# DEDICATORIA

*A mis padres por apoyarme y darme un ejemplo a seguir.*

*A mis hermanos por su contención y amistad.*

*A mi familia por cada palabra de aliento.*

*A mis amigos y compañeros por todo lo compartido en estos años.*

# AGRADECIMIENTOS

*A mis tutores Eduardo Dominguez y Jorge Naguil por su seriedad y dedicación.*

*Al Departamento de Ciencias Básicas por conseguir los recursos y espacio de trabajo.*

*Al Instituto Universitario Aeronáutico y los profesores que lo conforman por proveer las herramientas necesarias para la formación profesional.*

*A mis compañeros y amigos por ayudarme y compartir estos años de estudio.*

# IMPLEMENTACIÓN DE UN ALGORITMO DE DETECCIÓN DE MARCAS DE AGUA BASADO EN RESONANCIA ESTOCÁSTICA

*INSTITUTO UNIVERSITARIO AERONAUTICO – FACULTAD DE INGENIERIA*

**Aprobado por el Departamento de Electrónica y Telecomunicaciones en cumplimiento de los requisitos exigidos para otorgar el título de Ingeniero en Telecomunicaciones al Señor Lanfranco Sebastián – DNI 32599630.**

Revisado por:

| | |
|---|---|
| ..... | ..... |
| Dr. Eduardo Alfredo Dominguez<br>Tutor de Trabajo | Ing. Jorge Luis Naguil<br>Tutor de Trabajo |

.....

Ing. Marcela Busnardo

Director Dpto. Electrónica y Telecomunicaciones

.....

Ing. Héctor Carlos Riso

Director Dpto. Desarrollo Profesional

Tribunal Examinador

.....

Dr. Víctor Sauchelli

Presidente del Tribunal Examinador

.....

Dr. Roberto Guibert

Vocal del Tribunal Examinador

Córdoba, 30 de Julio de 2010

# *ÍNDICE*

## ACRÓNIMOS

| | |
|---|---|
| $A$ | Amplitud de la señal moduladora |
| $a$ | Coeficiente del término lineal del polinomio en $x$ |
| $B_i$ | Constante $i$ -ésima |
| $b$ | Coeficiente del término cúbico del polinomio en $x$ |
| $C$ | Constante ingresada al sistema de dos estados para el análisis cualitativo |
| $C_0$ | Valor crítico para el cual el sistema de dos estados tiene un único punto crítico |
| $C_x$ | Campo a cifrar |
| $C_{x.enc}$ | Campo cifrado |
| $c$ | Valores de $x$ con el potencial cuartico tiene sus mínimos |
| $D$ | Dispersión del ruido Gaussiano |
| $D_A$ | Dispersión del audio |
| $E$ | Constante |
| $E_b$ | Energía de bit |
| $E_{'1'}$ | Energía de bit para ‘1’ |
| $E_{'0'}$ | Energía de bit para ‘0’ |
| $F$ | Figura de ruido |
| $F_M$ | Figura de ruido máxima |
| $f_s$ | Frecuencia de muestreo |
| $f_{tr}$ | Frecuencia de trabajo |

| | |
|---|---|
| $G_{Lat}$ | Campo “grados” para Latitud |
| $G_{Lat.enc}$ | Campo “grados” para Latitud encriptado |
| $G_{Lon}$ | Campo “grados” para Longitud |
| $G_{Lon.enc}$ | Campo “grados” para Longitud encriptado |
| $H(t)$ | Tasa de transición de un estado a otro |
| $H_K$ | Tasa de Kramer |
| $K$ | Cociente entre la dispersión y la amplitud de la señal moduladora |
| $K_i$ | Ganancia del integrador discreto |
| $K_x$ | Valor en decimal del código ASCII de la letra de la llave en uso |
| $K_x{}'$ | Valor decimal para descifrar |
| $k$ | Ajuste de entrada el receptor basado en resonancia estocástica |
| $L$ | Límite |
| $M_{Lat}$ | Campo “minutos” para Latitud |
| $M_{Lat.enc}$ | Campo “minutos” para Latitud encriptado |
| $M_{Lon}$ | Campo “minutos” para Longitud |
| $M_{Lon.enc}$ | Campo “minutos” para Longitud encriptado |
| $N_i$ | Potencia de ruido de entrada |
| $N_o$ | Potencia de ruido de salida |
| $n_0$ | Condición inicial para la posición de la variable dinámica $x$ |
| $n_+$ | Probabilidad de que $x$ esté en el estado + |
| $n_-$ | Probabilidad de que $x$ esté en el estado - |

| | |
|---|---|
| $P_{Lat}$ | Precisión Latitud |
| $P_{Lon}$ | Precisión Longitud |
| $p(x)$ | Función densidad de probabilidad de $x$ |
| $R_{\eta\eta}$ | Función de auto-correlación |
| $S$ | Potencia de señal |
| $SNR$ | Relación señal-ruido |
| $S_i$ | Potencia de señal de entrada |
| $S_{Lat}$ | Campo “segundos” de Latitud |
| $S_{Lat.enc}$ | Campo “segundos” de Latitud encriptado |
| $S_{Lon}$ | Campo “segundos” de Longitud |
| $S_{Lon.enc}$ | Campo “segundos” de Longitud encriptado |
| $S_o$ | Potencia de señal de salida |
| $T_b$ | Tiempo de bit |
| $T_s$ | Periodo de muestreo |
| $T_{tr}$ | Periodo de la señal moduladora |
| $t_0$ | Tiempo inicial |
| $U$ | Función potencial |
| $U_0$ | Altura de la barrera de potencial |
| $U_1$ | Amplitud de la modulación de la barrera de potencial |
| $W$ | Proceso de Wiener |
| $W_{\pm}$ | Tasa de salto entre los estados + y - |

| | |
|---|---|
| $x$ | Variable dinámica |
| $x_i$ | Puntos críticos |
| $x_M$ | Tren de pulsos cuadrados |
| $x_+$ | Valor de $x$ para el estado + |
| $x_-$ | Valor de $x$ para el estado - |
| $z$ | Variable compleja |
| $\alpha$ | Parte real del número complejo |
| $\alpha_n$ | Coeficiente $n$-ésimo del polinomio propuesto para $H(t)$ |
| $\beta$ | Parte imaginaria del número complejo |
| $\delta$ | Delta de Dirac |
| $\delta_{x_0 c}$ | Función que toma el valor de 1 cuando $x_0 = c$ |
| $\eta$ | Ruido Gaussiano |
| $\eta_0$ | Fuerza de modulación de la señal al parámetro $\mu$ |
| $\mu$ | Relación entre la barrera potencial y el ruido |
| $\tau_K$ | Tiempo de Kramer |
| $\tau_r$ | Tiempo de relajación |
| $\Omega$ | Variable de frecuencia |
| $\omega_{tr}$ | Frecuencia de la señal moduladora en $\left[\frac{rad}{seg}\right]$ |

## RESUMEN

En el presente trabajo se analizó el mecanismo del fenómeno de resonancia estocástica con el objetivo de implementarlo en un sistema para la detección de marcas de agua digitales en archivos de audio.

A través de este análisis, se desarrolló un sistema de comunicaciones en el cual se envía, por medio del transmisor, un mensaje encriptado por un algoritmo simple y enmascarado por audio a través del método de marcas de agua. Una marca de agua digital es un mensaje (generalmente un código de identificación) que se inserta en un archivo multimedia de manera tal que sólo pueda ser detectado a través de un algoritmo específico y una llave. En el receptor, el cual trabaja bajo el principio de resonancia estocástica, se detecta dicha marca de agua digital y se recupera la información enviada.

La descripción de este receptor en códigos VHDL permitió dejar constancia de la viabilidad de la implementación de dicho sistema en una plataforma digital, quedando demostrado su buen desempeño a través de las simulaciones en Simulink.

# Capítulo 1: INTRODUCCIÓN

# 1 INTRODUCCIÓN

En el marco del convenio de colaboración recíproca firmado entre IUA – UTB/TSC (Universidad de Texas en Brownsville, EE.UU) surgió la necesidad de analizar la performance del método resonancia estocástica en relación con el proceso de detección de señales de ondas gravitatorias (señales de extremadamente reducida relación señal-ruido). Paralelamente, en el Departamento de ciencias Básicas de la Facultad de Ingeniería del IUA se están desarrollando actividades de I+D en el marco de un proyecto PIDDEF (Proyectos de Investigación y Desarrollo para la Defensa) del Ministerio de Defensa de la Nación en el área de Teoría y Análisis de Señales. En viste de la concurrencia de ambas temáticas de trabajo, se propuso adaptar el método de resonancia estocástica al área de telecomunicaciones a través del envío de información oculta en archivos de audio, lográndose entonces una aplicación altamente relacionada con las telecomunicaciones aplicadas a la Defensa.

El fenómeno de resonancia estocástica consiste en mejorar la relación señal-ruido (SNR) de un determinado sistema añadiendo ruido al mismo. Esto es, que la señal de entrada al sistema actúa en forma cooperativa con el ruido presente en el mismo, mejorando así la SNR. Este sistema permitirá la detección de señales débiles en entornos con mucho ruido, lo cual es ideal para ser aplicado en la tecnología de marcas de agua digitales.

Una marca de agua digital es un mensaje oculto en archivos multimedia (audio, imagen, video). Este mensaje (marca de agua) se inserta en el archivo multimedia (el cual será de audio en nuestro caso) modificándolo levemente, de manera tal que sea imperceptible al oído humano. Independientemente del método utilizado, se deberá garantizar que la marca de agua tenga las siguientes características: robustez, resistente a manipulaciones, imperceptible e indetectable y que posea baja probabilidad de error.

Para añadir más seguridad a los datos transportados a través de la marca de agua se incluirá un algoritmo de encriptación. El concepto de encriptación consiste en transformar datos de alguna forma que sean ilegibles, sin el conocimiento de la clave o algoritmo adecuado. El propósito de ésta es mantener oculta la información que consideramos privada a cualquier persona o sistema que no tenga permitido el acceso a la misma. De esta forma, se estaría incrementando el nivel de seguridad a los datos.

Se analizará la posibilidad de implementar un sistema compuesto por dos módulos, un transmisor donde se generará y transmitirá el audio marcado (audio portador de la marca de

agua) y un receptor que trabajará bajo el principio de resonancia estocástica con el cual se detectará la marca de agua digital y se recuperarán los datos. De esta manera se mostrará el proceso completo de envío y recepción de marcas de agua transportadas en archivo de audio.

Finalmente, a modo de ejemplo de utilización específica de la aplicación, se describirá el proceso de envío de posición sobre la superficie terrestre a través de coordenadas. La idea principal es poder enviar una posición en la superficie terrestre de forma secreta, representada por una palabra binaria, la cual será cifrada y desordenada de manera tal de añadir dificultad a un supuesto interceptor de la señal no autorizado.

# Capítulo 2: CONCEPTOS GENERALES

# 2 CONCEPTOS GENERALES

En este capítulo se hará una breve descripción de los pilares que sustentan este trabajo final. Se comenzará con un resumen histórico del fenómeno de resonancia estocástica, siguiendo por los conceptos de marca de agua digital, encriptación y esteganografía.

## 2.1 Breve reseña histórica de resonancia estocástica

El concepto de resonancia estocástica fue presentado por Benzi [1], con el objeto de modelar y argumentar el cambio de clima de la Tierra entre las edades de hielo y periodos de relativo calentamiento, los cuales se dan cada unos 100000 años, aproximadamente. La excentricidad de la órbita de la Tierra varía con ese periodo, pudiendo atribuirse a esto el cambio climático; sin embargo, de acuerdo a teorías actuales, dicha variación no es lo suficientemente intensa para provocar tales cambios. Considerando al clima terrestre como una variable que puede tomar dos estados (análogo a un potencial de dos estados, biestable), se sugirió que la existencia de un fenómeno cooperativo entre una variación periódica débil en la excentricidad de la Tierra (la "señal") y otras fluctuaciones aleatorias, podría explicar la fuerte periodicidad observada.

A pesar del nombre "resonancia estocástica" propuesto para este mecanismo, se advirtió que no se trata de un fenómeno de resonancia en el sentido clásico, esto es, un incremento de la respuesta de un sistema cuando una perturbación se encuentra sintonizada con las "frecuencias naturales" del mismo. Este mecanismo, sin embargo, es una útil analogía con la resonancia, ya que se maximiza la relación señal-ruido (la "respuesta") cuando algunos parámetros del "ruido" son apropiadamente sintonizados.

## 2.2 Marca de agua digital

### 2.2.1 Introducción

La marca de agua digital es un código de identificación que se inserta directamente en el contenido de un archivo multimedia (imagen, audio, video), para dificultar su percepción por el sistema perceptual humano, pero fácil de detectar usando un algoritmo dado y una clave.

Las marcas de agua digitales han sido propuestas principalmente como una solución eficiente para la protección de los derechos de copia y propiedad de los archivos de datos multimedia, posibilitando la identificación de la fuente, autor, propietario, distribuidor o consumidor autorizado, de imágenes digitales, grabaciones de audio o video. Otras de las aplicaciones de esta técnica son marcas de agua transaccionales (fingerprinting), monitorizado de transmisiones de radiodifusión y comunicaciones secretas (esteganografía), siendo esta última de interés para este trabajo.

La principal ventaja de estos sistemas consiste en que la marca es inseparable del contenido del archivo. Sin embargo, existen algunas cuestiones que necesitan ser resueltas, antes de que estas técnicas puedan ser eficazmente aplicadas en los escenarios de la vida real.

### 2.2.2 Características básicas

Independientemente de su aplicación, esta marca debe tener características tales como:

- Robustez: los archivos digitales de imágenes, audio y video, están expuestos a muchos tipos de modificaciones (o distorsiones) tales como pérdidas por compresión, cambios producidos por el mejoramiento de imágenes, amplificación de las señales de audio, etc. Una marca de agua se considera robusta si perdura después de esas operaciones y puede ser recuperada eficazmente. Para consolidar su robustez, los sistemas de marcas de agua deben insertar la marca en las regiones perceptualmente significativas de los archivos multimedia. La robustez no debe exigirse incondicionalmente, ya que un sistema de marcas de agua puede necesitar ser robusto respecto a determinados procesos y frágil respecto a otros.

- Resistencia a manipulaciones: este aspecto que puede relacionarse con la seguridad del mismo; se refiere a su resistencia frente a los ataques hostiles basados en el total conocimiento de los algoritmos de incrustado y detección y de los archivos marcados, excepto de la clave utilizada. Según la aplicación de que se trate, unos ataques serán más importantes que otros y en general, un ataque efectivo deberá eliminar la marca de agua sin cambiar la calidad perceptual del archivo en cuestión. Sin embargo, existen varias aplicaciones en las que la resistencia a determinadas manipulaciones es un aspecto indeseable.

- Imperceptibilidad e indetectabilidad: son dos conceptos que tienden a confundirse frecuentemente, aunque son muy distintos y no están relacionados entre sí. La imperceptibilidad o transparencia de la marca tiene como base el comportamiento del sistema perceptual humano. Una marca de agua es imperceptible (transparente), si la degradación que causa en los archivos donde se ha insertado es muy difícil de apreciar. La indetectabilidad está relacionada con el modelo estadístico del archivo antes y después de ser marcado. Se dice que la marca es indetectable si después de haberla insertado, el archivo marcado conserva las mismas propiedades estadísticas que su original. Lo que quiere decir que una persona no autorizada no podrá detectar la presencia de la marca utilizando métodos estadísticos. Esta propiedad es muy deseable en el caso de las comunicaciones encubiertas en las que el principal objetivo es ocultar la presencia del mensaje incrustado en el archivo.

- Baja probabilidad de error: la probabilidad de error al detectar una marca debe ser muy pequeña. Se denomina probabilidad de falso negativo a la probabilidad de que, habiendo estado presente una marca en determinado archivo, el detector asuma que no hay tal marca. Por otro lado, la probabilidad de falso positivo es la probabilidad de que no estando la marca presente en un archivo, el detector asuma que la marca está presente.

### 2.2.3 Aplicaciones

Debe considerarse que los requisitos que deben cumplir en la práctica los algoritmos de marcas de agua deben analizarse dentro de un entorno de trabajo del sistema y de acuerdo con la aplicación donde será utilizado. A continuación se enumera una seria de aplicaciones para el método de marcas de agua digitales:

- Marcas de agua como firmas: las marcas pueden utilizarse para firmar archivos multimedia. El propietario de uno de estos archivos insertará una marca de agua que lo identifique como tal.
- Marcas de agua transaccionales (fingerprinting): las marcas de agua también pueden utilizarse para identificar a los compradores de los archivos multimedia, lo que puede servir para la búsqueda del infractor en el caso de distribución de copias ilegales de un archivo dado.
- Marcas de agua para autenticación: existen muchas aplicaciones donde la veracidad de una imagen es crucial, por ejemplo: imágenes médicas. Las marcas utilizadas para la autenticación contendrán la información requerida que determinará la integridad de un archivo multimedia. Esta marca debe ser invisible y frágil (cualquier modificación de la imagen debe alterar la marca) ya que es muy deseable que pueda ofrecer información sobre los cambios ocurridos en las imágenes.
- Monitorizado de transmisiones de radiodifusión: al igual que en las firmas y las marcas transaccionales, las marcas de agua identificarán al propietario de los archivos multimedia y/o al comprador de una copia determinada de los mismos y serán detectadas por sistemas automatizados que rastrean las transmisiones de televisión y radiodifusión, las redes de ordenadores y otros canales de distribución para estar al tanto de cuándo y dónde se ha utilizado un archivo multimedia.
- Comunicaciones secretas: en esta aplicación, la marca incrustada en los archivos multimedia se utiliza por dos o más personas para comunicarse secretamente sin levantar la sospecha de terceros. Es la aplicación clásica de la esteganografía de comunicación por canales subliminales.

Destacamos la última aplicación señalada, ya que será la tomada para el desarrollo de este trabajo.

## 2.3 Criptografía

### 2.3.1 Definición

La criptografía es el arte o ciencia de cifrar y descifrar información mediante técnicas especiales y se emplea frecuentemente para permitir un intercambio de mensajes que sólo puedan ser leídos por personas a las que van dirigidos y que poseen los medios para descifrarlos.

### 2.3.2 Conceptos básicos

A continuación de describen una seria de conceptos básicos utilizados en la jerga de la criptografía:

- Texto plano o texto claro: se denomina así a la información original que debe protegerse.
- Cifrado: es el proceso de convertir el texto plano de manera que se torne ilegible, denominado texto cifrado o criptograma.
- Algoritmo de cifrado: su aplicación se basa en la existencia de una clave, la cual adapta dicho algoritmo para cada uso distinto.
- Descifrado: es el proceso inverso que recupera el texto plano a partir del criptograma y la clave.
- Tipos de cifrado: hay dos grandes grupos de cifras.
    - Cifrado Simétrico: son aquellos algoritmos que utilizan una única clave para realizar el cifrado y descifrado. A esta clave se la llama clave simétrica o clave privada.
    - Cifrado Asimétrico: son aquellos algoritmos que utilizan una clave para cifrar los mensajes y una distinta para descifrarlos. A este tipo de claves se las llama clave asimétrica o clave pública.

### 2.3.3 Métodos

Los cifrados se pueden dividir en dos grandes grupos según el método que se utilice para hacer el cifrado. El primero, llamado cifrado por sustitución, consiste en sustituir las unidades del texto original por otras, sin alterar el orden de los símbolos. El segundo, llamado cifrado por trasposición, se basa en crear un texto cifrado simplemente desordenando las unidades que componen el texto original.

A continuación se exponen ejemplos ilustrativos de las opciones de cifrado anteriormente descritas:

- Ejemplo de trasposición:

Ciframos RESONANCIA ESTOCASTICA. Se escribe la palabra rellenando una tabla de dos filas, entrando por la primer columna de arriba hacia abajo, luego por la segunda, y así sucesivamente.

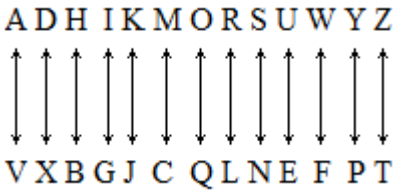
R S N N I E T C S I A
E O A C A S O A T C

Luego se coloca la segunda fila a continuación de la primera, quedando el texto cifrado de la siguiente manera:

R S N N I E T C S I A E O A C A S O A T C.

- Ejemplo de sustitución:

Ciframos SISTEMA BIESTABLE. En este algoritmo de cifrado por sustitución se divide el alfabeto en dos y se arma el siguiente gráfico:




Para armar el texto cifrado, cada letra se reemplaza por su par, quedándonos el ejemplo resuelto para SISTEMA BIESTABLE de la siguiente manera:

N G N Z U C V H G U N Z V H R U.

## 2.4 Esteganografía

### 2.4.1 Definición

La esteganografía es la disciplina en la que se estudian y aplican técnicas que permiten el ocultamiento de mensajes u objetos, dentro de otros, llamados portadores, de modo que no se perciba su existencia. Es una mezcla de artes y técnicas que se combinan para conformar la práctica de ocultar y enviar información sensible en un portador que pueda pasar desapercibido.

### 2.4.2 Diferencia con criptografía

Si bien la esteganografía suele confundirse con la criptografía, por ser ambas, parte de los procesos de protección de la información, son disciplinas bastante distintas tanto en su forma de implementar como en su objetivo mismo. Mientras que la criptografía se utiliza para cifrar o codificar información de manera que sea ininteligible para un probable intruso, a pesar del conocimiento de su existencia, la esteganografía oculta la información en un portador de modo que quede inadvertido el hecho mismo de su existencia y envío. De esta última forma, un probable intruso ni siquiera sabrá que se está transmitiendo información sensible.

Sin embargo, la criptografía y la esteganografía pueden complementarse, dando un nivel de seguridad extra a la información, es decir, es muy común (aunque no imprescindible) que el mensaje a esteganografiar sea previamente cifrado, de tal modo que a un eventual intruso no sólo le costará advertir la presencia misma de la mensajería oculta, sino que si la llegara a obtener, la encontraría cifrada.

# Capítulo 3: TEORÍA DE RESONANCIA ESTOCÁSTICA

# 3 TEORÍA DE RESONANCIA ESTOCÁSTICA

## 3.1 Introducción

A modo de introducción, podemos decir que para generar el fenómeno de resonancia estocástica, en su versión más elemental, se necesitan tres elementos básicos: a) Un sistema de dos estados, con un umbral energético que separe ambos estados, b) Una señal periódica débil que excite al sistema (referida usualmente como "señal"), c) Una fuente de ruido aleatorio (referido usualmente como "ruido"), la cual es parte de la dinámica del mismo sistema (en ocasiones, puede ser también adicionada a la señal). En presencia de estos tres elementos, el sistema exhibe un comportamiento del tipo resonante, para determinadas condiciones del nivel de ruido.

