Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

TEC | Tecnológico de Costa Rica

Implementación de algoritmo para calcular la correlación *bitstream* entre dos señales para detectar sonidos de disparos y motosierras en el bosque.

Informe de Proyecto de Graduación para optar por el título de

Ingeniero en Electrónica con el grado académico de Licenciatura

Josué Andrés Mora Castro

Cartago, 26 de agosto del 2016

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado Implementación de algoritmo para calcular la correlación bitstream entre dos señales para detectar sonidos de disparos y motosierras en el bosque, realizado por el señorJosué Andrés Mora Castro y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Ing. Luis Carlos RosalesAlpízar
Profesor lector

Ing. Leonardo Rivas Arce
Profesor lector

Ing. Alfonso Chacón Rodríguez
Profesor asesor

Cartago, 26 de agosto, 2016

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Josué Andrés Mora Castro

Cartago, 26 de agosto del 2016

Céd: 5-0390-0930

Resumen

La correlación cruzada de dos señales es una medida de que tan similares son éstas. Sin embargo los métodos tradicionales de implementación de este algoritmo requieren de muchos recursos lo que implica la búsqueda constante de alternativas de cálculo que simplifiquen ese proceso. Este trabajo pretende proponer una implementación prototípica funcional de la correlación utilizando la representación *bitstream* de los datos, simplificando así el hardware requerido para las operaciones, haciendo viable su utilización para el procesamiento de señales.

Se espera que este proyecto ayude con la protección ambiental contra la caza y tala ilegal en los bosques tropicales de Costa Rica al ser un medio para la detección de señales como sónido de disparos y motosierras.

Palabras clave: Correlación, *Bitstream*, Detección de Patrones, Modulación sigma-delta

Abstract

Cross correlation is a way of measuring how similar two signals are. However, traditional methods of implementing this algorithm require a lot of resources which promotes a constant search of alternative ways of performing it. This project intended to develop a prototipical implementation of a *bitstream* based correlation between to signals, simplifying the necessary hardware for the operations and making it feasible.

The results of this project are meant to be use to help in the battle against illegal hunting and logging of Costa Rican tropical forest by providing another way to detect shooting and chainsaws sounds.

Keywords: Correlation, Bitstream, Pattern Recognition

a mis queridos padres

Agradecimientos

A mis familiares por nunca dudar de que iba a lograr esta meta y por ser un ejemplo a seguir.

A mis amigos y personas cercanas que han sido de las principales fuentes de motivación durante todo el proceso, especialmente en los momentos en los que más se ha necesitado.

A mis compañeros con los que desarrollé tantos proyectos y compartí tantos ratos de estudio por ser una parte muy importante de este proceso y de los cuales he aprendido demasiado.

A mis profesores, principalmente a Alfonso Chacón, Anibal Coto, Eduardo Interiano, Juan José Pineda, Luis Carlos Rosales, Renato Rímolo, Roberto Pereira, Saúl Guadamuz y William Marín que además de brindar una formación académica de calidad han demostrado su verdadera preocupación y compromiso por formar profesionales de calidad.

Josué Andrés Mora Castro

Cartago, 6 de septiembre de 2016

Índice general

Índice de figuras

Índice de tablas

CC: Cuenta condicional.

CD: Corriente Directa.

CI: Cuenta incondicional.

DAC: Del inglés Digital to Analog Converter.

FSM: Del inglés Finite-State Machine.

OSF: Del inglés Oversampling Factor.

PCM: Del inglés Pulse Code Modulation.

RTL: Del inglés Register-Transfer Level.

SC: Salto condicional.

SNR: Del inglés Signal-to-Noise Ratio.

VI: Del inglés Virtual Instrument.

Capítulo 1

Introducción

De acuerdo con [13] Costa Rica posee 51 100 km^2 de superficie terrestre, lo que corresponde a un 0,03% de la superficie mundial. Por ello tan impresionante que se encuentre entre los 20 países con mayor biodiversidad del mundo. Las más de 500 mil especies que se encuentran en el país representa cerca del 4% del total de especies estimadas a nivel mundial.

Como se dice en [21] no es de extrañar entonces que Costa Rica sea ejemplo a nivel mundial por su preocupación por conservar el ambiente, convirtiendo el 29% del territorio nacional en áreas protegidas.

Aún así, de acuerdo con [18] solo hay 500 guardaparques encargados de la tala de los bosques protegidos, la caza de animales en peligro de extinción y la invasión de las áreas protegidas, por lo que cada guardaparque es responsable de aproximadamente 2654 hectáreas, el área equivalente a 37 veces el Parque La Sabana.

Por esta razón el DCILab (Laboratorio de Diseño de Circuitos Integrados) de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica ha estado desarrollando un proyecto que consiste en crear una red de sensores para detectar estas actividades ilegales. En particular se ha propuesto el desarrollo de un sistema de reconocimiento de patrones basado en Modelos Ocultos de Markov, llamado SiRPA, que se encuentra en sus etapas finales de prueba. Este sistema clasifica muestras de señales capturadas en el bosque para discriminar entre sonidos naturales del bosque, sonidos de motosierras o de disparos.

Para ello usa distintas técnicas de filtrado y generación de símbolos (ver [4, 5, 23]) para luego aplicar estos últimos al clasificador final (ver [1, 4] para detalles del sistema completo). No obstante, un clasificador es una unidad de procesamiento compleja, por lo que es tradicional introducir un detector más sencillo y de menor consumo de potencia antes, que despierte por así decirlo al clasificador cuando haya un fenómeno que requiere su análisis más avanzado.

En [7, 8] se muestra la evaluación de cinco algoritmos de pre-procesamiento para la detec-

ción de disparos, entre ellos la correlación, el uso de onditas o wavelets (ver [10] para una implementación parcial), y el uso de un filtro de mediana. De todos, el del correlación es el que teóricamente ofrece mayor calidad de detección, pero su costo de hardware lo coloca debajo de otros algoritmos en términos de consumo energético.

Además, como se menciona en [6,9,11], ha sido demostrado que la diferencia de fase entre señales capturadas de manera independiente por dos o más micrófonos puede ser utilizada para estimar en ángulo entre la fuente y el receptor, ya que la diferencia de fase entre dos señales se puede determinar calculando la correlación entre ellas [22].

Capítulo 2

Meta y Objetivos

2.1 Meta

Desarrollar un sistema de monitoreo ambiental orientado a la protección natural contra caza y tala ilegal.

2.2 Objetivo General

- Diseñar en un lenguaje de descripción de una prueba de concepto de un correlador *bitstream* capaz de procesar señales con un ancho de banda de 20Hz a 20kHz.

2.3 Objetivos Específicos

- Desarrollar un modelo de correlación *bitstream* en un lenguaje de alto nivel, para que sirva de patrón dorado.
- Diseñar en un lenguaje de descripción de hardware las operaciones numéricas *bitstream* en coma fija necesarias para realizar la operación de correlación *bitstream*.
- Diseñar en un lenguaje de descripción de hardware una prueba de concepto del módulo correlador.

2.4 Metodología

Para la consecución de los objetivos se siguió con la metodología del diseño en ingeniería. Así, una vez planteados los problemas a resolver para cada objetivo, se realizó una investigación bibliográfica del estado del arte sobre la correlación en formato *bitstream* y

las técnicas de modulación sigma-delta. Posteriormente, se propuso una solución en alto nivel de dicho algoritmo, que serviría posteriormente como modelo de referencia para la verificación final (este modelo funcional fue primero contrastado contra el modelo teórico de la correlación). Además, se utilizó un modelo de alto nivel de modulación sigma-delta necesario para producir la modulación *bitstream* con la cual alimentar el algoritmo de correlación *bitstream*.

Con estos modelos ya verificados, se procedió a desarrollar la unidad propuesta a nivel de RTL. Esta unidad se verificó mediante simulación RTL contra los resultados de los modelos de más alto nivel. Aspectos como sincronización e interfaz con los módulos que alimentarán a la unidad con la señal bitstream, y aquellos que tomarán los datos de correlación procesados, debieron incorporarse en esta etapa.

Capítulo 3

Marco teórico

3.1 Correlación cruzada

La correlación cruzada es una medida de que tan similares son dos señales, por lo que se puede utilizar para reconocer cuando dos señales son iguales o para calcular un desfase entre dos señales como en [22]. Esta se puede expresar de muchas maneras, siendo una de las formas de expresarla

$$R_i = \sum_{k=1}^{n} x(k)y(k-i) \tag{3.1}$$

donde $x(k)$ es el patrón y $y(k)$ la muestra, cada cálculo de correlación R_i implica un desplazamiento $y(k-i)$ en la muestra, una multiplicación uno a uno entre cada valor de las señales $x(k)$ y $y(k-i)$ y una sumatoria entre cada una de las multiplicaciones realizadas. Es por esta razón que la correlación es un calculo que normalmente consume muchos recursos.

3.2 Implementación de un algoritmo de correlación para señales moduladas sigma-delta (*bitstream*)

En la Figura 3.1 (tomada de [15]) se muestra un esquemático típico de la implementación del algoritmo de correlación de una señal contra un patrón. Para ello se utiliza un multiplexor para recorrer todos los datos en el registro de muestra y patrón. Conforme se recorren los datos se va realizando la multiplicación uno a uno y el resultado de cada multiplicación va a un acumulador donde se suma con los resultados de las multiplicaciones previas. Nótese que este correlador debe funcionar a n veces la tasa de adquisición de Nyquist, y que los datos deben venir codificados en algún estándar PCM (típicamente coma fija, en complemento a dos).

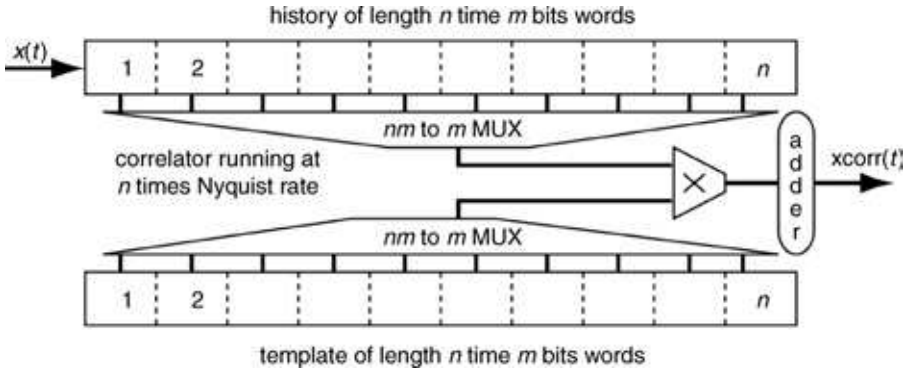


Figura 3.1: Funcionamento a nivel de bloques de la operación correlación. Tomado de [15].

Una forma alternativa de implementar este algoritmo es efectuar todas las multiplicaciones en paralelo, esto con el fin de reducir la cantidad de ciclos necesarios para realizar el cálculo de la correlación, y así disminuir la frecuencia de conmutación de las operaciones aritméticas. Sin embargo, hacer eso requiere de un aumento desmedido en el hardware ya que se deben utilizar n multiplicadores, que ya por sí mismos son circuitos complejos y de gran consumo de potencia. Esto limitaría entonces el tamaño del vector n también.

En [16, 17] (ver Figura 3.2) se realiza una implementación del algoritmo de correlación realizando todas las multiplicaciones en paralelo, con la diferencia de que los datos se representan en una señal modulada sigma-delta o *bitstream* (de ahora en adelante, tal como se explicará en la Sección 3.3, se considerará que siempre que se hable de una señal *bitstream*, ésta estará modulada sigma-delta). Esto implica un aumento en los registros de patrón y de muestra, a la vez que un aumento en la velocidad requerida para las operaciones, producto del sobremuestreo inherente a la modulación sigma-delta, mientras que el hardware necesario para el cálculo de la correlación se simplifica, ya que la multiplicación entre dos señales moduladas *bitstream*, como se explicará en la Sección 3.4 se puede realizar utilizando una compuerta lógica.

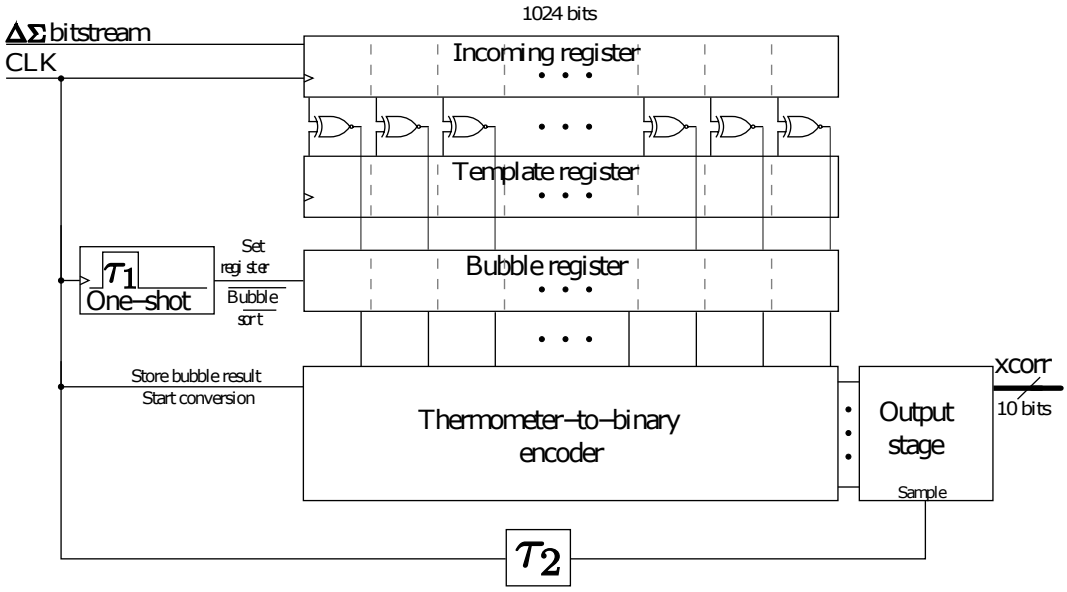
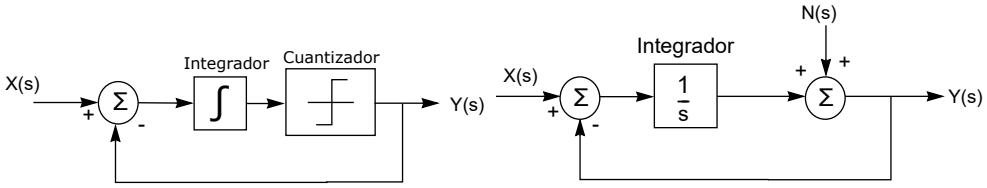


Figura 3.2: Funcionamento a nivel de bloques de la operación correlación utilizando la representación *bitstream*. Tomado de [17].

