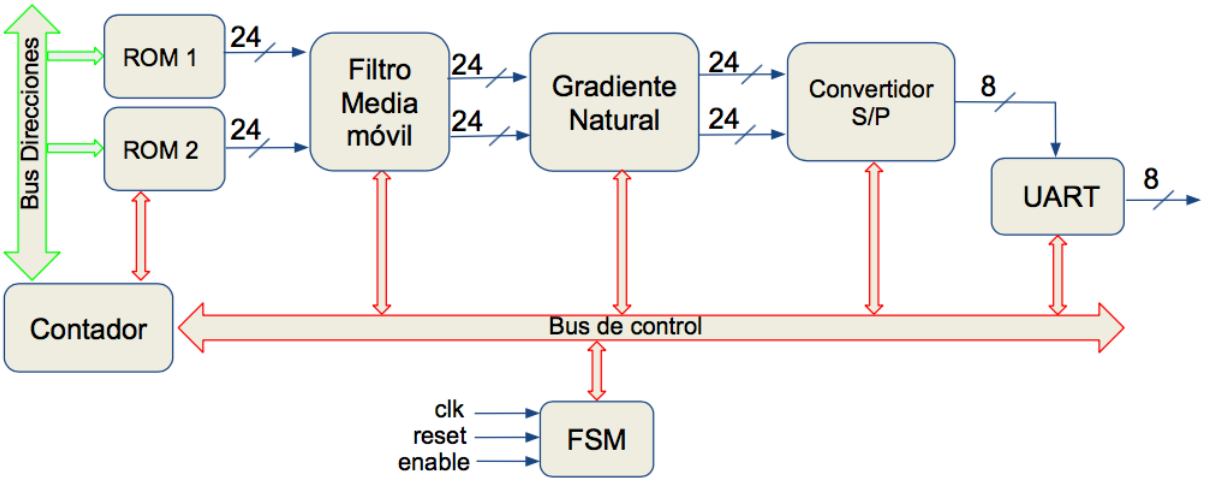


Figura 4.1: Diagrama de bloques del sistema completo implementado para las pruebas del algoritmo adaptativo gradiente natural, donde se resalta la inclusión de memorias ROM con las mezclas y un convertidor serie-paralelo para la extracción de las variables de salida de la FPGA a la PC.

La codificación de cada sub bloque se realizó siguiendo la estructura de datos mostrada en la figura 4.2. Para efectos prácticos, se codificaron por separado la etapa de centrado y el núcleo central, con el fin de ofrecer la posibilidad de incluir otro tipo de preprocesamiento o cambiar el filtro de los datos antes de ingresar al algoritmo mismo. En la parte izquierda de la figura 4.2(a) se muestra el flujo de datos para el bloque de centrado de datos (filtro EMA) y en la parte derecha 4.2(b) el núcleo central del algoritmo. Cabe señalar que esta estructura es escalable para diferente cantidad de mezclas de entrada y

representa un ejemplo de cómo partir la operación matricial en su representación como conjunto de ecuaciones, en este caso para dos y tres entradas.

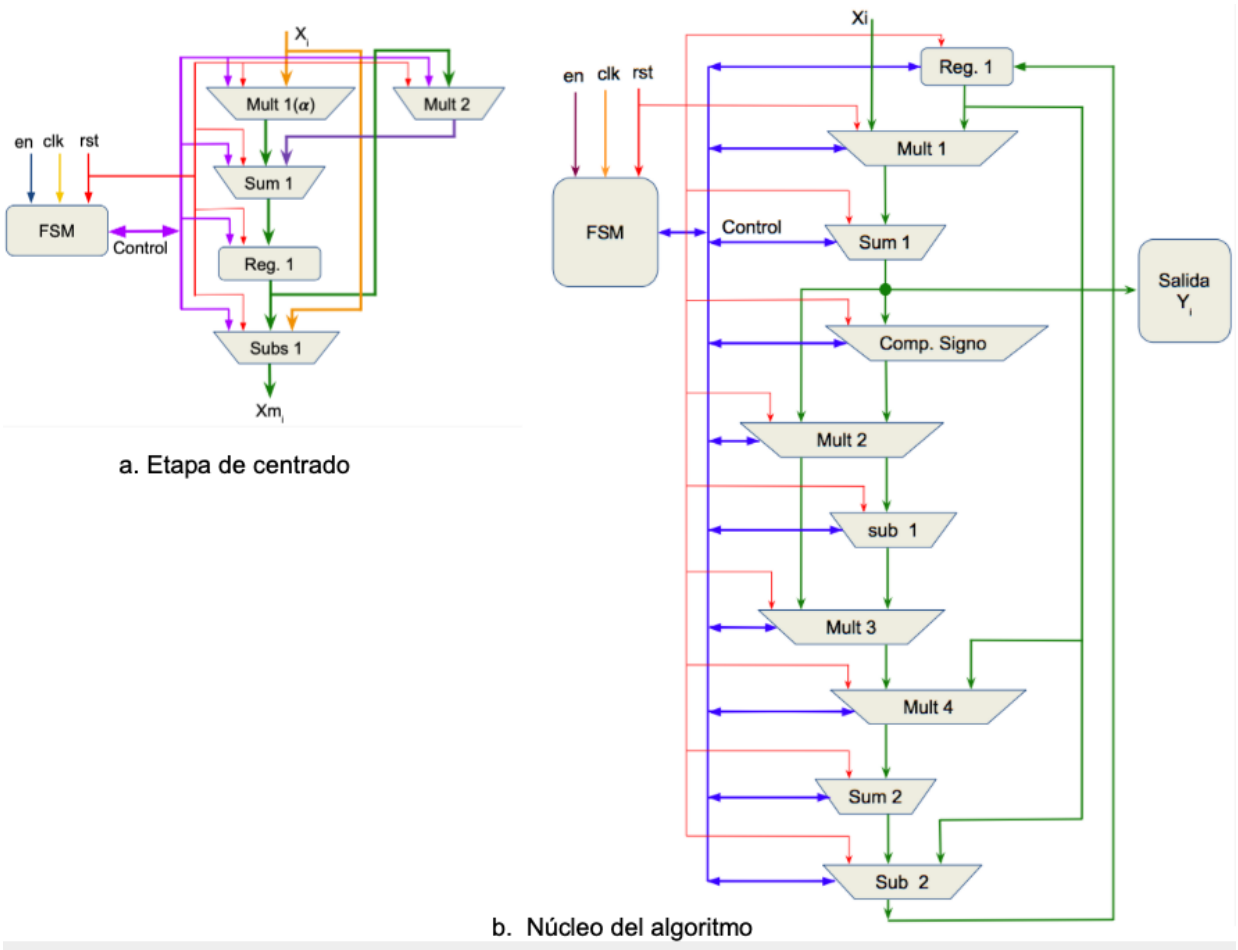


Figura 4.2: Diagrama de flujo de datos del algoritmo. Cada unidad de cálculo (suma, multiplicación, etc.) va a un registro para mejorar el rendimiento del sistema.

4.2.1. Validación y análisis de desempeño con dos entradas

Considerando dos mezclas X_{ij} y N cantidad de muestras, las matrices B y X que describen las ecuaciones 3.12 y 3.13, quedan representadas de la forma siguiente:

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$X = \begin{bmatrix} X_{11} & X_{12} & ... & X_{1N} \\ X_{21} & X_{22} & ... & X_{2N} \end{bmatrix}$$

El número de micrófonos o mezclas define la dimensión de la matriz B. En el primer caso evaluado, las pruebas iniciaron con dos mezclas, por lo que, B es una matriz cuadrada 2x2 y sus valores se actualizan en cada iteración, como se explicó en el capítulo anterior. La matriz X porta N cantidad de datos de cada mezcla de entrada (sonidos provenientes de micrófonos). Para recuperar las fuentes, primero se debe realizar una transformación lineal del modelo general y generar manualmente las ecuaciones que serán implementadas en un HDL. Para dos fuentes estimadas las ecuaciones de salida quedan definidas como:

$$y_1 = B_{11}X_{11} + B_{12}X_{21} \tag{4.1}$$

$$y_2 = B_{21}X_{11} + B_{22}X_{21} \tag{4.2}$$

Las cuatro ecuaciones para obtener el parámetro de actualización son de la forma:

$$B_{11} = B_{11}(k-1) - \mu[f(y_1)y_1 - 1]B_{11}(k-1) \tag{4.3}$$

$$B_{12} = B_{12}(k-1) - \mu[f(y_1)y_2]B_{12}(k-1) \tag{4.4}$$

$$B_{21} = B_{21}(k-1) - \mu[f(y_2)y_1]B_{21}(k-1) \tag{4.5}$$

$$B_{22} = B_{22}(k-1) - \mu[f(y_2)y_2 - 1]B_{22}(k-1) \tag{4.6}$$

donde μ es un valor constante, evaluado en el capítulo 3, $f(y) = sign(y)$ y $B(k-1)$ es el valor de B obtenido en la anterior iteración. Las condiciones iniciales de la matriz B se

definieron a partir de múltiples simulaciones, de lo cual se determinó que la alternativa con mejor desempeño es una matriz positiva, con los valores siguientes:

$$B = \begin{bmatrix} 0{,}1 & 0 \\ 0 & 0{,}1 \end{bmatrix}$$

Se codificaron las estructuras aritméticas en punto fijo para la verificación en RTL usando los datos digitalizados a 32 bits para obtener una mejor resolución y mayor precisión en los resultados. El código se segmentó en diez (10) ciclos de reloj para el núcleo del algoritmo y cuatro (4) para el filtro. El entorno de prueba fue parametrizado permitiendo así trabajar con cualquier cantidad de mezclas.

En la figura 4.3, se muestran las dos fuentes de audio tomadas de la internet (similares a las usadas en el análisis preliminar del capitulo 3) y sus respectivas mezclas. Las fuentes estimadas usando LabView (modelo de alto nivel) y Verilog (modelo digital) se muestran en la figura 4.4. Como se observa, existe una estrecha correlación entre las formas de onda que confirman la funcionalidad correcta de la arquitectura aritmética del algoritmo de gradiente natural.

A partir de los resultados obtenidos se puede observar que las señales estimadas con la FPGA muestran una atenuación debido a los efectos de la representación coma fija comparada con coma flotante en LabView, y a la discretización realizada a los datos de entrada en el pre y el pos procesamiento. Sin embargo, se resalta que las señales estimadas son similares a las originales.

Por otro lado, considerando el reporte de síntesis de la herramienta de simulación y la localización de los distintos elementos lógicos del diseño con su respectivo ruteo den-

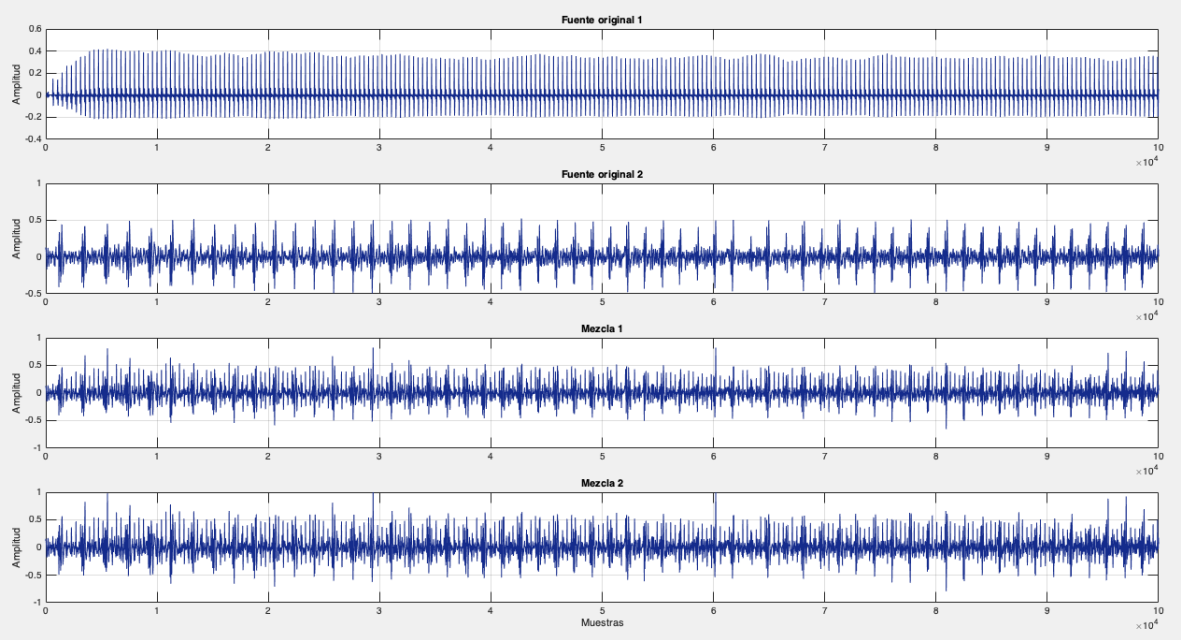
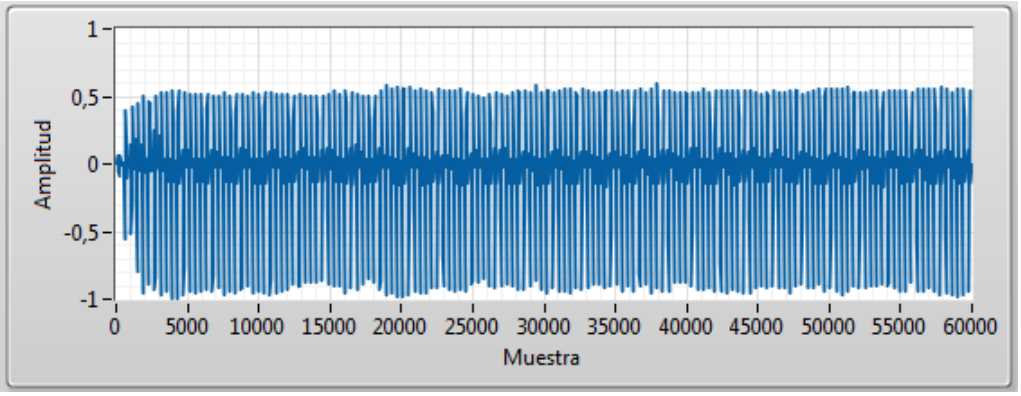


Figura 4.3: Sonidos utilizados para pruebas del algoritmo en la FPGA, (a)Bocina Camión (fuente original 1) y (b)Motor Tractor (fuente original 2), (c) Mezcla 1 y (d)Mezcla 2.