En ausencia de la señal periódica externa, el comportamiento del sistema es estocástico, por lo que su salida presentará un espectro de frecuencias continuo. La excitación del sistema mediante una señal periódica externa forzará el comportamiento periódico del mismo, verificándose un comportamiento cooperativo entre dicha señal y la mecánica propia de este sistema. Lo llamativo de este fenómeno es que bajo ciertas condiciones, el ruido presente en el sistema alimenta la señal aplicada mejorando así la relación señal-ruido y mostrando un pico en el espectro de potencia de la señal de salida, generándose entonces el fenómeno de resonancia estocástica.

De esta forma, mediante una configuración adecuada de los parámetros del sistema e intensidad del ruido se busca lograr dicho comportamiento, lo cual permite la obtención de una señal de salida periódica con frecuencia aproximadamente igual a la de la señal de excitación, mejorando así la relación señal-ruido.

Como se mencionó anteriormente, para materializar la descripción anterior, se tomará como objeto de estudio un sistema de dos estados con dos entradas y una salida, la cual será función de ambas entradas y la dinámica interna del sistema.

## 3.2 Mecanismo de resonancia estocástica: sistema de dos estados

Un sistema de dos estados, es un sistema cuya salida puede tomar dos valores estables, saltando de un valor (estado) al otro según sus entradas. El funcionamiento del sistema de dos estados que se utilizará para materializar el mecanismo de resonancia estocástica, responde a la siguiente ecuación diferencial no lineal:

$$dx = \left(ax - bx^3\right)dt + DdW$$
$$\frac{dx}{dt} = ax - bx^3 + D\frac{dW}{dt}$$

Donde:

$$\frac{dW}{dt} = \eta(t)$$

Esta ecuación establece que la velocidad de variación de la variable dinámica $x$ está gobernada por la derivada temporal del proceso de Wiener (el cual es un proceso estocástico de tiempo continuo cuya derivada temporal es igual al ruido blanco con distribución Gaussiana) y un polinomio en dicha variable que describe la mecánica interna del sistema. Dado que la derivada temporal del proceso de Wiener es igual al ruido Gaussiano, tenemos que la salida del sistema está gobernada por dicho término, por lo que su espectro de frecuencias será constante.

Por lo tanto nos queda el sistema con ruido Gaussiano ($\langle \eta \rangle = 0$ y $\langle \eta^2 \rangle = D^2$):

$$\frac{dx}{dt} = ax - bx^3 + D\eta(t)$$

Con el fin de analizar el efecto de resonancia, se suma una señal periódica, la cual será una función $A\cos(\Omega t)$, donde $A$ es la amplitud y $\Omega$ la frecuencia. Ahora la salida del sistema mostrará un comportamiento conjunto entre la señal y el ruido.

$$dx = \left(ax - bx^3 + A\cos\left(\Omega t\right)\right)dt + DdW$$
$$\frac{dx}{dt} = ax - bx^3 + A\cos\left(\Omega t\right) + D\eta(t) \qquad (3.1)$$

La ecuación diferencial no lineal (3.1) nos muestra que la velocidad de variación de $x$ ahora depende también de un término periódico. El análisis de aquí en adelante tendrá como objetivo encontrar los valores precisos de $D$, $a$, $b$ y $A$ y el dominio de $\Omega$ para los cuales se logra un comportamiento cooperativo entre los términos de ruido Gaussiano y señal periódica, es decir, para que se manifieste el mecanismo de resonancia estocástica.

### 3.2.1 Análisis cualitativo del sistema en ausencia de de señal y ruido

Antes de encontrar las soluciones que satisfacen a la ecuación (3.1) se realizará un análisis preliminar de carácter cualitativo, en ausencia de ruido y señal periódica, o sea para $A = 0$ y $D = 0$. La ecuación diferencial nos queda de la siguiente manera:

$$\frac{dx}{dt} = ax - bx^3 \qquad (3.2)$$

Con el análisis de la ecuación (3.2) se interpreta el funcionamiento de la mecánica del sistema. Lo que se busca es obtener un gráfico de $\frac{dx}{dt}$ en función de $x$ llamado *plano de fase*, y analizar como responde dicha mecánica ante estímulos externos, como el ruido y/o la señal periódica.

Para comenzar con el trazado del plano de fase, se deben encontrar en primera instancia los puntos críticos de (3.2), los cuales se obtienen igualando la derivada temporal a cero y despejando las raíces de la ecuación de tercer grado resultante:

$$\frac{dx}{dt} = 0$$

Llamamos al polinomio en $x$ como $f(x)$, por lo tanto nos queda lo siguiente:

$$f(x) = ax - bx^3 = 0 \qquad (3.3)$$

$$\Rightarrow xb\left(\frac{a}{b} - x^2\right) = 0$$

$$\Rightarrow xb\left(\sqrt{\frac{a}{b}} - x\right)\left(\sqrt{\frac{a}{b}} + x\right) = 0$$

$$x_1 = 0$$

$$x_2 = -\sqrt{\frac{a}{b}}$$

$$x_3 = \sqrt{\frac{a}{b}}$$

Dado que (3.3) una ecuación de tercer grado, se obtienen tres raíces, o sea, tres valores de $x$ para los cuales su derivada temporal es igual a cero. A estos valores se los llama *puntos críticos* o *puntos de equilibrio*.

Una vez encontrados los puntos críticos, se procede a realizar un gráfico aproximado de $f(x)$:

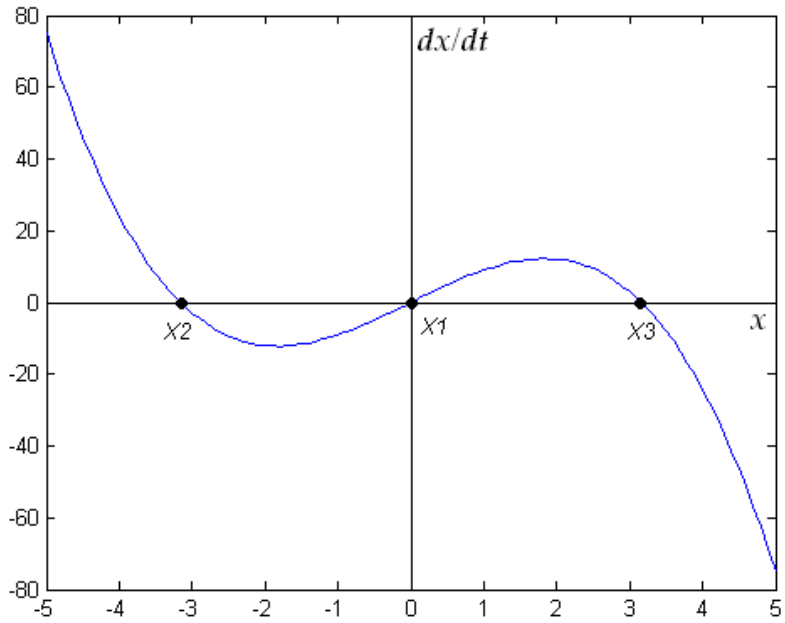



Figura 3.1. Plano fase de la ecuación (3.2).

Este gráfico (Figura 3.1) provee una idea preliminar de cómo varía el signo de la derivada alrededor de los puntos críticos de (3.2), observar si ésta es creciente o decreciente, y trazar diferentes curvas $x(t)$ las cuales dependen de las condiciones iniciales. A este conjunto de trayectorias $x(t)$ se lo denomina *retrato de fase*.

Realizamos a continuación un análisis preliminar analizando el signo de la derivada en base al gráfico obtenido (Figura 3.1). Observamos el comportamiento del sistema alrededor de los puntos críticos, obteniendo lo siguiente:

- Para $x > \sqrt{\frac{a}{b}} \Rightarrow \dot{x} < 0$, lo cual implica que $x$ decrecerá con $t$.
- Para $x < -\sqrt{\frac{a}{b}} \Rightarrow \dot{x} > 0$, lo cual implica que $x$ crecerá con $t$.
- Para $0 < x < \sqrt{\frac{a}{b}} \Rightarrow \dot{x} > 0$, lo cual implica que $x$ crecerá con $t$.
- Para $0 > x > -\sqrt{\frac{a}{b}} \Rightarrow \dot{x} < 0$, lo cual implica que $x$ decrecerá con $t$.

Por lo tanto $x = 0$ es un punto crítico *inestable* y $x = \pm\sqrt{\frac{a}{b}}$ son puntos críticos *estables*.

Nota: *la demostración matemática formal de la estabilidad de los puntos críticos está disponible en el Anexo 1.*

Obtenida la información de estabilidad de los puntos críticos, estamos en condiciones de trazar el retrato de fase (Figura 3.2).

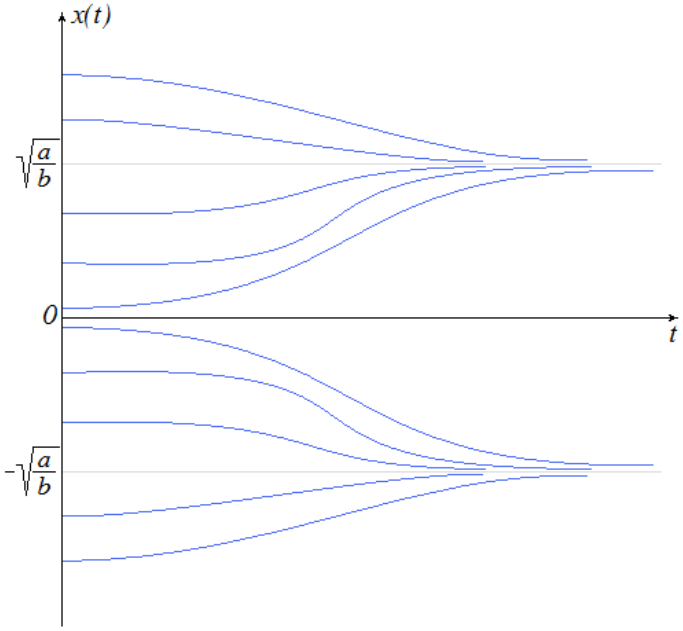



Figura 3.2. Retrato fase de la ecuación (3.2).

La ecuación diferencial hasta aquí analizada (ecuación (3.2)) describe la mecánica del sistema de dos estados que se utilizará para demostrar el fenómeno de resonancia estocástica, quedando señalado tras el análisis del plano de fase, que su salida se mantendrá alrededor de cualquiera de los dos puntos críticos estables, derivando de este hecho el nombre de *sistema biestable o sistema de dos estados*.

### 3.2.2 Análisis en presencia de señales externas

Una vez analizado el comportamiento del sistema sin estímulos externos, se procederá a insertar una señal a la entrada del mismo y analizar su funcionamiento.

El efecto que produce el tener una señal a la entrada del sistema se puede analizar al sumarle una constante en cada instante de tiempo, por lo que se optará por llevar a cabo el análisis de tal manera, con el fin de simplificar el cálculo. Sumamos una constante a la ecuación (3.2):

$$dx = \left(ax - bx^3 + C\right)dt$$
$$\frac{dx}{dt} = ax - bx^3 + C \qquad (3.8)$$

Nuevamente se buscarán los puntos críticos, esta vez con el fin de establecer ciertas condiciones que deberá cumplir la constante $C$ para incidir de manera determinante en el comportamiento del sistema.

Como se procedió anteriormente, se igualara la derivada temporal a cero:

$$\frac{dx}{dt} = 0$$
$$-bx^3 + ax + C = 0$$
$$-x^3 + \frac{a}{b}x + C = 0$$

Al igual que en el análisis de la ecuación (3.2) se despejan las raíces de la ecuación (3.8), para lo cual se utiliza Matlab. Las raíces obtenidas son las siguientes:

$$x_1 = \frac{1}{6}\sqrt[3]{108C + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C^2}} + \frac{2\left(\frac{a}{b}\right)}{\sqrt[3]{108C + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C^2}}}$$

$$x_{2,3} = -\frac{1}{12}\sqrt[3]{108C + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C^2}} - \frac{\frac{a}{b}}{\sqrt[3]{108C + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C^2}}}$$

$$\pm\frac{\sqrt{3}}{2}i\left(\frac{1}{6}\sqrt[3]{108C + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C^2}} - \frac{2\left(\frac{a}{b}\right)}{\sqrt[3]{108C + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C^2}}}\right)$$

Como paso siguiente se hace un análisis de las raíces obtenidas. Por inspección se puede observar que pueden existir ciertos valores de $C$ para los cuales las raíces $x_{2,3}$ pueden ser complejas conjugadas, reales e iguales y, reales y distintas. Obtendremos entonces el valor de $C$ que anula la parte imaginaria de las raíces $x_{2,3}$, las cuales expresaremos de la forma $x = \alpha \pm i\beta$. Para ello se utiliza Matlab y se obtiene la siguiente expresión:

$$x_{2,3} = \alpha + i\beta$$

$$\beta = 0$$

$$\Rightarrow \frac{1}{6}\sqrt[3]{108C_0 + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C_0^{\,2}}} - \frac{2\left(\frac{a}{b}\right)}{\sqrt[3]{108C_0 + 12\sqrt{-12\left(\frac{a}{b}\right)^3 + 81C_0^{\,2}}}} = 0$$

$$C_0 = \frac{2}{3\sqrt{3}}a\sqrt{\frac{a}{b}} \qquad (3.9)$$

Como podemos observar, $C_0$ es el valor de $C$ para el cual $\beta = 0$.

El efecto de $C$ en la ecuación (3.2) se puede observar gráficamente de manera sencilla. En efecto, esta constante hace que el gráfico del plano de fase de la ecuación (3.2) se desplace hacia arriba ($C > 0$, Figura 3.3) o hacia abajo ($C < 0$, Figura 3.4), observando también por inspección que al superar cierto valor (precisamente, $C_0$), la ecuación (3.2) pasa a tener un único punto crítico real.

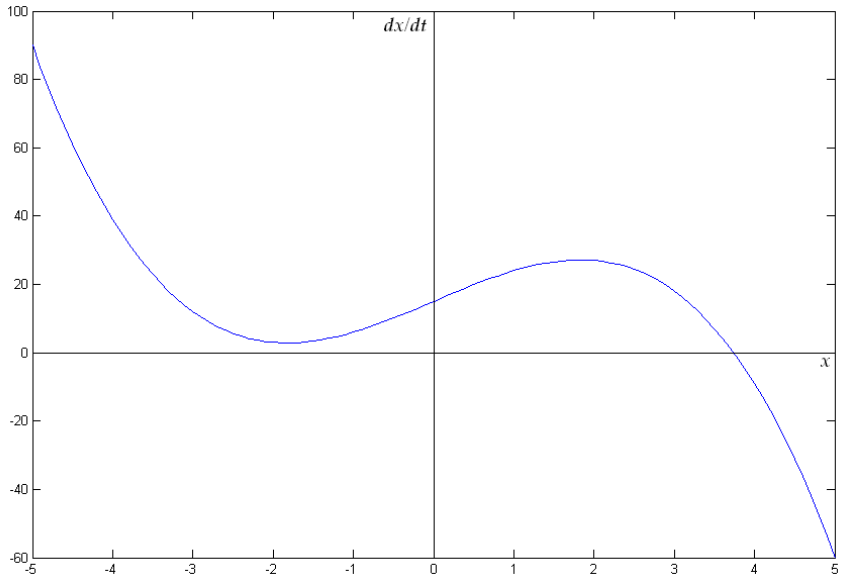



Figura 3.3. Plano fase de la ecuación (3.2) desplazado hacia arriba ($C > 0$ y $C > C_0$).

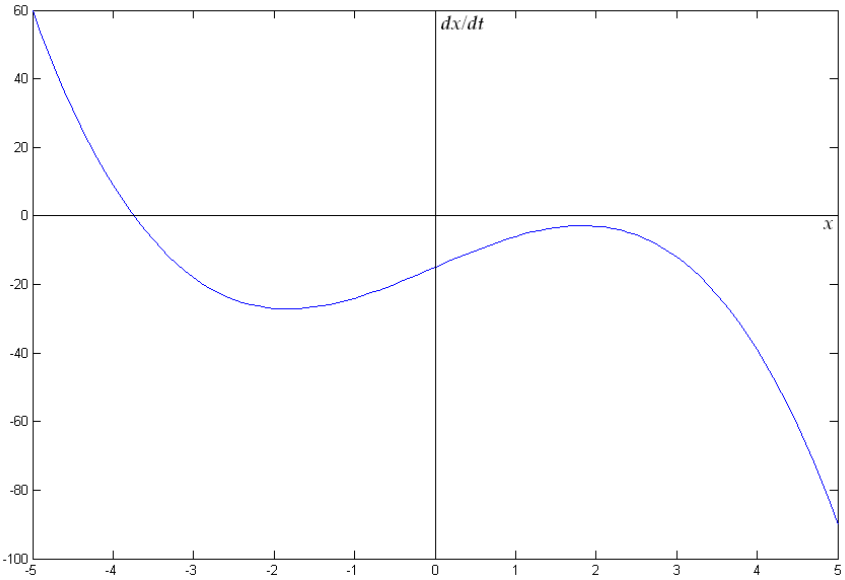



Figura 3.4. Plano fase de la ecuación (3.2) desplazado hacia abajo ( $C < 0$ y $C < C_0$ ).

La condición (3.9) es de gran importancia para el análisis de la mecánica de este sistema, ya que es la base para entender la razón por la cual, ante estímulos externos (con señales a la entrada del mismo) su salida permanece alrededor de un estado o el otro. Por lo tanto, dependiendo del valor de $C$ , como ya se menciono anteriormente, se puede tener tres casos diferentes:

- Para $-C_0 < C < C_0$ y $C \neq C_0$ , las raíces $x_{2,3}$ son reales y distintas, lo cual implica que el sistema tenga tres puntos críticos (Figura 3.5).

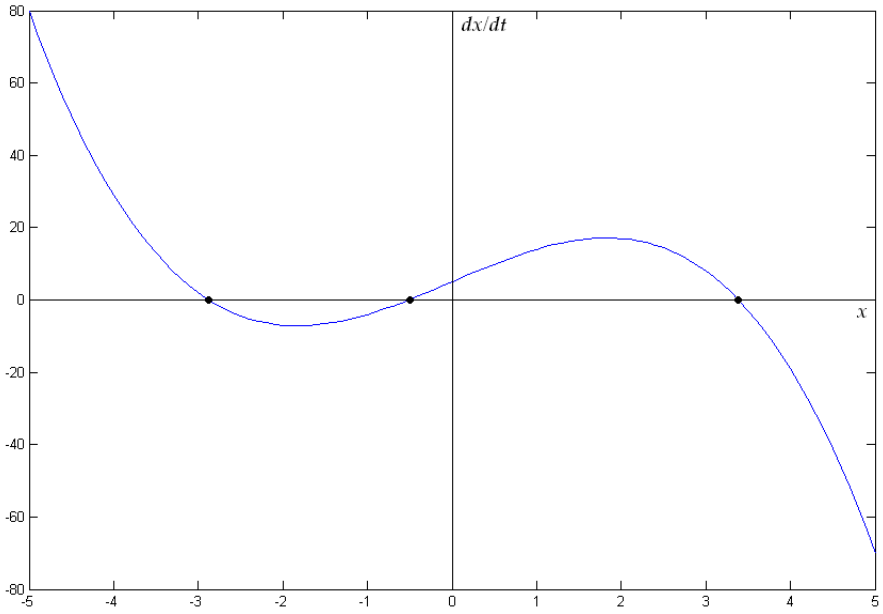



Figura 3.5. Con puntos negros se resaltan los tres puntos críticos (cruces por cero de la función).

- Para $C = C_0$, las raíces $x_{2,3}$ son reales iguales. Este es un caso crítico en el cual el sistema posee dos puntos críticos (Figura 3.6).

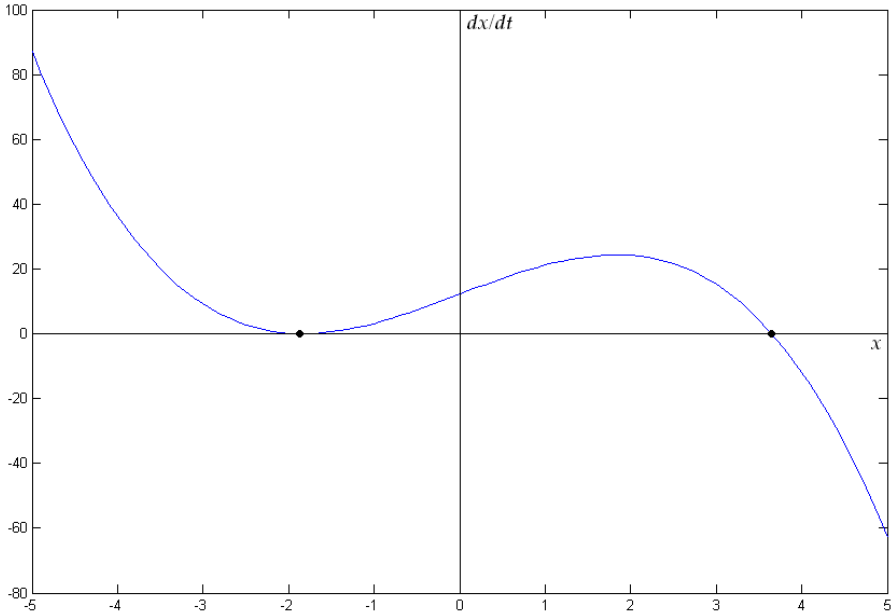



Figura 3.6. Con puntos negros se resaltan los dos puntos críticos (cruces por cero de la función).

- Para $C > C_0$ o $C < -C_0$, las raíces $x_{2,3}$ son complejas conjugadas, derivando en que el sistema tenga solo un punto crítico y estable (Figura 3.7).

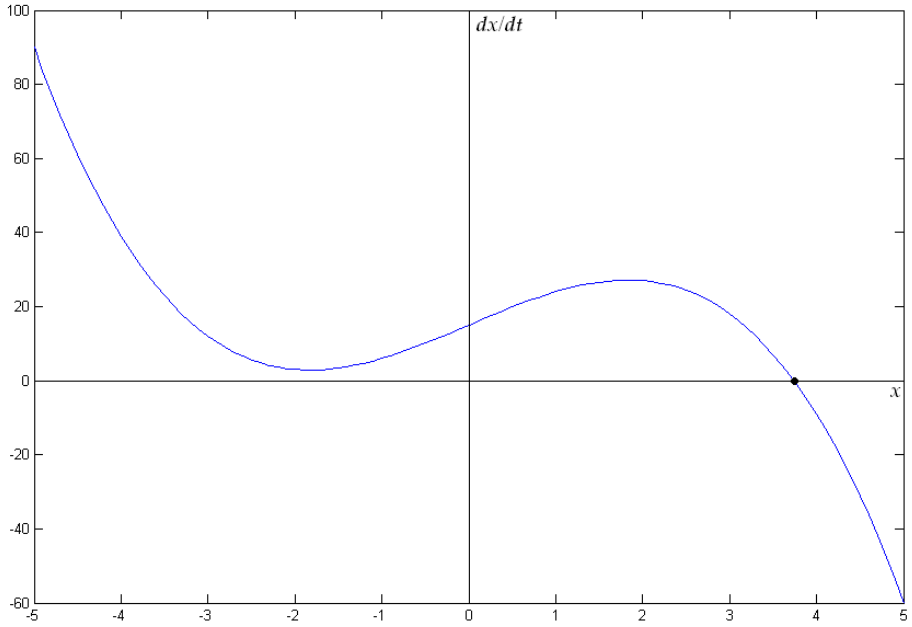



Figura 3.7. El punto negro nos señala el único punto crítico de la función.