3.3 Modulador Sigma Delta

En [20] se explica el funcionamiento del Modulador Sigma Delta; en la Figura 3.3(a) se puede observar que este consta de un integrador y un cuantizador. Además, en la Figura 3.3(b) se observa un diagrama de bloques equivalente al de la Figura 3.3(a), pero expresado en el dominio de la frecuencia, y se sustituye el cuantizador por un nodo sumador al lado derecho del integrador para agregar el ruido de cuantización.



(a) Componentes del Modulador.

(b) Modulador en el dominio de la frecuencia.

Figura 3.3: Diagrama de bloques del Modulador Sigma Delta. Tomado de [20].

Suponiendo un ruido de cuantización $N(s)$ igual a cero se tiene

$$Y(s) = \frac{(X(s) - Y(s))}{s}$$

despejando $\frac{Y(s)}{X(s)}$

$$\frac{Y(s)}{X(s)} = \frac{\frac{1}{s}}{1 + \frac{1}{s}}$$

y al simplificar se obtiene la función de transferencia del Modulador Sigma Delta:

$$\frac{Y(s)}{X(s)} = \frac{1}{s+1} \tag{3.2}$$

Partiendo nuevamente del diagrama de frecuencia del Modulador Sigma Delta, pero esta vez asumiendo que la entrada $X(s)$ es cero se tiene

$$Y(s) = -\frac{Y(s)}{s} + N(s)$$

despejando $\frac{Y(s)}{N(s)}$

$$\frac{Y(s)}{N(s)} = \frac{1}{1+\frac{1}{s}}$$

al simplificar se obtiene la función de transferencia del ruido de cuantización:

$$\frac{Y(s)}{N(s)} = \frac{s}{s+1} \tag{3.3}$$

De la Ecuación (3.2) se observa que la función del Modulador se comporta como un filtro pasobajo, mientras que la Ecuación (3.3) muestra como la función de transferencia del ruido de cuantización se comporta como un filtro pasoalto. Esto quiere decir que la señal muestreada se va a concentrar en frecuencias bajas, mientras que el ruido de cuantización se va a trasladar a las frecuencias más altas, tal y como se representa en la Figura 3.4.

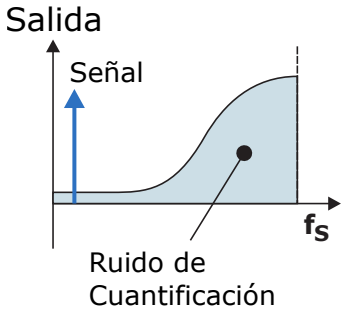


Figura 3.4: Análisis del Modulador Delta Sigma en la Frecuencia. Tomado de [3].

La cantidad de bits con los que se represente cada dato es igual al OSF y el valor que representa el dato en *bitstream* depende de la densidad de bits en uno que se encuentran en el *bitstream*; por ejemplo, en la Figura 3.5 se muestra la respuesta del Modulador Sigma Delta a una rampa, se observa que para valores mayor a cero la mayor cantidad de bits del *bitstream* son uno.

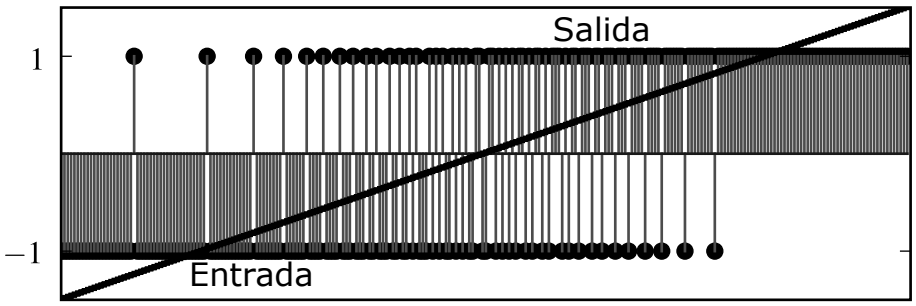


Figura 3.5: Respuesta del Modulador Delta Sigma a una Rampa. Tomado de [17].

3.4 Operaciones *Bitstream*

En [16] se menciona que la probabilidad de que una señal x de un modulador sea uno está dada por

$$P(X = 1) = \frac{1 + x}{2} \tag{3.4}$$

mientras que la probabilidad de que la señal x sea cero es

$$P(X = 0) = \frac{1 - x}{2} \tag{3.5}$$

Además, de acuerdo con [19] la probabilidad de que la *xnor* de dos señales x y y sea uno es

$$P(x\overline{\oplus}y = 1) = P(x = 1)P(y = 1) + P(x = 0)P(y = 0) \tag{3.6}$$

Utilizando la Ecuación (3.4) y (3.5) en la Ecuación (3.6) para calcular la probabilidad de que la *xnor* entre dos señales x y y moduladas sea uno, donde se representa como X y Y las señales moduladas, se obtiene que

$$P(X\overline{\oplus}Y = 1) = P(X = 1)P(Y = 1) + P(X = 0)P(Y = 0)$$

$$P(X\overline{\oplus}Y = 1) = (\frac{1 + x}{2})(\frac{1 + y}{2}) + (\frac{1 - x}{2})(\frac{1 - y}{2})$$

$$P(X\overline{\oplus}Y = 1) = \frac{(1 + x + y + xy) + (1 - x - y + xy)}{4}$$

$$P(X\overline{\oplus}Y = 1) = \frac{2 + 2xy}{4}$$

$$P(X\overline{\oplus}Y = 1) = \frac{1+xy}{2} \tag{3.7}$$

Por lo que la Ecuación (3.7) permite considerar el utilizar la compuerta *xnor* en dos señales moduladas sigma-delta para calcular de manera aproximada la multiplicación de dos señales.

3.5 Relación señal a ruido (SNR)

Como se menciona en [2] la relación señal a ruido está dada por

$$SNR = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \tag{3.8}$$

Suponiendo un ADC de N bits con 2^N niveles de cuantización se tiene que

$$\sigma_e^2 = \frac{\Delta}{12} = \frac{\left(\frac{2V}{2^N-1}\right)^2}{12} \approx \frac{\left(\frac{2V}{2^N}\right)^2}{12} \tag{3.9}$$

Sustituyendo 3.9 en 3.8 y simplificando se obtiene que la relación señal a ruido para un ADC de N bits es

$$SNR = 10\log\left(\frac{\sigma_x^2}{V^2}\right) + 4.77 + 6.02N \tag{3.10}$$

Como se observa en la ecuación anterior, por cada bit adicional en el ADC se obtiene una mejora de 6 dB en el SNR.

Además, de [2] se tiene que para un conversor con sobremuestreo se tiene que el SNR es

$$SNR = 10\log\left(\frac{\sigma_x^2}{\sigma_{ey}^2}\right) \tag{3.11}$$

y para un conversor sigma delta de primer orden

$$\sigma_{ey}^2 = \sigma_e^2\frac{\pi^2}{3}\left(\frac{2f_B}{fs}\right)^3 \tag{3.12}$$

Al sustituir 3.12 en 3.11 se obtiene

$$SNR = 10\log\left(\sigma_x^2\right) - 10\log\left(\sigma_e^2\right) - 10\log\left(\frac{\pi^2}{3}\right) + 30\log\left(\frac{f_s}{2f_B}\right) \tag{3.13}$$

Si se define el factor de sobremuestreo como

$$2^r = \frac{f_s}{2f_B} \tag{3.14}$$

Al sustituirlo en la ecuación 3.13 se obtiene que para un conversor sigma delta de primer orden que

$$SNR = 10\log\left(\sigma_x^2\right) - 10\log\left(\sigma_e^2\right) - 10\log\left(\frac{\pi^2}{3}\right) + 9.03r \tag{3.15}$$

De lo cual se puede observar que al duplicar el OSF se mejora el SNR 9 dB, que comparado con un ADC tradicional PCM como se vio en la Ecuación (3.10), sería equivalente a agregar 1.5 bits.

3.6 LabVIEW

Como se menciona en [14] LabVIEW, que recibe su nombre de *Laboratory Virtual Instrumentation Engineering Workbench*, es una plataforma de desarrollo creada por National Instrument que utiliza lenguaje de programación gráfico conocido como *G*. Los archivos creados utilizando LabVIEW se llaman VI, que es una abreviación para *Virtual Instrument*.

Al emplear un lenguaje de programación gráfico, las variables se representan por medio de cables, el color del cable indica el tipo de la variable y su grosor o forma la dimensión de la misma como se aprecia en la Figura 3.6.

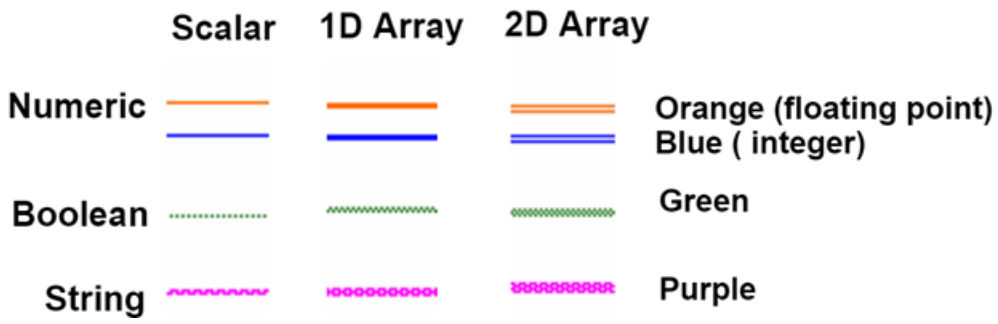


Figura 3.6: Tipo de datos en LabVIEW.

Una de las estructuras más utilizadas en LabVIEW es el *for loop*, como se muestra en la Figura 3.7 este cuenta con un *Count Terminal* en el cual se indica cuantas veces se ejecuta el código contenido en esta estructura, y también cuenta con un indicador de iteración con el valor de la iteración actual.

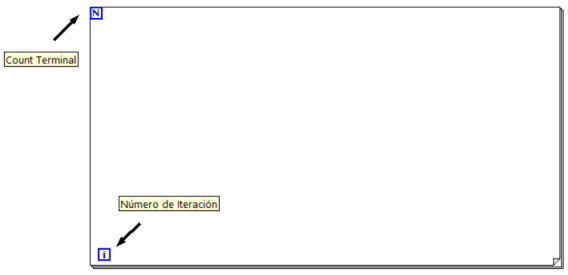
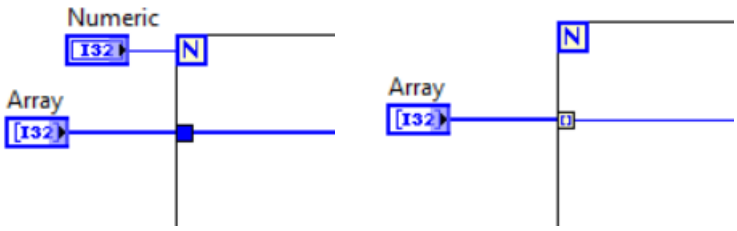


Figura 3.7: *For Loop* en LabVIEW.

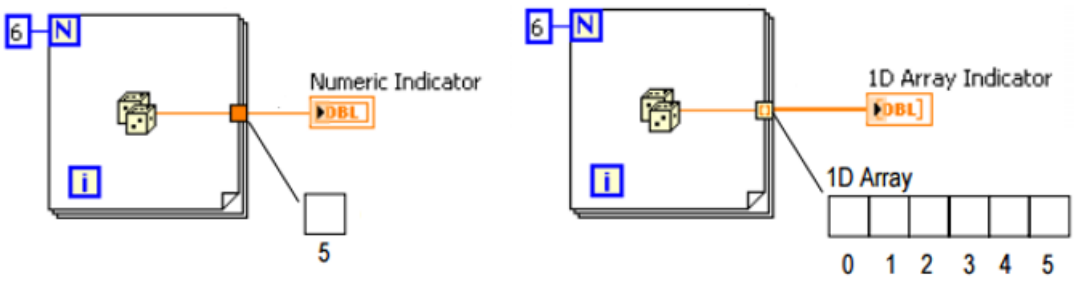
Cuando se conecta un *array* a la entrada de un *for loop* se puede escoger habilitar el auto-indexing, esto se realiza cuando se desea que los datos ingresen al *for loop* de uno en uno y no como un *array*. La Figura 3.8 muestra la conexión de un *array* a un *for loop*, notese que el cuadro donde se realiza la conexión es el indicador de si esta opción está habilitada. Si está habilitada no es necesario realizar ninguna conexión al *Count Terminal* ya que la cantidad de veces que se ejecuta el *for loop* está definida por el tamaño del array.



(a) Auto-indexing deshabilitado (b) Auto-indexing habilitado

Figura 3.8: Auto-indexing a la entrada de un *for loop*.

De igual manera cuando se tiene un *array* dentro del *for loop* se puede elegir si se habilita el auto-indexing, por ejemplo en la Figura 3.9(a) se tiene un *for loop* que se ejecuta seis veces en el cual se genera un número aleatorio, ya que se auto-indexing se encuentra deshabilitado a la salida del *for loop* se va a generar solo un elemento con el valor de la última iteración realizada. En la Figura 3.9(b) se tiene el mismo código, pero se habilita el auto-indexing lo que hace que se genere una *array* a la salida del *for loop* con los seis números aleatorios producto de cada iteración realizada.



(a) Auto-indexing deshabilitado (b) Auto-indexing habilitado

Figura 3.9: Auto-indexing a la salida de un *for loop*.

Además, en los *for loops* se pueden utilizar *shift register*, esto se hace cuando se desea utilizar un valor calculado en una iteración anterior. Por ejemplo el código de la Figura 3.10 se ejecuta cinco veces como se define en el *Count Terminal*, se observa que con cada iteración se incrementa en uno el valor calculado en la iteración anterior, el valor inicial de este cálculo es con el que se inicializa el shift register, en este caso cero. Por lo que el resultado final es cinco.

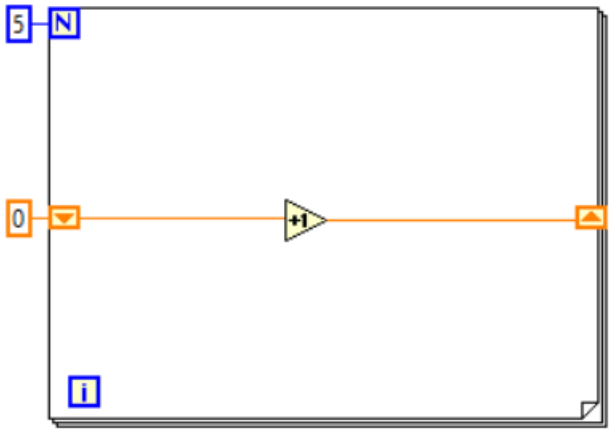


Figura 3.10: Registro de desplazamiento en LabVIEW.