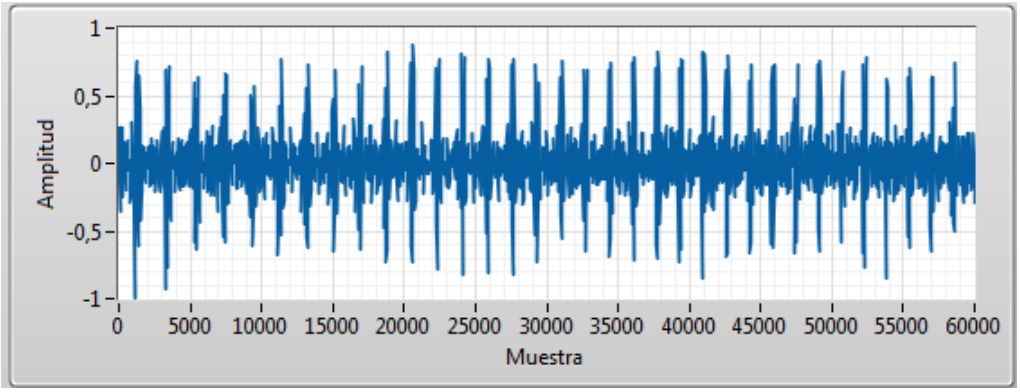
tro de la FPGA presentada en la figura 5.12, se determina que la implementación del algoritmo en la FPGA ocupa poca lógica en general. Esto permite disponer de recursos para la inclusión de otras tareas, relacionadas o no con el mismo algoritmo. De acuerdo al reporte de análisis de retardos críticos, la implementación del algoritmo puede ejecutarse en 9.956ns, con una latencia de catorce ciclos de reloj en total y una frecuencia máxima de funcionamiento de 100 MHz, lo que demuestra que el algoritmo funciona a la frecuencia establecida en los parámetros iniciales.

4.2.2. Validación y análisis de desempeño con tres entradas

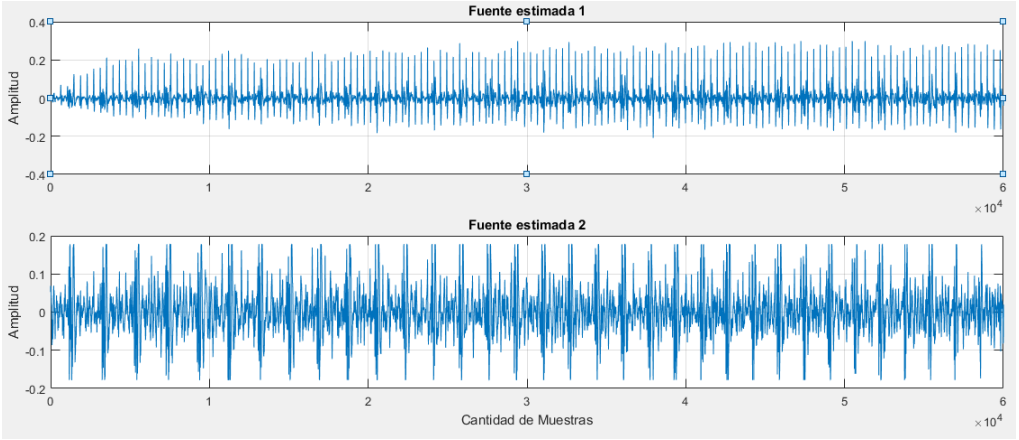
Consecuentemente, para tres fuentes estimadas B es una matriz cuadrada 3×3, las ecuaciones de salida quedan definidas como:



(a) Fuente estimada 1 en LabView



(b) Fuente estimada 2 en LabView



(c) Fuentes estimadas con la FPGA

Figura 4.4: Comparación de las fuentes estimadas (a) y (b) Resultados obtenidos de la simulación teórica en LabView y (c) Fuentes estimadas usando la FPGA

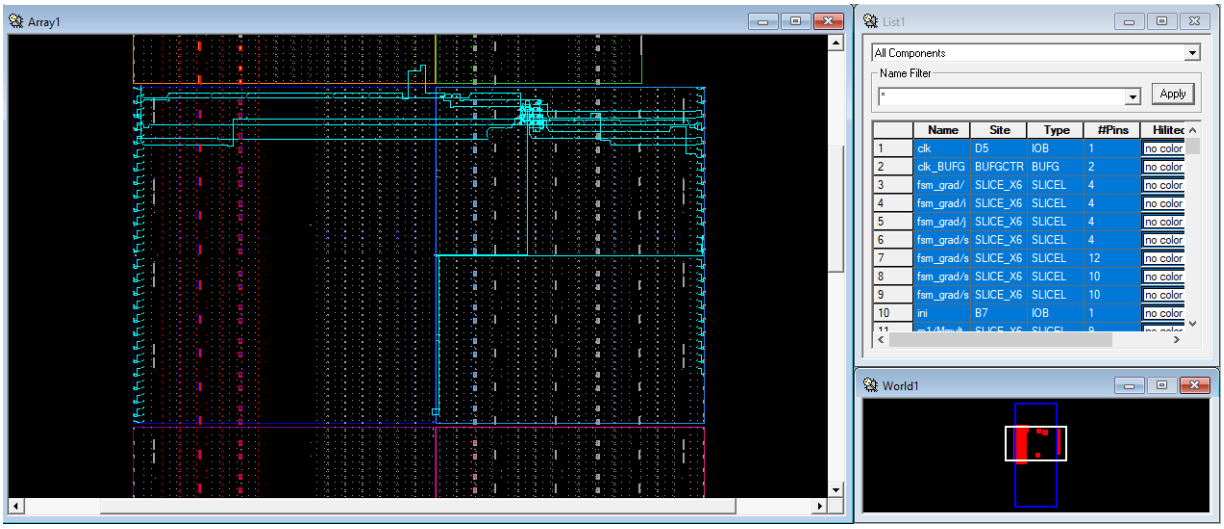


Figura 4.5: Locación de los distintos elementos lógicos del diseño y del respectivo ruteo del algoritmo dentro del FPGA, se observa los recursos disponibles para la inclusión de otros módulos.

$$y_1 = B_{11}X_{11} + B_{12}X_{21} + B_{13}X_{31} \tag{4.7}$$

$$y_2 = B_{21}X_{11} + B_{22}X_{21} + B_{23}X_{31} \tag{4.8}$$

$$y_3 = B_{31}X_{11} + B_{32}X_{21} + B_{33}X_{31} \tag{4.9}$$

Así mismo, las nueve ecuaciones para calcular el parámetro de actualización son de la forma:

$$B_{11} = B_{11}(k-1) - \mu[f(y_1)y_1 - 1]B_{11}(k-1) \tag{4.10}$$

$$B_{12} = B_{12}(k-1) - \mu[f(y_1)y_2]B_{12}(k-1) \tag{4.11}$$

$$B_{13} = B_{13}(k-1) - \mu[f(y_1)y_3]B_{13}(k-1) \tag{4.12}$$

$$B_{21} = B_{21}(k-1) - \mu[f(y_2)y_1]B_{21}(k-1) \tag{4.13}$$

$$B_{22} = B_{22}(k-1) - \mu[f(y_2)y_2 - 1]B_{22}(k-1) \tag{4.14}$$

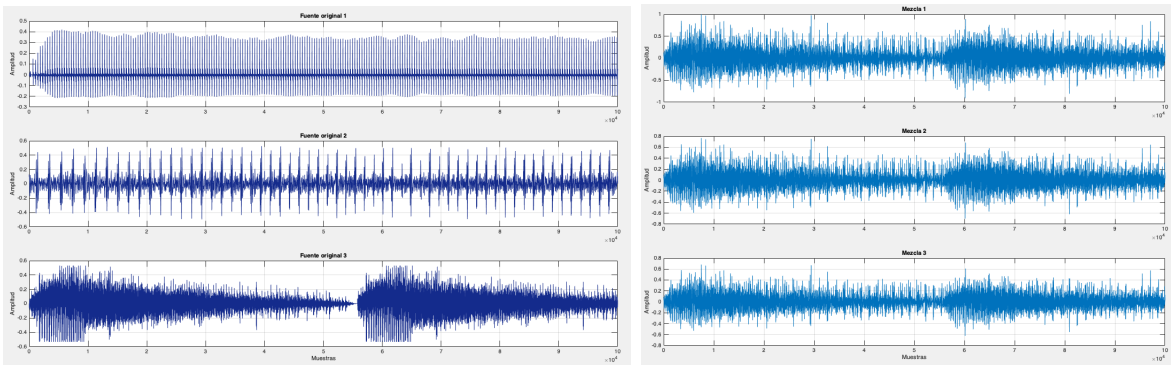
$$B_{23} = B_{23}(k-1) - \mu[f(y_2)y_3]B_{23}(k-1) \tag{4.15}$$

$$B_{31} = B_{31}(k-1) - \mu[f(y_3)y_1]B_{31}(k-1) \tag{4.16}$$

$$B_{32} = B_{32}(k-1) - \mu[f(y_3)y_2 - 1]B_{32}(k-1) \tag{4.17}$$

$$B_{33} = B_{33}(k-1) - \mu[f(y_3)y_3]B_{33}(k-1) \tag{4.18}$$

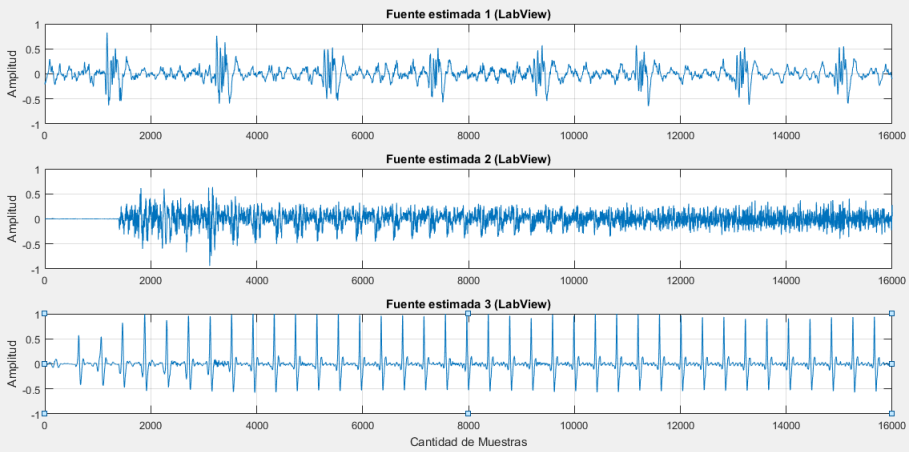
Cabe resaltar que las anteriores ecuaciones son aplicables a cada uno de N vectores de X, es decir, de mezclas capturadas. En la figura 4.6 se ilustra las fuentes originales y sus mezclas sintéticas.



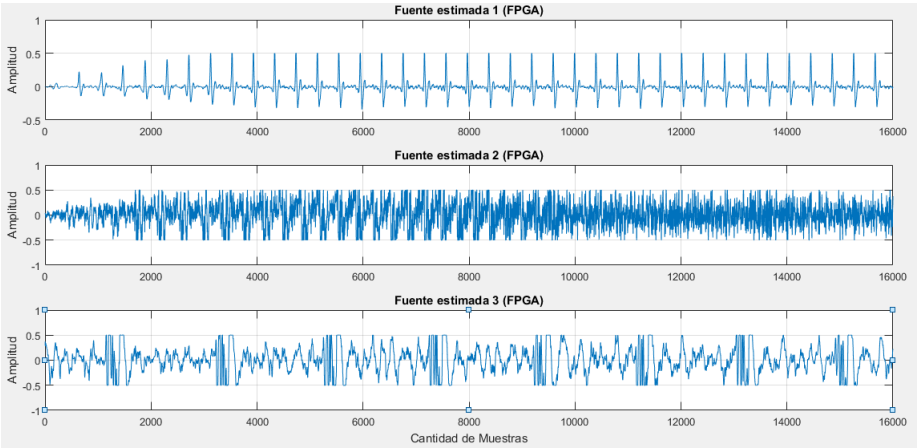
(a) Originales (b) Mezclas

Figura 4.6: Sonidos utilizados para pruebas del algoritmo en la FPGA, (a)Bocina Camión (fuente original 1), (b)Motor Tractor (fuente original 2), (c)Motosierra (Fuente original 3) y (d)Mezcla 1.

Los resultado obtenidos a partir del modelo de alto nivel (LabVIew) y de la FPGA se representan en la figura 4.7. Al igual que las pruebas realizadas con dos entradas, la estrecha correlación entre las formas de onda de las señales estimadas confirman la correcta operación de la arquitectura aritmética del algoritmo de gradiente natural.