Podemos concluir con este análisis cualitativo, que el sistema responde con cambios de estado (pasa de un estado estable al otro) ante estímulos externos, siempre y cuando estas señales aplicadas a su entrada tomen valores de amplitud para que el sistema tenga, en un determinado instante de tiempo, un único punto crítico. Por ejemplo, si a la entrada del sistema tenemos una señal periódica $A\cos(\Omega t)$, donde $A > C_0$, el plano fase se desplaza hacia arriba y abajo con una frecuencia $\Omega$, y para los instantes de tiempo que se cumpla $A > C_0$ (al menos para los máximos y mínimos de la señal, cada $n\pi$) el sistema tendrá alternadamente como único punto crítico alguno de sus puntos estables, derivando en cambios de estado con frecuencia también $\Omega$.

Analizada la mecánica del sistema de dos estados, estamos en condiciones de comenzar con el análisis del fenómeno de resonancia estocástica.

## 3.3 Demostración de la existencia del fenómeno de resonancia estocástica

Presentado el funcionamiento del sistema de dos estados, se prosigue con la demostración de la existencia de resonancia estocástica atacando dicho sistema con ruido y una señal periódica. Para el desarrollo de esta sección se analizó [2].

En primera instancia se busca demostrar la existencia de un pico en el espectro de potencias (existencia de resonancia estocástica) independizándonos de la dinámica del sistema. Para ello se asume conocida a la tasa de salto entre pozos (la cual depende de los parámetros de sistema). Cabe aclarar también que la variable dinámica (señal de salida del sistema, $x$) es tomada como discreta y es asumido que la frecuencia de la señal moduladora es menor a la inversa del tiempo propio de relajación del sistema $\tau_r$. Esto permite despreciar la existencia de transitorios provenientes de la dinámica intrínseca del mismo, llegándose entonces al equilibrio de probabilidad dentro del pozo (punto estable). Esta última condición es importante, ya que el hecho de que se alcance el equilibrio de probabilidad dentro del pozo nos garantiza que no se produzca un salto de estado en plena transición de un estado a otro, generando que la salida pierda la periodicidad que hace que esté presente un pico en el espectro de potencias, o sea, resuene.

Luego de este análisis, se procede establecer las condiciones que nos permiten adaptar y relacionar los resultados obtenidos con el análisis del sistema de potencial cuartico, el cual nos habilita a realizar el desarrollo del funcionamiento de resonancia estocástica en un sistema de dos estados tomando la variable dinámica $x$ como continua.

### 3.3.1 Variable dinámica discreta

Como se mencionó, se busca obtener un pico en el espectro de potencias independizándonos de la dinámica del sistema. Dado que se trata de un sistema biestable, los estados posibles para éste son sólo dos, a saber, + y -, siendo $x_-$ y $x_+$ los valores que la variable dinámica adopta en dichos estados. Designaremos a la probabilidad de que $x$ esté en el estado + o – como sigue:

$$n_{\pm} = prob(x = x_{\pm})$$

$$n_- = 1 - n_+ = \int_{-\infty}^{x'} p(x)dx$$

Donde $p(x)$ es la función densidad de probabilidad.

En virtud de las hipótesis asumidas, definimos la función densidad de probabilidad $p(x)$ de la siguiente manera:

$$p(x,t) = n_+(t)\delta(x - x_+) + n_-(t)\delta(x - x_-)$$

De esta manera la variable $x$ esta sólo en un estado o el otro, como se mencionó, es tomada como discreta. Por simplicidad, asumiremos que el sistema tiene las siguientes características:

- Simétrico respecto $x = 0$.
- $x_+ = -x_- = c$.

La varianza para el sistema no modulado (sin señal periódica) resulta entonces:

$$n_+ = n_- = \frac{1}{2}$$

$$\left\langle x^2 \right\rangle = \int_{-\infty}^{\infty} x^2 p(x)dx = x_+^2 n_+ + x_-^2 n_-$$

$$\left\langle x^2 \right\rangle = c^2$$

Otra magnitud que introduciremos es la tasa de transición de un estado a otro, $H(t)$, la cual tiene una componente periódica correspondiente a la señal moduladora del sistema actuando en conjunto con el ruido.

Se inicia el desarrollo planteando una ecuación diferencial para la probabilidad $n(t)$, la cual indica que la tasa de cambio de la probabilidad que el sistema este en el estado + ( $n_+$ ) es igual a la diferencia entre la tasa de *ingreso* al estado + ( $H_-(t)n_-$ ) y la tasa de *egreso* del estado – ( $H_+(t)n_+$ ):

$$\frac{dn_+}{dt} = -\frac{dn_-}{dt} = H_-(t)n_- - H_+(t)n_+ = H_-(t)(1 - n_+) - H_+(t)n_+$$

$$\frac{dn_+}{dt} = H_-(t) - \left[H_-(t) + H_+(t)\right]n_+ \qquad (3.10)$$

De esta manera obtenemos una ecuación diferencial de primer grado con coeficientes periódicos (ecuación (3.10)) para $n_+$, cuya solución, como es bien conocida, se puede escribir en la forma:

$$n_+(t\,|\,x_0, t_0) = g^{-1}(t)\left[ n_+(t_0)g(t_0) + \int_{t_0}^{t} H_-(t')g(t')dt' \right] \qquad (3.11)$$

Donde $n_+(t|x_0,t_0)$ es la probabilidad condicional de que el sistema ocupe el estado + al tiempo $t$, dado que en el tiempo $t_0$ ocupaba el estado $x_0$; y:

$$g(t) = e^{\int_{t0}^{t}[H_+(t')+H_-(t')]dt'}$$

Como ya se mencionó anteriormente, la tasa de transición $H(t)$ depende de parámetros que se ven afectados por el ruido y la señal periódica; proponemos entonces el siguiente modelo paramétrico para $H(t)$:

$$H_{\pm}(t) = f(\mu \pm \eta_0 \cos(\omega_{tr} t))$$

Donde:

- $\mu$: es un parámetro que relaciona la barrera de potencial que se debe superar para que se produzca un cambio de estado (lo cual está relacionado con la condición (3.9)) y el ruido.
- $\eta_0$: representa la intensidad con la que la señal periódica modula al parámetro $\mu$.
- $\omega_{tr}$: es la frecuencia en radianes por segundo de la señal moduladora.

Esta expresión para $H_{\pm}(t)$ es del tipo del modelo de Kramer para $\eta_0 = 0$; en nuestro caso tomaremos $\eta_0 \ll \mu$ para describir los pequeños apartamientos que representará la inclusión de la señal en una tasa de transición afectada sólo por el ruido aleatorio. Asumiendo que $f$ es desarrollable en serie de potencias alrededor de $\mu$, podemos escribir:

$$H_{\pm}(t) = \frac{1}{2}\left[\alpha_0 \mp \alpha_1 \eta_0 \cos(\omega_{tr} t) + \alpha_2 \eta_0^2 \cos^2(\omega_{tr} t) \mp \ldots\right]$$

Donde:

$$\frac{1}{2}\alpha_0 = f(\mu)$$

$$\frac{1}{2}\alpha_n = \frac{(-1)^n}{n!}\frac{d^n f(\mu)}{d\eta^n}$$

Por simplicidad y dado que $\eta_0 \ll \mu$ tomamos sólo los dos primeros términos de la serie:

$$H_{\pm}(t) = \frac{1}{2}\left[\alpha_0 \mp \alpha_1 \eta_0 \cos(\omega_{tr} t)\right]$$
$$H_{+}(t) + H_{-}(t) = \alpha_0$$

Reemplazamos en (3.11) y resolvemos:

$$g(t) = e^{\int_{t_0}^{t} \alpha_0 dt'}$$
$$\Rightarrow g(t) = e^{\alpha_0 (t - t_0)}$$

$$g(t_0) = 1$$
$$n_{+}(t_0) = \delta_{x_0 c} = \begin{cases} 1 \rightarrow x_0 = c \\ 0 \rightarrow x_0 = -c \end{cases}$$
$$n_{+}(t \,|\, x_0, t_0) = e^{-\alpha_0 (t - t_0)} \left[ \delta_{x_0 c} + \frac{1}{2} \int_{t_0}^{t} \left[\alpha_0 + \alpha_1 \eta_0 \cos(\omega_{tr} t')\right] e^{\alpha_0 (t' - t_0)} dt' \right]$$
$$n_{+}(t \,|\, x_0, t_0) = \frac{1}{2} e^{-\alpha_0 (t - t_0)} \left[ 2\delta_{x_0 c} + \int_{t_0}^{t} \alpha_0 e^{\alpha_0 (t' - t_0)} dt' + \int_{t_0}^{t} \alpha_1 \eta_0 e^{\alpha_0 (t' - t_0)} \cos(\omega_{tr} t') dt' \right]$$

Teniendo en cuenta que:

$$\int e^{au} \cos(bu)\, du = \frac{e^{au}}{a^2 + b^2} \left(a \cos(bu) + b \sin(bu)\right) + C$$

$$a \sin x + b \cos x = \sqrt{a^2 + b^2} \sin\left( x + \arctan\left(\frac{b}{a}\right)\right)$$

La solución para la probabilidad condicional de que $x = x_{+}$ dado $t_0$ y $x_0$ (teniendo en cuenta que este último debe ser $+c$ o $-c$) nos queda de la siguiente manera:

$$n_{+}(t \,|\, x_0, t_0) = \frac{1}{2} \left[ e^{-\alpha_0 (t - t_0)} \left( 2\delta_{x_0 c} - 1 - \frac{\alpha_1 \eta_0 \cos\left(\omega_{tr} t_0 - \phi\right)}{\sqrt{\alpha_0^2 + \omega_{tr}^2}} \right) + 1 + \frac{\alpha_1 \eta_0 \cos\left(\omega_{tr} t_0 - \phi\right)}{\sqrt{\alpha_0^2 + \omega_{tr}^2}} \right] \quad (3.12)$$

Donde:

$$\phi = \arctan\left(\frac{\omega_{tr}}{\alpha_0}\right)$$

Dado lo propuesto al principio de esta sección, podemos expresar $x(t)$ en función de $n_{\pm}$, sabiendo que se tomó como discreta. Entonces planteamos lo siguiente:

$$x(t) = cn_{+}(t|x_0,t_0) - cn_{-}(t|x_0,t_0)$$
$$x(t+\tau) = cn_{+}(t+\tau|x_0,t_0) - cn_{-}(t+\tau|x_0,t_0)$$

Esto nos indica que la señal tomará únicamente los valores $\pm c$ con una probabilidad condicional $n_{\pm}(t|x_0,t_0)$.

Obtenida la expresión para $x(t)$ en función de $n_{\pm}$, buscamos la función de auto-correlación, cuya transformada de Fourier nos permitirá obtener una expresión para el espectro de potencias.

Planteamos entonces la función de auto-correlación:

$$\langle x(t)x(t+\tau)|x_0,t_0\rangle = +c^2 n_{+}\left(t+\tau|+c,t\right)n_{+}\left(t|x_0,t_0\right) - c^2 n_{+}\left(t+\tau|-c,t\right)n_{-}\left(t|x_0,t_0\right)$$
$$-c^2 n_{-}\left(t+\tau|+c,t\right)n_{+}\left(t|x_0,t_0\right) - c^2 n_{-}\left(t+\tau|-c,t\right)n_{-}\left(t|x_0,t_0\right)$$

Recordando que $n_{-}(t) = 1 - n_{+}(t)$, podemos expresar esta última función de $n_{+}\left(t|x_0,t_0\right)$ (que fue calculada anteriormente):

$$\langle x(t)x(t+\tau)|x_0,t_0\rangle = c^2\left[2n_{+}\left(t+\tau|+c,t\right)-1+2n_{+}\left(t+\tau|-c,t\right)-1\right]n_{+}\left(t|x_0,t_0\right) - \left[2n_{+}\left(t+\tau|-c,t\right)-1\right] \quad (3.13)$$

En virtud de las hipótesis asumidas, el sistema se encuentra en condiciones estacionarias, lo cual simplifica enormemente el cálculo, puesto que entonces debemos tomar límite para $t_0 \to -\infty$.

Reemplazando (3.12) en (3.13) y tomando el límite antes mencionado, obtenemos lo siguiente:

$$\lim_{t_0\to-\infty}\left[\langle x(t)x(t+\tau)|x_0,t_0\rangle\right] = c^2 e^{-\alpha_0|\tau|}\left[1 - \frac{\alpha_1^2\eta_0^2\cos^2\left(\omega_{tr}t-\phi\right)}{\alpha_0^2+\omega_{tr}^2}\right] + \frac{c^2\alpha_1^2\eta_0^2\left\{\cos\left(\omega_{tr}\tau\right)+\cos\left[\omega_{tr}\left(2t+\tau\right)+2\phi\right]\right\}}{\alpha_0^2+\omega_{tr}^2}$$

Entonces:

$$\langle x(t)x(t+\tau)\rangle = c^2 e^{-\alpha_0|\tau|}\left[1-\frac{\alpha_1^2\eta_0^2\cos^2(\omega_{tr}t-\phi)}{\alpha_0^2+\omega_{tr}^2}\right]+\frac{c^2\alpha_1^2\eta_0^2\left\{\cos(\omega_{tr}\tau)+\cos\left[\omega_{tr}(2t+\tau)+2\phi\right]\right\}}{\alpha_0^2+\omega_{tr}^2}$$

Donde en esta última expresión hemos llamado:

$$\lim_{t_0\to-\infty}\langle x(t)x(t+\tau)|x_0,t_0\rangle = \langle x(t)x(t+\tau)\rangle$$

El espectro de potencias de $x(t)$ que deseamos calcular, es función tanto de $\Omega$ como de $t$. Destacable es su dependencia con $t$ ya que, por ejemplo, en una simulación de este sistema, $t$ representa el tiempo en el cual se comienzan a tomar los datos (muestreo de la señal). Podemos considerar a dicha variable como aleatoria uniformemente distribuida en un periodo de la señal, ya que se puede comenzar el muestreo en cualquier instante de tiempo dentro de dicho periodo. Sabiendo que el espectro de potencia se obtiene tomando la transformada de Fourier de la función de auto-correlación, podemos promediar en $t$ a lo largo de un periodo de la señal y luego obtener la transformada, llegando así al mismo resultado que transformando y luego promediando.

Promediamos entonces a los largo de un periodo de la señal moduladora:

$$\langle\langle x(t)x(t+\tau)\rangle\rangle_t = \frac{\omega_{tr}}{2\pi}\int_0^{2\pi/\omega_{tr}}\langle x(t)x(t+\tau)\rangle dt$$

Obteniendo la siguiente expresión:

$$\langle\langle x(t)x(t+\tau)\rangle\rangle_t = c^2 e^{-\alpha_0|\tau|}\left[1-\frac{\alpha_1^2\eta_0^2}{2\left(\alpha_0^2+\omega_{tr}^2\right)}\right]+\frac{c^2\alpha_1^2\eta_0^2\cos(\omega_{tr}\tau)}{2\left(\alpha_0^2+\omega_{tr}^2\right)}$$

A continuación, aplicamos transformada de Fourier a la función de auto-correlación promediada en $t$:

$$\mathbb{F}\left\{\langle\langle x(t)x(t+\tau)\rangle\rangle_t\right\} = \langle S(\Omega)\rangle_t = \frac{\omega_{tr}}{2\pi}\int_{-\infty}^{\infty}\langle\langle x(t)x(t+\tau)\rangle\rangle_t e^{-i\Omega\tau}d\tau$$

Y se obtiene:

$$\langle S(\Omega)\rangle_t = \frac{2c^2\alpha_0}{\alpha_0^2+\Omega^2}\left[1-\frac{\alpha_1^2\eta_0^2}{2\left(\alpha_0^2+\omega_{tr}^2\right)}\right]+\frac{\pi c^2\alpha_1^2\eta_0^2}{2\left(\alpha_0^2+\omega_{tr}^2\right)}\left[\delta(\Omega-\omega_{tr})+\delta(\Omega+\omega_{tr})\right]$$

Se usará la notación $S(\Omega)$ para referirnos sólo a la parte del espectro promediado en *t* para frecuencias positivas, entonces:

$$S(\Omega) = \langle S(\Omega) \rangle_t + \langle S(-\Omega) \rangle_t$$

$$S(\Omega) = \frac{4c^2\alpha_0}{\alpha_0^2 + \Omega^2}\left[1 - \frac{\alpha_1^2\eta_0^2}{2\left(\alpha_0^2 + \omega_{tr}^2\right)}\right] + \frac{\pi c^2 \alpha_1^2 \eta_0^2}{\alpha_0^2 + \omega_{tr}^2}\delta\left(\Omega - \omega_{tr}\right) \qquad (3.14)$$

Donde $\delta$ es la conocida distribución de Dirac. Esta es entonces la expresión para el espectro de potencias de $x(t)$. En ella se reconocen básicamente dos términos. El primero de ellos representa al ruido presente en el sistema, el cual tiene la forma de la función $h(x) = \frac{a}{a^2 + x^2}$ (Figura 3.8) centrada en $\Omega = 0$ ante la ausencia de señal moduladora ($\eta_0 = 0$), multiplicado por un factor de corrección que representa el efecto de la señal sobre éste. El segundo término es el de interés, ya que nos indica la existencia de un pico (función $\delta$) para $\Omega = \omega_{tr}$ en el espectro de potencias de la señal de salida del sistema, lo cual señala un comportamiento periódico de $x(t)$, *quedando demostrado así el efecto de resonancia estocástica* para este sistema.

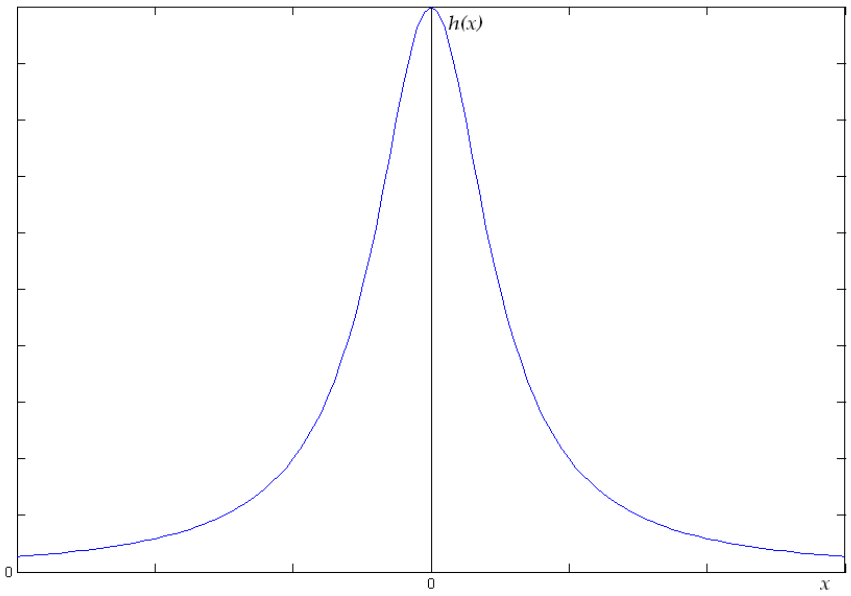



Figura 3.8. Función $h(x)$.

### 3.3.2 Variable dinámica continua. Resonancia estocástica en sistema de dos estados

El sistema de potencial cuártico será utilizado en esta sección ya que es apropiado para representar el comportamiento de un sistema de dos estados (biestable), cuando el sistema es del tipo variable continua. En este caso, la variable dinámica que caracteriza el estado del sistema será entonces continua, y la designaremos $x(t)$. En presencia de una señal moduladora, el potencial cuártico $U(x,t)$ está gobernado por la siguiente expresión:

$$U(x,t) = -\frac{a}{2}x^2 + \frac{b}{4}x^4 - Ax\cos(\omega_{tr}t) \qquad (3.15)$$

Donde $A$ y $\omega_{tr}$ son la amplitud y frecuencia respectivamente de la señal moduladora.

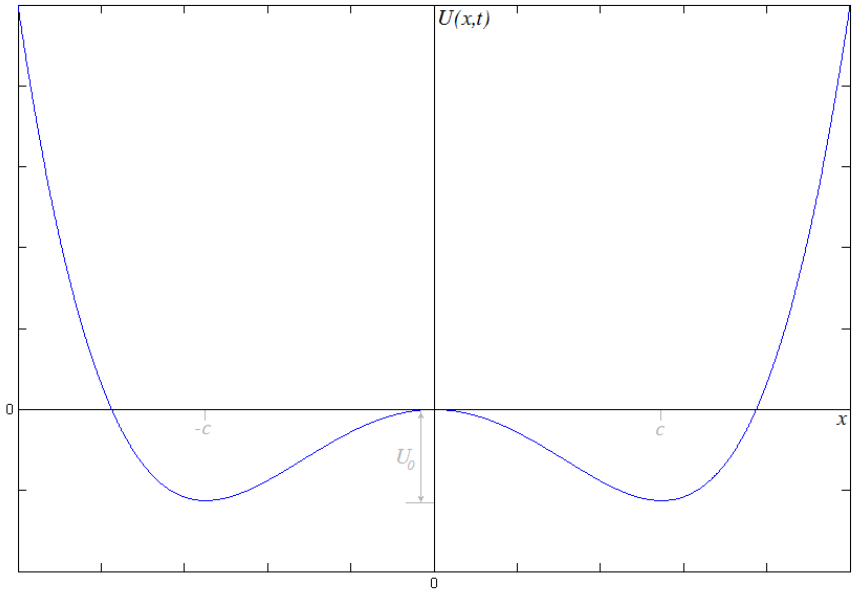



Figura 3.9. Gráfico del potencial cuartico $U(x,t)$ en ausencia de la señal moduladora ( $A=0$ ).