Capítulo 4

Correlación *Bitstream*

4.1 Implementación en alto nivel

El VI de LabVIEW que implementa el conversor de los datos a *bitstream* de la Figura 4.1 es uno de los módulos principales, ya que sin este no se podría probar el funcionamiento de la función de correlación. El *for loop* exterior se encarga de recorrer todos los elementos de la señal que se desea convertir, los cuales se representan con un *array*; éste se ejecuta n veces, donde n es la cantidad de datos que conforman la señal. El *for loop* interior se ejecuta OSF veces ya que como se explicó en la Sección 3.3 cada dato al convertirse en *bitstream* se representa con OSF bits. En el interior de este *for loop* se encuentra la lógica para el Modulador Sigma Delta de primero orden, este se puede comparar con el modulador de la Figura 3.3; se observa que cuenta con el integrador encargado de trasladar el ruido debido a la cuantización a frecuencias más altas [3]. Como cuantizador se utiliza un comparador de un bit, se agrega un DAC en el lazo de realimentación y por último se puede apreciar que también cuenta con el nodo sumador encargado de restar el valor anterior, en este caso la salida del DAC, al dato que se desea muestrear.

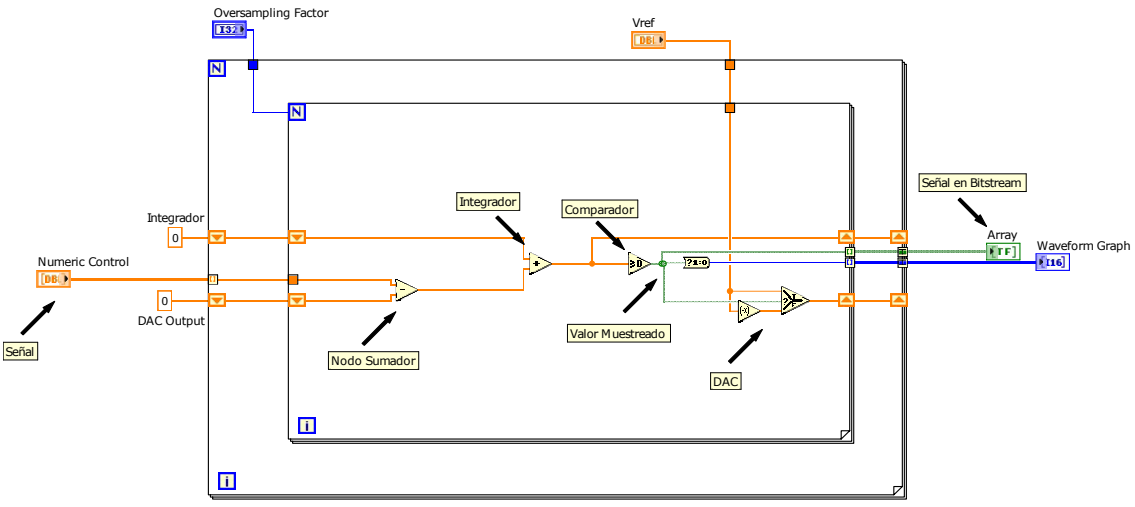


Figura 4.1: VI de conversor *bitstream*.

En la Figura 4.2 se encuentra el VI desarrollado en este proyecto para el algoritmo de correlación *bitstream*. Este contiene un *for loop* que se encarga de realizar los desplazamientos en la señal de muestra. Dentro de este ciclo el patrón entra al conversor *bitstream* sin ser modificado, mientras que la muestra primero pasa por un módulo que selecciona cuales son los datos que deben entrar al conversor. La salida de estos conversores es un array de tamaño OSF, y van a un módulo de *xnor* donde cada bit es multiplicado uno a uno. Después, estos datos van a un sumador para obtener el resultado final de correlación.

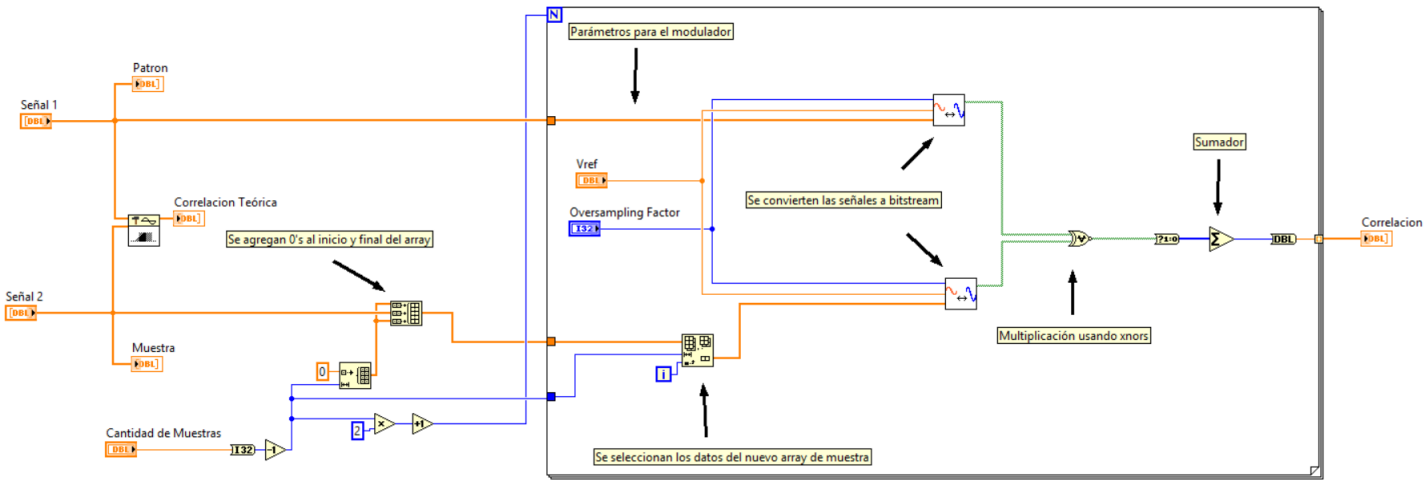


Figura 4.2: VI desarrollado para el algoritmo de correlación *bitstream*.

Debido a que se está utilizando la compuerta lógica *xnor* para aproximar la multiplicación de dos señales, se produce un escalamiento a la correlación *bitstream* además de agregarle un valor medio, como muestra la Ecuación (3.7), lo que hace que la correlación teórica y la correlación *bitstream* no sean comparables. El VI de la Figura 4.3 corrige este fenómeno, al remover de la correlación teórica su promedio, es decir:

$$T = t - \bar{t} \tag{4.1}$$

donde T es el nuevo valor de correlación teórico. A la correlación *bitstream* también se le aplica la corrección

$$\xi = e - \bar{e} \tag{4.2}$$

donde ξ es el nuevo valor de la correlación *bitstream*; sin embargo, este valor aún debe corregirse por ser de diferente amplitud a la correlación teórica, por lo que el valor definitivo es

$$E = \frac{T_{MAX}}{\xi_{MAX}}\xi \tag{4.3}$$

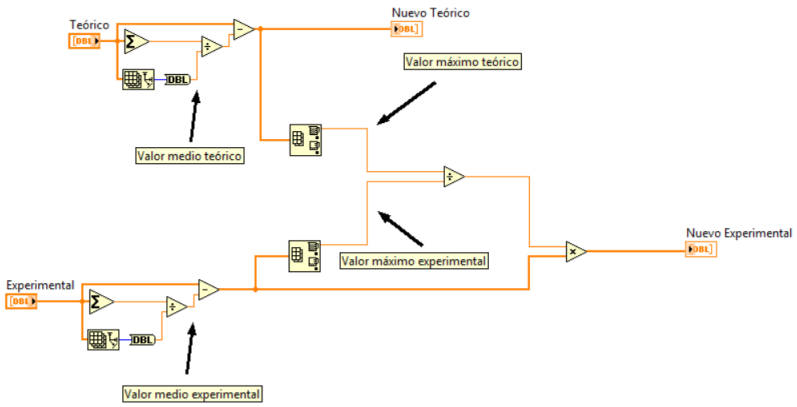


Figura 4.3: Acondicionador de señal necesario para corregir la escala de la salida de correlación *bitstream*.

4.2 Desarrollo de la solución RTL del algoritmo de correlación *bitstream*

En la Figura 4.4 muestra un diagrama de bloques RTL de la solución propuesta para calcular la correlación sobre una señal en formato *bitstream*. La Tabla 4.1 contiene una descripción de las entradas y salidas de este módulo. La parte superior de la imagen describen el control y sincronización necesarias del sistema, que se explicarán más adelante.

Tabla 4.1: Descripción de entradas y salidas de la solución RTL.

Nombre de la señal	Tipo de Señal	Descripción
Bitstream	Input	Por esta señal se ingresa el *bitstream* con el que se representan las señales a correlacionar
Ready	Input	Indica que se debe leer el dato de la señal *bitstream*
Correlacion	Output	Contiene el valor final de correlación calculado
Flag	Output	Indica que el valor de correlación está listo

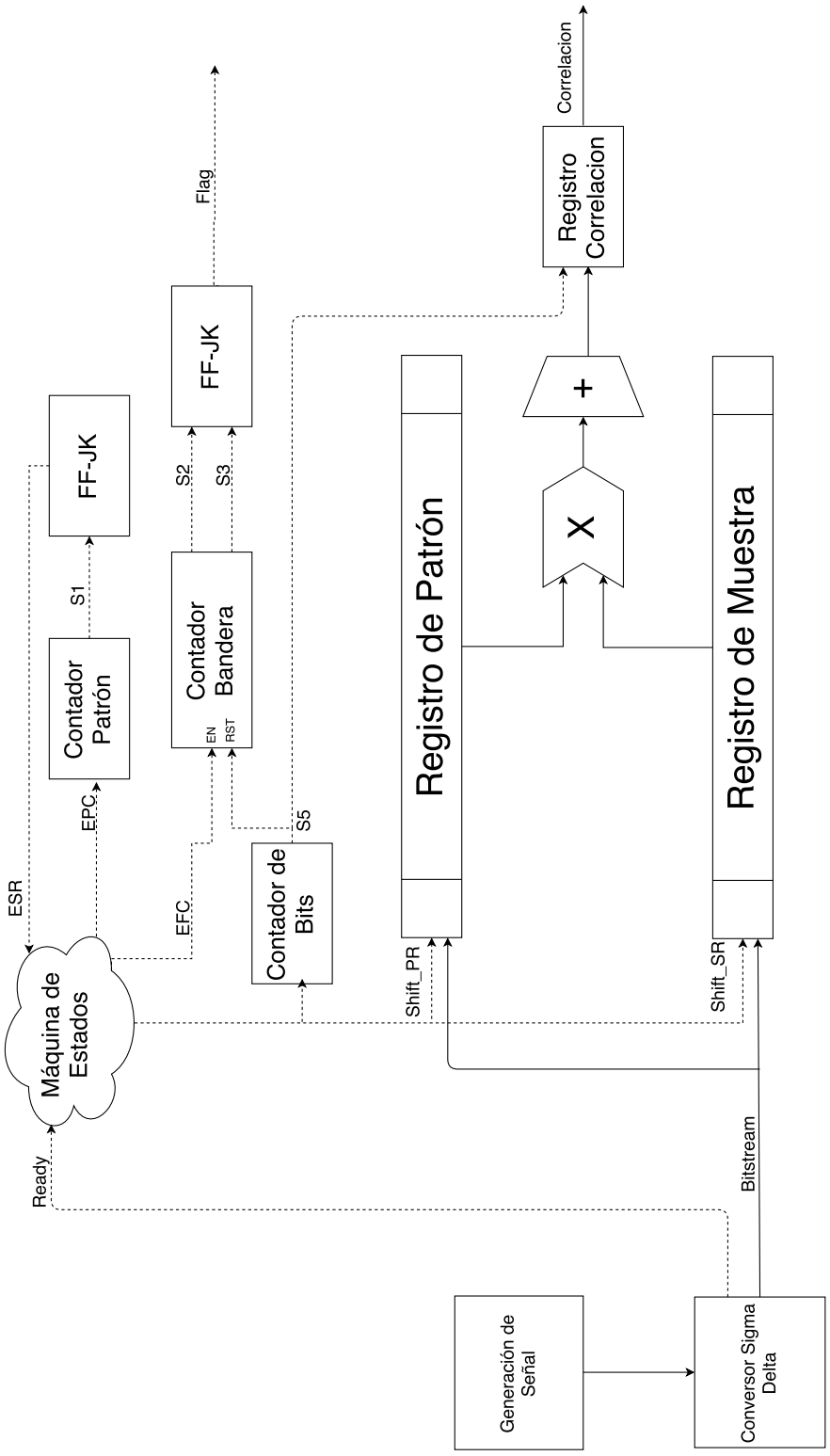


Figura 4.4: Diagrama de bloques RTL de la solución propuesta.

En el *Registro de Patrón* se almacena la plantilla contra la que se correlaciona y el *Registro de Muestra* almacena la señal de entrada que ingresa a una tasa igual a la de la frecuencia de sobre muestro del modulador. Sobre los bits de estos registros de va calculando la correlación aplicando las operaciones *bitstream* multiplicación y suma.

El *Registro de Patrón* y *Registro de Muestra* son registros de desplazamiento de tamaño $nxOSF$, donde n es la cantidad de muestras y OSF el factor de sobremuestreo del modulador delta-sigma. Para este proyecto se utiliza como punto de partida un n=128 y un OSF=8 que es el mismo utilizado en la implementación realizada en [16]. La *Máquina de Estados* es la que habilita el desplazamiento de estos registros por medio de las señales $Shift\_PR$ y $Shift\_SR$.

Para el multiplicador *bitstream* se utilizan compuertas *xnor* tal y como se sugiere en [16] y [15] donde las operaciones se realizan bit a bit.

Finalmente para el cálculo de la correlación *bitstream* se deben sumar todos los bits 1 del resultado de la multiplicación, lo cual se realiza en el bloque de sumatoria que se divide en dos partes. La primera parte está compuesta por unos decodificadores como los de la Figura 4.5 puestos en paralelo, que toman un bus de datos de 8 bits y a la salida se obtiene la cantidad de esos bits que son 1's. Cada uno de estos decodificadores cumple una función equivalente al *Bubble register* y al $Thermometer - to - binaryencoder$ de la Figura 3.2, al actuar tanto como sumador como filtro pasa bajo de la señal.