(a) Fuentes estimadas con LabView



(b) Fuentes estimadas con FPGA

Figura 4.7: Comparación de las fuentes estimadas (a) Obtenidas de la simulación teórica en LabView y (b) con FPGA

Con el fin de determinar si las señales estimadas corresponde a la fuente original, se realizó el análisis frecuencial mediante el calculo de la transformada rápida de Fourier, presentado en las figuras 4.8, 4.9 y 4.10. Este análisis permitió determinar las diferentes frecuencias presentes en las señales y establecer una relación de energía para cada banda de frecuencia dentro del espectro sonoro. Es importante resaltar que las frecuencias de las fuentes estimadas se concentran en la misma banda de la frecuencia nominal de cada señal original (100 - 800 Hz), mostrando la similitud de las fuentes estimadas con su original correspondiente.

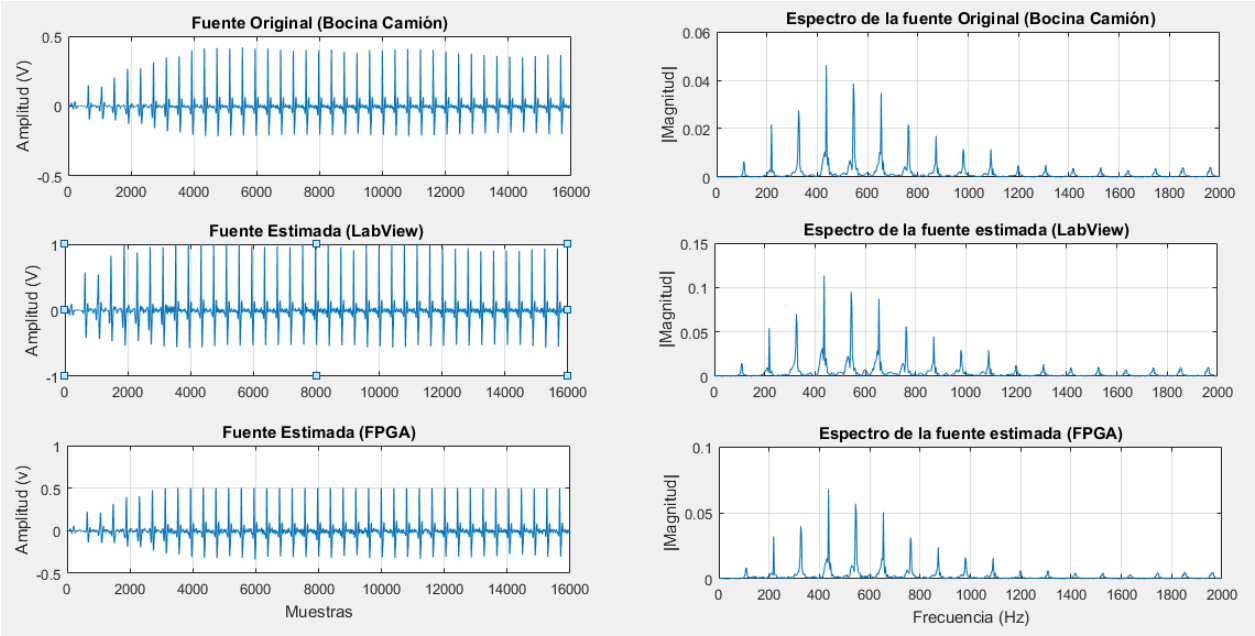


Figura 4.8: Comparación de los espectros de las tres señales estimadas con labView y FPGA de la fuente Bocina Camión.

Después de evaluar el desempeño del diseño digital propuesto para dos y tres señales de entrada y demostrar que la estructura aritmética se ejecuta correctamente, se finaliza con el análisis de la eficiencia del algoritmo de separación de fuentes de gradiente natural en función del criterio de calidad SIR planteado en la ecuación 2.25. Los resultados contenidos en la tabla 4.1 muestran que el modelo de referencia desarrollado en LabView

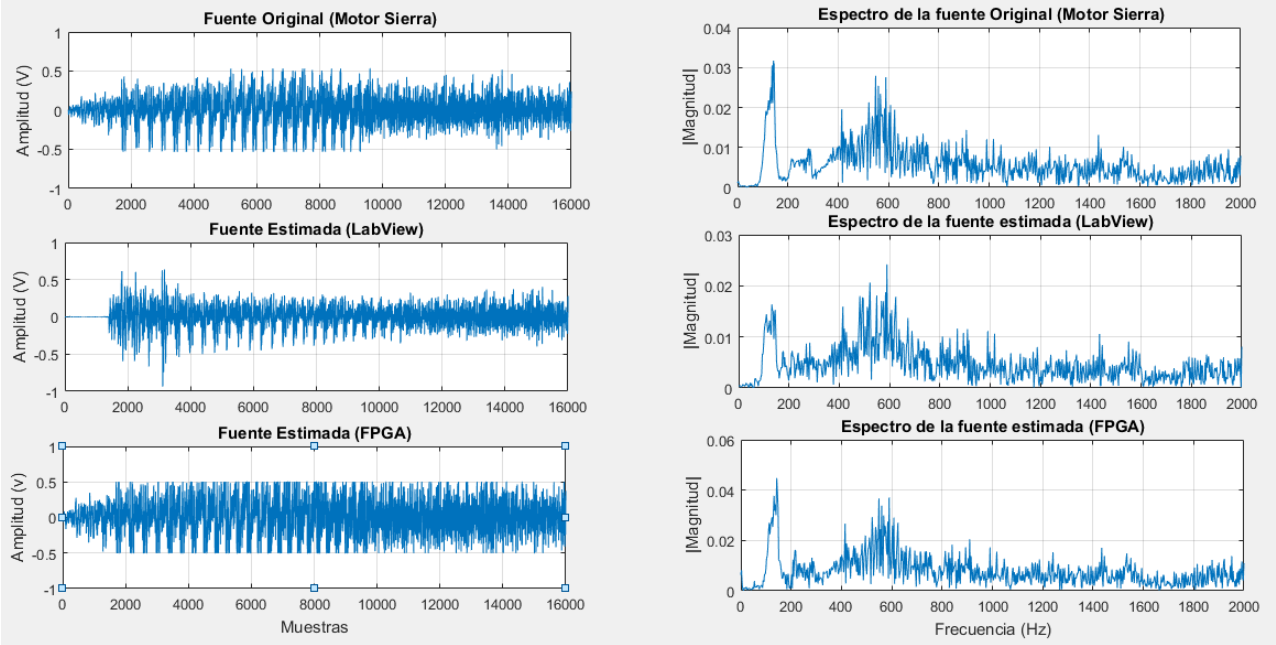


Figura 4.9: Comparación de los espectros de las tres señales estimadas con labView y FPGA de la fuente Motosierra.

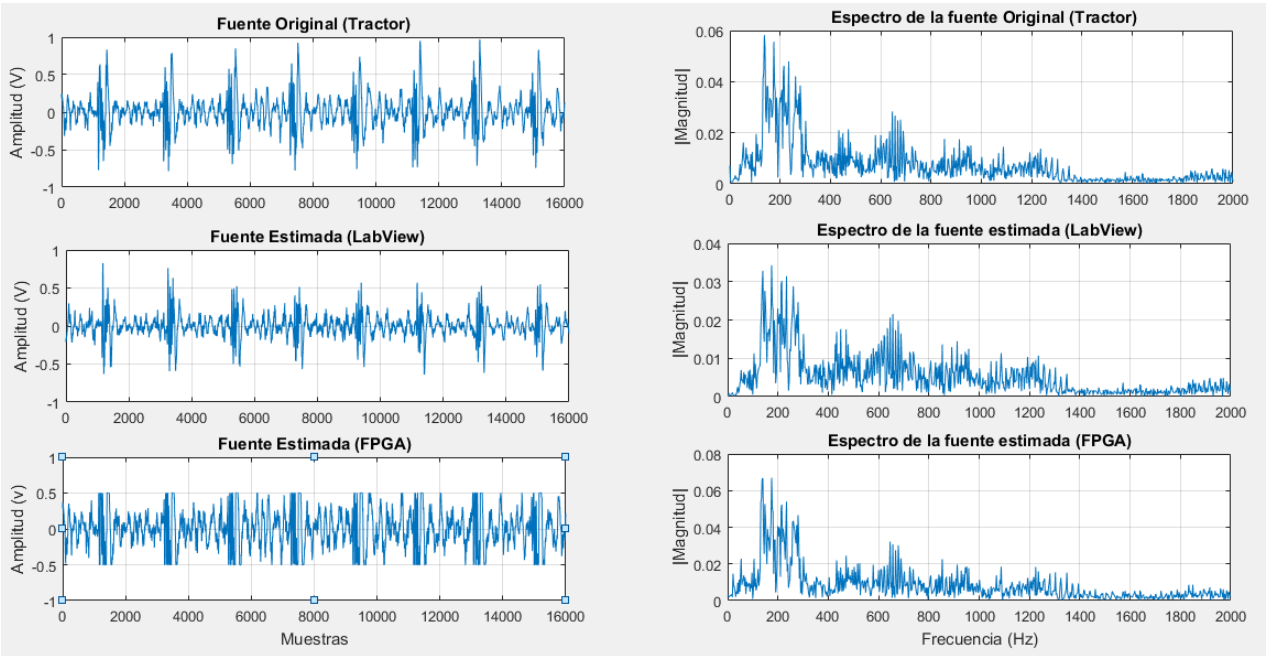


Figura 4.10: Comparación de los espectros de las tres señales estimadas con labView y FPGA de la fuente Motor Tractor

permite determinar la calidad de separación del algoritmo con promedio de 12 dB, para las fuentes de prueba utilizadas. La calidad de las estimaciones obtenidas con la FPGA son bajas debido al proceso propio de la discretización de los datos y en la conversión y la transferencia de los mismos desde la FPGA a la PC.

Tabla 4.1: Evaluación de la calidad de separación usando la ecuación 2.25, el modelo dorado en LabView (punto flotante) presenta la referencia de la eficiencia del algoritmo, en FPGA se reduce la calidad debido al proceso de discretización de las señales en punto fijo.

Cantidad de mezclas	**LabView SIR (dB)**	**FPGA SIR (dB)**	**Reducción %**
2	15	5	33
	11	8	72
3	18	13	72
	22	15	68
	10	3	31

Para finalizar la tabla 4.2 resume el consumo de los recursos lógicos utilizados por la FPGA en la implementación del algoritmo de separación de fuente de gradiente natural para los dos casos evaluados de señales de entrada, resaltando los pocos recursos de hardware que utiliza el mismo, lo que ratifica lo expuesto anteriormente.

4.2.3. Síntesis física en un proceso CMOS de 180 nm

Siguiendo el flujo de diseño digital, para la última etapa se hizo uso de software EDA para evaluar las características de potencia y área del IC antes de la posible fabricación del prototipo. La síntesis lógica o front end produce descripciones del diseño estandarizado

Tabla 4.2: Resumen de los recursos lógicos utilizados por la FPGA en la implementación del algoritmo de separación de fuente de gradiente natural para diferente cantidad de señales de entrada, se resalta los pocos recursos de hardware que utiliza el algoritmo.

Recursos	**2 x 2**		**3 x 3**	
	Total	%	**Total**	%
Flip Flops	406	1	627	2
LUTs	333	1	614	1
IOBs	100	47	102	48

que compila a invocaciones de celdas estándar, sin considerar la tecnología de las mismas. Las celdas implementan la lógica u otras funciones electrónicas que utilizan la tecnología del circuito integrado particular.

La fase final de la implementación digital del algoritmo de separación de fuentes basado en gradiente natural consistió en trasladar el código validado a nivel de RTL a una tecnología CMOS, con el fin de evaluar la factibilidad de una posible integración del IC que ejecuta el algoritmo en un sistema completo para localización de una fuente de audio especifica en un ambiente industrial. Es deseable por tanto conocer si en una tecnología no tan nueva (de 180 nm), el diseño es aún funcional y eficiente. Con el objetivo anterior, se buscó el trazado físico o layout del diseño que ha sido llevado a una estructura digital. El layout es generado a partir de la lista de nodos a nivel de compuertas (Gate Level Netlist) obtenida en la síntesis RTL, este proceso se realizó únicamente para el conjunto de tres mezclas. Se genera el circuito sintetizado mediante la herramienta EDA Design Compiler, de Synopsys, con una biblioteca de celdas estándar provista por el proveedor comercial XFAB (con propósitos estrictamente académicos).

Por medio del IC Compiler de Synopsys, se realizó la síntesis física, está síntesis es ya

fabricable en principio. En la figura 4.11 se muestra un detalle del plano final obtenido para una potencial fabricación. La tabla 4.3 resume las características de área, consumo de potencia y temporizado del módulo sintetizado, obtenidos de las dos síntesis: lógica (Front End) y física (Back End) realizadas con la herramienta, donde se demuestra la factibilidad de integración del algoritmo de separación de fuentes de gradiente natural dentro de un circuito integrado para una aplicación específica.