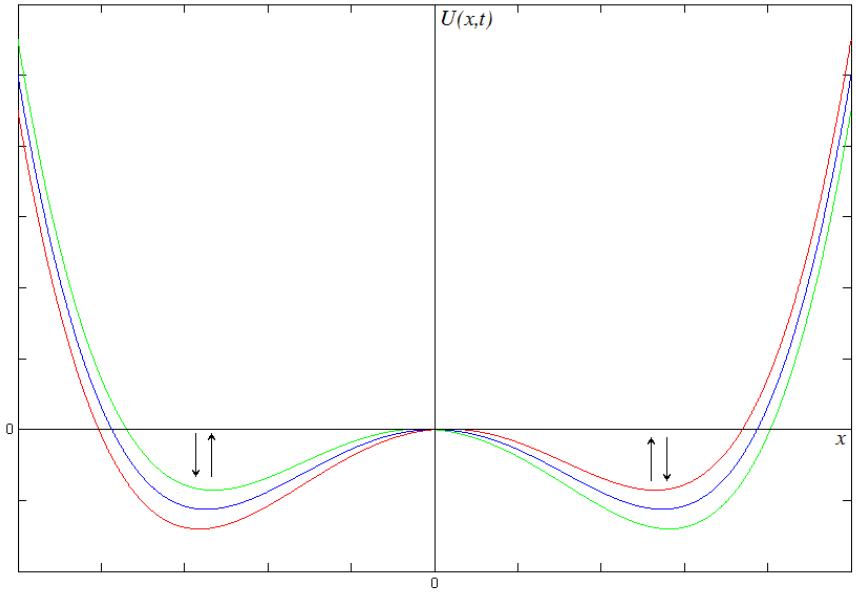



Figura 3.10. Gráfico del potencial cuartico $U(x,t)$ en presencia de la señal moduladora ( $A \neq 0$ ). Aquí se observa como la señal deforma los pozos de la función de potencial.

En la Figura 3.9 tenemos el gráfico de la función $U(x,t)$ en ausencia de la señal moduladora. En gris se señala los valores de $x$ donde aparecen los mínimos ( $\pm c$ ) y la barrera de potencial $U_0$. En la Figura 3.10 vemos el efecto de la señal moduladora sobre el potencial, deformando sus pozos y desplazando los mínimos hacia arriba y abajo.

A continuación se buscará obtener expresiones para la altura de la barrera de potencial, ubicación de los mínimos y amplitud de la modulación de la altura de la barrera.

Expresamos (3.15) de la siguiente manera:

$$U(x,t) = U_0\left[-2\left(\frac{x}{c}\right)^2 + \left(\frac{x}{c}\right)^4\right] - U_1\left(\frac{x}{c}\right)\cos(\omega_{tr}t) \qquad (3.16)$$

Donde:

- $U_0 = \frac{a^2}{4b}$ ; es la altura de la barrera de potencial.
- $\pm c = \pm\sqrt{\frac{a}{b}}$ ; es la ubicación en $x$ de los mínimos sin modulación ( $A = 0$ ).

- $U_1 = A\sqrt{\frac{a}{b}} = Ac$; cuando $A \neq 0$, es la amplitud de la modulación de la altura de la barrera.

Presentaremos a continuación la ecuación que rige el movimiento de una partícula en dicho potencial, sometida a un forzamiento aleatorio, lo cual representa el ruido que debe estar presente en todo el sistema que exhiba resonancia estocástica.

$$\dot{x} = -\partial_x U(x,t) + D\eta(t) \qquad (3.17)$$

Donde, como ya se ha mencionado anteriormente, $\eta(t)$ representa al ruido Gaussiano (ruido blanco con distribución Gaussiana).

Resolviendo (3.17) obtenemos lo siguiente:

$$\frac{dx}{dt} = ax - bx^3 + A\cos(\omega_{tr} t) + D\eta(t)$$

Expresión que gobierna la respuesta de un sistema de dos estados cuyas entradas son ruido Gaussiano y una señal periódica (señal moduladora).

Para relacionar lo desarrollado en la sección anterior con el sistema de dos estados presentado a través del potencial cuártico, se busca encontrar una expresión para la tasa de salto del sistema sin modulación (en ausencia de la señal periódica). Para ello se utilizará el tiempo de Kramer $\tau_K$, el cual indica el tiempo medio para el cual se produce el primer salto, estando el sistema sometido a un forzamiento aleatorio. Este tiempo está dado por la siguiente ecuación:

$$\tau_K = H_K^{-1} = \frac{2\pi e^{2U_0/D}}{\sqrt{|U''(0)|U''(c)}}$$

Resolviendo nos queda lo siguiente:

$$\tau_K = H_K^{-1} = \frac{\sqrt{2}\pi}{a} e^{2U_0/D} \qquad (3.18)$$

Como se puede observar, el tiempo de Kramer depende solo de la altura de la barrera, y de la curvatura en los máximos y mínimos de la función de potencial. Además, en el cálculo del tiempo de Kramer se asume que la densidad de probabilidad dentro de cada pozo está esencialmente en equilibrio, o sea, posee una distribución Gaussiana centrada en el mínimo. En vista a esto, la expresión de la tasa de Kramer en (3.18) puede ser generalizada para incluir la perturbación de la señal de la siguiente manera:

$$H_{\pm}(t) = \frac{a}{\sqrt{2\pi}} e^{-2(U_0 \pm U_1 \cos \omega_{tr} t)/D},$$

para lo cual será necesario que la frecuencia de la señal sea mucho menor que una tasa característica (para mantener el equilibrio de probabilidad dentro del pozo). Dicha tasa está dada justo por la curvatura en el mínimo dentro del pozo, por lo que la aproximación anterior es válida bajo la siguiente condición:

$$\omega_{tr} \ll U''(\pm c) = 2a$$

Ahora estamos en condiciones de comparar la tasa de salto planteada en la sección anterior, con la obtenida en ésta, con el fin de obtener una expresión para dicha tasa en función de los parámetros del sistema de dos estados.

Recordamos la expresión propuesta en la sección anterior:

$$H_{\pm}(t) = f(\mu \pm \eta_0 \cos(\omega_{tr} t)) \qquad (3.19)$$

Nada nos impide asumir $f(u) = e^{-2u}$ y proponer las siguientes igualdades:

- $\mu = \frac{U_0}{D}$
- $\eta_0 = \frac{U_1}{D} = \frac{Ac}{D}$

Ya que de esta manera, estaríamos estableciendo que la tasa de salto del sistema ante una modulación periódica sigue la forma de la expresión obtenida a partir del tiempo de Kramer (recordando las hipótesis propuestas tras generalizar (3.18) para incluir la perturbación de la señal moduladora), lo que nos posibilita expresar dicha tasa en función de los parámetros de sistema.

Tras lo expuesto, tenemos que:

$$f(\mu \pm \eta_0 \cos \omega_{tr} t) = \frac{a}{\sqrt{2\pi}} e^{-2(\mu \pm \eta_0 \cos \omega_{tr} t)}$$

Planteada esta analogía, podemos dejar la expresión obtenida para $S(\Omega)$ también en función de los parámetros del sistema de dos estados. Para ello debemos reemplazar $\alpha_0$ y $\alpha_1$ por las siguientes expresiones:

- $\alpha_0 = 2f(\eta = 0) = \frac{\sqrt{2}a}{\pi} e^{-2U_0/D}$

- $\alpha_1 = -2\frac{df}{d\eta}(\eta = 0) = 2\frac{\sqrt{2}a}{\pi} e^{-2U_0/D} = 2\alpha_0$

Reemplazamos en la ecuación de $S(\Omega)$:

$$S(\Omega) = \frac{\frac{4\sqrt{2}c^2 a}{\pi} e^{-2U_0/D}}{\frac{2a}{\pi^2} e^{-4U_0/D} + \Omega^2} \left[ 1 - \frac{\frac{4a^2A^2c^2}{\pi^2 D^2} e^{-4U_0/D}}{\frac{2a}{\pi^2} e^{-4U_0/D} + \omega_{tr}^2} \right] + \frac{\frac{8a^2A^2c^4}{\pi D^2} e^{-4U_0/D}}{\frac{2a}{\pi^2} e^{-4U_0/D} + \omega_{tr}^2} \delta(\Omega - \omega_{tr}) \qquad (3.20)$$

Habiendo dejado $S(\Omega)$ en función de los parámetros del sistema, podemos despejar la relación señal-ruido ($SNR$), integrando en todo el espectro el término correspondiente a la señal (segundo término de la ecuación (3.20)) y dividiéndolo por la integral en todo el espectro de la ecuación (3.20). De esa manera se obtiene la siguiente expresión para $SNR$:

$$SNR = \frac{\sqrt{2}aA^2c^2}{D^2} e^{-2U_0/D} \left[ 1 - \frac{\frac{4a^2A^2c^2}{\pi^2 D^2} e^{-4U_0/D}}{\frac{2a}{\pi^2} e^{-4U_0/D} + \omega_{tr}^2} \right]^{-1} \qquad (3.21)$$

Notar que el segundo factor en (3.21) representa una fracción de la potencia total en todo el espectro de ruido. Lo que se le sustrae a 1, es la parte coherente de la señal de salida. Teniendo en cuenta que la señal moduladora tiene solo una pequeña fracción de la potencia total, la expresión para la $SNR$ se puede aproximar y simplificar de la siguiente manera:

$$SNR \approx \frac{\sqrt{2}aA^2c^2}{D^2} e^{-2U_0/D} \qquad (3.22)$$

A continuación se hará un análisis gráfico de la expresión obtenida para la relación señal-ruido. Para ello tomaremos el ejemplo citado en el trabajo de R. Benzi, A. Sutera y A. Vulpiani. “The mechanism of stochastic resonance”. 1981.

Valores utilizados:

$$a = 1$$
$$b = 1$$
$$A = 0.12[V]$$
$$\omega_{tr} = 10^{-3}\left[\frac{rad}{seg}\right]$$
$$D = 0.25$$
$$U_0 = \frac{a^2}{4b} = 0.25$$

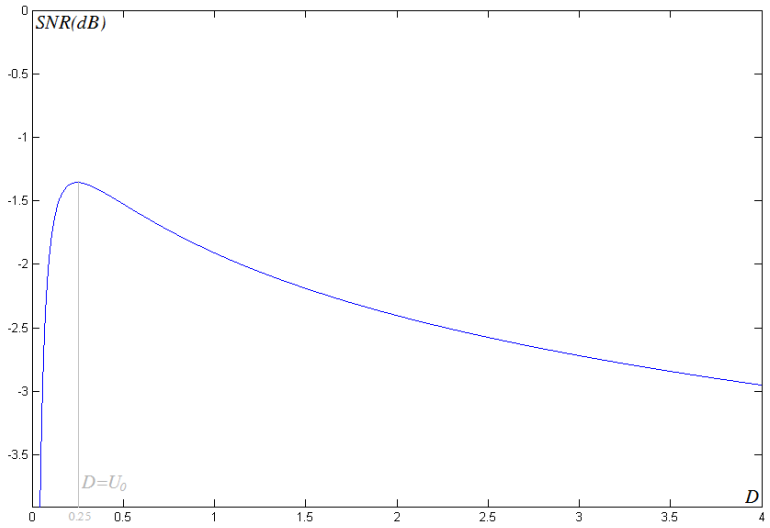



Figura 3.11: $SNR(dB)$ (relación señal-ruido expresada en decibeles) vs. $D$ (dispersión del ruido en voltios).

Como se aprecia en la Figura 3.11, la relación señal-ruido presenta un máximo para $D = U_0$. Se puede comprobar analíticamente la existencia de un máximo para $D = U_0$ de manera sencilla:

$$\frac{\partial SNR}{\partial D} = \frac{\partial\left(\frac{\sqrt{2}aA^2c^2}{D^2}e^{-2U_0/D}\right)}{\partial D} = \sqrt{2}aA^2c^2\left(\frac{2U_0e^{-2U_0/D} - 2De^{-2U_0/D}}{D^4}\right)$$

$$\frac{\partial SNR}{\partial D} = 0 \Rightarrow D = U_0$$

Observando el gráfico de la Figura 3.11, notamos que la relación señal-ruido alcanza un valor máximo cuando $D$ (dispersión del ruido) se acerca al valor $U_0$ (altura de la barrera de

potencial) demostrándose claramente la manifestación del fenómeno de resonancia estocástica, el cual recordamos, consistía en la maximización de $SNR$ sintonizando ciertos parámetros (como la dispersión del ruido) cerca de determinado valor (el cual debe ser $U_0$). De esta manera queda expuesta la existencia de dicho fenómeno, siendo primordial que la dispersión del ruido alcance el valor señalado para obtener una plena cooperación entre la señal moduladora y el ruido aleatorio.

Notemos también, que la condición para la frecuencia de la señal $\omega_{tr} \ll U''(\pm c) = 2a$, se cumple de manera trivial, dado que:

$$a = 1$$
$$2a = 2$$
$$\omega_{tr} \ll 2$$

### 3.3.3 Optimización de sistema

Analizada la teoría del método de resonancia estocástica y presentado el sistema de dos estados, en el cual se manifiesta este método, se procede a establecer expresiones para una serie de parámetros a partir de los cuales se desempeñará este sistema con vistas a la aplicación en la cual se basa este trabajo final.

Se busca obtener una seria de expresiones que relacionen los parámetros del sistema con el fin de poder configurar el mismo para que trabaje de la forma deseada. Teniendo en cuenta que la intención es enviar información enmascarada con ruido, o en nuestro caso con audio, se deberá trabajar con una relación señal-ruido de entrada al sistema baja, de manera tal que un filtro convencional tenga dificultades para recuperar la señal que contiene la información. También es importante definir la frecuencia a la cual se trabajará (frecuencia de la señal moduladora), ya que de esto dependerá la cantidad de información (bits por segundo) que se podrán enviar, y en base a este parámetro se configura el sistema.

El análisis comienza estableciendo ciertos requerimientos que se pretenden al sistema, tales como una relación señal-ruido de entrada y una frecuencia de trabajo. Definimos entonces:

- Relación señal-ruido de entrada: $\frac{S_i}{N_i}$
- Frecuencia de trabajo: $f_{tr}$

Renombraremos la expresión de relación señal-ruido de salida del sistema obtenida en la sección anterior de la siguiente manera:

$$SNR = \frac{S_o}{N_o}$$

Definimos la figura de ruido del sistema:

$$F = \frac{S_o / N_o}{S_i / N_i}$$

Lo que se busca es maximizar $F$ con el fin de obtener la mayor relación posible entre la relación señal-ruido de entrada y la de salida (optimizar) "sin importar" la $\frac{S_i}{N_i}$, lo que posibilitará maximizar la ganancia del sistema.

Buscando la expresión para la relación señal-ruido de entrada, tenemos que:

- La energía de un pulso cuadrado está dada por:

$$x_M(t) = \begin{cases} A\prod\left(\frac{t}{T_b}\right), a_k = '1' \\ 0, a_k = '0' \end{cases}$$

$$E_b = 0.5E_{'1'} + 0.5E_{'0'}$$

$$E_{'1'} = \int_{-T_b/2}^{T_b/2} \left|x_M(t)\right|^2 \partial t = A^2 T_b$$

$$E_{'0'} = 0$$

$$E_b = \frac{A^2 T_b}{2}$$

Donde llamamos $x_M(t)$ a la señal moduladora, $E_b$ es la energía de bit.

Ahora calculamos la potencia en un periodo:

$$T_{tr} = T_b$$

$$S_i = \frac{E_b}{T_{tr}} = \frac{A^2}{2}$$

La potencia del ruido blanco con distribución Gaussiana es la siguiente:

Tenemos que el ruido Gaussiano tiene $\langle \eta \rangle = 0$ y $\langle \eta^2 \rangle = D^2$. Su función de auto-correlación $R_{\eta\eta}$, como es sabido, está dada por una función delta:

$$R_{\eta\eta}(\Delta) = D^2 \delta(\Delta)$$

Para obtener la potencia del ruido blanco, se debe aplicar transformada de Fourier a la función de auto-correlación:

$$N_i = \mathbb{F}\left\{R_{\eta\eta}(\Delta)\right\} = \mathbb{F}\left\{D^2\delta(\Delta)\right\} = D^2$$

Ahora se hace el cociente entre $S_i$ y $N_i$ para obtener la expresión para la relación señal-ruido de entrada:

$$S_i = \frac{A^2}{2}$$
$$N_i = D^2$$
$$\Rightarrow \frac{S_i}{N_i} = \frac{A^2}{2D^2}$$

Donde $A$ es la amplitud de la señal moduladora y $D$ la dispersión de ruido.

- La relación señal-ruido de salida del sistema se obtuvo anteriormente y está dada por la ecuación (3.22):

$$\frac{S_o}{N_o} = \frac{\sqrt{2}aA^2c^2}{D^2} e^{-2U_0/D}$$
$$c = \sqrt{\frac{a}{b}}$$
$$\Rightarrow \frac{S_o}{N_o} = \frac{\sqrt{2}a^2A^2}{bD^2} e^{-2U_0/D}$$

Por lo tanto $F$ nos queda de la siguiente manera:

$$F = \frac{2D^2}{A^2} \times \frac{\sqrt{2}a^2A^2}{bD^2} e^{-2U_0/D}$$

$$F = \frac{2\sqrt{2}a^2}{b} e^{-2U_0/D}$$

$$U_0 = \frac{a^2}{4b}$$

$$\Rightarrow F = 8\sqrt{2}U_0 e^{-2U_0/D}$$

A continuación se busca para que valor de $U_0$ se hace máximo $F$ ( $F = F_M$ ):

$$\partial_{U_0} F = 8\sqrt{2}\left( U_0 e^{-2U_0/D} - \frac{2}{D} U_0 e^{-2U_0/D} \right)$$

$$\partial_{U_0} F = 0$$

$$\Rightarrow U_0 = \frac{D}{2} \rightarrow F = F_M$$

Podemos observar que $D$ (dispersión de ruido) debe duplicar a $U_0$ (altura de la barrera de potencial) para obtener una figura de ruido máxima del sistema. Esto se observa en la igualdad (3.23).

$$D = 2U_0 \qquad (3.23)$$

Obtenida esta relación, expresamos uno de los requerimientos que vamos a tener con el sistema, el cual consiste en proponer una relación señal-ruido de entrada lo suficientemente baja para hacer obsoleto a cualquier filtro convencional o al menos dificultar su desempeño. Se plantea lo siguiente:

$$A = \frac{D}{K}$$

$$\Rightarrow \frac{S_i}{N_i} = \frac{1}{2K^2} \qquad (3.24)$$

El otro requerimiento pasa por establecer para que frecuencia queremos que trabaje nuestro sistema. Sabiendo que el sistema en ausencia de señal moduladora produce saltos con un periodo de tiempo medio llamado tiempo de Kramer ( $\tau_K$ ), podemos hacer coincidir dicho tiempo con el periodo de esta señal, de manera tal que haya cierta cooperación para alcanzar un máximo valor de entrada que le permita al sistema pasar de un estado al otro siguiendo la

frecuencia de la señal moduladora. Podemos relacionar ambos periodos de la siguiente manera:

$$2\tau_K = T_{tr}$$

Donde $T_{tr} = \frac{1}{f_{tr}}$, es el periodo de la señal moduladora.

Esta relación establece que cada dos tiempos de Kramer, el sistema tiende a volver hacia el mismo estado, cooperando con la modulación de la señal. A la vez se cumple que $\omega_{tr} \ll U''(\pm c) = 2a$, ya que tomando $2\tau_K = T_{tr}$, $\omega_{tr}$ es aproximadamente diez veces menor a $2a$.

De la ecuación (3.18) y proponiendo una frecuencia de trabajo, se puede obtener una expresión para el parámetro de sistema $a$, lo cual nos permite ir configurando el sistema según los requerimientos planteados:

$$\frac{1}{2 f_{tr}} = \frac{\sqrt{2}\pi}{a} e^{2U_0/D}$$
$$2U_0 = D$$

$$\Rightarrow a = 2\pi f_{tr} \sqrt{2} e \qquad (3.25)$$

El parámetro que nos falta buscar es $b$. Con la relación (3.9) se estableció el nivel de señal mínimo que permite el salto de un estado al otro. A partir de esta ecuación se despejará el parámetro restante, estimando que el sistema trabaja con el nivel de señal de entrada límite ($C_0$):

$$A + D = \frac{2}{3\sqrt{3}} ac$$
$$c = \sqrt{\frac{a}{b}}$$

$$D = 2U_0 = \frac{a^2}{2b} \qquad (3.26)$$

$$A = \frac{D}{K} \qquad (3.27)$$

$$\frac{D}{K}+D=\frac{2}{3\sqrt{3}}a\sqrt{\frac{a}{b}}$$

$$\frac{a^2}{2b}\left(\frac{K+1}{K}\right)=\frac{2}{3\sqrt{3}}a\sqrt{\frac{a}{b}}$$

$$\Rightarrow b=\frac{4}{3\sqrt{3}}\left(\frac{K}{K+1}\right)a \qquad (3.28)$$

Finalmente se obtuvieron ecuaciones como la (3.25) y (3.28) donde se establecen parámetros del sistema según los requerimientos.

Establecido lo anterior, queda aún un aspecto por revisar, el cual tiene que ver con cuantas veces debe ser superada la condición (3.9) para que el sistema funcione de forma óptima. Esto consiste en encontrar $k$ tal que:

$$k(A+D)>\frac{2}{3\sqrt{3}}ac$$

El valor de este parámetro se sintonizará empíricamente, dejando la búsqueda de una ley o expresión exacta como objetivo para trabajos futuros complementarios a éste.

### 3.3.4 Simulaciones

En esta sección se hacen simulaciones con la herramienta de Matlab llamada Simulink. Se arma el sistema con bloques y se configuran los mismos según las especificaciones requeridas.

Para estos ejemplos, se requiere una frecuencia de trabajo $f_{tr}=2[Hz]$, y se probarán diversos valores de relación señal ruido de entrada. En la Figura 3.12 se observa en detalle el bloque del sistema basado en resonancia estocástica y en la Figura 3.13 se observa el diagrama en bloques del modelo en Simulink del sistema completo.

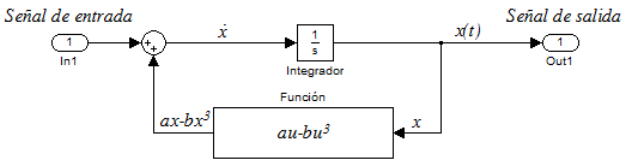



Figura 3.12. Sistema basado en resonancia estocástica, en bloques Simulink.