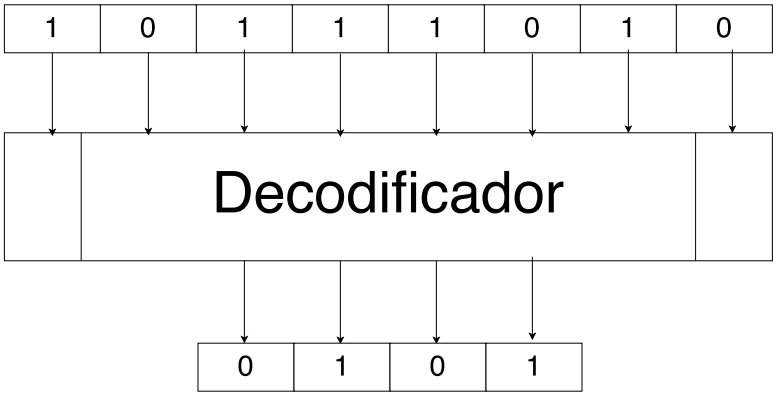


Figura 4.5: Diagrama de bloques RTL del decodificador necesario para realizar la suma.

La segunda sección del circuito es un arreglo de sumadores que tienen como entrada todas las salidas de los decodificadores. El resultado de esta etapa es la suma de todos los datos decodificados, lo que corresponde al valor parcial de correlación, ahora en formato PCM. Todo este cálculo es puramente combinacional, por lo que puede realizarse en un solo ciclo de reloj. El valor parcial de correlación se carga al *Registro de Correlación* con el resultado final una vez que se ingrese una cantidad OSF de bits correspondiente a un dato nuevo de la señal muestreada.

Tabla 4.2: Descripción de señales auxiliares para el control y sincronización del sistema.

Nombre de la señal	Descripción
S1	Realiza un Set al FF encargado de activar la señal ESR
S2	Realiza un Set al FF encargado de activar la señal Flag
S3	Realiza un Reset al FF encargado de activar la señal Flag
S5	Realiza un Reset al Contador de Bandera, además carga el valor de correlación parcial al Registro Correlacion

Para poder interfazar esta unidad de correlación, se crearon los bloques de *Registro de Correlación*, *Contador de Bits* y *Contador de Bandera*, junto con sus respectivos flip flops que se encargan de mantener la señal que activan los contadores. En la Tabla 4.2 se encuentra la descripción de las principales señales de estos bloques.

El *Contador de Bandera*, como se puede intuir por su nombre, se encarga de activar una bandera que indica cuando el valor de correlación es válido, así el módulo posterior al algoritmo de correlación *bitstream* sabe cuándo está disponible este dato a su salida.

También se crea el *Contador de Patrón* para determinar cuando se ha terminado de escribir en el *Registro de Patrón*, de manera que los primeros datos solo se escriben en el *Registro de Patrón*.

4.2.1 Diseño de la máquina de estados de control (FSM)

Tabla 4.3: Descripción de las señales de entrada y salida de la máquina de estados de control.

Nombre de la señal	Tipo de Señal	Descripción
Read	Input	Indica que se debe leer el dato de la señal bitstream
ESR	Input	Indica el fin de la escritura en el Registro de Patrón
EPC	Output	Habilita el Contador de Patrón
EFC	Output	Habilita el Contador de Bandera
Shift_PR	Output	Habilita el ingreso de datos en el Registro de Patrón
Shift_SR	Output	Habilita el ingreso de datos en el Registro de Muestra, también realiza un Reset al Contador de Bits

La Tabla 4.3 contiene una descripción de las señales que componen la máquina de estados propuesta para el control de la unidad de correlación. Además en la Figura 4.6 puede verse el diagrama de estados que explica como responde la máquina de estados ante esas señales. La Tabla 4.4 se resume el diagrama de estados.

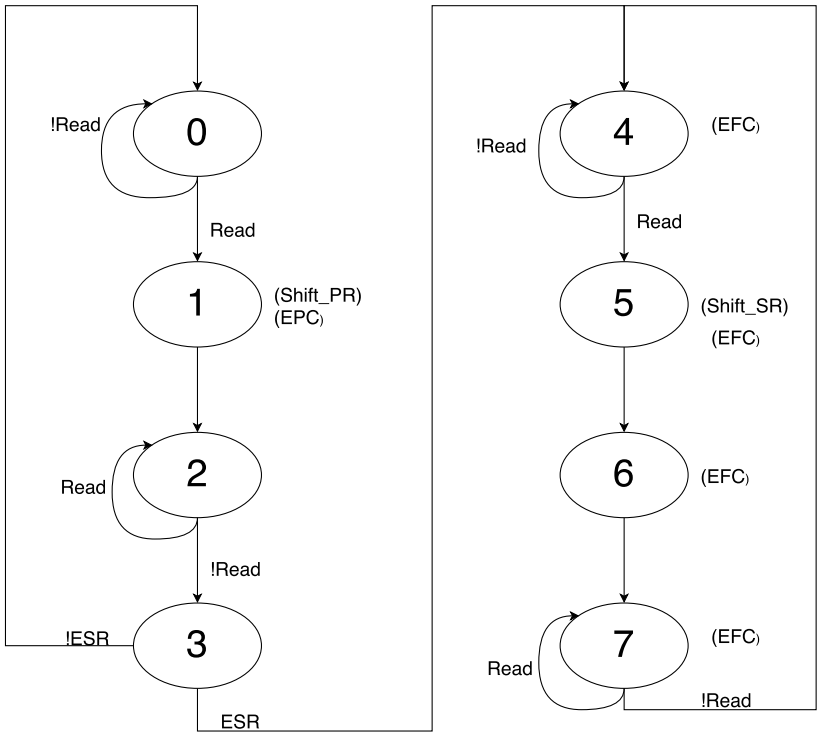


Figura 4.6: Diagrama de estados desarrollado para el control de la operación.

Tabla 4.4: Descripción de los estados de la FSM de control, estados siguientes y operaciones realizadas en cada estado.

Estado	Variable	Operación	Estado Siguiente		Salida
			Verdadera	Falsa	
0	Read	CC	1	0	(Shift_PR)(EPC)
1	-	CI	2		-
2	Read	CC	2	3	-
3	ESR	SC	4	0	-
4	Read	CC	5	4	(EFC)
5	-	CI	6		(Shift_SR)(EFC)
6	-	CI	7		(EFC)
7	Read	SC	7	4	(EFC)

Capítulo 5

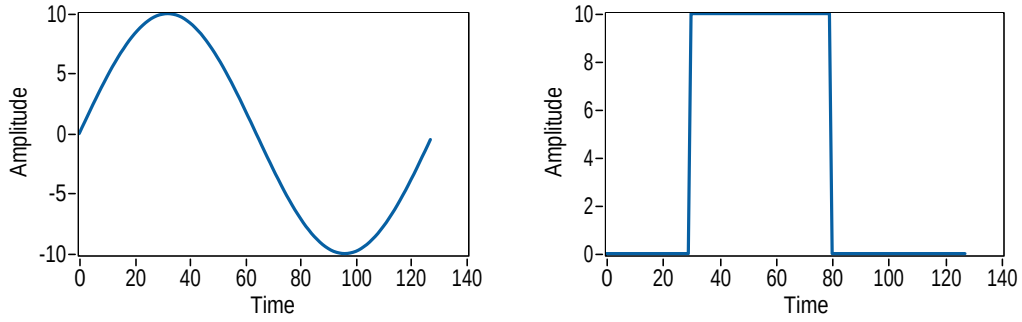
Resultados y análisis

5.1 Resultados y análisis de la implementación en alto nivel.

En esta sección se presentan los resultados obtenidos al realizar tres pruebas diferentes con diferentes combinaciones de patrón y muestra, y se comparan los resultados obtenidos con el algoritmo de correlación *bitstream* en alto nivel con diferentes valores de sobremuestreo contra el resultado teórico de correlación.

5.1.1 Primera prueba

En la primera prueba se usó como patrón una señal senoidal, con un solo periodo de la señal como la de la Figura 5.1(a). Como señal muestreada se utiliza una señal cuadrada como la de la Figura 5.1(b), ambas señales cuentan con 128 datos.



(a) Patrón

(b) Muestra

Figura 5.1: Señales utilizadas para la primera prueba.

La Figura 5.2 contiene las formas de ondas de correlación del modelo de alto nivel y la teórica, luego de realizar un acondicionamiento a las señales obtenidas. La correlación

teórica es la señal de color azul, la señal roja es la correlación *bitstream* con un OSF de 8, la verde con un OSF de 16, la morada con un OSF de 32 y por último la anaranjada con un OSF de 64. Estos colores se mantienen para todas las gráficas de las pruebas del modelo de alto nivel exceptuando a la de la Figura 5.11.

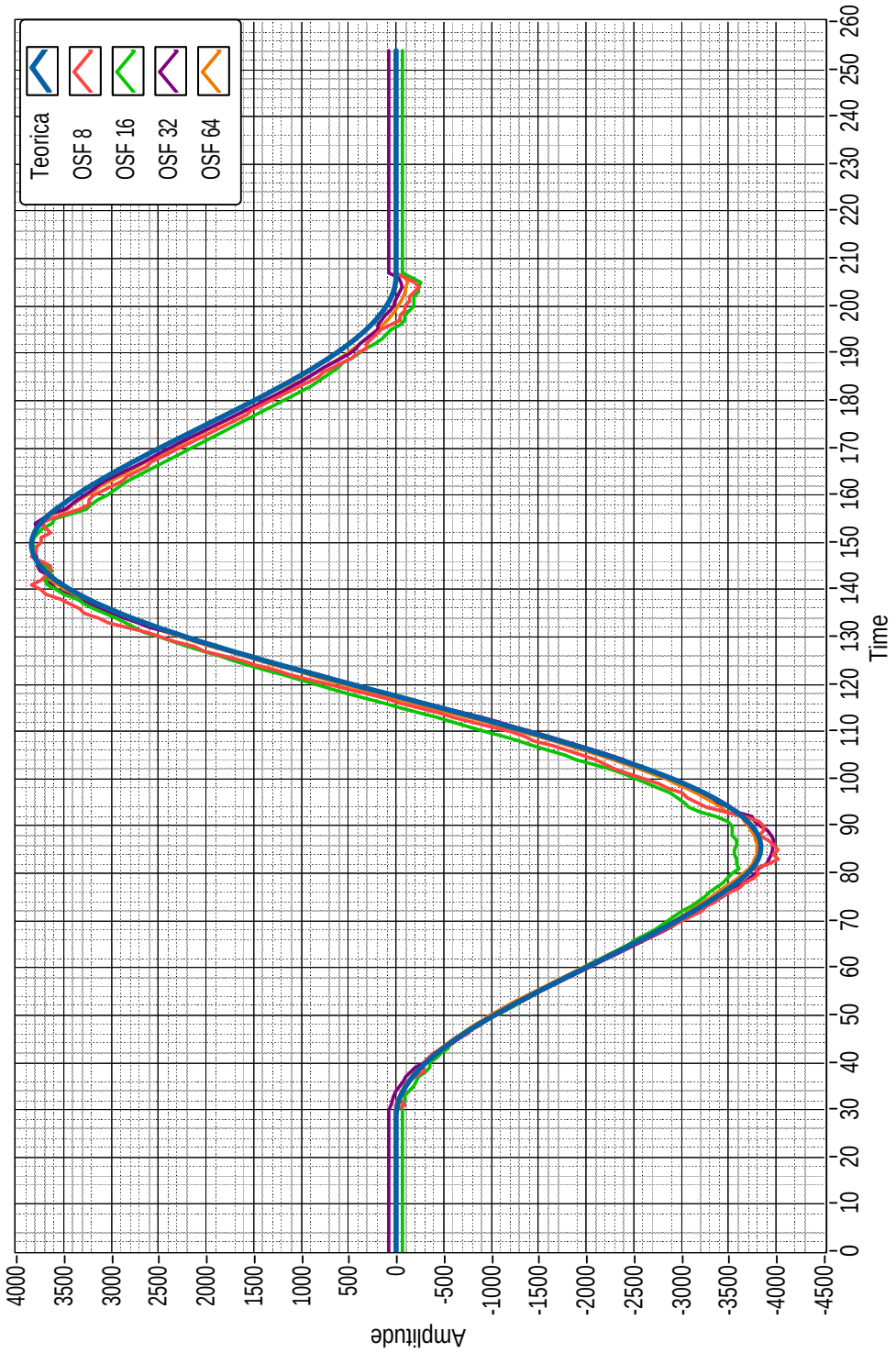


Figura 5.2: Primera prueba de correlación para cuatro valores de OSF.

Los porcentajes de error obtenidos al utilizar el algoritmo *bitstream* se presentan en la Figura 5.3. Se puede apreciar de la Figura 5.2 que el valor del modelo de alto nivel se apega fielmente a la correlación teórica, pero hay momentos en que los porcentajes de error alcanzan valores muy altos, incluso superior al 100%. Esto es producto del modelo de error usado (ver Ecuación (5.1)), que obviamente se indefine en aquellos casos en que teóricamente la correlación entre dos señales es cero. Se sabe que el error en sistemas de representación numérica son función de la precisión de dicho sistema [12]. Según la teoría de modulación sigma-delta, la precisión es una función directa de la relación señal a ruido (SNR), es decir del sobremuestreo sobre la señal a convertir (a diferencia de la conversión estándar PCM, donde la precisión se mapea directamente con la resolución de la representación usada). Para averiguar si el valor de error es aceptable en estos casos extremos, se grafica la precisión del algoritmo evaluado, según la Ecuación (5.2) que mide el error absoluto en vez del relativo, tal como se aprecia en la Figura 5.4.