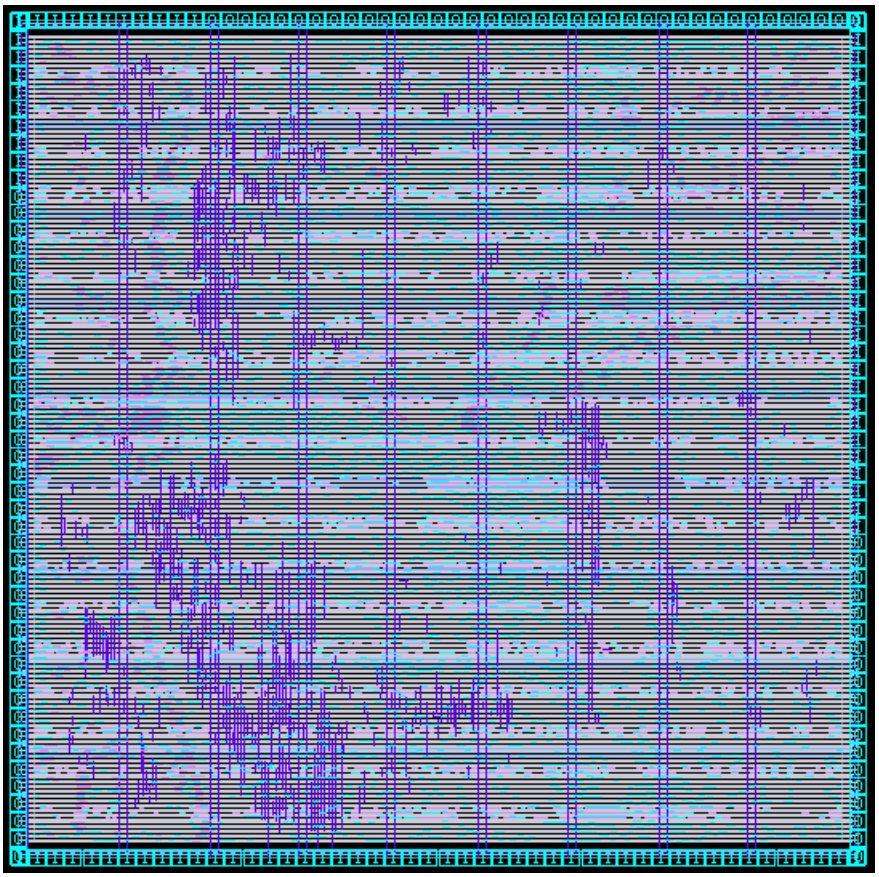


Figura 4.11: Layout final generado por la herramienta IC Compiler de Synopsys del algoritmo de separación de fuentes basado en Gradiente Natural.

Para finalizar, la tabla 4.4 detalla la comparación de resultados obtenidos en otros trabajos sobre rendimiento de diferentes implementaciones de algoritmos fundamentados en métodos ICA para diseños analógicos y digitales publicado en [7]. De estos datos podemos

Tabla 4.3: Resumen de las características de área y consumo de potencia generado por la herramienta IC Compiler de Synopsys, como resultado de la síntesis del algoritmo Gradiente Natural para tres señales de entrada, deduciendo que el IC es de bajo consumo de potencia.

Variable	**Front End**	**Back End**
Área $(\mu\mathbf{m})^2$	1774599,84	1796965,69
Potencia Interna (mW)	1,543	2,501
Pot. Conmutación (mW)	12,881	49,297
Pot. Dinámica Total(mW)	14,425	51,798

resaltar que para las tecnologías utilizadas el consumo de potencia es bajo y la frecuencia de operación está ajustada a los requerimientos de cada diseño.

Tabla 4.4: Resultados de la implementación de diferentes algoritmos fundamentados en métodos ICA [7].

Referencia	**Esta tesis**	**[7]**	**[62]**	**[63]**
Tecnología	180nm CMOS	0.5μ CMOS	90nm CMOS	90nm CMOS
Algoritmo	NG ICA	NG ICA	FastICA	FastICA
Frecuencia (MHz)	50	0.16	11	100
Potencia	51.79mW	192μW	86μW	16.35mW
Diseño	Digital	Analógico	Digital	Digital

4.3. Conclusiones

Se propuso una estructura microelectrónica de punto fijo con eficiencia de hardware para el algoritmo de separación de fuentes basado en gradiente natural estudiado. La detallada descripción arquitectónica mostró que es posible reducir la complejidad del hardware mediante la optimización de diferentes unidades aritméticas, a partir la conversión de estructuras matriciales en simple ecuaciones con sumas y multiplicaciones. En la arquitectura aritmética presentada, la reducción en el hardware conduce al bajo consumo de energía, de acuerdo a los resultados obtenidos de las síntesis del diseño digital.

Los experimentos han demostrado que el rendimiento del algoritmo de gradiente natural está en función del ajuste de menores tasas de aprendizaje, como es de esperarse en este tipo de algoritmos. Es importante reconocer que existen múltiples fuentes de audio en ambientes reales con distintos comportamientos estadısticos. Las fuentes que se utilizaron en los experimentos funcionan como un simple caso demostrativo, por lo que la respuesta de los algoritmos no puede considerarse como una absoluta generalización. Además, por su mismo carácter adaptativo los resultados presentan variaciones aún en pruebas consecutivas con las mismas mezclas de entrada.

Si bien la solución en FPGA es práctica y flexible, en dicha situación, seria conveniente evaluar la viabilidad de integrar la estructura microelectrónica propuesta en un sistema de localización o rastreo redundante, como el inicialmente planteado en la investigación que dio origen a esta tesis, donde la calidad de separación de la fuente de interés no sea un factor decisivo en el resultado de la ubicación del objeto en estudio y con un limite de fuentes a estimar. Es por ello que se demostró la integrabilidad del algoritmo en un proceso comercial CMOS, evaluación que resultó ser positiva en cuanto al consumo de potencia y de área.

Capítulo 5

ALGORITMO DE ESTIMACIÓN DE PARÁMETROS ELÉCTRICOS

5.1. Introducción

Como en cualquier sistema generador de energía, en los sistemas fotovoltaicos PVS (en inglés, Photovoltaic systems) es de suma importancia la eficiencia energética [64]. Para comprender todas las variables que inciden sobre dicha eficiencia, se debe considerar el funcionamiento del sistema en todos sus extremos, desde las curvas características de los paneles generadores, hasta los efectos el procesamiento energético asociado, y el efecto de la carga en el sistema. En el caso particular de los paneles fotovoltaicos, deben analizarse los parámetros internos de los mismos en su estado de operación, y contrastarlos con la ayuda de un modelo matemático aproximado que describe su comportamiento eléctrico no ideal [64]. Existe gran variedad de paneles fotovoltaicos; los más comunes utilizan la unión semiconductora P-N, donde la energía se genera por medio del efecto fotovoltaico y de la radiación electromagnética proveniente del sol, que incide sobre la capa del semiconductor. Cuando los fotones tienen mayor energía que la banda prohibida el semiconductor crea un

par electrón-hueco, y el campo eléctrico generado en la unión P-N mueve los electrones (portadores), produciendo una corriente que es directamente proporcional a la radiación del sol. Debido a este proceso el funcionamiento de un panel se describe por medio de un modelo exponencial, similar al modelo de un diodo de silicio [64].

Un modelo de estimación de parámetros eléctricos de un PVS está compuesto por un conjunto de ecuaciones diferenciales que es posible implementar en lenguaje de descripción de hardware (HDL). El análisis del algoritmo propuesto incluye la verificación del comportamiento dinámico de un sistema de generación fotovoltaica orientado a evaluar el rendimiento en los puntos de máxima potencia, a partir de una única ecuación que estima la relación corriente-voltaje.

Este capitulo presenta el análisis y la evaluación de la implementación en FPGA de un modelo lineal para la estimación de parámetros eléctricos de un PVS, teniendo en cuenta la portabilidad en hardware de estructuras aritméticas de aplicaciones reales, ya sea en un FPGA o un ASIC de bajo costo. El diseño físico del algoritmo de estimación es validado utilizando tecnología CMOS de 180 nm.

5.2. Características eléctricas de un PVS

El rendimiento de un panel fotovoltaico esta estrictamente relacionado con la eficiencia de la conversión de la luz solar en electricidad. Considerando que no toda la luz es 100 % aprovechada por la unión $p - n$ del material semiconductor utilizado, debido a la cantidad y tipo de impurezas adicionadas en el proceso de dopado del silicio, típicamente la eficiencia de un PV se sitúa entre el 15 y 20 %, esto es, casi un sexto de la luz incidente en las celdas. Debido a que la potencia generada por los PVS viene dada en función de la temperatura y la radiación solar, es importante incluir en el sistema un bloque que estime

la eficiencia, tomando en cuenta las características eléctricas del mismo [29].

Para garantizar la eficiencia energética de los PVS, se hace necesario la monitorización y el control de sus variables internas, para lo cual existe una variedad de herramientas que permiten optimizar y diagnosticar la relación corriente-voltaje (I-V), que depende de numerosos factores: la magnitud de la radiación incidente, su espectro, el ángulo de incidencia, la temperatura, y otros parámetros [65].

Las pruebas de rendimiento de un PVS es la medición de la máxima energía eléctrica generada por un panel por metro cuadrado. Este valor depende de la tecnología del módulo, que es el principal factor a tener en cuenta. En general, la caracterización de un panel fotovoltaico se obtiene bajo condiciones estándar de prueba conocidas como STC (en inglés, Standard Test Conditions), siendo la principal prueba del rendimiento de un módulo fotovoltaico, utilizada por la mayoría de los fabricantes. En dicha prueba se establece una determinada cantidad de irradiancia, de 1000 W/m^2 y una temperatura de 25ºC en la superficie del panel [65, 66].

Los PVS son sistemas dinámicos no lineales que no cuentan con un mecanismo que les permita predecir el punto óptimo de máxima potencia. De acuerdo con la figura 5.1, se ha modelado un PVS como una red de dos puertos, con una corriente circulante a través del dispositivo, I_{PV}, y una tensión presente en sus terminales, V_{PV} [23]. Estas variables están relacionadas por medio de una función que depende de la temperatura, T, la incidencia de la radiación del sol, S, y un conjunto de parámetros constantes provenientes del material semiconductor y del proceso de fabricación. Sin embargo, algunos parámetros de los modelos comerciales de PV no difieren en gran medida en sus valores numéricos, pero si, en la forma en que T y S afectan su comportamiento eléctrico [29].

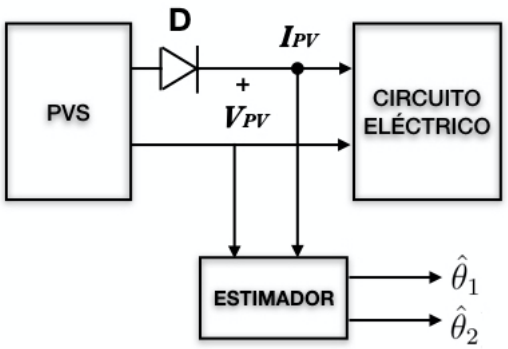


Figura 5.1: Diagrama de bloques general del problema de estimación considerado, donde $\hat{\theta}_1$ y $\hat{\theta}_2$ son los parámetros estimados y D es un diodo de bloqueo que no permite el retorno de la corriente hacia el panel.

La curva caracterıstica de un PVS que muestra la figura 5.2, se obtiene manteniendo fijos los parámetros de irradiancia y temperatura, en condiciones controladas. La potencia entregada solo depende del valor de la carga, de manera que si la carga es pequeña el panel se comportará como una fuente de corriente, pero si la carga es grande se comportará como una fuente de tensión [66]. Al monitorear los parámetros eléctricos de los PVS es posible conocer la máxima potencia entregada por el sistema. Sin embargo, el factor más importante que determina la calidad real del panel es la corriente eléctrica producida idealmente, es decir, con el sol incidiendo de forma perpendicular encima de la superficie. En consecuencia, es necesario implementar sistemas automáticos confiables, que permitan el monitoreo y optimizaciones de las variables eléctricas del PVS, con retroalimentación del rendimiento del mismo en cada instante de tiempo [66].

La máxima potencia puede ocurrir para una gran cantidad de valores de corriente, I_{PV}, y voltaje V_{PV}, al incrementar la carga resistiva desde el corto circuito I_{SC} hasta un valor muy alto cercano a circuito abierto, V_{OC}, donde es posible determinar la potencia máxima $Pmax$, es decir, el punto donde se genera la máxima salida que la celda es capaz de proveer a un cierto nivel de radiación [23].

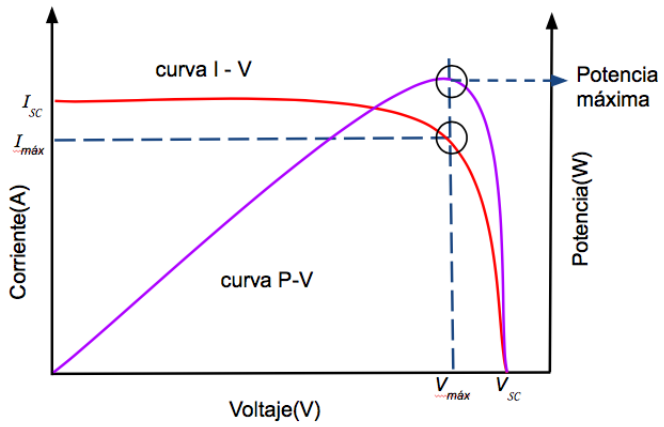


Figura 5.2: Curva característica de corriente, I_{PV}, voltaje V_{PV} y $Pmax$ para un panel PV [9], donde se indica el punto de máxima potencia.

En [29], se plantea la ecuación (5.1) como un modelo de estimación que ofrece una aproximación de dos componentes independientes a partir de la relación $I_{PV} - V_{PV}$, en función de la temperatura, T y la radiación solar S, para una gran cantidad de paneles fotovoltaicos (mono y policristalinos).

$$I_{PV} = \xi(S, T, \rho) - \varphi_1(S, V_{PV}, I_{PV}, \rho) - \varphi_2(S, V_{PV}, I_{PV}, \rho) \tag{5.1}$$

donde $\xi(\cdot) \geq 0$ y $\varphi(\cdot) \geq 0$, son funciones dependientes de la tecnología del panel y $\rho = (\rho_1, \rho_2, ..., \rho_n)$ son los parámetros asociados a los materiales de fabricación contenidos en las hojas de especificaciones.

A partir de las anteriores consideraciones, los resultados teóricos han demostrado que la estimación de los parámetros eléctricos de un PVS radica en la convergencia de un algoritmo compuesto por dos ecuaciones diferenciales [29]. Este algoritmo permite estimar dos incógnitas en una expresión tal como:

$$y = \omega\theta_1 + \theta_2 \tag{5.2}$$

donde θ_1 y θ_2 son los parámetros estimados, ω y y son variables medidas. Para obtener el valor de las dos incógnitas a partir de una única expresión, es necesario que la señal ω sea de excitación permanente. Señales de este tipo pueden ser periódicas o ruido [29]. Para el caso de estudio se utiliza una señal senoidal.