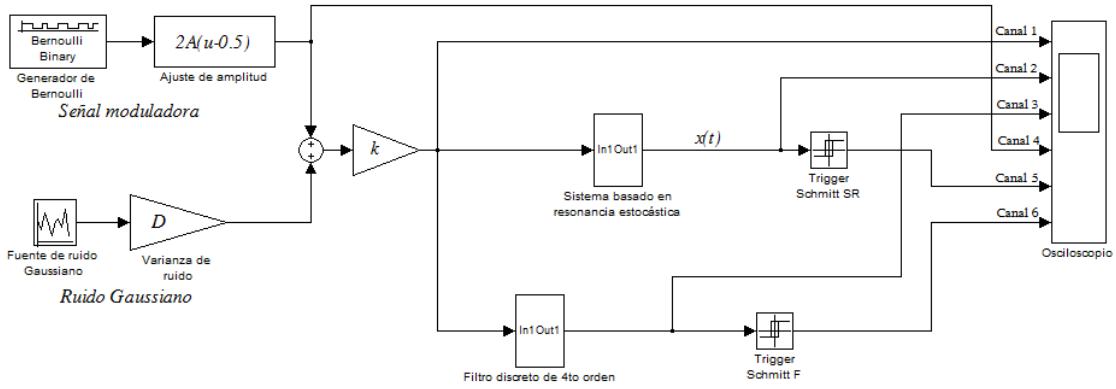



Figura 3.13. Modelo en Simulink del sistema completo.

La señal de entrada al sistema basado en resonancia estocástica está compuesta por la suma del ruido Gaussiano normalizado entre 1 y -1 multiplicado por la dispersión $D$, y la salida del generador de Bernoulli, el cual genera pulsos aleatorios, multiplicado por el bloque de ajuste de amplitud que lleva la señal moduladora a valores de amplitud entre $A$ y $-A$.

*Aclaración:* para las siguientes simulaciones se utilizará como señal de moduladora un generador de onda cuadrada aleatorio (generador de Bernoulli). Esta fuente no es periódica (es aleatoria) lo cual no coincide con lo desarrollado hasta aquí, pero se podrá observar con el correr de las simulaciones que la cooperación entre las señales existe y que el fenómeno de resonancia estocástica se manifiesta tras haber ajustado el tiempo de bit ($T_b$) del generador, al periodo de la frecuencia de trabajo ( $f_{tr}$ ).

Esta señal de entrada es sumada a la salida del bloque “Función”, el cual tiene como entrada la variable $x$. El resultado de esta suma es $\frac{dx}{dt}$, el cual entra al integrador y así se obtiene la señal de salida $x(t)$ (Figura 3.12), la cual está conectada a un circuito Trigger de Schmitt (bloque “Trigger Schmitt SR”) con el fin de reconstruir el pulso cuadrado.

El bloque que se observa en la parte inferior de la Figura 3.13 es un filtro discreto de cuarto orden, el cual es normalmente utilizado para el filtrado en audio, cuya salida está conectada también a un circuito Trigger de Schmitt (bloque “Trigger Schmitt F”). Por último, el bloque de la derecha es un osciloscopio con seis canales, los cuales nos permitirán obtener los gráficos de las forma de onda arrojadas por el modelo. El Canal 1 nos muestra la señal de entrada a ambos sistemas, el 2 la salida del sistema basado en resonancia estocástica, el 3 la salida del filtro, el 4 la señal moduladora (señal original), el 5 el pulso reconstruido a partir del sistema basado en resonancia estocástica y el 6 el pulso reconstruido a partir del filtro.

Con este modelo se llevan a cabo las simulaciones para diferentes valores de relación señal-ruido de entrada, dejando fija la frecuencia de trabajo. Con esto se intenta bajar dicha relación hasta un valor en el cual el filtro tenga un desempaño inferior al sistema basado en resonancia estocástica, intentando demostrar la superioridad de este sistema.

Utilizando las ecuaciones (3.24), (3.25), (3.26), (3.27) y (3.28), y habiendo fijado una frecuencia de trabajo de 2Hz, se configuran los parámetros del sistema para los diferentes $K$:

- Para $K=10$:
    - De la ecuación (3.24):

$$\frac{S_i}{N_i}=\frac{1}{2K^2}=\frac{1}{200}$$

$$\left.\frac{S_i}{N_i}\right|_{dB}=-23.01[dB]$$

    - De la ecuación (3.25):

$$a=2\pi f_s\sqrt{2}e=48.308$$

    - De la ecuación (3.28):

$$b=\frac{4\sqrt{3}}{9}\left(\frac{K}{K+1}\right)a=33.807$$

    - De la ecuación (3.26):

$$D=2U_0=\frac{a^2}{2b}=34.514[V]$$

    - De la ecuación (3.27):

$$A=\frac{D}{K}=3.451[V]$$

Sintonizando $k$ a un valor óptimo para la recuperación de la señal (en este caso $k=11$), se obtuvieron los siguientes gráficos (Figura 3.14):

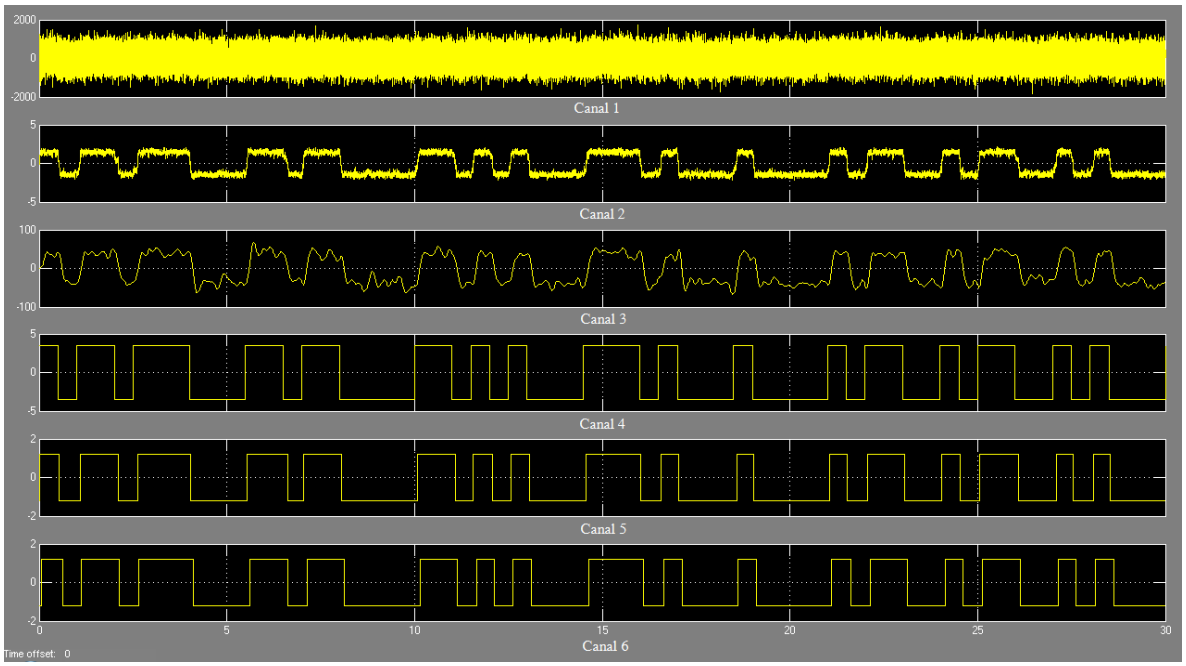



Figura 3.14. Como se menciono en la descripción del modelo, el Canal 1 muestra la señal de entrada a ambos sistemas (señal moduladora sumada al ruido), el Canal 2 la señal recuperada por el sistema basado en resonancia estocástica, el Canal 3 la señal recuperada por el filtro, el Canal 4 muestra la señal moduladora (datos), el Canal 5 los datos reconstruidos a partir de la salida del sistema basado en resonancia estocástica y el Canal 6 los datos reconstruidos a partir del filtro.

Observando los pulsos reconstruidos a partir de las salidas de ambos sistema, tanto el filtro como el sistema basado en resonancia estocástica, se puede apreciar que se recuperan los datos fielmente para este valor de relación señal-ruido de entrada.

- Para $K = 20$:
  - De la ecuación (3.24):

$$\frac{S_i}{N_i} = \frac{1}{2K^2} = \frac{1}{800}$$

$$\left.\frac{S_i}{N_i}\right|_{dB} = -29.03[dB]$$

  - De la ecuación (3.25):

$$a = 2\pi f_s \sqrt{2}e = 48.308$$

- De la ecuación (3.28):

$$b = \frac{4\sqrt{3}}{9}\left(\frac{K}{K+1}\right)a = 35.417$$

- De la ecuación (3.26):

$$D = 2U_0 = \frac{a^2}{2b} = 32.946[V]$$

- De la ecuación (3.27):

$$A = \frac{D}{K} = 1.647[V]$$

Sintonizando $k$ a un valor óptimo para la recuperación de la señal (en este caso $k = 13$), se obtuvieron los siguientes gráficos (Figura 3.15):

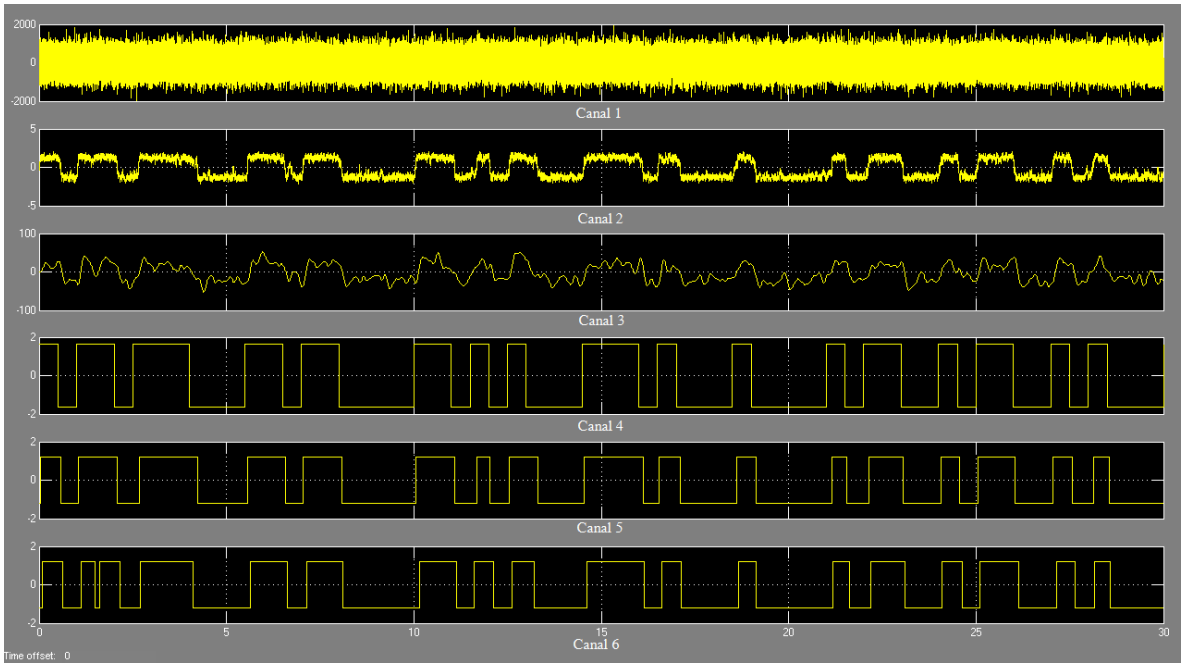



Figura 3.15. Descripción Ídem Figura 3.14.

- Para $K = 30$:

  - De la ecuación (3.24):

$$\frac{S_i}{N_i} = \frac{1}{2K^2} = \frac{1}{1800}$$

$$\left.\frac{S_i}{N_i}\right|_{dB} = -32.56[dB]$$

- De la ecuación (3.25):

$$a = 2\pi f_s \sqrt{2}e = 48.308$$

- De la ecuación (3.28):

$$b = \frac{4\sqrt{3}}{9}\left(\frac{K}{K+1}\right)a = 35.988$$

- De la ecuación (3.26):

$$D = 2U_0 = \frac{a^2}{2b} = 32.423[V]$$

- De la ecuación (3.27):

$$A = \frac{D}{K} = 1.081[V]$$

Sintonizando $k$ a un valor óptimo para la recuperación de la señal (en este caso $k = 14.5$), se obtuvieron los siguientes gráficos (Figura 3.16):

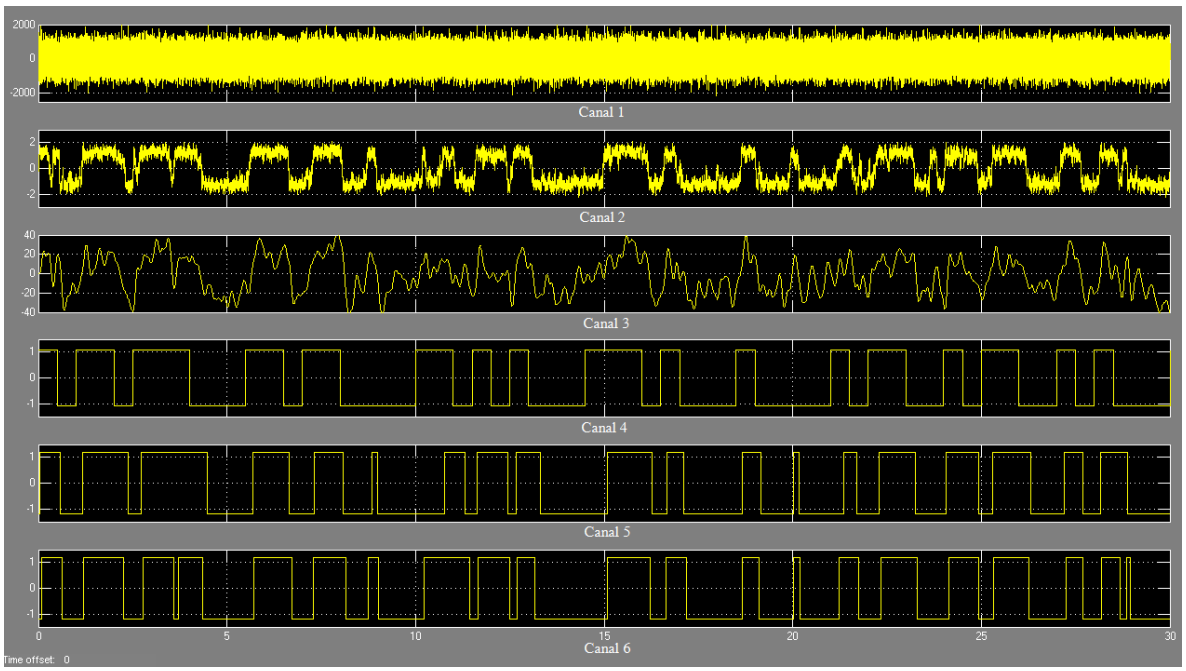



Figura 3.16. Descripción Ídem Figura 3.14.

Para este caso, se observa que ambos sistemas comienzan a introducir bits erróneos, con una leve superioridad de desempeño del sistema basado en resonancia estocástica que se observa al comienzo y al final del muestreo.

- Para $K = 40$:
  - De la ecuación (3.24):

$$\frac{S_i}{N_i} = \frac{1}{2K^2} = \frac{1}{3200}$$

$$\left.\frac{S_i}{N_i}\right|_{dB} = -35.06\left[dB\right]$$

  - De la ecuación (3.25):

$$a = 2\pi f_s \sqrt{2}e = 48.308$$

  - De la ecuación (3.28):

$$b = \frac{4\sqrt{3}}{9}\left(\frac{K}{K+1}\right)a = 36.281$$

- De la ecuación (3.26):

$$D = 2U_0 = \frac{a^2}{2b} = 32.161[V]$$

- De la ecuación (3.27):

$$A = \frac{D}{K} = 0.804[V]$$

Sintonizando $k$ a un valor óptimo para la recuperación de la señal (en este caso $k = 15.5$), se obtuvieron los siguientes gráficos (Figura 3.17):

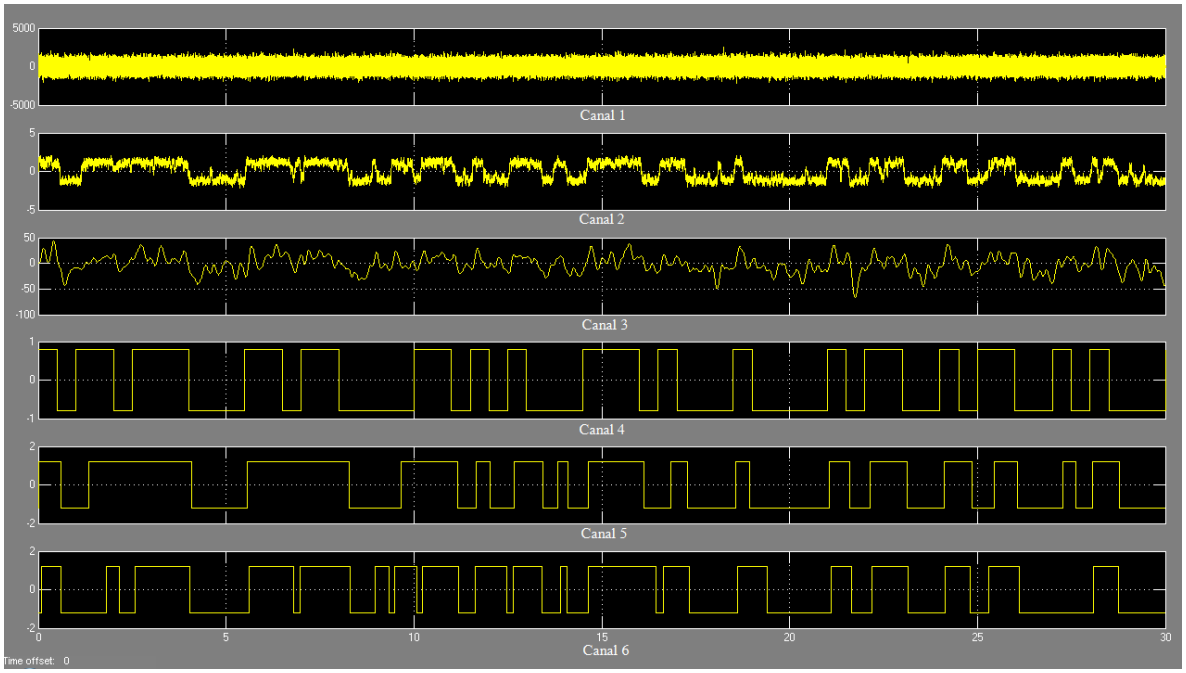



Figura 3.17. Descripción Ídem Figura 3.14.

A simple vista observamos que la tasa de error aumentó en ambos sistema, los errores cometidos entre los segundos 10 y 15 son comunes en ambas salidas, mientras que el desempeño del sistema basado en resonancia estocástica a grandes rasgos sigue siendo levemente superior al del filtro, lo cual se puede observar a partir del segundo 15.

Como conclusión de esta comparación, podemos señalar que con una fina sintonización de los parámetros del sistema basado en resonancia estocástica, se podría obtener un desempaño superior al filtro de cuarto orden. Tener en cuenta que el sistema basado en resonancia estocástica tiene la desventaja que trabaja a muy baja frecuencia, pero por contrapartida,

tiene las ventajas de sencillez a la hora de su implementación y selectividad en frecuencia, siendo esta última característica muy útil para varias aplicaciones.

Con este resultado se puede argumentar la viabilidad de la implementación de un sistema basado en el método de resonancia estocástica para la detección de señales débiles en entornos ruidosos, siendo efectivo bajo una relación señal-ruido de aproximadamente -30dB.

# Capítulo 4: APLICACIÓN: MARCA DE AGUA DIGITAL EN ARCHIVO DE AUDIO

# 4 APLICACIÓN: MARCA DE AGUA DIGITAL EN ARCHIVO DE AUDIO

En este capítulo se desarrolla la aplicación que le daremos al sistema basado en resonancia estocástica.

## 4.1 Descripción

El sistema basado en resonancia estocástica hasta aquí analizado se puede utilizar como detector de marcas de agua digitales en archivos de audio, lo cual se busca implementar en este trabajo final. La idea es generar, en el bloque transmisor, un archivo de audio al cual se le adhiere la marca de agua digital (datos). En el otro extremo, o sea en el bloque receptor, se pretende utilizar el sistema basado en resonancia estocástica para detectar la marca de agua y recuperar los datos.

A continuación se muestra un esquema general (Figura 4.1) de un sistema de comunicaciones propiamente dicho, con un transmisor, un canal de comunicación (el cual será el aire en nuestro caso) y un receptor.

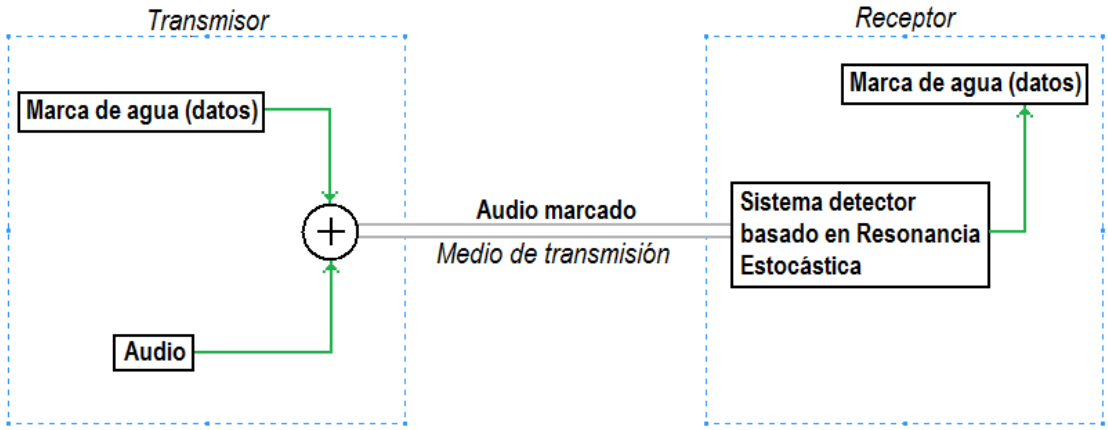



Figura 4.1.

El desarrollo de este capítulo tiene como objetivo pasar del sistema en tiempo continuo, a un modelo en tiempo discreto, ya que la idea es analizar la posibilidad de implementación del sistema en una plataforma digital (FPGA). También, se reemplazará la fuente de ruido Gaussiano por una de audio, ya que será éste el que enmascare a la marca de agua.

## 4.2 De sistema continuo a sistema discreto

Analizada la performance del sistema de dos estados en tiempo continuo, se llevará a cabo un análisis del mismo en tiempo discreto usando el software Simulink, teniendo en cuenta que se buscará analizar la posibilidad de implementarlo en una plataforma digital. Para ello se

reemplazó el bloque integrador por su análogo en tiempo discreto (Figura 4.2), y se configuró el resto de los bloques para que trabajen en "Sample Time" (tiempo de muestreo), donde también se deberá configurar dicho tiempo al valor que corresponda.