$$Porcentaje\,Error = \left| \frac{ValorTeórico - Valor\,Experimental}{Valor\,Teórico} \right| \tag{5.1}$$

$$Error\,Absoluto = |Valor\,Teórico - Valor\,Experimental| \tag{5.2}$$

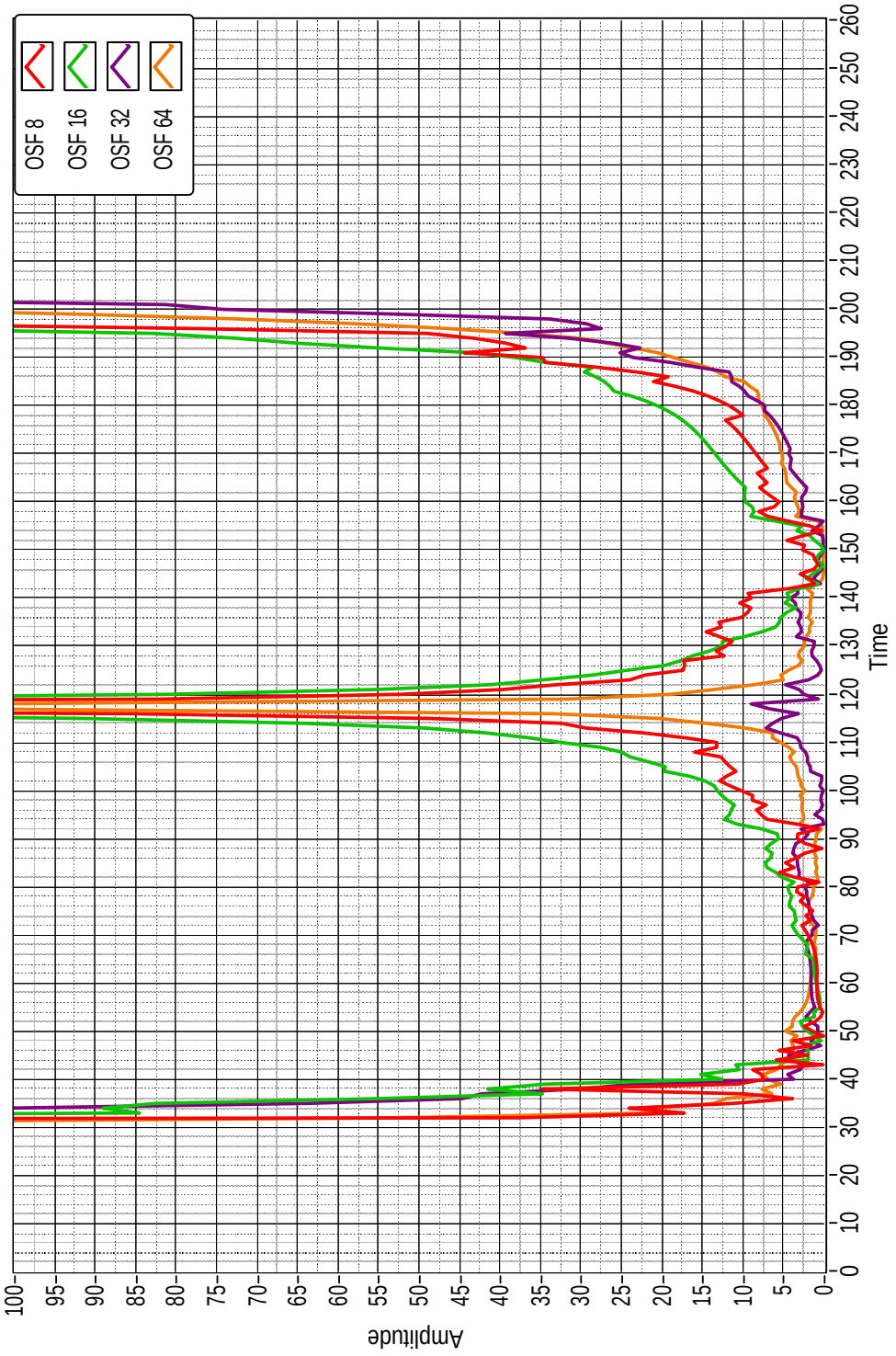


Figura 5.3: Graficación del porcentaje de error para la primera prueba(ver ecuación (5.1)). Nótese la divergencia del error en los casos en que teóricamente la correlación es cero.

Aunque en el gráfico de la Figura 5.2 no se percibe gran diferencia al variar el sobremuestreo para calcular la correlación, ya que todos los modelos se asemejan en gran medida a

la respuesta teórica, en la Figura 5.4 es claro que la señal roja y la verde que representan un OSF de 8 y de 16 respectivamente, tienen un error mucho mayor, respecto al obtenido con un OSF de 32 y 64 representado con las señales morada y anaranjada.

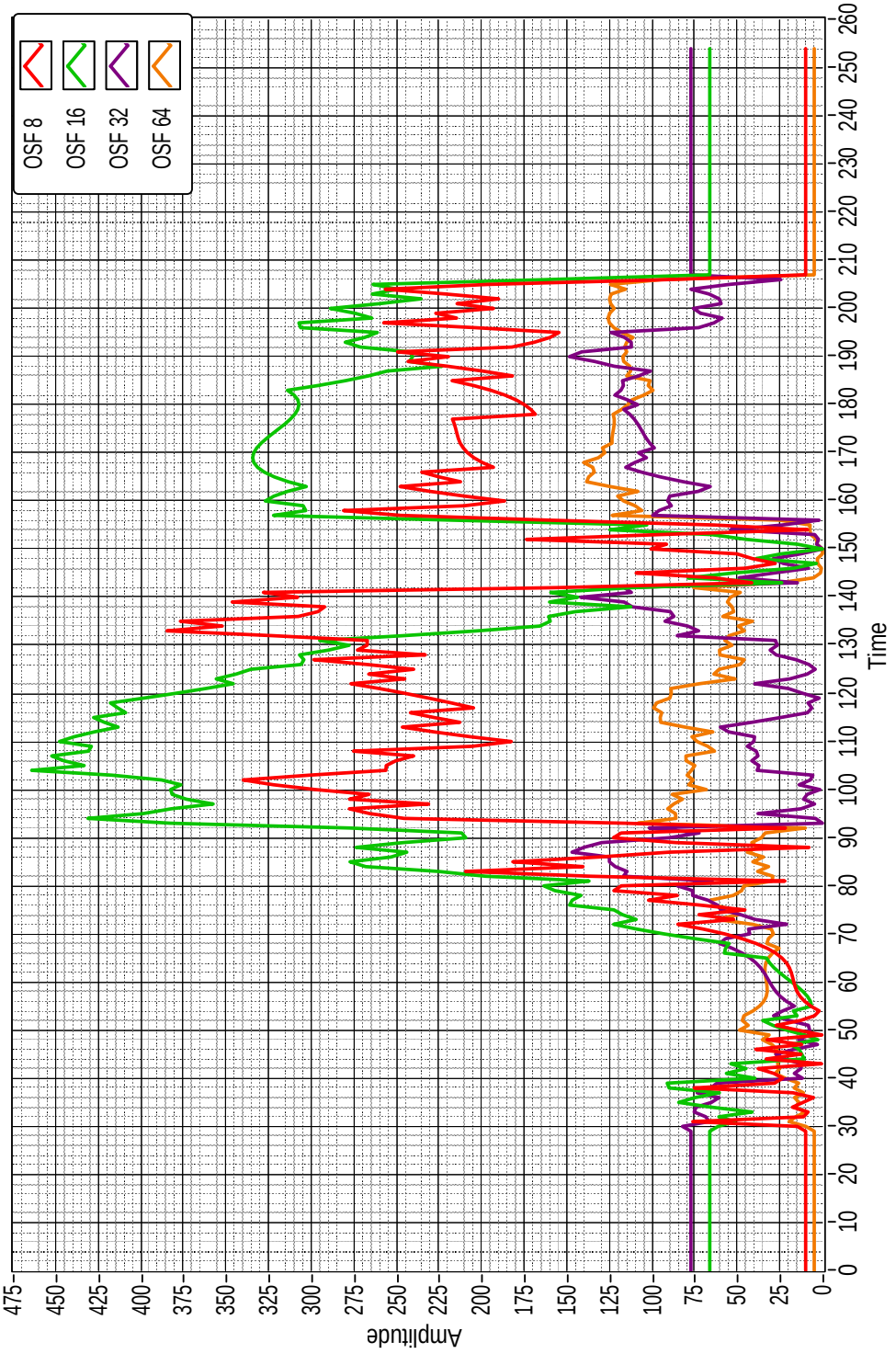


Figura 5.4: Error absoluto del algoritmo de correlación *bitstream* de la primera prueba contra su modelo teórico.

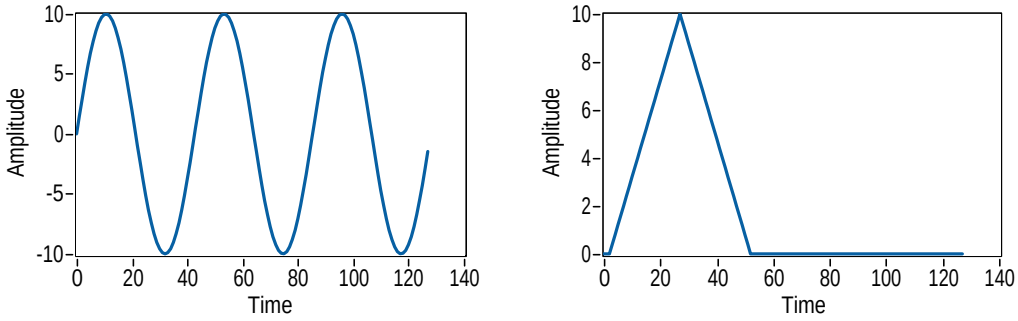
Tabla 5.1: Error absoluto promedio para la primera prueba.

$OSF8$	$OSF16$	$OSF32$	$OSF64$
111,45	169,15	65,02	47,43

La Tabla 5.1 contiene los resultados obtenidos al promediar los datos del gráfico de la Figura 5.4. Se confirma entonces que el OSF de 32 y OSF de 64 son los que brindan un menor error, llegando en el último caso a un error tres veces menor que el de un OSF de 16.

5.1.2 Segunda prueba

Para la segunda prueba se utilizó como patrón una señal senoidal de tres periodos correlacionada contra una señal triangular. Ambas señales cuentan con 128 datos y se observan en la Figura 5.5.



(a) Patrón (b) Muestra

Figura 5.5: Señales utilizadas para la segunda prueba.

En la Figura 5.6 se encuentra el gráfico de las correlaciones. Se destaca la señal con un OSF de 8 ya que a simple vista es la que señal que más difiere con la señal teórica, al apreciarse mucho ruido comparada con las otras señales, principalmente entre el dato 140 y 180. Además en algunos máximos y mínimos no se alcanza el valor debido como es el caso del dato 25, 100 y 120.

Al utilizar un OSF de 16 se nota un comportamiento similar al que se comentó para un OSF de 8, pero con una reducción en el ruido. Las señales obtenidas al utilizar un OSF de 32 y 64 siguen mucho mejor a la señal teórica que las otras dos señales, la mayor mejora que se observa en la gráfica es entre el dato 140 y 180 que es donde se da la mayor reducción del ruido, aún asi al utilizar un OSF de 64 se tienen problemas para alcanzar algunos valores máximos y mínimos como en las cercanias del dato 80 y 120, siendo la señal que se obtuvo con un OSF de 32 la que brinda la mejor aproximación a la señal teórica.

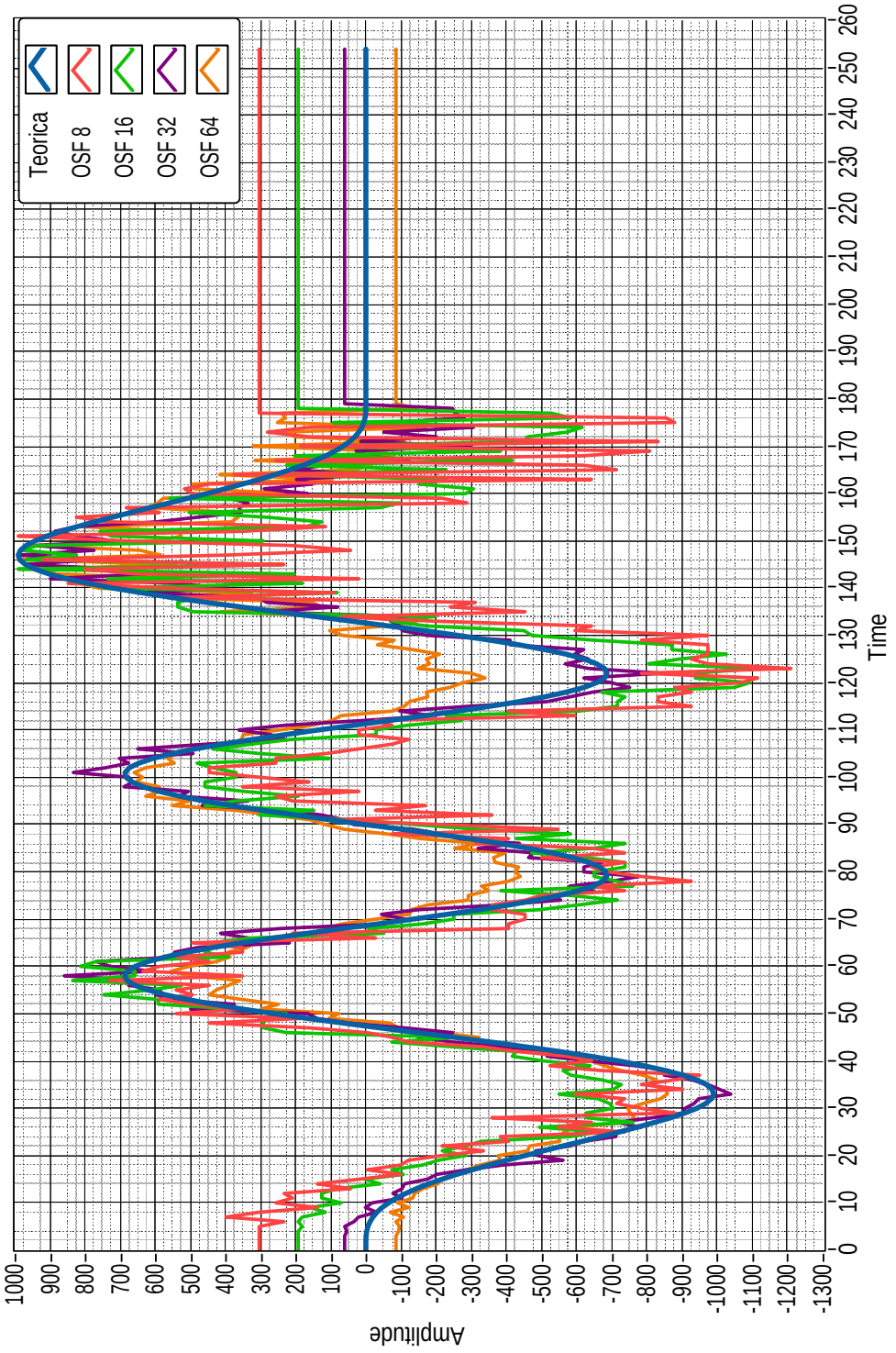


Figura 5.6: Segunda prueba de correlación para cuatro valores de OSF.

De la Figura 5.7 que contiene el error absoluto promedio para la segunda prueba realizada, se observa a simple vista que se mantiene el patrón esperado donde a menor OSF mayor el error en los resultados obtenidos.