5.3. Modelo del algoritmo estimador

El comportamiento eléctrico de un panel fotovoltaico, es representado con el siguiente modelo exponencial, el cual puede escribirse de la forma de la ecuación 5.2, si se cumplen las igualdades listadas a continuación:

$$\begin{aligned} I_{PV} &= I_g - I_s e^{\alpha V_{PV}} - 1 && (5.3)\\ y &= Ln(I_{PV} - I_g) && (5.4)\\ \omega &= V_{PV} && (5.5)\\ \theta_1 &= \alpha && (5.6)\\ \theta_2 &= ln(I_s) && (5.7) \end{aligned}$$

en donde I_{PV} es la corriente generada, V_{PV} es la tensión generada e I_s, I_g y α, son parámetros propios del panel. Se supone que I_g es conocido y mucho mayor que I_S [29, 66].

El anterior modelo fue derivado con base en el comportamiento intrínseco de los convertidores conmutados de potencia [23, 29]. En el diagrama de bloques de figura 5.1, se observa que el estimador debe estar conectado un circuito eléctrico activo, con el fin de

obtener las dos variables de entrada, la corriente, I_{PV}, y la tensión V_{PV}, y ser capaz de estimar correctamente los dos parámetros desconocidos, $\hat{\theta}_1$ y $\hat{\theta}_2$. Definiendo $\hat{\theta} = \hat{\theta}_1\ \hat{\theta}_2]^T$ como los valores estimados de θ, se obtiene la ecuación diferencial 5.8 que describe el algoritmo estimador [29].

$$\dot{\hat{\theta}} = \Gamma\Phi\left(y - \Phi^T\hat{\theta}\right) \tag{5.8}$$

donde Γ es una matriz definida positiva, de la forma:

$$\Gamma = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22} \end{bmatrix}$$

Las otras condiciones específicas de operación del algoritmo estimador de parámetros se definen a partir de las señales de entrada del sistema: la onda senoidal, y, y la corriente, I_{PV}, identificada por ω. El algoritmo estimador de parámetros se representa mediante las ecuaciones diferenciales (5.9) y (5.10) [29].

$$\dot{\hat{\theta}}_1 = (\Gamma_{11}\omega + \Gamma_{12})(y - \hat{y}) \tag{5.9}$$

$$\dot{\hat{\theta}}_2 = (\Gamma_{21}\omega + \Gamma_{22})(y - \hat{y}) \tag{5.10}$$

donde $\hat{y}$ se define por la relación:

$$\hat{y} = \hat{\theta}_1\omega + \hat{\theta}_2 \tag{5.11}$$

Estas ecuaciones son implementables de manera directa en un sencillo circuito secuencial, siguiendo el ejemplo de una implementación similar de un filtro lineal de Kalman,

realizada en el DCILab y reportada en [17, 21], y sin poseer la restricción en el caso actual de minimizar la operación de multiplicación que, en los casos citados debió implementarse como una unidad aparte según el algoritmo secuencial de Booth.

5.4. Definición de condiciones de operación del algoritmo estimador

Con miras a diseñar un sistema automático, y además confiable, que permita el monitoreo y optimización de las variables eléctricas del PVS, con retroalimentación del rendimiento del mismo, esta sección se centra en evaluar las condiciones de operación del módulo de estimación de parámetros, partiendo de las variables de entrada previamente acondicionadas. Como muestra la figura 5.3, el PVS general está compuesto por dos etapas de acondicionamiento de las señales de entrada, I_{PV} y V_{PV}.

La etapa de preprocesamiento, puede implementarse por medio de un circuito acondicionador analógico logarítmico (con su respectivo aislamiento galvánico, al requerirse muestrear señales con decenas de voltios de amplitud), para realizar la linealización de los datos que se han descrito como un modelo exponencial. Después de ser estimados los dos parámetros de interés, es necesaria una etapa de pos-procesamiento (particularmente, la des-linealización de los mismos), para obtener los valores estimados reales. Etapa que puede realizarse también en la FPGA mediante alguna estructura de procesamiento como un algoritmo CORDIC (ver ejemplos del mismo en [4]). La implementación de éstas dos etapas no son el interés central en este trabajo, y principalmente dado que un diseño ya que resuelve el punto anterior se ha presentado por investigadores del DCILab en [19].

Note que el sistema de ecuaciones planeadas en la sección anterior, es lineal, y particularmente, la ecuación 5.11 es la de una recta. Por tal razón, para evaluar experimentalmente

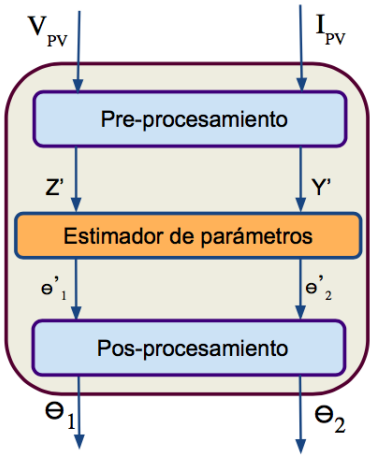


Figura 5.3: Diagrama general del sistema fotovoltaico, donde se indica el flujo que sigue cada una de las variables en los módulos de pre y pos-procesamiento.

el estimador, se utilizó un simple circuito eléctrico en serie como el mostrado en la figura 5.4, que permite modelar el comportamiento linealizado de un módulo fotovoltaico. Por analogía con las ecuaciones 5.9 y 5.10, se tiene que y es equivalente a $Ln(I_{PV} - I_g)$, R $= \alpha$ y $-E = Ln(\omega)$. La corriente que circula por R es equivalente a la variable (ω) en la ecuación 5.2. La amplitud de la señal alterna se considera igual al valor máximo de la fuente de corriente continua, E.

Para ajustar las condiciones iniciales de la señal alterna y, primero se realizaron simulaciones del algoritmo implementado en MatLab con tres señales periódicas: sinodal, triangular y cuadrada, con amplitud 3V a 100 KHz, de donde se evidenció que no hay una diferencia sustancial en el tiempo de convergencia del algoritmo al utilizar las diferentes señales.

Finalmente, para la implementación digital se considera el diagrama general del estimador de la figura 5.5, donde se muestran todas las variables de entrada y salida del sistema completo, $\hat{\theta}_1$ y $\hat{\theta}_2$ son las señales de salida o valores estimados, los cuales deben

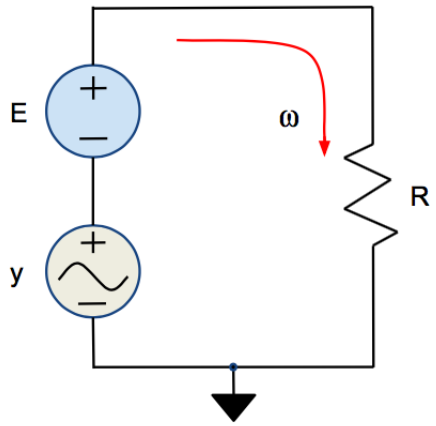


Figura 5.4: Circuito serie con las señales de entrada al estimador, donde ω representa la corriente proveniente del módulo preprocesamiento y la variable y es la amplitud de la fuente alterna.

converger al valor de la resistencia R y a $-E$, respectivamente.

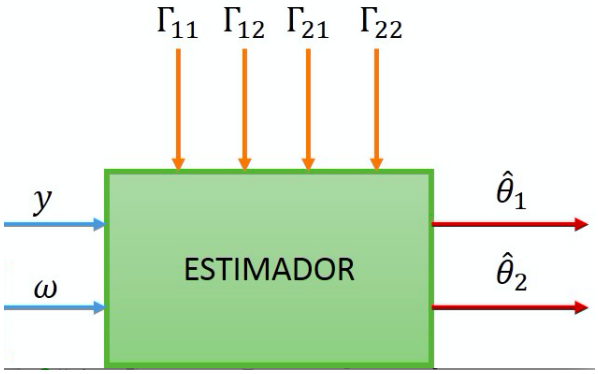


Figura 5.5: Diagrama general del estimador de parámetros, donde, y es la tensión de entrada, la corriente ω provenientes del PVS y Γ la matriz de ajuste.

5.5. Ajuste de parámetros para el diseño digital

Siguiendo el diagrama de bloques general de la figura 5.6, se implementaron digitalmente las ecuaciones que han sido previamente discretizadas. Además, considerando el caso que V_{PV} es constante y que la matriz Γ es uno de los parámetros fundamentales para la convergencia del algoritmo estimador [29], se asumen valores reales y positivos en la diagonal principal y ceros para los demás, es decir, $\Gamma_{12} = \Gamma_{21} = 0$ y $\Gamma_{11} = \Gamma_{22}$. Se evaluó el algoritmo con diferentes valores Γ, donde se observó que el aumento de este parámetro influye en el tiempo de convergencia del algoritmo. En función de los resultados obtenidos, para este estudio, se utilizó $\Gamma_{11} = \Gamma_{22} = 8000$. Con este ajuste las ecuaciones se reduce a dos variables de entrada, ω y y.

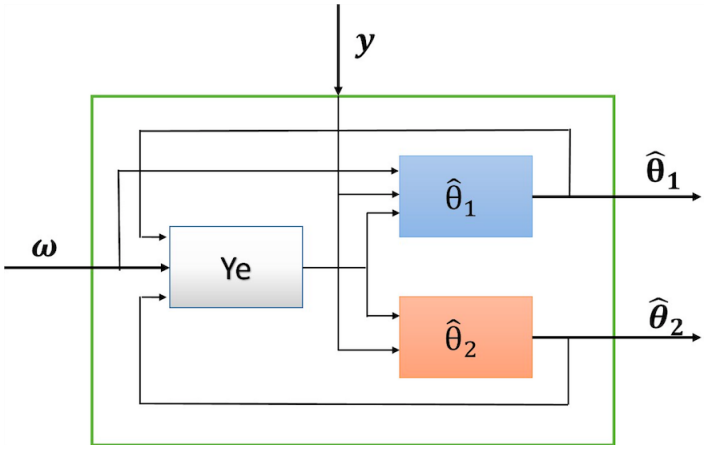


Figura 5.6: El sub bloque *ye*, que realiza la función descrita en la ecuación (5.11), y dos sub bloques que ejecutan las ecuaciones (5.9) y (5.10). Todas la variables de entrada y salida del algoritmo son manejadas en punto fijo.

Para efectos prácticos, el algoritmo se dividió en tres sub bloques como se muestra en la figura 5.6. La codificación de cada sub bloque del algoritmo se realizó a partir de las estructuras de datos mostradas en la figura 5.7. La implementación del algoritmo se hizo en Verilog. Para mejorar la velocidad de convergencia del algoritmo, se segmentó (pipeline)

el código en seis (6) ciclos de reloj, adicionando registros en cada salida de las diferentes operaciones aritméticas, esto permite también, incrementar el rendimiento del sistema. La segmentación permitió el análisis de todos los caminos sincrónicos y así verificar que el algoritmo funcione a la frecuencia requerida. Los resultados obtenidos son presentados en la siguiente sección.

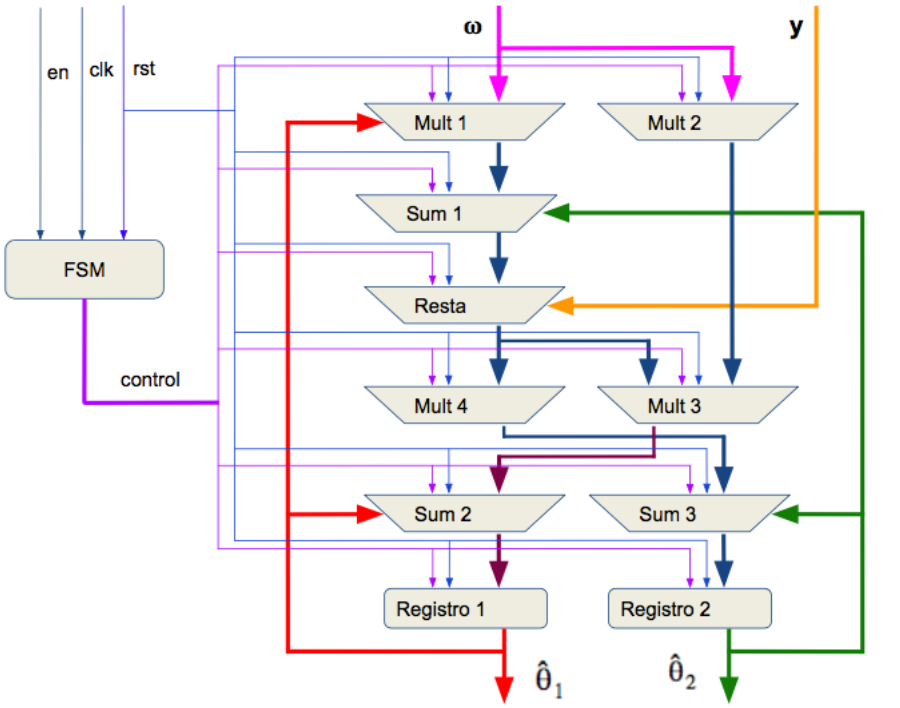


Figura 5.7: Diagrama de flujo de datos del algoritmo estimador de parámetros. Cada unidad de cálculo (suma, multiplicación) está registrada a la salida, para mejorar el rendimiento del sistema.