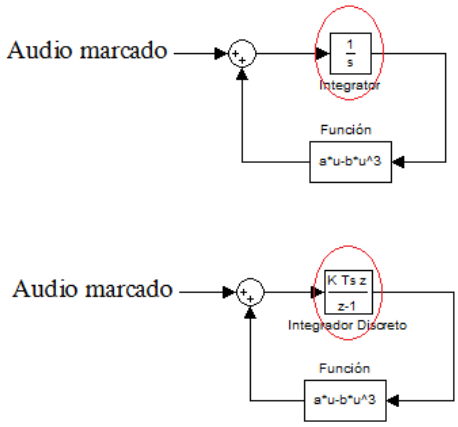



Figura 4.2. Se resalta con un círculo rojo los bloques integradores, el modelo superior con el integrador en tiempo continuo y el inferior con el de tiempo discreto.

### 4.2.1 Integrador discreto

Breve descripción del bloque integrador discreto.

Para la integración discreta se utilizó el método de Euler hacia atrás ya que corresponde a un sistema realimentado. Para ello, $\frac{1}{s}$ es reemplazado por $\frac{KTz}{z-1}$, donde $T$ es el tiempo de muestreo. La expresión resultante a la salida del bloque para el paso $n$ es la siguiente:

$$y(n) = y(n-1) + K.T.u(n)$$
$$x(n) = y(n-1)$$

Donde $K$ es la ganancia del bloque.

## 4.3 Comparación de desempeño entre el sistema en tiempo continuo y discreto

Reemplazado el bloque integrador y configurado el sistema para trabajar en modo discreto, se procede a realizar una rápida comparación entre el sistema en tiempo continuo y el de tiempo discreto. Se demostrará gráficamente que eligiendo una frecuencia de muestreo adecuada, los desempeños son muy similares.

En la próxima sección se hará el reemplazo de la fuente de ruido por la de audio, por lo que la frecuencia de muestreo del sistema será justamente la frecuencia a la cual estará muestreado

el audio. Como el audio estará representado por un archivo WAV con frecuencia de muestreo $f_s = 11.025[kHz]$, se utilizará dicha frecuencia para la siguiente simulación.

En la Figura 4.3 se muestra el modelo en Simulink, donde el sistema superior es el de tiempo continuo y el inferior el de tiempo discreto. El bloque de la derecha es un osciloscopio, el cual nos reproduce las formas de onda arrojadas por ambos sistemas, las cuales se observan en la Figura 4.4.

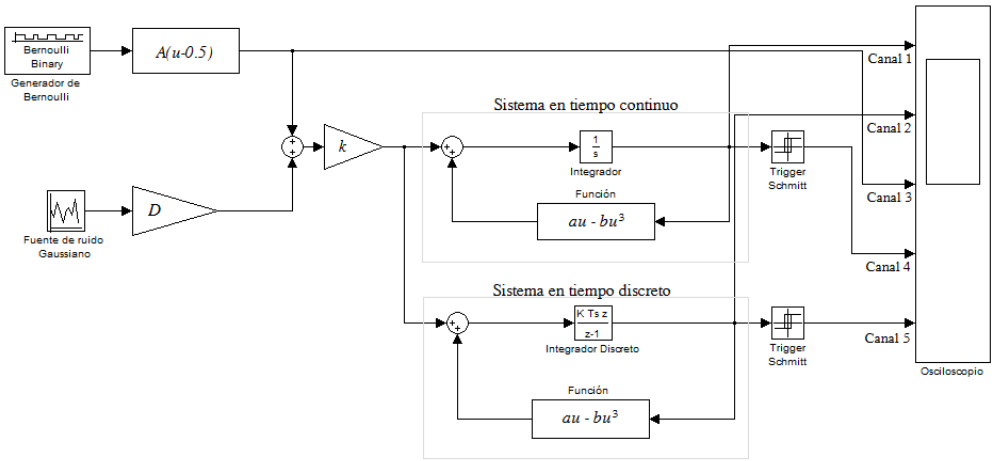



Figura 4.3. Modelo de comparación en Simulink.

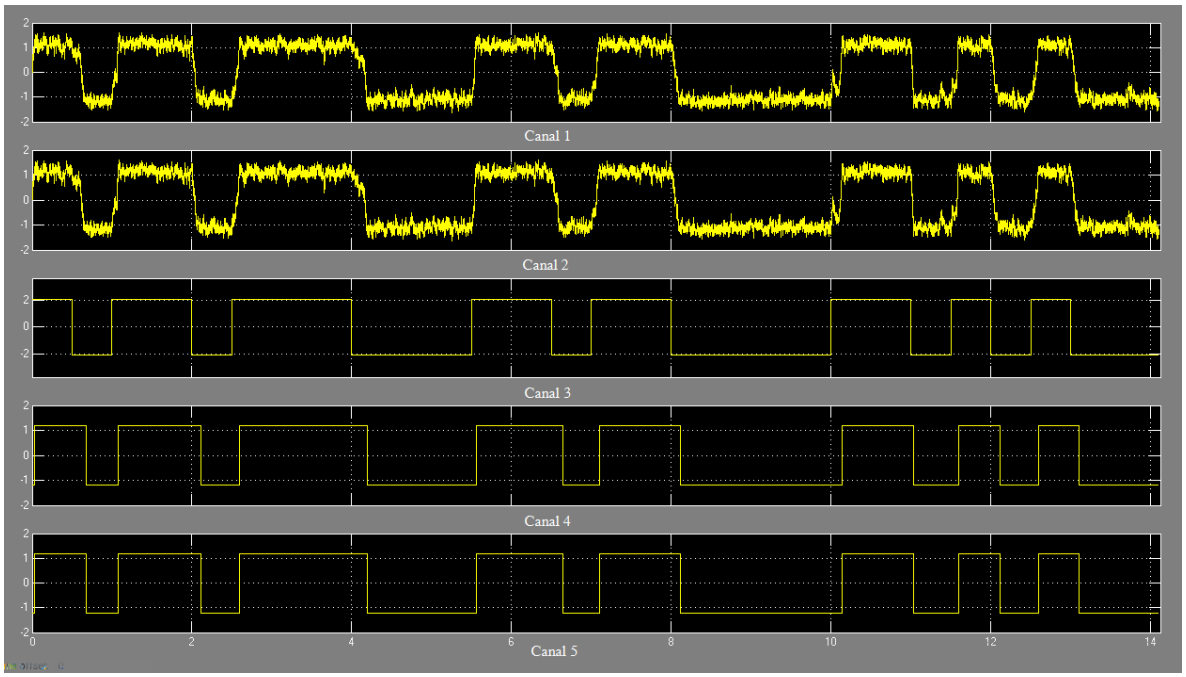


Figura 4.4. En los canales 1 y 2 se observan las formas de onda de salida de los sistemas en tiempo continuo y tiempo discreto respectivamente. El Canal 3 representa la marca de agua original y los canales 4 y 5 los pulsos reconstruidos (marca de agua digital detectada) a partir de las salidas de los sistemas en tiempo continuo y tiempo discreto respectivamente.

Como se puede observar en la Figura 4.4 las formas de onda y la marca de agua recuperada por ambos sistemas poseen diferencias prácticamente imperceptibles. Se realizaron las simulaciones con $f_{tr} = 2[Hz]$ y $K = 10$.

## 4.4 Reemplazo de la fuente de ruido por la fuente de audio

Habiendo demostrado el funcionamiento del sistema en tiempo discreto, se procederá a reemplazar la fuente de ruido Gaussiano por una de audio. De esta forma el sistema va tomando la forma de la aplicación que se le desea dar a la teoría hasta aquí analizada, *envío de información oculta a través de marcas de agua digitales en archivo de audio.*

Es importante que la fuente de audio utilizada tenga un espectro relativamente similar al de la fuente de ruido, o sea, como se trataba de ruido blanco, el audio debe presentar un espectro lo más parecido al continuo posible.

Para las siguientes simulaciones y desarrollos de ejemplos se escogió un archivo de audio llamado *m_park.wav* el cual represente el sonido de un bosque. En la Figura 4.5 se observan los espectros de la fuente de ruido (Figura 4.4-a) y de la fuente de audio para el archivo seleccionado (Figura 4.4-b).

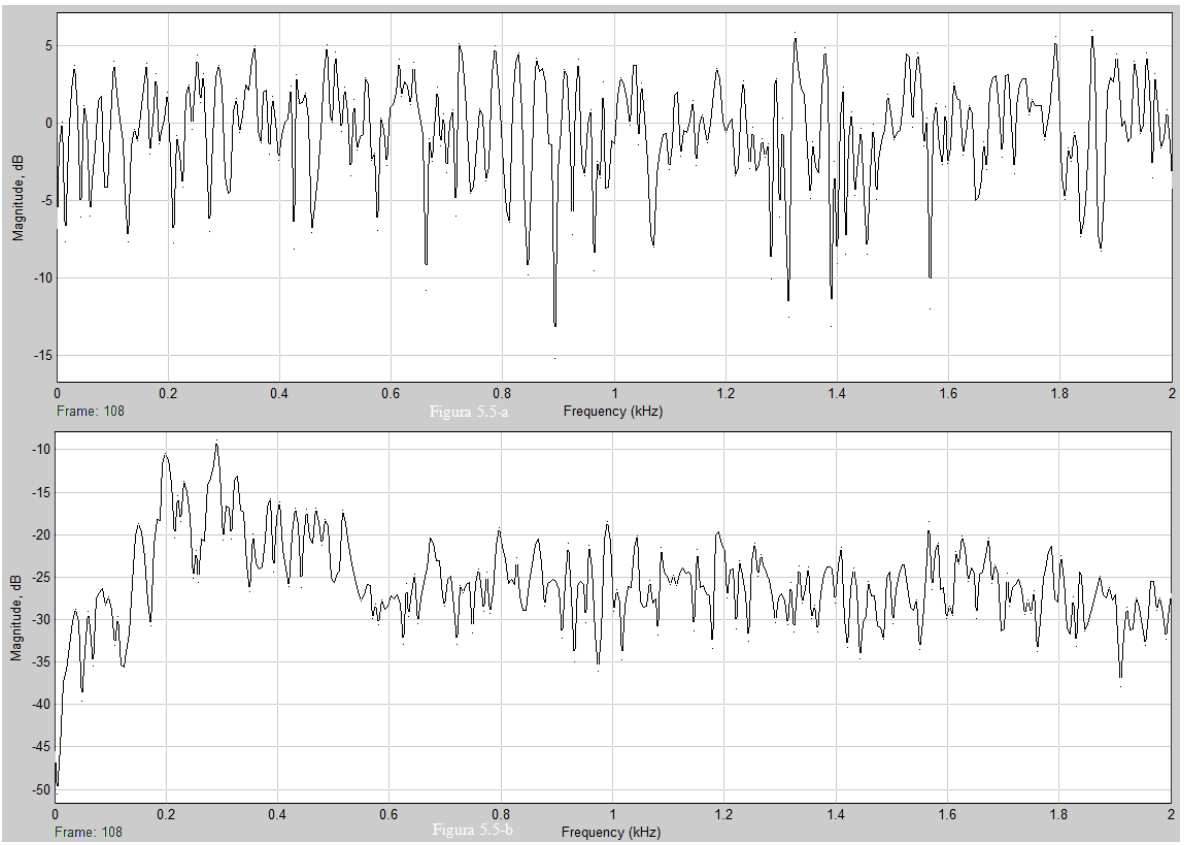



Figura 4.5. Se observan los espectros de frecuencias de la fuente de ruido (Figura 4.5-a) y la fuente de audio (Figura 4.5-b), en un ancha de banda de 0 a 2kHz.

Datos del archivo de audio:

- Frecuencia de muestreo: 11.025[kHz]
- Bits por muestra: 8 bits
- Velocidad: 88.2[kbps]
- Duración: 8[seg]

A continuación, se reemplaza la fuente de ruido por la de audio en el modelo de Simulink del sistema basado en resonancia estocástica (Figura 4.6). Se agrego un bloque llamado “From Wave File” donde se detalla la ubicación del archivo de audio a utilizar (ejemplo: C:\Users\Garbarino\Desktop\Seba\facu - IUA\Tesis - Resonancia Estocastica\m_park.wav) al cual debe conectarse a la salida un bloque llamado “Unbuffer” el cual, como su nombre lo indica, realiza la función contraria a un Buffer, tomando las muestras y reconstruyendo la señal de audio en tiempo continuo. También se agrega una ganancia que ajusta la amplitud de salida

de la fuente de audio entre $\pm 1[V]$. De esta manera la fuente de audio compuesta por estos dos bloques, devuelve una señal de características similares al ruido Gaussiano.

Antes de comenzar con las simulaciones, se debe tener en cuenta que se configuró todo el sistema para que trabaje en "Sample time" (de forma discreta) con una frecuencia de muestreo igual a la del archivo de audio ( $f_s = 11.025[kHz]$).

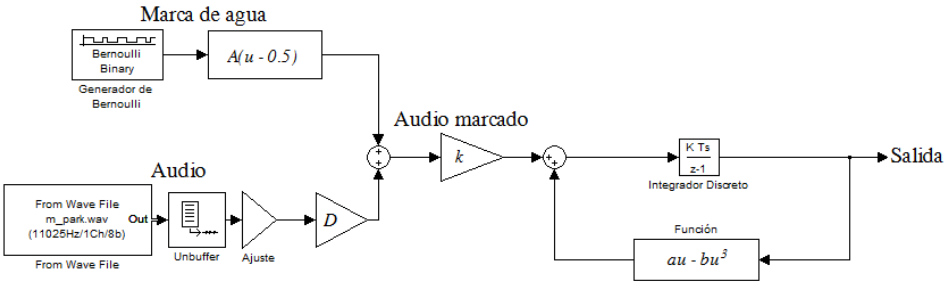



Figura 4.6. Modelo en Simulink del sistema basado en resonancia estocástica con la fuente de audio incorporada.

De esta manera estamos en condiciones de iniciar las simulaciones con la fuente de audio incorporada, para las cuales se deberá reajustar el parámetro $k$ .

## 4.5 Simulaciones

En esta parte del Capítulo 4, se realizan las simulaciones del sistema en tiempo discreto y con la fuente de audio como portadora de la marca de agua digital. El objetivo es comparar que el hecho de sumar la marca de agua al audio, éste no se modifica de manera considerable, y mostrar el buen funcionamiento del sistema basado en resonancia estocástica con estos cambios.

Como se dijo anteriormente, se compara en primera instancia el audio marcado con el original, se observan las formas de onda (Figura 4.7):

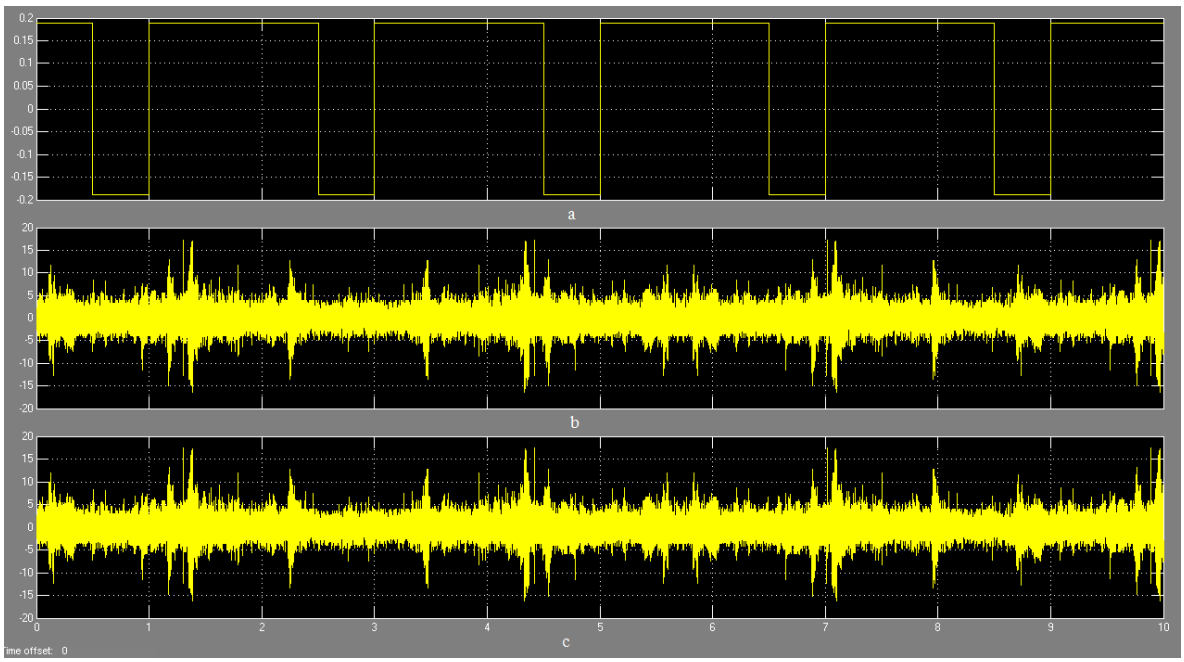



Figura 4.7. En el gráfico 4.7-a se observa la marca de agua digital, en el 4.7-b el audio original y en el 4.7-c el audio marcado, el cual es el resultado de la suma de los dos anteriores.

Notar que las diferencias entre las formas de onda de los gráficos 4.7-b y 4.7-c son casi imperceptibles, lo cual también se puede demostrar de forma auditiva. Tener en cuenta las amplitudes, tanto del audio como de la marca de agua, necesarias para que el sistema detecte esta última de manera eficiente.

A continuación se demuestra gráficamente como el sistema recupera la marca de agua digital sin inconvenientes. El modelo en Simulink utilizado es el de la Figura 4.8, en el cual se configuraron los siguientes parámetros:

$$K = 100$$
$$f_{tr} = 2[Hz]$$
$$A = 0.1878[V]$$
$$D_A = 18.78[V]$$
$$a = 48.31$$
$$b = 62.14$$

Donde $D_A$ es la dispersión del audio.

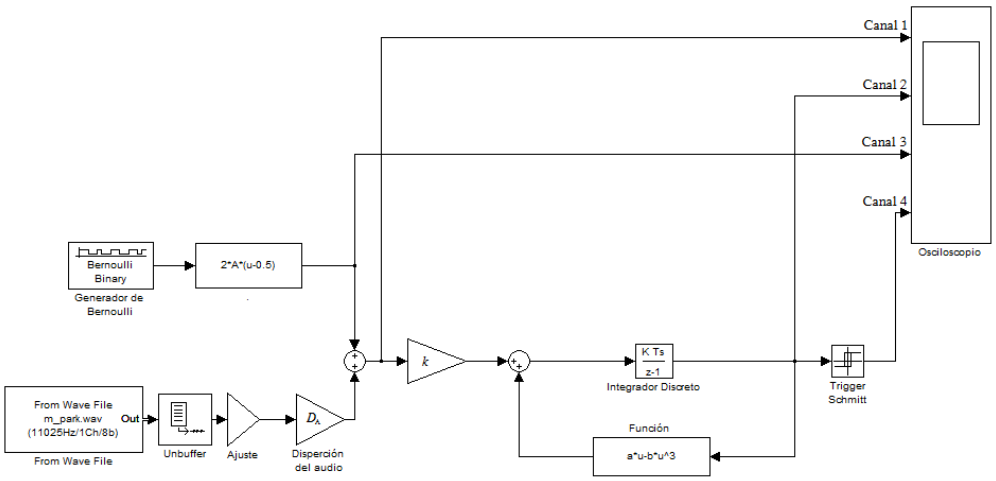



Figura 4.8. Aquí se muestra el modelo en Simulink.

Como se observa en la Figura 4.8, a la salida del sistema basado en resonancia estocástica se colocó un circuito Trigger de Schmitt, lo que permite reconstruir el pulso cuadrado y compararlo con la marca de agua original. En la Figura 4.9 se muestran las formas de onda obtenidas del osciloscopio del simulador, el cual posee 4 canales: en el Canal 1 se conecta la señal de entrada al sistema (audio marcado), en el Canal 2 la señal de salida del sistema, en el Canal 3 se conecta la salida del generador de Bernoulli representado a la marca de agua digital (datos), y en el Canal 4 se conecta la salida de circuito Trigger de Schmitt (datos recuperados).

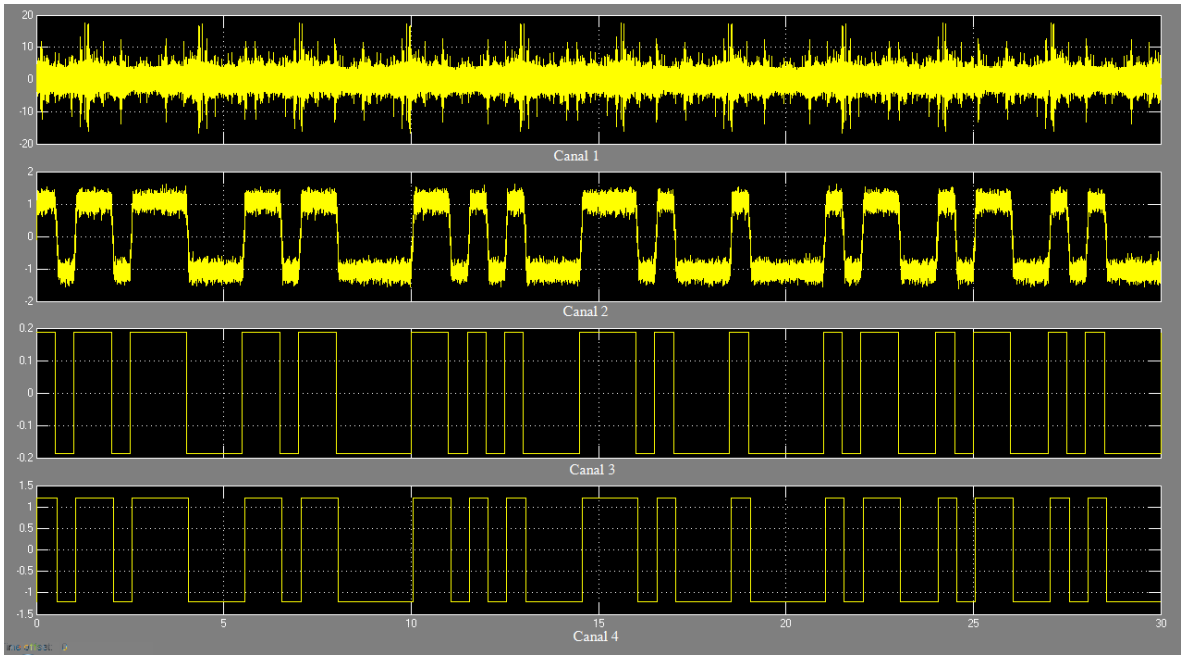



Figura 4.9. Señales obtenidas del osciloscopio del modelo en Simulink.