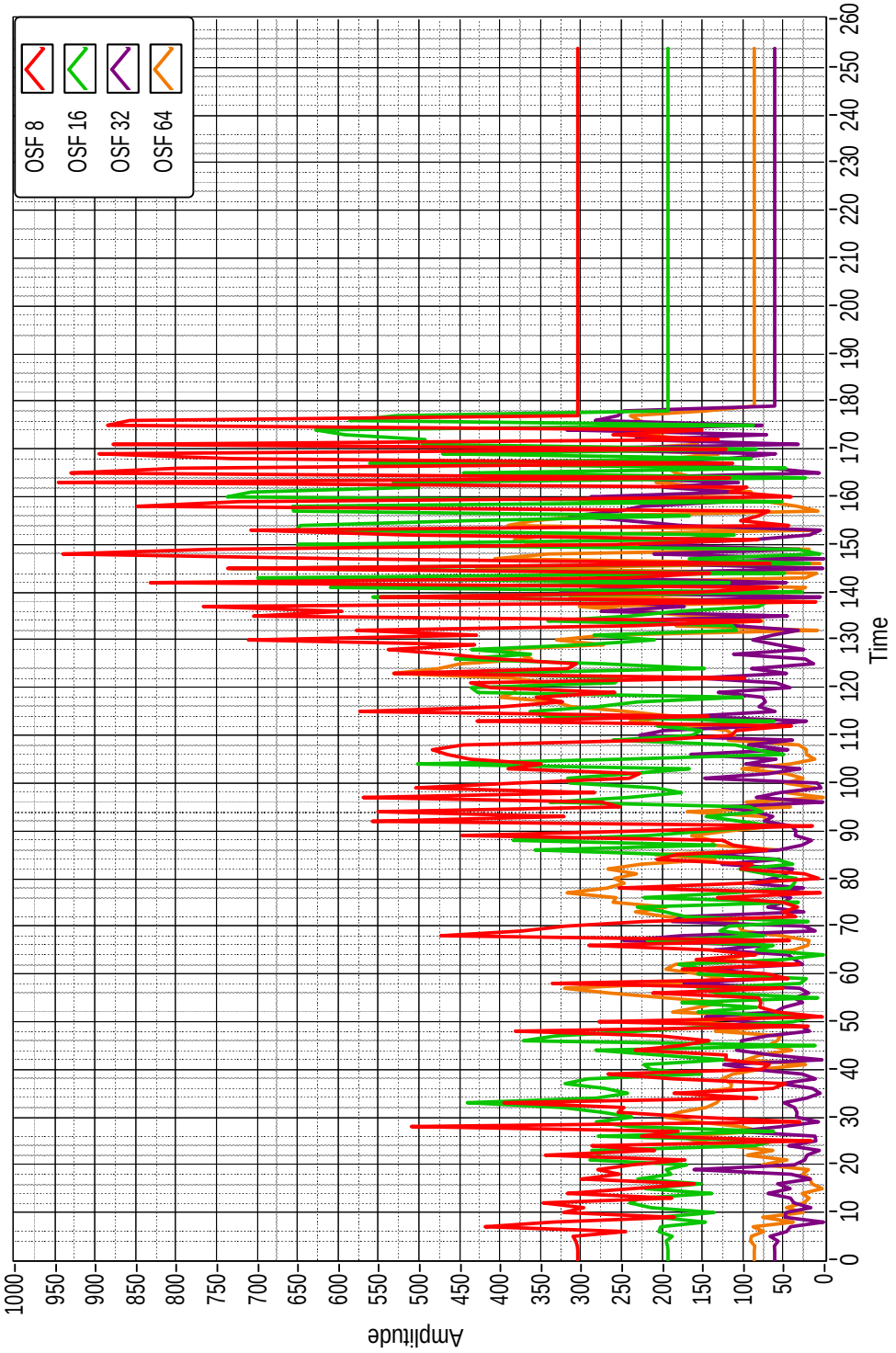


Figura 5.7: Error absoluto del algoritmo de correlación *bitstream* de la segunda prueba contra su modelo teórico.

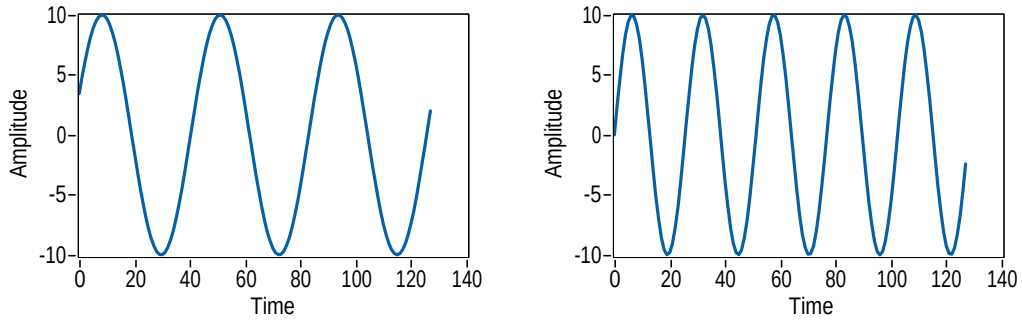
Tabla 5.2: Error absoluto promedio para la segunda prueba.

$OSF8$	$OSF16$	$OSF32$	$OSF64$
294,72	220,84	82,03	134,72

La Tabla 5.2 el promedio del error de la Figura 5.7. Los resultados son similares a los obtenidos en la primer prueba, donde el mayor error se obtuvo para un OSF de 8 y de 16 y el menor error con un OSF de 32 y 64. Sin embargo en este caso el error promedio obtenido con un OSF de 64 supera al obtenido con un OSF de 32.

5.1.3 Tercera prueba

Para la tercera prueba se usó como patrón una señal senoidal, con tres periodos de la señal y como señal muestreada una señal senoidal con cinco periodos. Ambas señales cuentan con 128 datos y se observan en la Figura 5.8.



(a) Patrón (b) Muestra

Figura 5.8: Señales utilizadas para la tercera prueba.

La Figura 5.9 contiene las gráficas resultantes al calcular la correlación entre las señales anteriores, en este caso se puede ver como a pesar de que la correlación teórica es la más compleja de las 3 pruebas, debido a que esta tiene mayor cantidad de cambios en la monotonía. Todas las correlaciones obtenidas por el algoritmo de correlación *bitstream* brindan una buena aproximación de este comportamiento.

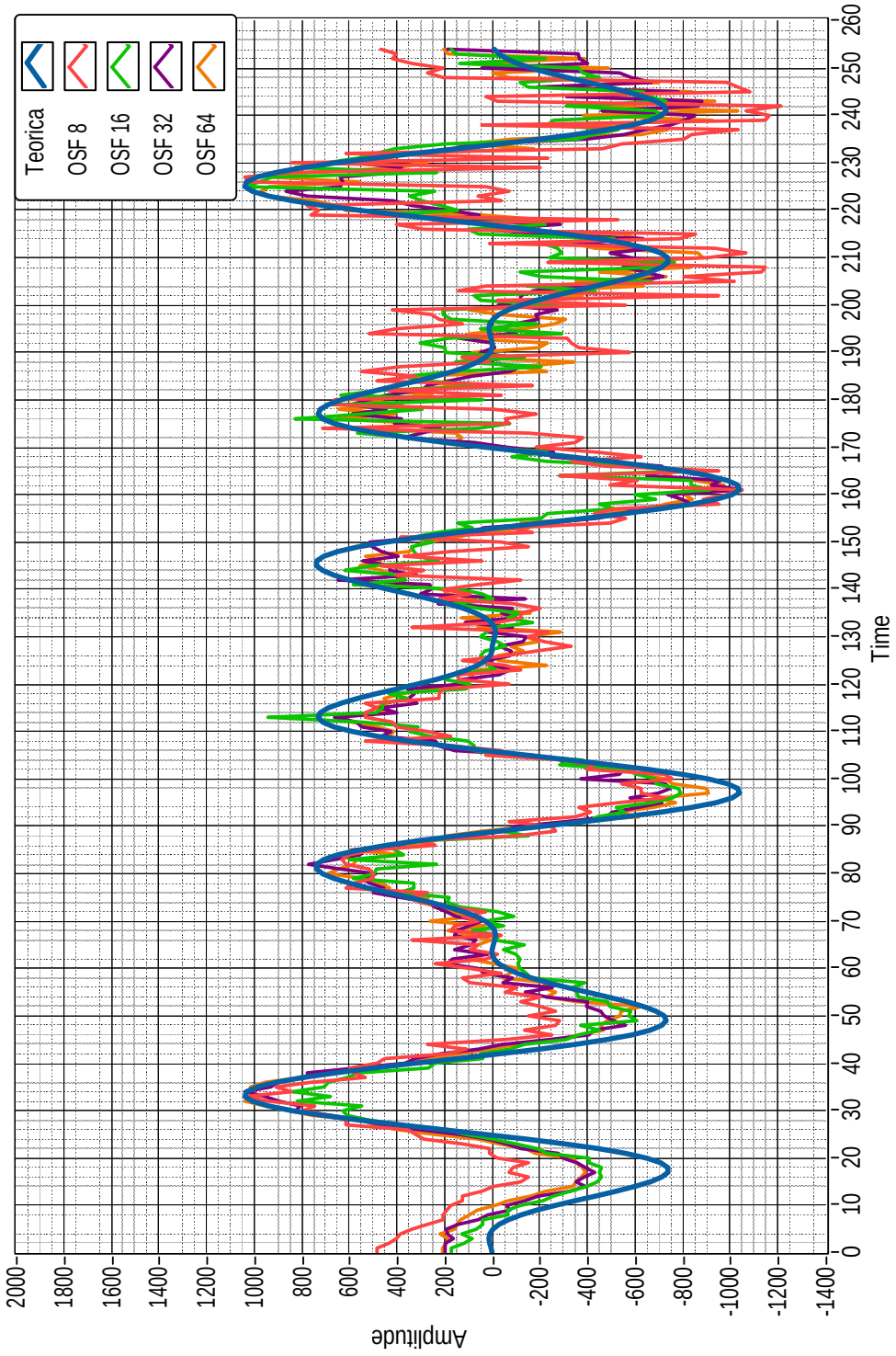


Figura 5.9: Tercera prueba de correlación para cuatro valores de OSF.

La Figura 5.10 presenta el gráfico del error absoluto promedio obtenido para la tercera prueba para los cuatro valores de OSF utilizados para realizar el cálculo de correlación.

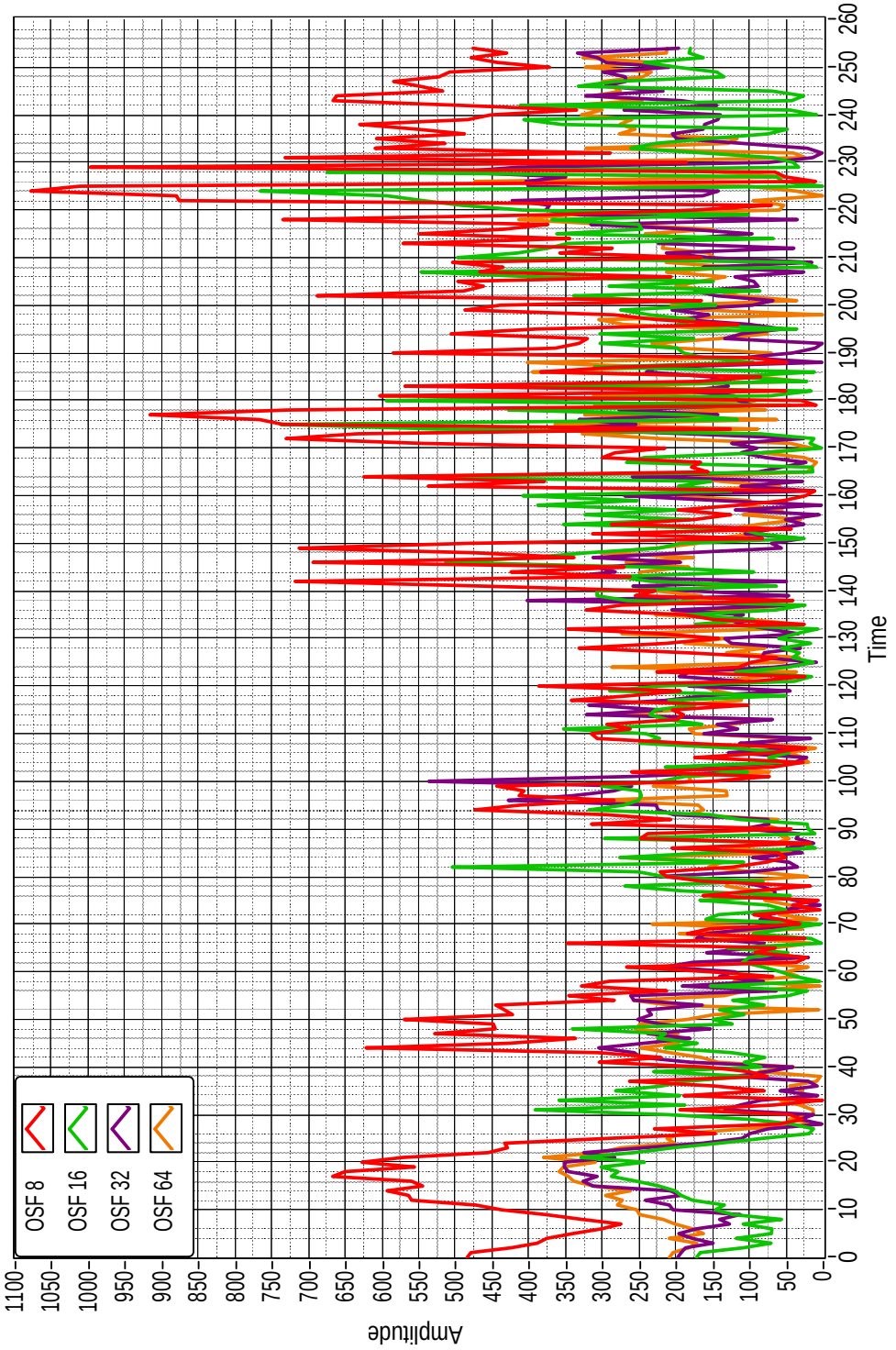


Figura 5.10: Error absoluto del algoritmo de correlación *bitstream* de la tercera prueba contra su modelo teórico.

Tabla 5.3: Error absoluto promedio para la tercera prueba.

$OSF8$	$OSF16$	$OSF32$	$OSF64$
321,21	183,577	157,58	159,86

De la Tabla 5.3 se puede destacar que el error promedio de las señales de la Figura 5.10 fue similar para la correlación que utiliza un OSF de 16, 32 y 64. Cabe notar no obstante que en este caso el error al usar un OSF de 8 no fue tan alto como en las pruebas anteriores, con respecto a OSF más altas.

5.1.4 Resultados generales

En la Figura 5.11 se encuentra un gráfico que resume los resultados obtenidos en las pruebas anteriores. Se puede notar que los mejores resultados para todas las pruebas se obtienen con un OSF de 32 y de 64. Dada la proximidad de los errores en estos dos casos, se favorece el uso de una OSF de 32 bits, que requirá cerca de la mitad de hardware con respecto al uso de un OSF de 64, con una pérdida no determinante de precisión.