Posteriormente se discretizaron las ecuaciones 5.9 y 5.10 para ser implementadas en HDL, que quedaron de la forma:

$$\hat{\theta_1}(t_i) = \hat{\theta_1}(t_i - 1) + T_S[\Gamma_{11}\omega(t_i - 1) + \Gamma_{12}][y(t_i - 1) - \hat{y}(t_i - 1)] \tag{5.12}$$

$$\hat{\theta_2}(t_i) = \hat{\theta_2}(t_i - 1) + T_S[\Gamma_{21}\omega(t_i - 1) + \Gamma_{22}][y(t_i - 1) - \hat{y}(t_i - 1)] \tag{5.13}$$

Si se supone que $\Gamma_{12} = \Gamma_{21} = 0$, las ecuaciones 5.12 y 5.13 se transforman en:

$$\hat{\theta_1}(t_i) = \hat{\theta_1}(t_i - 1) + T_S[\Gamma_{11}\omega(t_i - 1)][y(t_i - 1) - \hat{y}(t_i - 1)] \tag{5.14}$$

$$\hat{\theta_2}(t_i) = \hat{\theta_2}(t_i - 1) + T_S[\Gamma_{22}][y(t_i - 1) - \hat{y}(t_i - 1)] \tag{5.15}$$

5.6. Resultados de Implementación del estimador

5.6.1. Simulación teórica

A partir del modelo en tiempo continuo del algoritmo desarrollado en MatLab, se realizaron simulaciones en tiempo continuo y tiempo discreto (suponiendo un tiempo de muestreo Ts = 1 μs), para ajustar de la dinámica del estimador. Las gráficas de la figura 5.8 muestran los resultados de la conversión de las ecuaciones del algoritmo de tiempo continuo (rojo) a tiempo discreto (azul), de donde se determina que hay un error promedio de 0.1 %, en los dos parámetros de salida: $\hat{\theta_1}$, y $\hat{\theta_2}$. Este error es el esperado producto de la discretización del algoritmo estimador, para lo que se utilizó el método de Euler.

Para la verificación RTL se realizó la conversión analógico a digital de las muestras de los datos provenientes del circuito serie, todo en punto fijo. Las gráficas de la figura 5.9 muestra los resultados de las simulaciones de la convergencia de las dos variables de

salida del algoritmo estimador de parámetros. En rojo lo obtenido en Matlab y en azul la simulación de algoritmo implementado en FPGA. A partir de los datos presentados se puede determinar que existe una mínima diferencia entre los dos resultados obtenidos. La divergencia entre ellos es menor al 4 %. Este error se debe al uso de punto fijo en la implementación en FPGA (contra el uso de coma flotante en Matlab). Este error es en todo caso, despreciable, y puede minimizarse simplemente aumentando la resolución del estimador en punto fijo. Algo sencillo de realizar pues el código RTL usado es totalmente parametrizable.

Al comparar los resultados obtenidos de la FPGA con el modelo de alto nivel implementado en MatLab (ver fig. 5.9), se observa un comportamiento muy similar en la convergencia del algoritmo. El error promedio aumenta levemente, 4.5 %. Este error se atribuye al proceso de extracción desde la FPGA a la PC: en primer lugar, no existe una forma de mapear de manera exacta fracciones en binario a decimal; en segundo lugar, al utilizar una UART en modo ASCII para la extracción de los datos de la FPGA, existe un error extra en la conversión de de binario a ASCII, y de ASCII a decimal (error que no estará presente en el procesamiento mismo en binario dentro del sistema).

5.6.2. Validación del algoritmo en FPGA

La figura 5.10 muestra el flujo del sistema completo desarrollado en Verilog para validar el desempeño del algoritmo. Los módulos de memoria ROM contienen los datos de las variables de entrada, que se cargan con cada ciclo de reloj por medio del contador de direcciones. Un módulo de conversión serie a paralelo fue adicionado para transmitir los datos desde la FPGA a la computadora. Se utilizó una placa de desarrollo del fabricante DIGILENT Inc., la cual consta un chip de lógica programable Spartan 6, producido por la empresa Xilinx. La figura 5.11 muestra todo el sistema de pruebas utilizado para obtener los resultados desde la FPGA, tomando los datos del circuito serie indicado con 1. La

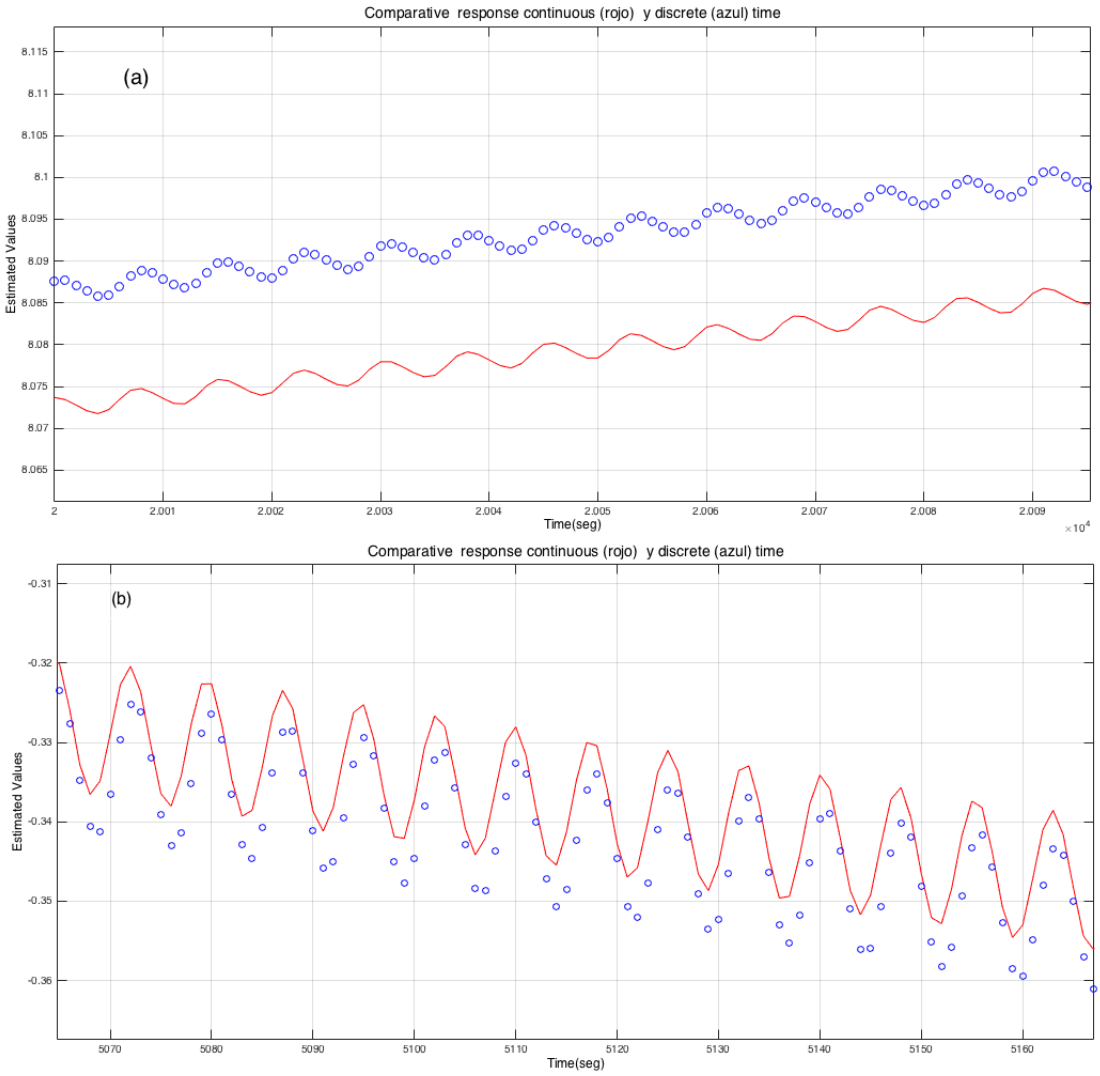


Figura 5.8: Gráfica comparativa de las ecuaciones diferenciales de continuo (rojo) y tiempo discreto (azul), (a) $\hat{\theta}_1$, (b) $\hat{\theta}_2$. El error promedio de la conversión es 0.1 %.

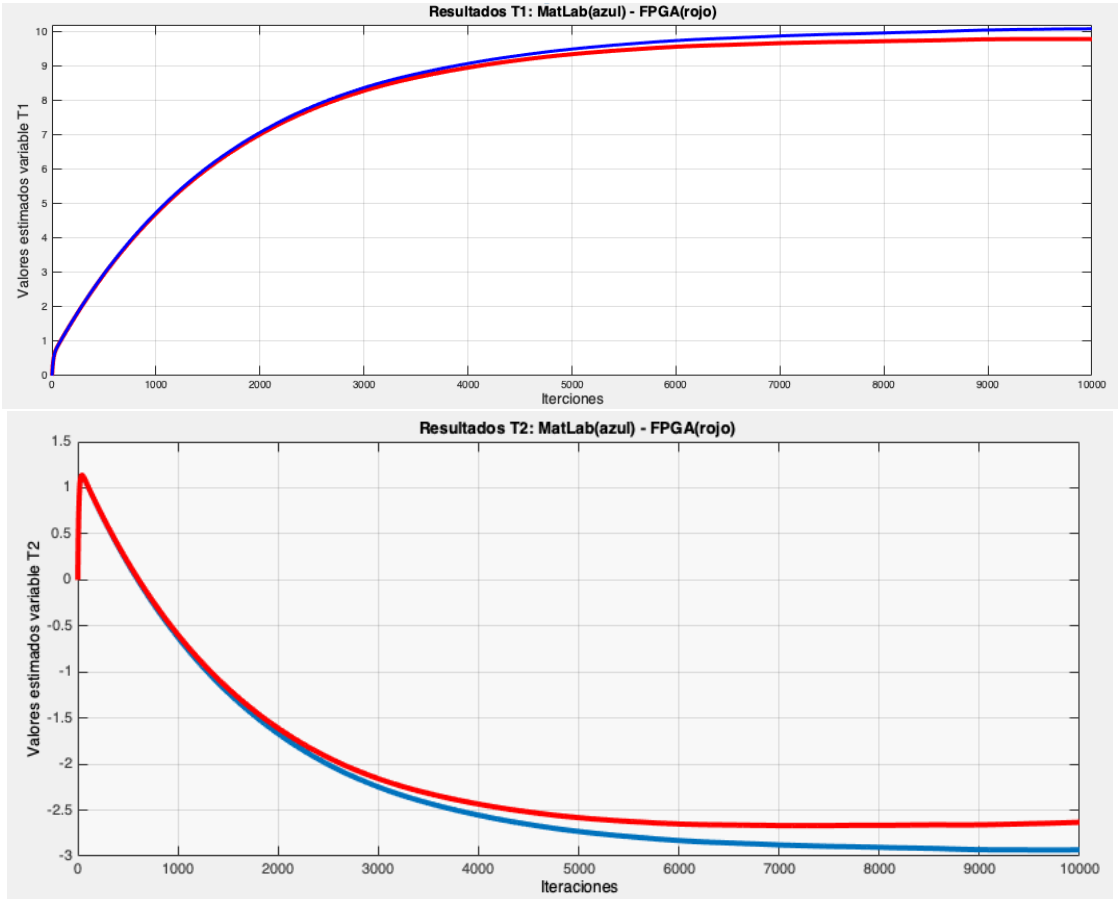


Figura 5.9: Resultados de la simulación en Matlab (rojo) y de la implementación en FPGA (azul), T1 $= \hat{\theta}_1$, T2 $= \hat{\theta}_2$. La divergencia entre ellas menor al 4 %, debido a que el algoritmo en la FPGA trabaja en punto fijo, y MatLab en punto flotante. El mismo puede disminuirse aumentando la resolución en la FPGA si fuese necesario (resultados de 24 bits).

conversión analógico a digital de los datos de entrada fue realizada por medio del PMOD AD5 de alta precisión de Digilent, indicado en la figura 5.11 con el número 2.

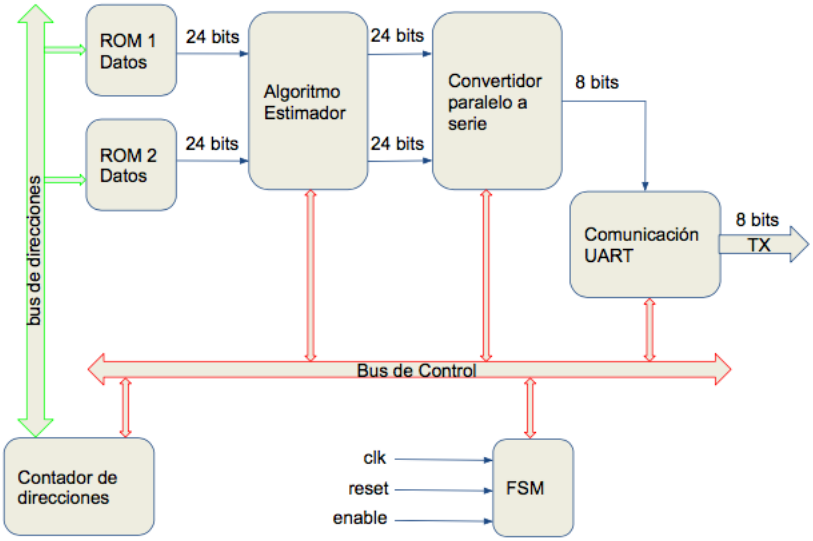


Figura 5.10: Sistema para validar el funcionamiento del algoritmo estimador de parámetros en la FPGA Spartan 6 de Xilinx. Los datos de entrada son cargados en las memorias ROM y los datos de salida se transmiten a la PC

Por otro lado, considerando el reporte de síntesis de la herramienta, se determina que la implementación del algoritmo estimador en la FPGA ocupa poca lógica en general, lo que permite disponer de recursos para la inclusión de otras tareas, relacionadas o no con el mismo. De acuerdo al reporte de análisis de retardos críticos, la implementación del algoritmo puede ejecutarse en 9.956ns, con una latencia de seis (6) ciclos de reloj y una frecuencia máxima de funcionamiento de 100 MHz, lo que demuestra que el algoritmo funciona a la frecuencia establecida en los parámetros iniciales.