Comparando el Canal 3 (datos originales, marca de agua digital) con el 4 (datos recuperados, marca de agua digital detectada), concluimos en que el sistema basado en resonancia estocástica en tiempo discreto detecta fielmente la marca de agua digital aun habiendo cambiando la fuente de ruido por la de audio.

En el siguiente capítulo se analizará la posibilidad de implementar el sistema con una plataforma digital realizando los ajustes necesarios.

# Capítulo 5: IMPLEMENTACIÓN EN PLATAFORMA DIGITAL

# 5 IMPLEMENTACIÓN EN PLATAFORMA DIGITAL

En este capítulo se describe la implementación del receptor basado en resonancia estocástica en una plataforma digital (FPGA, Field Programmable Gate Array).

El objetivo es generar un archivo de audio con una marca de agua digital y reproducirlo, para detectarlo con un receptor implementado en una FPGA que muestree el audio, detecte la marca de agua (datos) y devuelva la misma para su interpretación.

El archivo de audio marcado se genera en un modelo desarrollado en Simulink, el cual levanta el audio original, genera la marca de agua y los suma con los ajustes necesarios de amplitud. Por otro lado tenemos el receptor, basado en resonancia estocástica, que se describe en lenguaje VHDL (VHSIC Hardware Description Language) con el fin de programar la FPGA.

En la Figura 5.1 se muestra un dibujo del sistema completo.

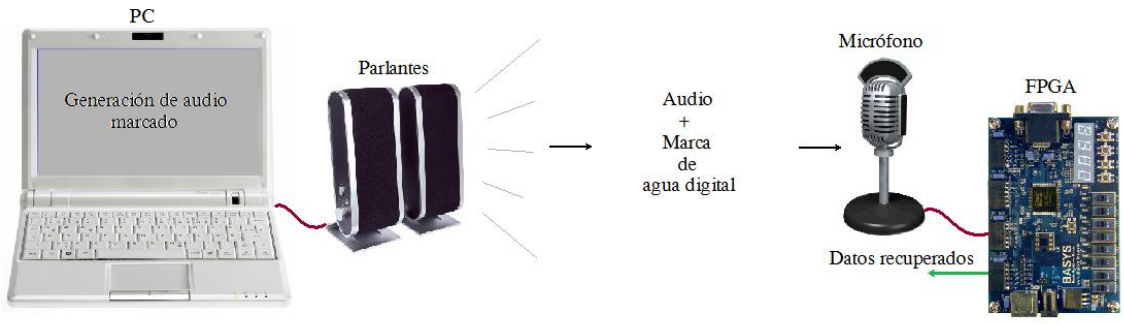



Figura 5.1.

El modelo de la Figura 5.1 tiene un desarrollo en software y en hardware que se describen a continuación por separado, detallando el transmisor y receptor.

## 5.1 Software

Se comienza por la descripción del software que se utiliza para la implementación del sistema mostrado en la Figura 5.1. Este se divide en dos partes principales:

- Transmisor: generación del archivo de audio marcado
- Receptor: sistema receptor basado en resonancia estocástica

### 5.1.1 Transmisor: generación del archivo de audio marcado

Para generar y adherir la marca de agua digital al audio original se utiliza un modelo desarrollado en Simulink. A modo de prueba del sistema, se generó una marca de agua de solo cuatro bits que se repite hasta cubrir toda la duración del audio. Como se observa en la Figura

5.2-a, que muestra el modelo en Simulink, en la parte superior izquierda hay un bloque (Figura 5.2-b) que genera la marca de agua de cuatro bits formado por un multiplexor de cuatro canales en los cuales se conectan generadores de tensión continua de $\pm 1[V]$. En la entrada "sel" del multiplexor, se conecta un contador ascendente de 0 a 3 (dos bits) con un periodo de cuenta de 0.5 segundos, lo que causa que el valor de continua conectado a cada canal se mantenga por dicho tiempo, generando una señal de onda cuadrada de $\pm 1[V]$ y $T_b = 0.5[s]$. La salida de este bloque pasa por un bloque de ganancia que le da la amplitud ( $A$ ) a la marca de agua.

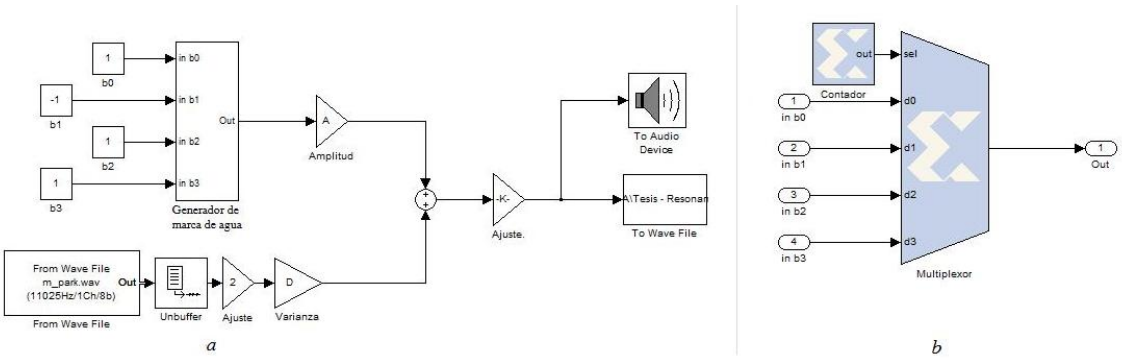



Figura 5.2.

Por otro lado, en la parte inferior izquierda de la Figura 5.2-a, se levanta el archivo de audio (en este caso llamado *m_park.wav*) con el bloque llamado "From Wave File", en el cual se especifica el directorio donde se encuentra dicho archivo. La salida pasa por un bloque llamado "Unbuffer" que toma la muestras de audio y genera una señal continua en el tiempo, la cual alimenta dos bloques de ganancia, uno ajusta la amplitud a $\pm 1[V]$ y el segundo le da el valor de dispersión ( $D$ ) correcto.

En el punto de suma, se adhiere la marca de agua digital al audio y se genera el audio marcado, el cual pasa por otro ajuste de amplitud que la normaliza a $\pm 1[V]$, lo cual permite que el bloque llamado "To Wave File" genere un archivo de audio llamado *audio_marcado.wav* para su posterior reproducción.

### 5.1.2 Receptor: sistema receptor basado en resonancia estocástica

En la parte del receptor, se implementó el sistema basado en resonancia estocástica en una plataforma digital (FPGA), la cual se observa en la Figura 5.3. Para ello se describió el hardware necesario para el muestreo del audio marcado y detección de la marca de agua digital (recuperación de los datos) en lenguaje VHDL, lo que permite programar la FPGA para que se desempeñe de la manera deseada.

El receptor tiene tres partes principales:

- Conversión analógica a digital del audio
- Sistema detector de marca de agua basado en resonancia estocástica
- Conversión digital a analógica de la marca de agua detectada

Basado en estas partes se desarrolló un código en VHDL para cada una de ellas, para luego ser unidas en uno principal. Cada una de los códigos se guardó como componente en un paquete (components y packages) para luego poder ser utilizados en el código principal.

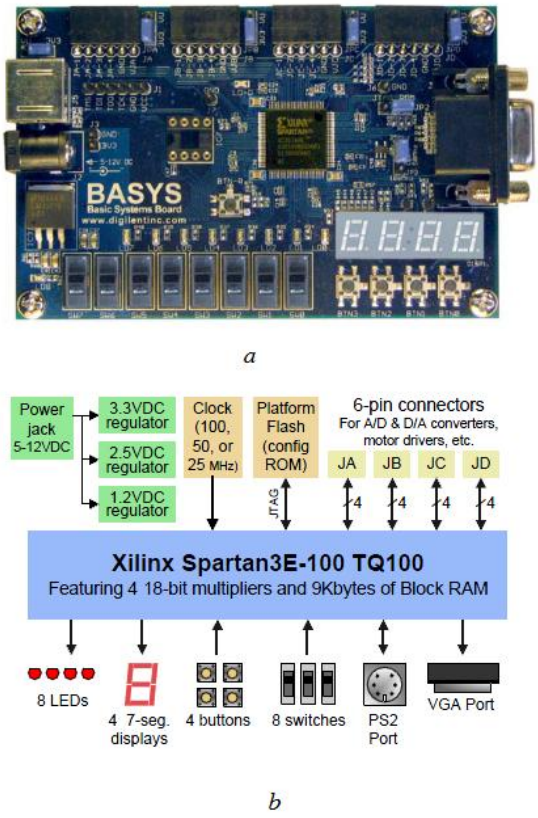



Figura 5.3. a) Fotografía de la Basys Board de Digilent basada en la FPGA Spartan 3E. b) Diagrama en bloques de la Basys Board.

### *5.1.2.1 Conversión analógica a digital*

Para llevar a cabo la conversión AD (analógica a digital) se utilizó un módulo Pmod ADCS7476 (cuyo diagrama en bloques se aprecia en la Figura 5.4) el cual es un dispositivo externo que se conecta a la plataforma digital y realiza la conversión analógica a digital. Para esta conversión, el módulo requiere de dos señales que deben ser generadas a través del código en VHDL (llamado *ad.vhd*) que lo manejan, una de ellas llamada "CS" (Chip Select, P1 de J1) que habilita al modulo para que haga la conversión, y la otra llamada "Clk" (clock, P4 de J1) que le da el sincronismo. Con estas señales el Pmod AD toma muestras de la señal analógica de entrada

conectada a P2 de J1 (DATA1) y devuelve una palabra binaria por muestra de 12 bits seriales, los cuales a través del código son pasados a paralelo para su posterior procesamiento. En el Anexo se encuentra el código VHDL que comanda el Pmod AD (archivo *ad.vhd*).

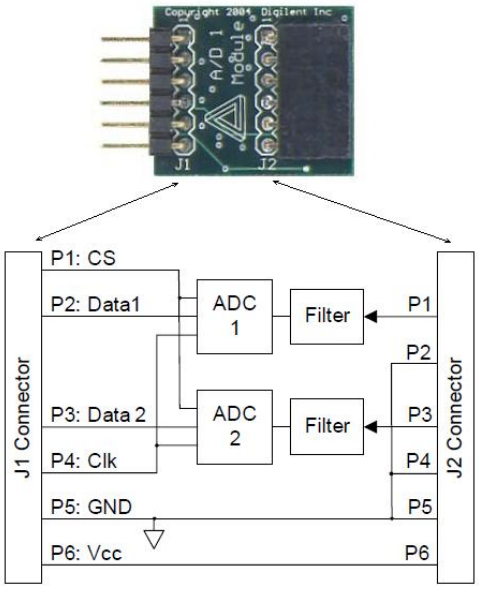



Figura 5.4.

### *5.1.2.2 Sistema detector basado en resonancia estocástica*

El bloque central del receptor es el detector basado en resonancia estocástica desarrollado en base al sistema de dos estados descripto en el Capítulo 3. Para el desarrollo del código, se describió en VHDL el sistema de dos estados implementado con bloques elementales de la Figura 5.5. Dicho código describe un componente que tiene como entrada para su procesamiento la muestra en paralelo arrojada por el bloque de conversión AD.

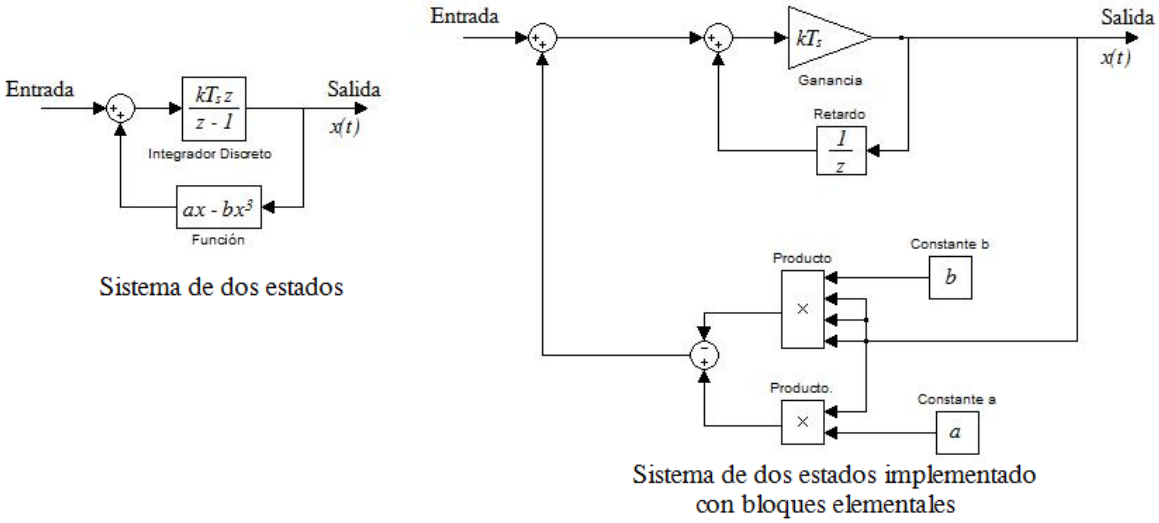



Figura 5.5.

El código principal de este componente (llamado *fpga_detector_basadosr_rx_clk6250ns_06_cw.vhd*) maneja los componentes VHDL que describen el funcionamiento de cada bloque elemental, todos sincronizados por dos señales llamadas "sysclk" (clock principal del sistema) y "sysce" (chip enable del sistema). Cada muestra es tomada con una frecuencia $f_s = 11.025[kHz]$ generada en el componente llamado *default_clock_driver_x0* a través de las señales recién nombradas.

Cada muestra de entrada es procesada y enviada al puerto de salida como una muestra representada por 12 bits en paralelo, la es entrada el bloque conversor DA, conformando la marca de agua digital recuperada. El código VHDL del sistema basado en resonancia estocástica llamado *fpga_detector_basadosr_rx_clk6250ns_06_cw.vhd* está disponible en el Anexo.

### *5.1.2.3 Conversión digital a analógica*

Así como se implementó el conversor AD también se utilizó dispositivo externo a la FPGA, módulo Pmod DAC121S101 (diagrama en bloques en la Figura 5.6), para la conversión digital a analógica. Análogamente al Pmod AD, el DA necesita dos señales para funcionar llamadas "Sync" y "clock". La primera habilita al módulo para la conversión, y la segunda le da el sincronismo.

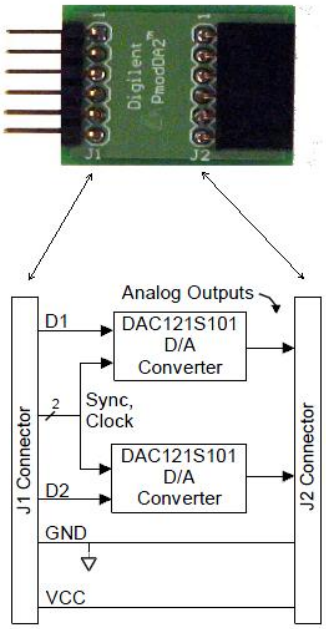



Figura 5.6.

Este componente es descripto por el código llamado *da.vhd* (disponible también en el Anexo), toma las muestras de 12 bits en paralelo arrojadas en por el bloque detector basado en resonancia estocástica, las convierte a seriales y las saca por el puerto P1 de J1. De esta forma, el Pmod DA toma las muestras seriales y reconstruye la señal analógica la cual es tomada del puerto P1 de J2 para ser observada en un osciloscopio. De esta forma se completa la conversión digital a analógica.

### 5.1.3 Sistema completo

En la Figura 5.7 se observa un diagrama esquemático del sistema completo. Se pueden observar los tres bloques que lo componen y las señales que los relacionan, tanto entre ellos como con los dispositivos de hardware que controlan. Cabe aclarar que las señales de sincronismo de los bloques de conversión derivan de la señal "sysclk", de esta forma los tres bloques trabajan de forma sincronizada, a una frecuencia de muestro $f_s = 11.025[kHz]$.

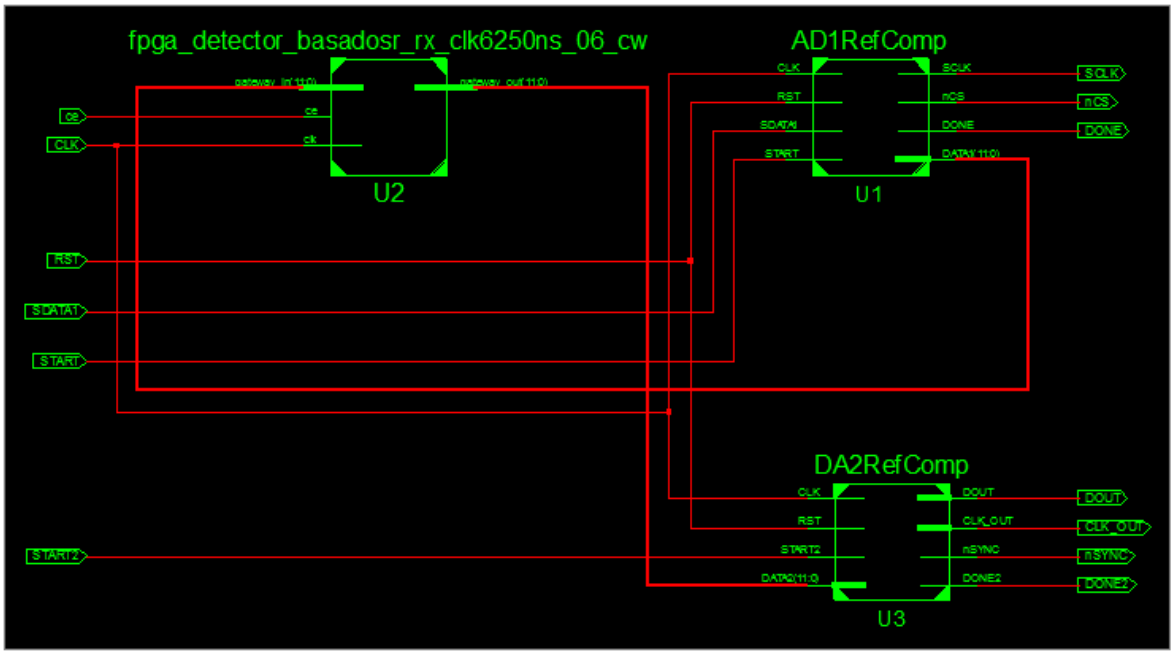



Figura 5.7.

Como se puede observar en la Figura 5.7, la cual exhibe un diagrama esquemático del receptor basado en resonancia estocástica descripto en lenguaje VHDL, hay siete puertos de entrada (input) al sistema y siete puertos de salida (output). Comenzando por los de entrada de arriba hacia abajo, tenemos los dos primeros alimentado al bloque detector, “ce” (chip enable) lo habilita para que procese la muestra y “CLK” es la señal de reloj proveniente de la FPGA que le da el sincronismo. Continuando observamos la señal “RST” (reset) que reinicia el procesamiento de los bloques conversores, “SDATA1” corresponde a la entrada serial de datos proveniente del conversor AD y las señales “START” y “START2” que activan los bloques conversores. En la parte derecha del esquemático, también comenzando de arriba hacia abajo, tenemos las señales (puertos) de salida al sistema, las cuales controlaran los bloques conversores. Las dos primeras, “SCLK” y “nCS” son el reloj (sincronismo) y el chip select (habilitación para la conversión) del conversor AD mientras que la señal “DONE” avisa cuando se tomó cada muestra. Luego observamos la señal “DOUT” que corresponde a la salida serial del sistema, la cual es entrada del conversor DA. Las dos siguientes controlan dicho conversor, “CLK_OUT” (sincronismo) y “nSYNC” (habilitación), y por último tenemos “DONE2” que avisa ante cada conversión DA. Existen dos señales internas que relaciones los tres componentes del sistema, llamadas “x” e “y”. La primera conecta la salida del AD con la entrada del sistema basado en resonancia estocástica, mientras que la segunda conecta la salida de este último con la entrada del DA. Estas señales son definidas en el código principal del receptor llamado *detector_bloque_principal.vhd* (disponible en el Anexo) donde se lleva a cabo el mapeo de puertos para relacionar los tres componentes antes descriptos.

## 5.2 Hardware

A continuación se describe el hardware que se utiliza para la implementación. Este se divide en dos partes, el transmisor y el receptor. Los ajustes de volumen se hacen de forma manual.

### 5.2.1 Transmisor

El transmisor se implementa de forma sencilla. Una vez generado el archivo de audio marcado, este se reproduce a través de parlantes de PC ajustando el volumen del mismo para que no sature al receptor.

### 5.2.2 Receptor

En la parte receptora esta el desarrollo más importante. El sonido proveniente de los parlantes es captado por un micrófono tipo electret el cual se adapta a la entrada del Pmod AD a través de un circuito pre amplificador como el mostrado en la Figura 5.8.

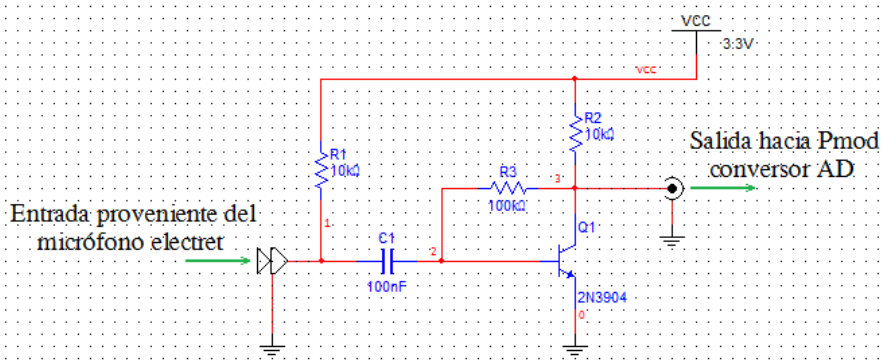



Figura 5.8. Circuito pre amplificador para adaptar el micrófono electret a la entrada del Pmod del conversor AD.

Este circuito permite adaptar la señal proveniente del micrófono electret a la entrada del Pmod ADCS7476 cuya señal de entrada debe estar entre 0 y 3.3[V]. La salida es conectada a P2 de J1 del conversor AD. La Figura 5.9 muestra el esquema del receptor completo, donde se puede apreciar la interconexión de las diferentes partes del receptor con las señales que controlan cada módulo.