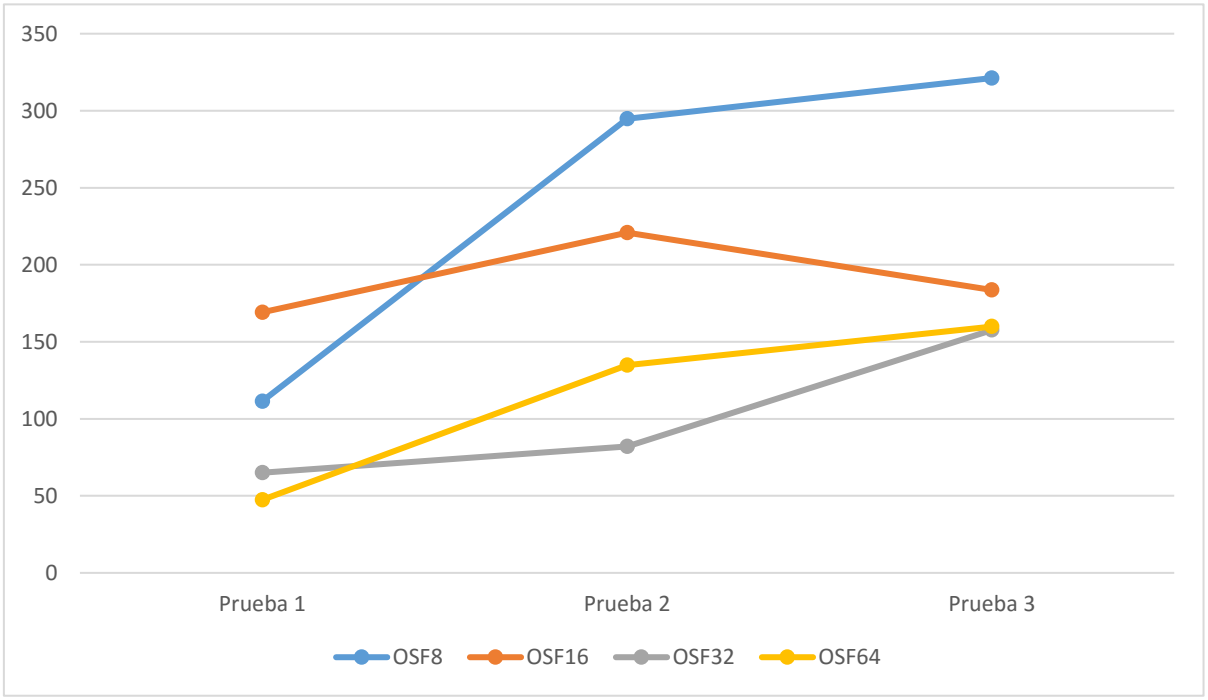


Figura 5.11: Gráfico resumen del error promedio en pruebas de alto nivel.

5.2 Análisis de la implementación RTL del algoritmo de correlación *bitstream*

Se realizaron dos pruebas en las que se comparan los resultados obtenidos con el módelo de alto nivel contra los de la implementación RTL utilizando un OSF de 8. Para esta aplicación la frecuencia de muestreo está dada por

$$F_s = K \times OSF \times F_{Max} \tag{5.3}$$

donde K es la constante de Nyquist, de acuerdo con el teorema de muestreo de Nyquist la frecuencia de muestreo tiene que ser mayor a 2 veces la máxima frecuencia de la señal a muestrear para que la señal sea recuperable

$$F_s > 2 \times F_{Max} \tag{5.4}$$

Partiendo de un K igual a 4 para cumplir con el teorema de muestreo de Nyquist y una frecuencia máxima de 20kHz ya que es de interés el saber si se puede utilizar el algoritmo de correlación *bitstream* para detectar sonidos de disparo y motosierras (y que estas señales tienen un ancho de banda de 20Hz a 20kHz), se puede calcular la frecuencia de muestreo mínima que se debe utilizar para la implementación al sustituir en la Ecuación (5.5).

$$F_s = 4 \times 8 \times 20 \times 10^3 = 640kHz \tag{5.5}$$

Debido a que el reloj que se utiliza para la simulación es de 100MHz, se elige utilizar una frecuencia de muestro de 8.3MHz. Despejando la constante de Nyquist de la Ecuación 5.5 se tiene

$$K = \frac{F_s}{OSF \times F_{Max}} \tag{5.6}$$

Si se utiliza la ecuación anterior para calcular la constante de Nyquist y se mantiene la frecuencia de muestreo y se cambia el OSF a 64 que es el mayor OSF que se utiliza en este documento, se obtiene

$$K = \frac{8.3 \times 10^6}{64 \times 20 \times 10^3} = 6.48 \tag{5.7}$$

Con lo que se cumple con el Teorema de Muestreo de Nyquist y se comprueba que esta implementación puede ser utilizada para la aplicación deseada incluso para un OSF de 64.

5.2.1 Comportamiento del modelo RTL

En la Figura 5.12 se verifica el correcto ingreso de datos en el *Registro de Patrón*. El valor del $ContadorPatron$ será la encargada de indicar a la FSM cuándo esta operación haya finalizado. Recuérdese que este registro actúa como plantilla contra la que se realizará la correlación de la señal de entrada en el detector de señales acústicas en que se busca aplicar este módulo.

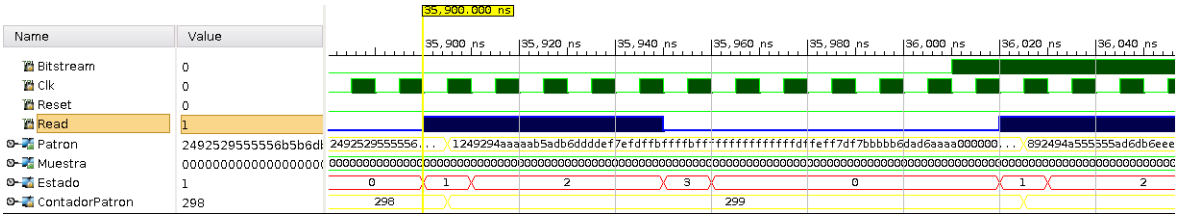


Figura 5.12: Comprobación del correcto almacenamiento en el *Registro de Patrón*.

En el ejemplo mostrado, puesto que la OSF es de 8 y el *Registro de Patrón* contiene 128 muestras quiere decir que el tamaño del registro es de 1024 bits. Al alcanzar el $ContadorPatron$ entonces el valor de 1024, se activa la señal $S1$ y la FSM sabrá que debe proceder al estado siguiente, dando por iniciado el proceso de escritura en el *Registro de Muestra* (ver Figura 5.13) donde la señal irá entrando a la F_s definida.

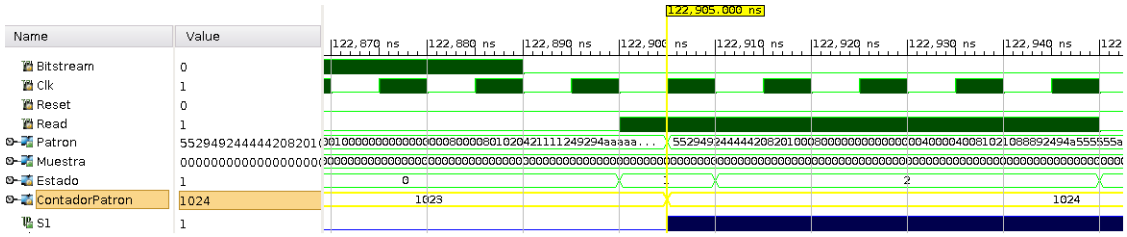


Figura 5.13: Comprobación de la habilitación de la escritura en el *Registro de Muestra*.

En la Figura 5.14 se comprueba el ingreso de datos en el *Registro de Muestra*, este se da de manera similar a la escritura en el *Registro de Patrón* mostrado en la Figura 5.12.

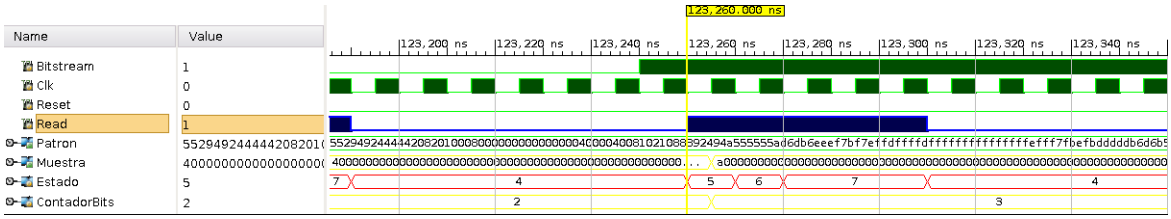


Figura 5.14: Comprobación del correcto almacenamiento en el *Registro de Muestra*.

La función de la señal $ContadorBits$ es determinar cuándo se ha escrito un dato completo en el sistema, para proceder a indicar que el cálculo de correlación está listo. Por la teoría de sobremuestreo en que se basa el convertidor sigma-delta, para obtener el equivalente a un cálculo de correlación para un dato codificado en PCM, se debe esperar a que el $ContadorBits$ sea igual a la OSF usada. Es decir, se deberá calcular un número OSF de correlaciones *bitstream* parciales, el último valor calculado nos da la correlación equivalente en PCM.

De la Figura 5.15 se puede observar que cuando el valor de $ContadorBits$ es igual al OSF, en este caso 8, se activa la señal $S5$. Esta señal hace que se cargue el valor parcial de correlación en el *Registro de Correlación* y que inicie la cuenta en el *Contador de Bandera*.

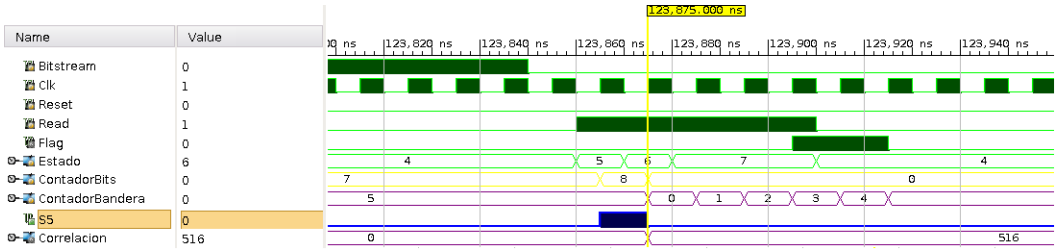


Figura 5.15: Comprobación del correcto funcionamiento del *Contador de Bits*.

El *Contador de Bandera* es el que se encarga de activar y desactivar la bandera que indica que ya el dato de correlación calculado es válido. Además determina cuanto tiempo después de que se carga correctamente el valor final de correlación al *Registro de Correlación* se activa la bandera y cuanto tiempo permanece la bandera activada. En la Figura 5.16 se aprecia que cuando $ContadorBandera$ es igual a 2 se activa la señal $S2$ que es la encargada de que la señal $Flag$ sea igual a 1, y cuando $ContadorBandera$ es igual a 4 se activa la señal $S3$ que se encarga de devolver a la señal $Flag$ a su valor inicial de 0.

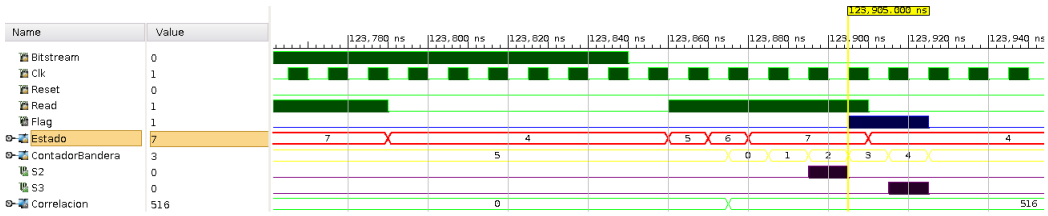
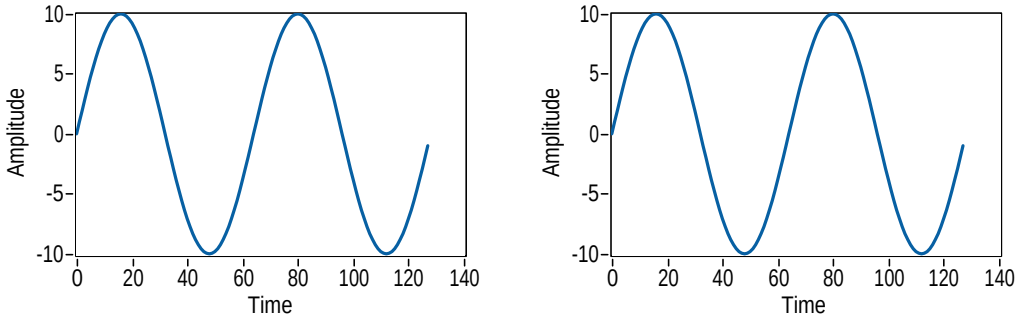


Figura 5.16: Comprobación de la activación de la señal de *Flag*.

Con lo anterior se verifica el comportamiento del diseño, para comprobar que los resultados obtenidos son los correctos se realizan dos pruebas en las cuales se compara el resultado del modelo de bajo nivel con el de alto nivel.

5.2.2 Primera prueba

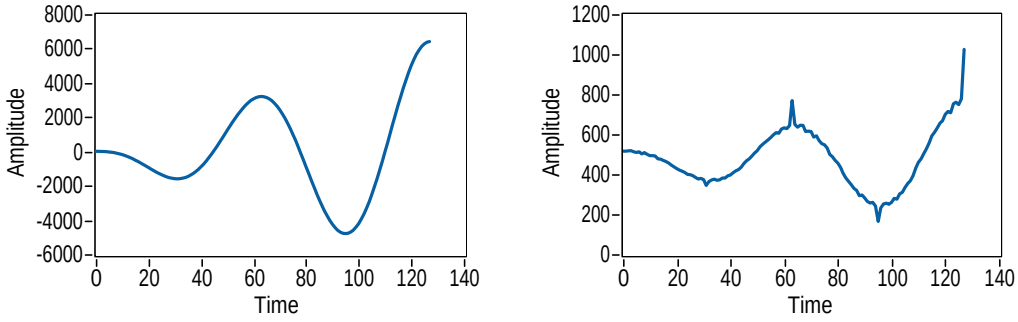
En la primera prueba se utiliza la misma señal como patrón y como muestra: dos senoidales idénticas de 128 datos como se puede ver en la Figura 5.17.



(a) Patrón (b) Muestra

Figura 5.17: Ondas usadas para la comprobación durante la primera prueba del correlador *bitstream*. Dos senoidales idénticas, de 128 datos de longitud.

En la Figura 5.18(a) se observan las salidas de la correlación teórica y la *bitstream*, para un OSF de 8. Nótese la relativa exactitud del algoritmo *bitstream*, excepto en los lugares que la correlación es máxima en magnitud. Deber recordarse que en el algoritmo implementado existe un relativo escalamiento y cambio en valor medio que debe tenerse en cuenta para aplicaciones posteriores.



(a) Correlación teórica (b) Correlación *bitstream*

Figura 5.18: Resultados de la salida de la correlación teórica y *bitstream* para las senoidales de la Figura 5.17. Nótese la relativa exactitud del algoritmo *bitstream*, excepto en los lugares que la correlación es máxima en magnitud.