Adicional, al observar la localización de los distintos elementos lógicos del diseño y su respectivo ruteo dentro del FPGA de la figura 5.12 y el total de recursos en la tabla 5.1, se demuestra el bajo porcentaje de hardware utilizado por el algoritmo estimador, dando

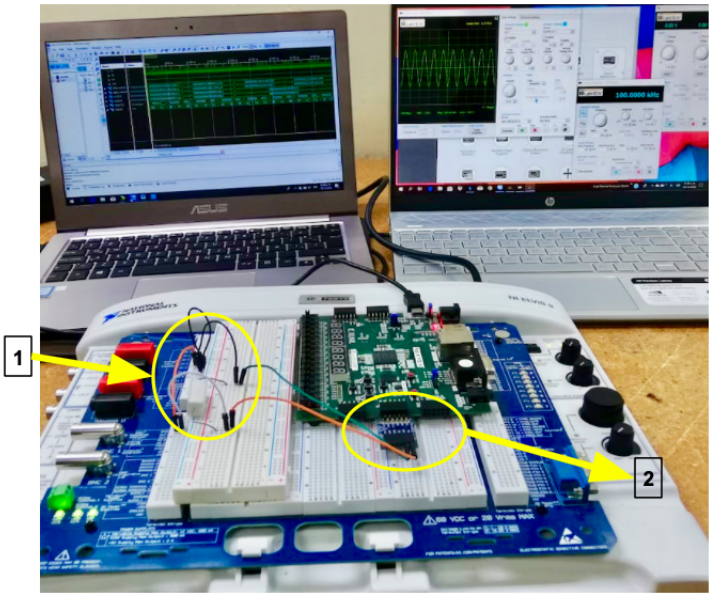


Figura 5.11: Sistema de pruebas utilizado para obtener los resultados desde la FPGA.

lugar a la inclusión de otros módulos de pre y pos procesamiento para el caso de un sistema completo de monitoreo de la máximo potencia en un panel fotovoltaico.

Tabla 5.1: Resumen de los recursos lógicos utilizados por la FPGA en la implementación del algoritmo de estimación de parámetros eléctricos de un panel fotovoltaico, se resalta los pocos recursos de hardware que utiliza el mismo.

Recursos	**Total**	**Utilización (%)**
Flip Flops	253	1
LUTs	154	1
IOBs	100	43

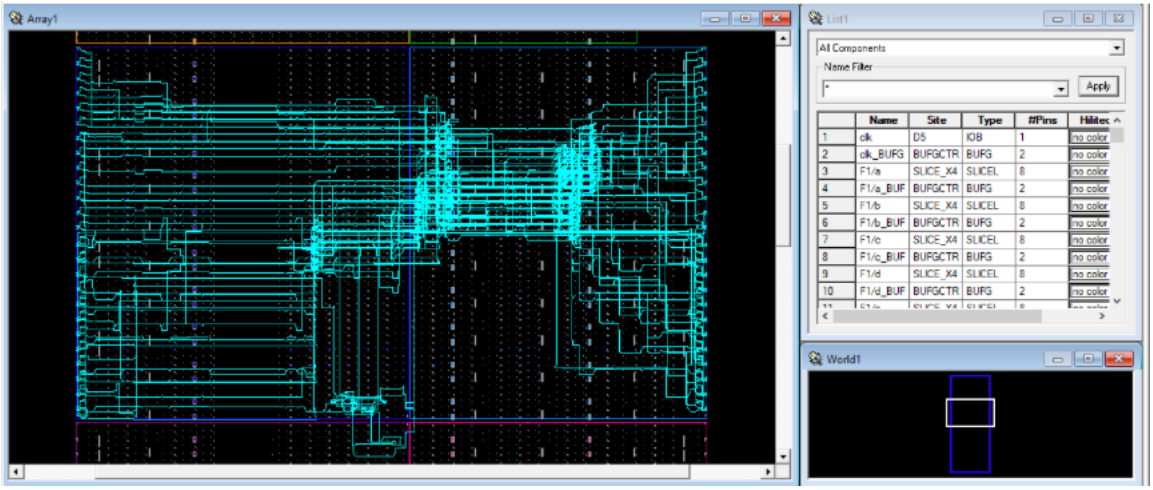


Figura 5.12: Locación de los distintos elementos lógicos del diseño y del respectivo ruteo del algoritmo estimador dentro del FPGA, se observa los recursos disponibles para la inclusión de otros módulos.

5.6.3. Síntesis física en proceso CMOS de 180 nm

Cómo última etapa, se propuso trasladar el código validado RTL a una tecnología CMOS. Ello con la intención de evaluar la factibilidad de la integración de un sistema completo de estimación sobre el mismo panel fotovoltaico (pues en principio es posible integrar el mismo circuito sobre el sustrato de silicio sobre el que se fabrican las celdas solares, o interconectarlo en un montaje híbrido directamente sobre el mismo). Es deseable por tanto conocer si en una tecnología no tan nueva (de 180 nm), el diseño es aún funcional y eficiente.

Con el objetivo anterior, se buscó obtener el trazado físico del diseño que ha sido llevado a una estructura digital. El layout es generado a partir de la lista de nodos a nivel de compuertas (Gate Level Netlist) obtenida en la síntesis RTL, realizada en la verificación funcional en HDL del paso anterior. Se genera el circuito sintetizado mediante la herramienta EDA Design Compiler, de Synopsys, con una biblioteca de celdas estándar provista por el proveedor comercial XFAB (con propósitos estrictamente académicos). Por

medio del programa IC Compiler de Synopsys, se realizó la síntesis física, está síntesis es ya fabricable en principio (aunque faltará aún toda la estructura de interconexión externa, no necesaria para esta prueba de concepto).

La tabla 5.2 resume las características de área y consumo de potencia del módulo sintetizado. Los resultados de las dos síntesis: lógica (Front End) y física (Back End), indican que este algoritmo puede correr a 100 MHz, con un consumo de potencia dinámica total de solo 4.41 mW, un área de 0.07563 mm^2. A partir de las dos síntesis realizadas se demuestra la factibilidad de integración del algoritmo estimador de parámetros dentro de un circuito integrado.

Tabla 5.2: Resumen de las características de área y consumo de potencia del algoritmo estimador de parámetros.

Variable Física	**Front End**	**Back End**
Área (μm)^2	75634.591351	75447.165131
Potencia (mW)	5.6039	4.4251

En la figura 5.13 se muestra un detalle del plano final obtenido para una potencial fabricación.

Es importante resaltar que este algoritmo solo ha sido implementado en simulación ya que es parte de un complejo sistema de control de un PVS. Por lo tanto, no existen pruebas de diseño digital del mismo, para realizar las comparaciones pertinentes de área y consumo de potencia del prototipo propuesto.

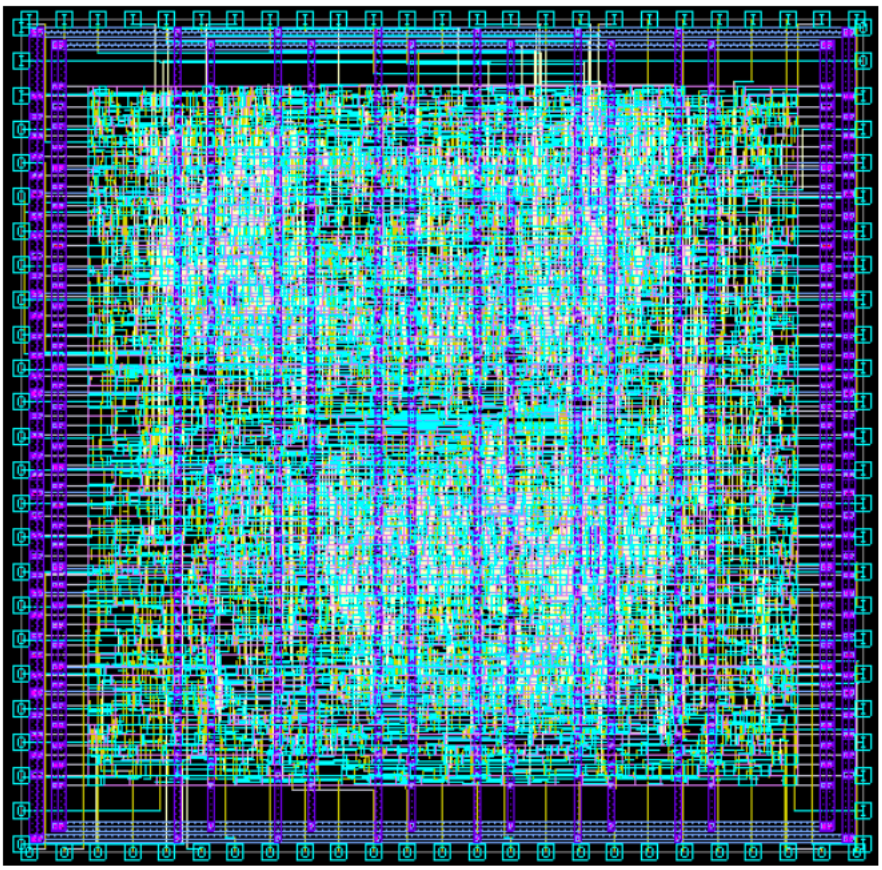


Figura 5.13: Layout final generado por la herramienta IC Compiler de Synopsys.

5.7. Conclusiones preliminares

Se ha demostrado la factibilidad de incorporar un algoritmo estimador de parámetros para un PVS en una FPGA, con porcentajes de error del orden del 4 % con respecto a su modelo de referencia en Matlab. Este trabaja a una frecuencia de 100 MHz, lo que permite el monitoreo de señales de alta frecuencia, muestreadas a una mega muestra por segundo.

La estructura implementada es altamente flexible, lo que significa que puede incorporarse dentro de un sistema de monitoreo en tiempo real de un PVS, con miras a la administración automática de una red inteligente compuesta por decenas o centenas de paneles.

Se ha mostrado además que el mismo podría integrarse de manera sencilla dentro de un ASIC en caso de buscarse la escalabilidad del sistema. Un siguiente paso implicaría ya el desarrollo de las interfaces de usuario y su conectividad dentro del sistema completo.

Capítulo 6

CONCLUSIONES

En este trabajo se han mostrado dos casos en particular de aplicaciones reales de procesamiento numérico local. Estos dos sistemas, representados por medio de ecuaciones diferenciales no lineales, fueron evaluados en función de la transformación de dichas ecuaciones en modelos descritos únicamente con operaciones aritméticas básicas, para ser implementadas de forma digital y así comprobar la viabilidad de llevar dichas arquitecturas a una FPGA o un ASIC.

La detallada descripción arquitectónica aplicada en el análisis de los dos casos mostró que es posible reducir la complejidad del hardware mediante la optimización de diferentes unidades aritméticas, a partir la conversión de estructuras matriciales en simple ecuaciones compuestas de sumas, restas y multiplicaciones. Considerando los resultados obtenidos, dicha reducción conduce a ICs de bajo consumo de energía.

El desarrollo de sistemas electrónicos avanzados es una tarea que requiere de herramientas y técnicas de diseño concienzudas que puedan lidiar con la complejidad de los sistemas y de los modelos que los caracterizan. En el caso de esta tesis, el uso adecuado de

herramientas de software de diseño automatizado y el seguir una adecuada metodología de implementación ha ayudado en la síntesis de sistemas electrónicos funcionales con una aplicación específica sobre FPGAs. Ese mismo proceso ha sido repetido para una potencial integración en ASIC. Esto ha implicado conocer de manera detallada el comportamiento teórico del sistema a implementar, lo que permitió generar los modelos dorados o de referencia necesarios para verificar funcionalmente el sistema tras la síntesis lógica y física del circuito.

Considerando las conclusiones presentadas al final del capítulo de los dos casos analizados, podemos afirmar que los objetivos propuestos fueron cubiertos satisfactoriamente. Además, los resultados obtenidos cumplen con las expectativas esperadas en cada uno.

A partir de los resultados generales obtenidos en ésta investigación se recomienda:

- Cuando se utiliza ICA para trabajar con aplicaciones reales es importante tener en cuenta las limitaciones que tienen los algoritmos, considerando que existen múltiples fuentes de audio en los ambientes reales con distintos comportamientos estadísticos, éstas características deben tenerse en cuenta en el momento de elegir un algoritmo BSS para implementarlo en hardware.

- La propuesta estructura microelectrónica del algoritmo de separación de fuentes basado en gradiente natural demostró eficiencia de hardware, la solución es práctica y flexible, es conveniente evaluar la viabilidad de integrarla en un sistema de localización o rastreo redundante, como el inicialmente planteado en la investigación que dio origen a esta tesis, donde la calidad de separación de la fuente de interés no sea un factor decisivo en el resultado de la ubicación del objeto en estudio.

- La estructura microelectrónica implementada para la estimación de los parámetros eléctricos de un sistema fotovoltaico es altamente flexible, lo que significa que puede incorporarse dentro de un sistema de monitoreo en tiempo real de un PVS, con miras a la administración automática de una red inteligente compuesta por decenas o centenas de paneles.

Trabajo Futuro

Para finalizar, se sabe que es necesario realizar un proceso de verificación de los sistemas estudiados que incluya pruebas en campo de los prototipos aquí propuestos. Si bien el tema de verificación y validación no ha sido parte integral de esta tesis, será necesario para continuar con el tema, conducir en esa dirección los siguientes esfuerzos.

Lo que resta son las pruebas finales de los prototipos fabricados físicamente (para lo cual, desgraciadamente, no se contó con los fondos debido al problema mencionado en la introducción con el proyecto de investigación internacional que dio arranque a esta tesis). Esto, sumado a la verificación en campo, y el desarrollo de otras funciones útiles a incorporar en los sistemas, tales como adquisición de datos y despliegue de los mismos, son labores a seguir. De ser posible, además, la consecución de fondos para la fabricación de estos prototipos en ASIC, abriría las puertas a otras potenciales aplicaciones siguiendo la metodología acá usada para el desarrollo de los prototipos demostrados.

Bibliografía

[1] I. Jaramillo, *Tendencias en diseño digital CMOS - VLSI*, ch. 6: Iterative algorithms. Universidad Nacional de Colombia, 1ra ed., 2012.

[2] N. Weste, *CMOS VLSI Design: A circuits and systems perspective.* PEARSON, 3rd ed., 2005.

[3] C. Sisterna, *Diseño de sistemas digitales Avanzados con VHDL FPGA.* Guía de Referencia Rápida. http://hdl-fpga.blogspot.com.

[4] J. Deschamps, *Synthesis of Arithmetic Circuits - FPGA, ASIC and Embedded Systems.* John Wiley and Sons, Inc., 2006.

[5] C. Lozano, A. Gómez, and A. Chacón-Rodríguez, "*Analysis of source separation algorithms in industrial acoustic environments*," *6th IEEE Latin America Symposium on Circuit and Systems*, 2015.

[6] A. Gómez, "*Análisis y validación de algoritmos de separación de fuentes sonoras para su aplicación en entornos de robótica industrial*," 2014. Instituto Tecnológico de Costa Rica, Escuela de Ingeniería Electrónica.

[7] M. Stanacevic, S. Li, and G. Cauwenberghs, "Micropower mixed-signal vlsi independent component analysis for gradient flow acoustic source separation," *IEEE Transactions on Circuit and Systems*, vol. 63, Julio 2016.

[8] F. White, *Mecánica de los fluidos.* McGraw-Hill, Madrid: España, 1983.

[9] R. Guerequeta and A. Vallecillo, *Técnicas de Diseño de Algoritmos*. Servicio de Publicaciones de la Universidad de Málaga-España, 1998.

[10] R. Pereira, *A methodology for automated design and implementation of complex analog and digital CMOS integrated circuits applying a genetic algorithm and a CAD tool for multiobjective optimization*. PhD thesis, Instituto Tecnológico de Costa Rica, Escuela de Ingeniería Electrónica, 2014.

[11] A. Chacón-Rodriguez, F. Martín-Pirchio, P. Julián, and P. Mandolesi, "A verilog HDL digital architecture for delay calculation," *Latin American applied research*, vol. 37, no. 1, pp. 41–45, 2007.

[12] C. Salazar-García, L. Alfaro-Hidalgo, M. Carvajal-Delgado, J. Montero-Aragón, R. Castro-Gonzalez, J. A. Rodríguez, A. Chacon-Rodríguez, and P. Alvarado-Moya, "Digital integrated circuit implementation of an identification stage for the detection of illegal hunting and logging," in *2015 IEEE 6th Latin American Symposium on Circuits Systems (LASCAS)*, pp. 1–4, Feb 2015.

[13] K. Alfaro-Badilla, A. Chacón-Rodríguez, G. Smaragdos, C. Strydis, A. Arroyo-Romero, J. Espinoza-González, and C. Salazar-García, "Prototyping a biologically plausible neuron model on a heterogeneous CPU-FPGA board," in *2019 IEEE 10th Latin American Symposium on Circuits Systems (LASCAS)*, pp. 5–8, Feb 2019.

[14] F. M. Pirchio, P. Julián, P. S. Mandolesi, and A. Chacon-Rodriguez, "An adaptive cross-correlation derivative algorithm for ultra-low power time delay measurement," in *2007 IEEE International Symposium on Circuits and Systems*, pp. 4016–4019, IEEE, 2007.

[15] F. Martin-Pirchio, A. Chacon-Rodriguez, P. Julian, and P. Mandolesi, "A comparison of low power architectures for digital delay measurement," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, pp. 498–499, March 2007.

[16] A. Chacon-Rodriguez, F. Martin-Pirchio, S. Sanudo, and P. Julian, "A low-power integrated circuit for interaural time delay estimation without delay lines," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, pp. 575–579, July 2009.

[17] A. Chacon-Rodriguez, P. Julian, and F. Masson, "Fast and low power integrated circuit for impulsive sound localisation using Kalman filter approach," *Electronics Letters*, vol. 46, pp. 533–534, April 2010.

[18] R. Cerdas-Robles, A. Rodríguez, A. Chacon-Rodríguez, and P. Julián, "Design of an IDM-based determinant computing unit for a 130nm low power CMOS ASIC acoustic localization processor," in *2015 IEEE 6th Latin American Symposium on Circuits Systems (LASCAS)*, pp. 1–4, Feb 2015.

[19] A. Cervantes, F. Lopez, J. Quiros, D. Rodriguez, C. Salazar-Garcia, C. Meza, and A. Chacon-Rodriguez, "Implementation of an open core IEEE 754-based FPU with non-linear arithmetic support," in *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*, pp. 1–6, Nov 2016.

[20] C. Salazar-García, R. Castro-González, and A. Chacón-Rodríguez, "RISC-V based sound classifier intended for acoustic surveillance in protected natural environments," in *2017 IEEE 8th Latin American Symposium on Circuits Systems (LASCAS)*, pp. 1–4, Feb 2017.

[21] A. Chacón-Rodríguez, *Circuitos Integrados de Bajo Consumo de Potencia para Detección y Localización de Disparos de Armas de Fuego*. PhD thesis, Universidad Nacional de Mar del Plata Argentina, Argentina, 2009.

[22] P. Julián, , F. Masson, and A. Chacon-Rodriguez, *3D Gigascale Integrated Circuits for Nonlinear Computation, Filter and Fusion with Applications in Industrial Field Robotics*. Project Technical Report.

[23] B. Baritompa and E. Hendrix, "*On the investigation of stochastic global optimization algorithms*," *IEEE Trans. Signal Processing*, vol. 31, 2005.

[24] Z. Gajic, *Linear dynamic systems and signals*, ch. III. Linear Systems In Electrical Engineering. Prentice Hall, Pearson Education, 1ª ed., 2003.

[25] O. M., *Sistemas dinámicos en tiempo continuo: Modelado y simulación.*, ch. 2, 3. OmniaScience (Omnia Publisher SL), universidad politécnica de victoria, méxico. ed., 2015.

[26] A. Scott, *Nonlinear Science: Emergence and Dynamics of Coherent Structures.* Oxford University Press. UK., 2a edition ed., 2003.

[27] J. Hérault and J. Jutter, "*Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture*," *IEEE Trans. Signal Processing*, vol. 24, pp. 1–10, 1991.

[28] T. H. Chang, "*Numerical Analysis*," 2008.

[29] C. Meza and R. Ortega, "*On-line estimation of the temperature dependent parameters of photovoltaic generators*," *IEEE Trans. Signal Processing*, vol. 11th IFAC International, Workshop on Adaptation and Learning in Control and Signal Processing, pp. 567–578, 2013.

[30] G. Burel, "*Blind separation of sources: A non-linear neural algorithm*," *IEEE Trans. Neural Networks*, vol. 5, pp. 937–947, 1992.

[31] J. Cardoso, "*Blind signal separation: statistical principles*," *IEEE Trans. Signal Processing*, vol. 86, pp. 2009–2025, 1998.

[32] S. Houcke and A. Chevreuil, "*Blind source separation of a mixture of communication sources emitting at various baud-rates*," *IEEE International conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 3053–3056, 2002.

[33] J. Cardoso, "*Estimating equations for source separation*," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3449–3452, 1997.

[34] E. Vincent and R. Gribonval, "*Performance Measurement in Blind Audio Source Separation*," *IEEE Trans. Speech and Audio Proc.*, 2005.

[35] J. Cichocki and S. Amari, *Adaptive Blind Signal And Image Processing.* John Wliey and Sons, Inc., New York, 2002.

[36] P. Comon and C. Jutten, *Handbook of Blind Source Separation*, ch. 1: Introduction, pp. 1–22. Elsevier Ltd, Ed 1., 2010.

[37] D. Pham, *Handbook of Blind Source Separation*, ch. 2: Information, pp. 23–63. Elsevier Ltd, 1 ed., 2010.

[38] R. Litovsky and S. Colburn, "*The precedence effect and its possible role in the avoidance of interaural ambiguities J.A.S.A.*," *IEEE International conference on Acoustics, Speech, and Signal Processing*, pp. 1633–1654, 1999.

[39] A. V. Schaik and L. Shih-Chii, "*AER EAR: A Matched Silicon Cochlea Pair with Address Event Representation Interface*," *Trans. Circuits Syst. I*, pp. 4213–4216, 2007.

[40] C. Cherry, "*On human communication a review, a survey, and a criticism*," *M. I. T. Press in Cambridge, Mass.*, 1966.

[41] V. Zarzoso and A. Hyvärinen, *Handbook of Blind Source Separation*, ch. 6: Iterative algorithms, pp. 179–225. Elsevier Ltd, 1 ed., 2010.

[42] M. Welling, "*Robust Higher Order Statistics*," 2005. School of Information and Computer Science University of California Irvine.

[43] A. Belouchrani and A. Cichocki, "*A Robust Whitening Procedure in Blind Source Separation Context*," *Electronics Letters*, vol. 36, pp. 2050–2051, 2000.

[44] A. Hyvärinen and E. Oja, "*Independent Component Analysis: Algorithms and Applications*," *Journal Neural Networks*, vol. 13, pp. 411–430, May-June 2000.

[45] A. Taha, "*FPGA Implementation of Blind Source Separation using FastICA*," Master's thesis, University of Windsor, 2010.

[46] E. Moreau and P. Comon, *Handbook of Blind Source Separation*, ch. 3: Contrasts, pp. 65–105. Elsevier Ltd, 1 ed., 2010.

[47] D. Schobben and K. Torkkola, "*Evaluation Of Blind Signal Separation Methods*," *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, 2000.

[48] A. Chen, "*Emerging research devices and architectures for more-than- Moore applications*," *Electrochemical Society (ECS) Transactions*, vol. 50, pp. 3–10, Oct. 2012.

[49] A. Chen, "*Emerging research device roadmap and perspectives*," *IEEE International Conference on IC Design & Technology*, pp. 28–30, Mayo 2014.

[50] G. Sutter, *Reducción de Consumo en FPGAs.* Universidad Autónoma de Madrid, Departamento de Ingeniería Informática, 2005.

[51] E. Kitagawa and et al, "*Impact of ultra low power and fast write operation of advanced perpendicular MTJ on power reduction for highperformance mobile CPU*," *IEDM Tech*, vol. 46, pp. 677–680, Dic. 2012.

[52] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs.* Oxford University Press, New York, 1999.

[53] S. M. Celik, A. and G. Cauwenberghs, "Mixed-signal real-time adaptative blind source separation," *Proc. IEEE Int. Symp. Circuit and Systems, (ISCAS)*, 2004.

[54] M. Stanacevic and G. Cauwenberghs, "*Gradient Flow Adaptive Beamforming and Signal Separation in a Miniature Microphone Array*," in *Proc. IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP'2002), (Orlando FL*, pp. 13–17, 2002.

[55] K. Zhang and L. Chan, "*Dimension Reduction Based on Orthogonality -a Decorrelation Method in ICA*," *Artificial Neural Networks and Neural Information Processing*, pp. 132–139, 2003.

[56] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Journal Neural Networks*, vol. 13, pp. 411–430, May-June 2000.

[57] K. Shyu and M. Li, eds., *FPGA Implementation of FastICA based on Floating-Point Arithmetic Design for Real-Time Blind Source Separation*, July 2006.

[58] S. Amari, A. Cichocki, and H. H. Yang, "*A New Learning Algorithm for Blind Signal Separation*," in *Advances in Neural Information Processing Systems*, pp. 757–763, MIT Press, 1996.

[59] M. Bastian, *Neural Networks and the natural gradient*, pp. 33–54. UTAH STATE UNIVERSITY, 2009.

[60] V. Zarzoso and A. Hyvärinen, *Handbook of Blind Source Separation*, ch. 6: Iterative algorithms, pp. 179–225. Elsevier Ltd, 1 ed., 2010.

[61] R. Salomon, "Evolutionary algorithms and gradient search: Similarities and differences," *IEEE Transactions on Evolutionary Computation*, vol. 2, Julio 1998.

[62] C. Yang, Y. Shih, and H. Chiueh, "An 81.6 μw fastica processor for epileptic seizure detection," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, pp. 60–71, 2015.

[63] L. Van, D. Wu, and C. Chen, "Energy-efficient fastica implementation for biomedical signal separation," *IEEE Trans. Neural Network*, vol. 22, pp. 1809–1822, 2011.

[64] C. Bello and all, "*Relevador portátil de curvas IV de paneles fotovoltaicos como herramienta de diagnóstico in situ de sistemas de generación fotovoltaica.*," *Avances en Energías Renovables y Medio Ambiente.*, vol. 13, pp. 77–83, 2009.

[65] A. Fahrenbruch and R. Bube, *Fundamentals of solar cells: photovoltaic solar energy conversion*, ch. 2, 3. Academic Press, 1a ed., 2012.

[66] C. Chiang, C. Tung-Sheng, and H. Hou-Sheng, "*Modeling a photovoltaic power system by CMAC-GBF.* ," *IEEE, Photovoltaic Energy Conversion*, vol. Proceedings of 3rd World Conference on, 3, 2003.