En la Figura 5.19 se encuentran los resultados obtenidos al simular el comportamiento del diseño del RTL en la herramienta de simulación de Vivado. En la Figura 5.20 pueden verse con detalle los resultados obtenidos en la simulación a nivel RTL contrastados contra el resultato del modelo de alto nivel. Puede notarse entonces que el error visto de la Figura 5.17 es producto no de la implementación RTL sino del mismo algoritmo, lo que significa que las cotas de error se mantienen.

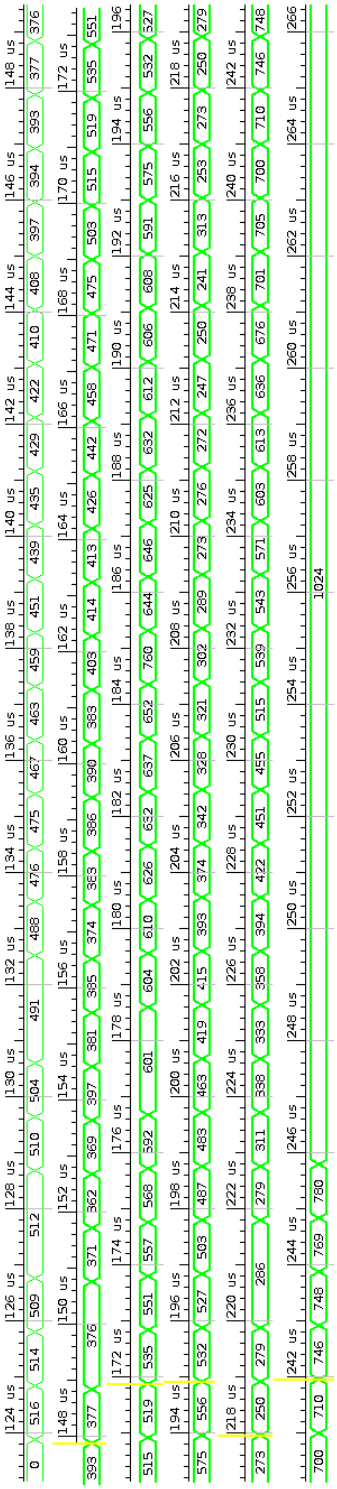


Figura 5.19: Detalle de la simulación RTL del algoritmo de correlación *bitstream* para la primera prueba. Se muestra los valores finales de correlación calculados.

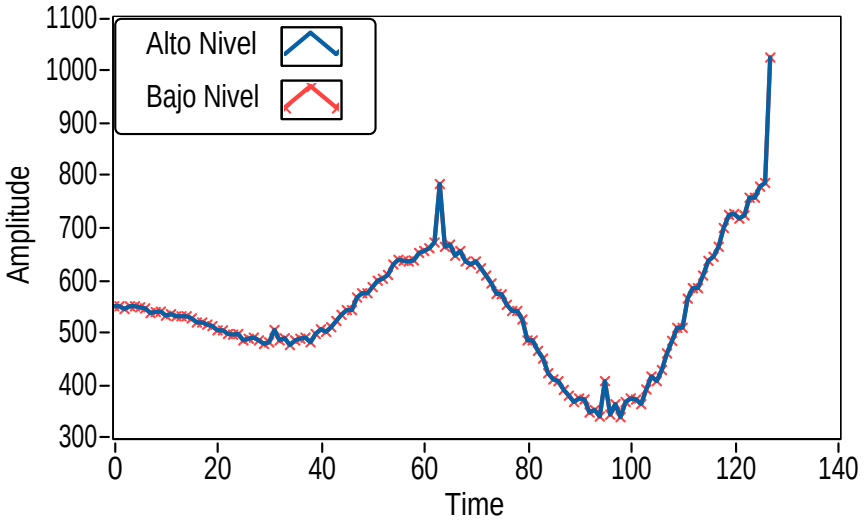
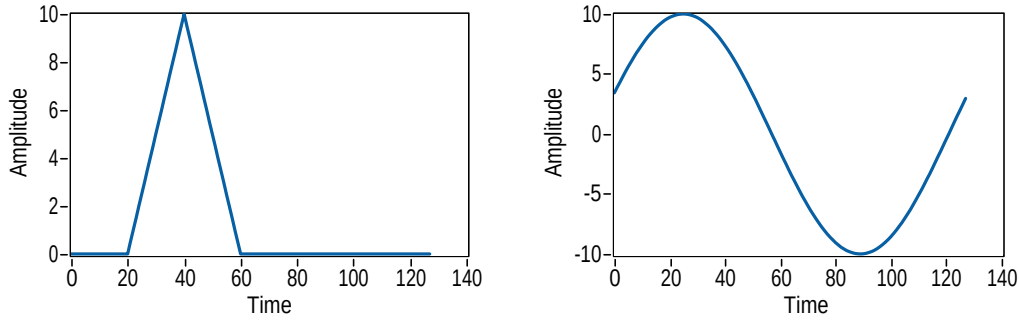


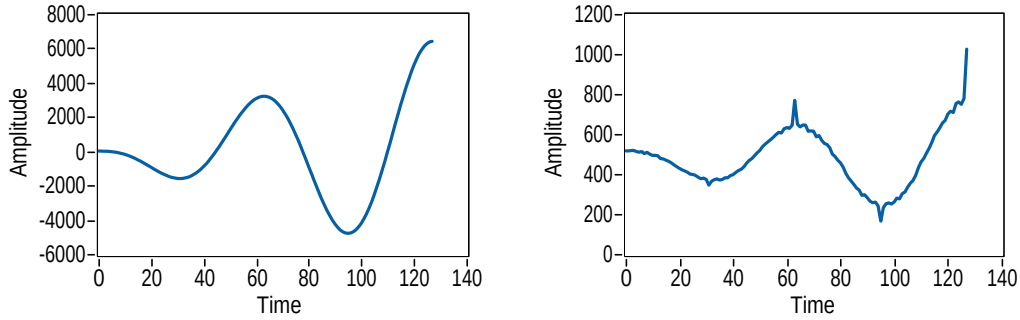
Figura 5.20: Contraste entre resultado del modelo de alto nivel de la correlación *bitstream* y el modelo RTL. Puede verse que el algoritmo RTL ofrece resultados idénticos al de alto nivel (y, por tanto, su error con respecto a la correlación teórica es el mismo.)

5.2.3 Segunda prueba



(a) Patrón (b) Muestra

Figura 5.21: Ondas usadas para la comprobación durante la segunda prueba del correlador bitstream



(a) Correlación Teórica (b) Correlación *Bitstream*

Figura 5.22: Resultados de la salida de la correlación teórica y *bitstream* para las señales de la Figura 5.21.

En la Figura 5.23 se encuentran los resultados obtenidos al simular el comportamiento del diseño del RTL en la herramienta de simulación de Vivado. La Figura 5.24 contiene el gráfico de los datos obtenidos con el modelo de bajo nivel, junto con los del modelo de alto nivel, se observa que la forma de ambas señales es igual como se dio en la prueba anterior.

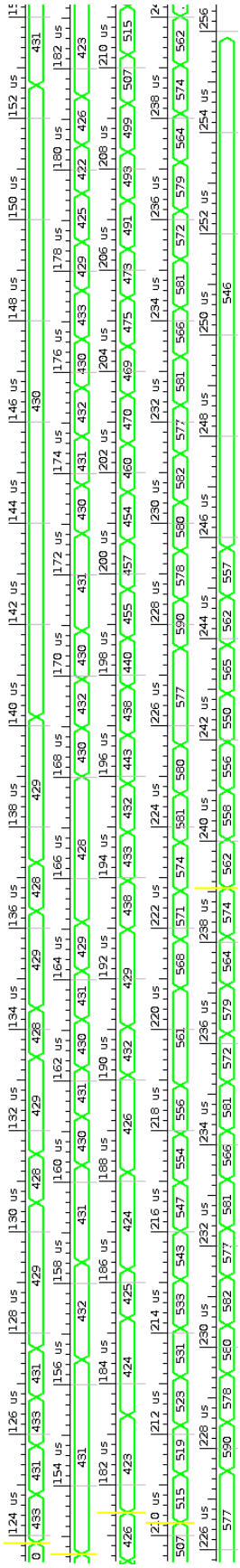


Figura 5.23: Detalle de la simulación RTL del algoritmo de correlación *bitstream* para la segunda prueba.

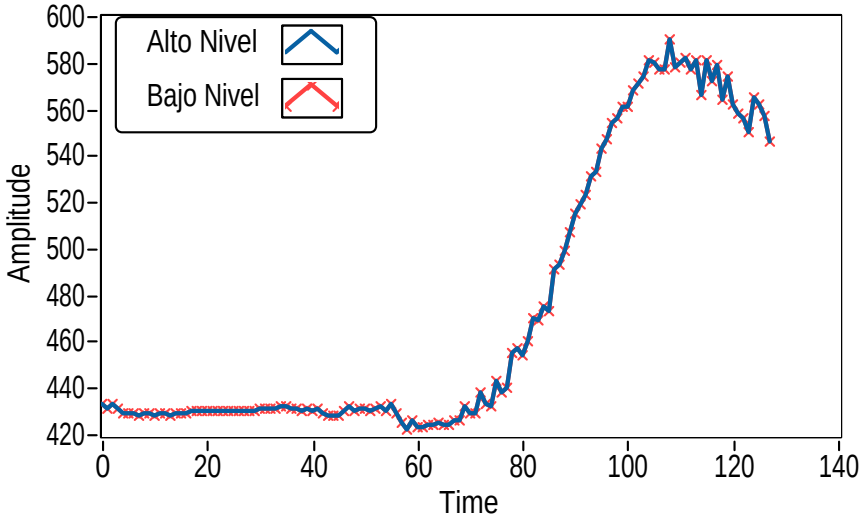


Figura 5.24: Contraste entre resultado del modelo de alto nivel de la correlación *bitstream* y el modelo RTL para la segunda prueba.

Capítulo 6

Conclusiones

- La representación *bitstream* permite calcular la correlación entre dos señales sin consumir tantos recursos con métodos tradicionales.
- Al utilizar la representación *bitstream* es posible aproximar la multiplicación entre dos datos utilizando simples compuertas *xnor*.
- Con un factor de sobremuestreo igual o superior a 32 se obtiene una aproximación satisfactoría de correlación comparada con su valor teórico.
- La mejor implementacción del algoritmo de correlación *bitstream* se obtiene utilizando un factor de sobremuestro de 32, ya que se obtienen resultados tan favorables como al utilizar un factor de sobremuestreo de 64, con la diferencia de que simplifica aún más el hardware requerido para su implementación.
- El algoritmo de correlación *bitstream* se puede utilizar con señales de con un ancho de banda de 20Hz a 20kHz.

Bibliografía

[1] Chacón-Rodríguez A. Alvarado Moya, J.P. Sistema electrónico integrado en chip (soc) para el reconocimiento de patrones de disparos y motosierras en una red inalámbrica de sensores para la protección ambiental, 2014. URL http://www.ie.itcr.ac.cr/achacon/VIE/sirpa-doc1.pdf.

[2] P. M. Aziz, H. V. Sorensen, and J. vn der Spiegel. An overview of sigma-delta converters. *IEEE Signal Processing Magazine*, 13(1):61–84, Jan 1996.

[3] Bonnie Baker. How delta-sigma adcs work, part 1. 2011.

[4] Salazar García C. Implementación de un microprocesador de aplicación específica para la ejecución del algoritmo de modelos ocultos de markov para el reconocimiento de patrones acústicos. Master's thesis, 2015.

[5] O. Villalta-Gutierrez A. Chacón-Rodriguez P. Alvarado-Moya. C. Gomez-Viquez, L.A. Li-Huang. An embedded test system for acoustic patternrecognition intended for environmental monitoringand protection in tropical rain forest reserves. In *Embedded Technologies Conference 2012*, 2012.

[6] 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 *Circuits Systems (LASCAS), 2015 IEEE 6th Latin American Symposium on*, pages 1–4, Feb 2015.

[7] A. Chacon-Rodriguez. *Circuitos integrados de bajo consumo para detección y localización de disparos de armas de fuego*. PhD thesis, 2009.

[8] A. Chacon-Rodriguez, P. Julian, L. Castro, P. Alvarado, and N. Hernandez. Evaluation of gunshot detection algorithms. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(2):363–373, Feb 2011.

[9] A. Chacon-Rodriguez, P. Julian, and F. Masson. Fast and low power integrated circuit for impulsive sound localisation using kalman filter approach. *Electronics Letters*, 46(7):533–534, April 2010.

[10] A. Chacon-Rodriguez, S. Li, M. Stanaćević, L. Rivas, E. Baradin, and P. Julian. Low power switched capacitor implementation of discrete haar wavelet transform. In

Circuits and Systems (LASCAS), 2012 IEEE Third Latin American Symposium on, pages 1–4, Feb 2012.

[11] 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*, 56(7):575–579, July 2009.

[12] R. Curtis. Dally, William J.; Harting. *Digital Design: A Systems Approach.* 1 ed. edition, 2012.

[13] Instituto Nacional de Biodiversidad. Biodiversidad en Costa Rica. 2016. URL http://www.inbio.ac.cr/es/biod/bio_biodiver.htm.

[14] HANS-PETTER HALVORSEN. *Introduction to LabVIEW*, 2014.

[15] T. S. Lande, T. G. Constandinou, A. Burdett, and C. Toumazou. Running cross-correlation using bitstream processing. *Electronics Letters*, 43(22), Oct 2007.

[16] O. E. Liseth, H. A. Hjortland, and T. S. "Bassen" Lande. Power efficient cross-correlation beat detection in electrocardiogram analysis using bitstreams. In *2009 IEEE Biomedical Circuits and Systems Conference*, pages 237–240, Nov 2009.

[17] Olav E. Liseth. Low power bitstream running cross-correlator / convolver. Master's thesis, 2009.

[18] V. Loaiza. Áreas silvestres protegidas expuestas a cacería, tala e invasiones. 2008. URL http://www.nacion.com/ln_ee/2008/mayo/23/pais1541047.html.

[19] F. Maloberti and P. O'Leary. Processing of signals in their oversampled delta-sigma domain. In *Circuits and Systems, 1991. Conference Proceedings, China., 1991 International Conference on*, pages 438–441 vol.1, Jun 1991.

[20] Sangil Park. *Principles of Sigma-Delta Modulation for Analog-to- Digital Converters.* 2008.

[21] A. Pauchard. La Experiencia de Costa Rica en Áreas Protegidas. 2000.

[22] F. N. M. Pirchio, P. Julian, 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*, pages 4016–4019, May 2007.

[23] 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 *Circuits Systems (LASCAS), 2015 IEEE 6th Latin American Symposium on*, pages 1–4, Feb 2015.