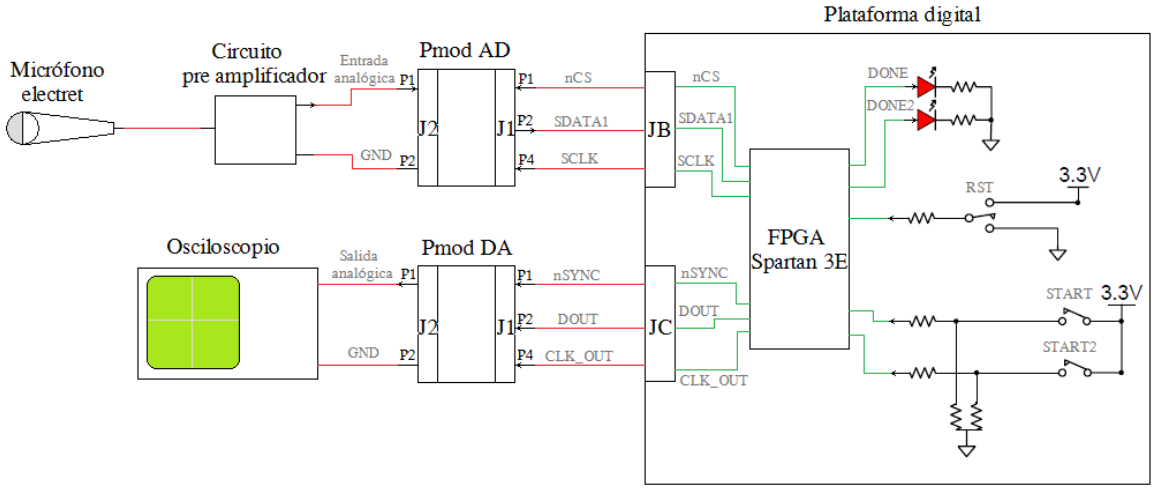



Figura 5.9.

Notar en la Figura 5.9, donde se puede observar el sistema completo, que la señal “RST” es controlada por un pulsador, que envía un ‘1’ lógico para el reinicio del sistema. La señales “START” y “START2” son mapeadas a dos llaves, las cuales permiten habilitar y deshabilitar los módulos de conversión a través de hardware. Por último tenemos las señales “DONE” y “DONE2” que encienden un diodo Led cada vez que cada conversor se ejecuta.

## 5.3 Conclusión

La sintaxis de los códigos VHDL que controlan cada Pmod y que describe cada componente del sistema ha sido chequeada y es correcta, pero debido a problemas de compatibilidad de las versiones de software disponibles en el laboratorio de la facultad y las placas (Basys Boards) no se ha podido probar el funcionamiento del sistema desarrollado.

Los componentes utilizados en para la descripción del hardware en VHDL no son son incompatibles con las placas disponibles en el laboratorio, lo cual no permite crear el archivo .BIT necesario para programar la FPGA, a pesar de que la sintaxis de los componentes VHDL son correctos.

Debido a esto, el funcionamiento del sistema fue analizado a través de simulaciones, las cuales son presentadas en el capítulo siguiente.

# Capítulo 6: SIMULACIONES DE LA IMPLEMENTACIÓN

# 6 SIMULACIONES DE LA IMPLEMENTACIÓN

En el presente capítulo se desarrolla un modelo en Simulink con la finalidad se realizar las simulaciones necesarias para demostrar el funcionamiento del sistema presentado en el Capítulo 5.

Se describe en primera instancia el sistema transmisor, en el cual se genera el archivo de audio marcado, posteriormente se describe el receptor, el cual trabaja bajo el principio de resonancia estocástica, detectando la marca de agua digital, recuperando así los datos.

Para las subsiguientes simulaciones se fija $K = 100$, $f_{tr} = 2[Hz]$ y se aplican las ecuaciones desarrolladas en el Capítulo 3.

## 6.1 Transmisor

Como se expuso en el Capítulo 5, la generación del archivo de audio marcado se lleva a cabo con el modelo de la Figura 6.1, en el cual se añade al audio original una marca de agua digital de 4 bits que se repite cíclicamente hasta igualar la duración del audio. El transmisor tiene cuatro salidas a través de las cuales se puede observar su desempeño en detalle. La salida 1 permite la visualización de la marca de agua en su formato original, la 2 el audio original, la 3 muestra la salida hacia el receptor con su amplitud original y por último la 4, la salida normalizada entre $\pm 1[V]$. En la Figura 6.2 se pueden apreciar las formas de onda de las cuatro salidas recién descriptas. Observando el Canal 1 y las entradas al bloque "generación de marca de agua" de la Figura 6.1, se aprecia que la marca de agua digital es el número binario '1011'. Comparando visualmente los canales 2 y 3, podemos comprobar el impacto casi imperceptible de la marca de agua sobre el audio original.

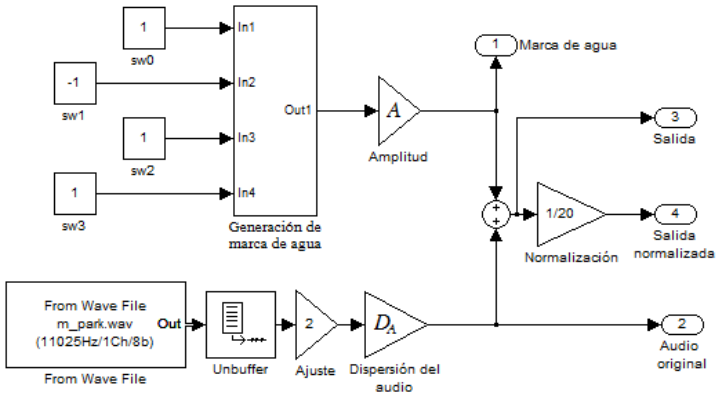



Figura 6.1. Sistema transmisor generador del audio marcado desarrollado en Simulink.

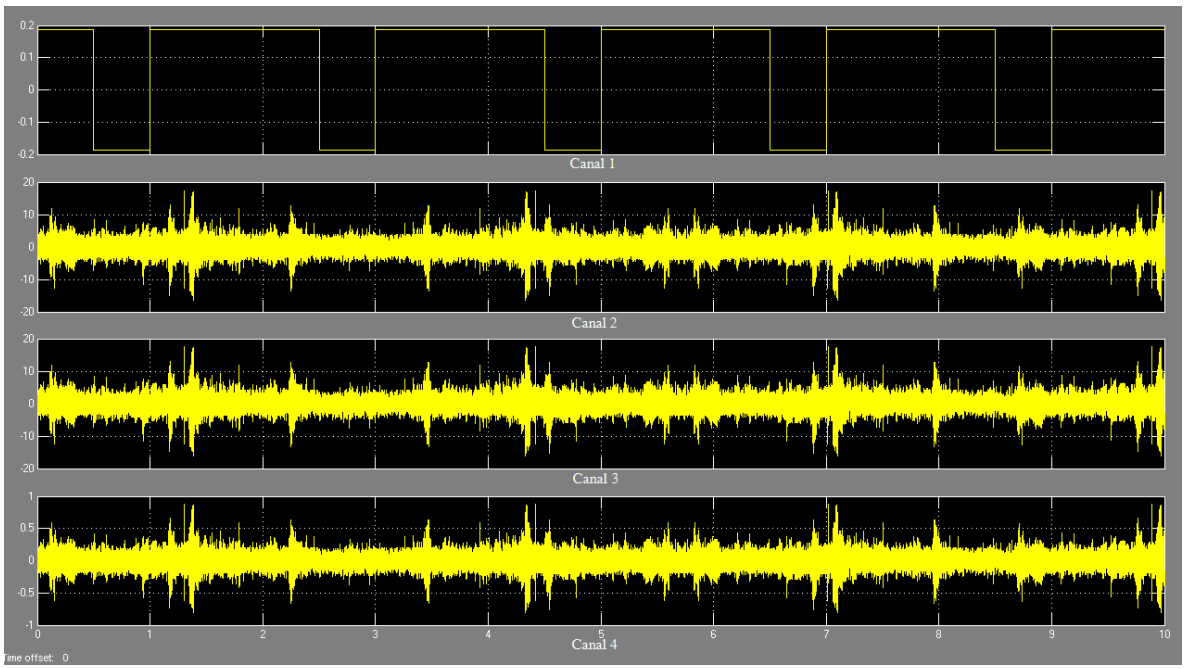



Figura 6.2. Formas de onda arrojadas por el transmisor. El Canal 1 muestra la marca de agua digital (salida 1), el Canal 2 el audio original (salida 2), el Canal 3 el audio marcado con su amplitud original (salida 3) y el Canal 4 el audio marcado con su amplitud normalizada (salida 4).

Analizada la performance del transmisor, pasamos al análisis del sistema receptor basado en resonancia estocástica.

## 6.2 Receptor

El modelo del receptor basado en resonancia estocástica mostrado en la Figura 6.3, es el utilizado para desarrollar los códigos VHDL en el Capítulo 5. Este receptor está desarrollado con bloques elementales para facilitar su implementación, por tal motivo se reemplazó el bloque correspondiente al polinomio en $x$ (bloque "función") por el circuito de la Figura 6.4-a, y el integrador discreto por el circuito de la Figura 6.4-b. Cada bloque del receptor se configura para que trabaje en tiempo discreto a una frecuencia de muestro $f_s = 11.025[kHz]$ (el valor de frecuencia viene dado por la frecuencia de la digitalización del audio crudo $f_s$)

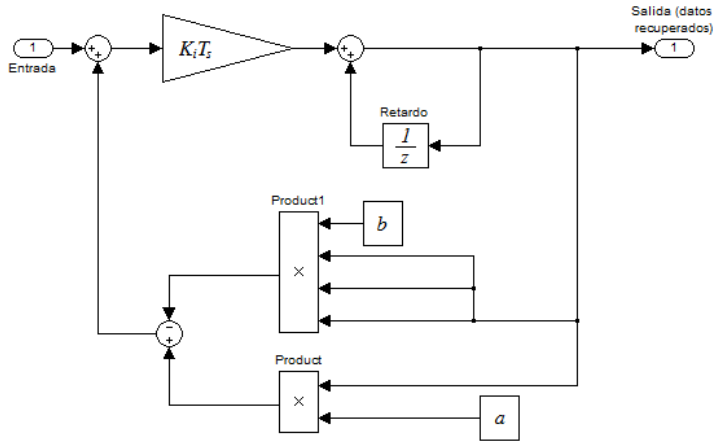



Figura 6.3. Sistema receptor basado en resonancia estocástica implementado con bloques elementales.

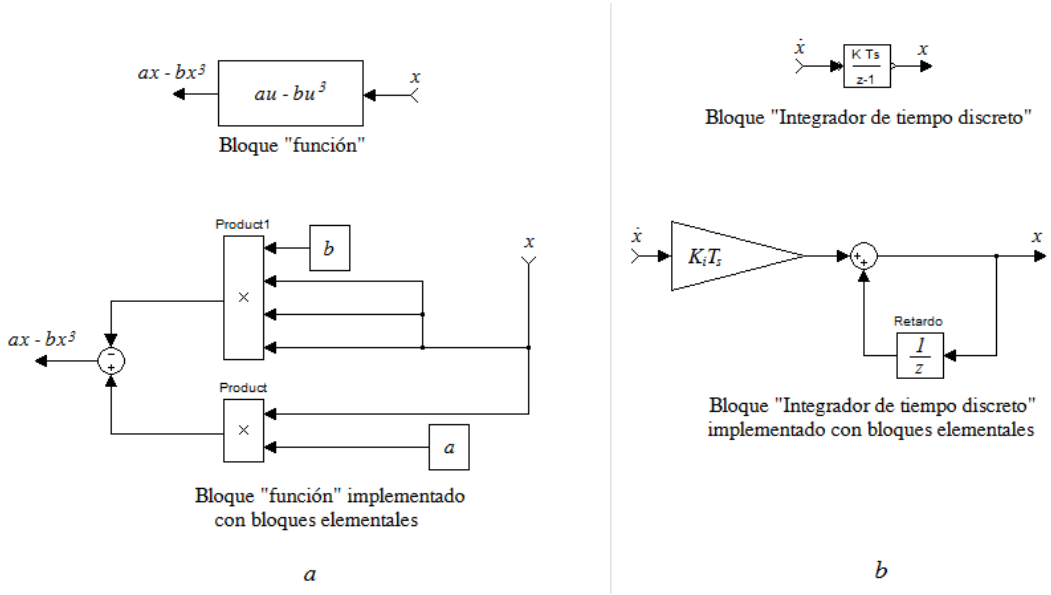



Figura 6.4.

Para analizar el desempeño del receptor se conectan ambos modelos (transmisor y receptor) para formar el sistema de encriptación completo de envío, recepción y detección de la marca de agua digital. La Figura 6.5 muestra el modelo en Simulink de dicho sistema, en el cual se toma la salida normalizada del transmisor para generar el archivo de audio marcado (bajo el nombre de *audio_marcado.wav*) a través del bloque “To Wave File” con la finalidad de realizar una comparación auditiva con el audio original. En la Figura 6.5 se puede observar que los sistemas transmisor (Figura 6.1) y receptor (Figura 6.3) se integraron en dos bloques, bloque “TX” (transmisor) y bloque “RX basado en res. estocástica” (receptor). Notar que a la entrada

de este último hay un bloque de ganancia llamado “Ajuste RX” el cual llave a cabo la función del parámetro $k$ de ajustar la amplitud de entrada al sistema basado en resonancia estocástica para que se produzca dicho fenómeno. A la salida se conecta un circuito Trigger de Schmitt, el cual permite reconstruir el pulso cuadrado que representa a la marca de agua detectada.

En la Figura 6.6 se muestran las formas de onda obtenidas a la entrada y la salida del receptor. En el Canal 1 se observa el audio marcado, el cual representa la señal que viaja desde el transmisor al receptor y transporta los datos enmascarados. En el Canal 2 se muestra la señal de salida del sistema receptor, donde ya se aprecian los pulsos detectados. En el Canal 3 y 4 se compara la marca de agua original con la reconstruida a través del circuito Trigger de Schmitt respectivamente.

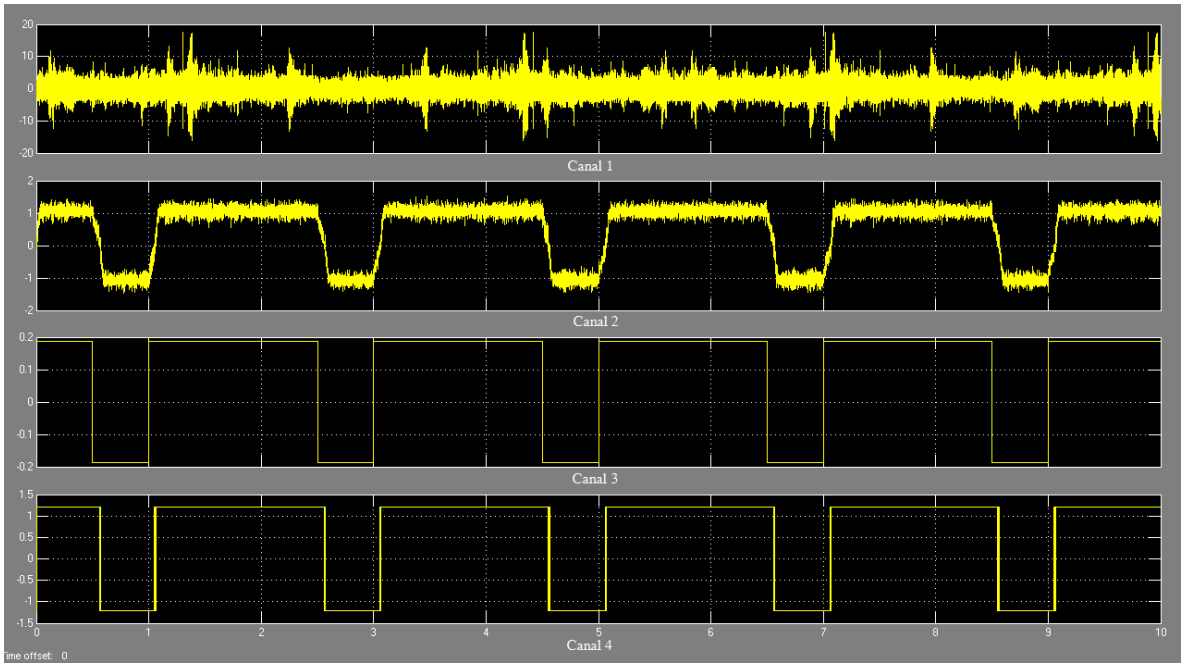



Figura 6.6. Canal 1: audio marcado, Canal 2: salida del sistema basado en resonancia estocástica, Canal 3: marca de agua digital original, Canal 4: marca de agua digital detectada.

## 6.3 Conclusión

Tras lo observado en la Figura 6.6, podemos demostrar que el sistema basado en resonancia estocástica recupera fielmente la marca de agua digital enmascarada por el audio. Notar que en el proceso del modelo anterior la marca de agua cumple con la característica de imperceptibilidad, lo cual es muy importante en el envío de información oculta o enmascarada por un portador, como ser en este caso, el audio.

De esta manera probamos la viabilidad de la implementación del sistema receptor y detector de marcas de agua digitales en archivos de audio basado en un sistema de dos estados el cual, tras la fina sintonización de sus parámetros, funciona bajo el principio de resonancia estocástica. Cabe destacar que la sintonización de los parámetros es fundamental en este tipo de transmisores/receptores no lineales, es decir que el juego de parámetros y estructura del decodificador no lineal actúan como llave del sistema de desencriptación.

# Capítulo 7: ENVÍO SECRETO DE COORDENADAS

# 7 ENVÍO SECRETO DE COORDENADAS

## 7.1 Introducción

En este capítulo se le dará un uso preciso y concreto a la aplicación desarrollada hasta aquí. Utilizando la técnica de marcas de agua digitales en archivos de audio se enviará de forma secreta alguna posición en la superficie terrestre, especificando Latitud y Longitud con una precisión determinada. Lo que se hará, es comunicar a través de información oculta la posición de un determinado objetivo a través de una palabra binaria previamente codificada. Para aumentar la seguridad de los datos se complementará la técnica con la implementación de un algoritmo de encriptación.

## 7.2 Utilización de marcas de agua digitales en esteganografía

Una de las tantas posibles aplicaciones de la técnica de marcas de agua digitales en archivos de audio es el envío de información secreta, esteganografía. Esta disciplina combina diferentes técnicas para conformar la práctica de ocultar y enviar información en un portador que pueda pasar desapercibido.

En nuestro caso, el portador será el archivo de audio, al cual se le sumará una señal de pulsos que representará la palabra binaria, previamente cifrada, que contendrá la información que deseamos enviar.

## 7.3 Información: Latitud y Longitud

En esta sección se presentan los conceptos de Latitud y Longitud.

### 7.3.1 Latitud

**Latitud**: es la distancia angular entre el Ecuador y un punto determinado del planeta medida a lo largo del meridiano que pasa por ese punto.

La Latitud se mide en grados, de 0 a 90, y puede representarse de dos formas:

- Indicando a que hemisferio pertenece la coordenada.
- Añadiendo valores negativos (Norte) y positivos (Sur).

### 7.3.2 Longitud

**Longitud**: es la distancia angular entre un punto dado en la superficie terrestre y el meridiano que se tome como 0º (en la actualidad, meridiano de Greenwich) medida a lo largo del Ecuador.

La longitud se mide en grados y existen varias formas de expresarla:

- Entre 0º y 360º, aumentando hacia el Este del meridiano 0º.
- Entre 0º y 180º, indicando a que hemisferio pertenece.

- Entre 0º y 180º, añadiendo valores negativos (Oeste) y positivos (Este).

## 7.4 Representación de la información

Introducidos los conceptos de Latitud y Longitud, se detalla en esta sección como será representada dicha información. Se comienza especificando la precisión que tendrá la posición que se enviará. Luego se estiman los bits necesarios para representar de forma binaria los datos. Obtenidas las palabras binarias se procede a aplicar el algoritmo de encriptación elegido. Por último, se ordenan los bits de tal manera para ser enviados de manera tal de proteger los más significativos.

### 7.4.1 Precisión

Se calcula la precisión en metros que tendrá la posición enviada, teniendo en cuenta que se enviarán las coordenadas en grados, minutos y segundos.

#### *7.4.1.1 Latitud*

Sabiendo que un meridiano junto con su correspondiente antimeridiano forma una circunferencia de 40007km de longitud, el salto de 1º corresponde a 111.1305km. Si añadimos minutos y segundos a dicha posición, se estaría especificando la coordenada de Latitud con la siguiente precisión ( $P_{Lat}$ ):

$$P_{Lat} = \frac{111.1305}{3600} = 30.87m$$

#### *7.4.1.2 Longitud*

En un cálculo parecido al realizado para la coordenada de Latitud, buscamos la precisión que se obtendría especificando la posición en Longitud con grados sexagesimales, minutos y segundos. Sabiendo que el Ecuador forma una circunferencia de 40076km, un salto de 1º corresponde a 111.3222km. La precisión para la Longitud ( $P_{Lon}$ ) es:

$$P_{Lon} = \frac{111.1322}{3600} = 30.92m$$

### 7.4.2 Codificación

A continuación se determinan la cantidad de bits necesaria para codificar los grados, minutos y segundos para la precisión calculada anteriormente.

#### *7.4.2.1 Latitud*

Se expresará con valores de 0º a 90º añadiendo valores negativos (Norte) y positivos (Sur). Se añaden subdivisiones que corresponden a los minutos y segundos.

- Signo:

| Bit $A_{19}$ | Referencia |
|---|---|
| '0' | Sur |
| '1' | Norte |

- Grados: de 0º a 90º. Son necesarios 7 bits.

$2^7 = 128$

Se representará el número de grados con su equivalente binario.

| Grados | $A_{18} - A_{12}$ |
|---|---|
| 0 | 0000000 |
| 1 | 0000001 |
| … | … |
| 90 | 1011010 |

- Minutos: de 0' a 60'. Son necesarios 6 bits.

$2^6 = 64$

Se representará el número de minutos con su equivalente binario.

| Minutos | $A_{11} - A_6$ |
|---|---|
| 0 | 000000 |
| 1 | 000001 |
| … | … |
| 60 | 111100 |

- Segundos: de 0'' a 60''. Son necesarios 6 bits.

$2^6 = 64$

Se representará el número de minutos con su equivalente binario.

| Segundos | $A_5 - A_0$ |
|---|---|
| 0 | 000000 |
| 1 | 000001 |
| … | … |
| 60 | 111100 |

### *7.4.2.2 Longitud*

Se expresará con valores de 0º a 180º añadiendo valores negativos (Oeste) y positivos (Este). Se añaden subdivisiones que corresponden a los minutos y segundos.

- Signo:

| Bit $B_{20}$ | Referencia |
|---|---|
| '0' | Este |
| '1' | Oeste |

- Grados: de 0º a 180º. Son necesarios 8 bits.

$2^8 = 256$

Se representará el número de grados con su equivalente binario.

| Grados | $B_{19} - B_{12}$ |
|---|---|
| 0 | 00000000 |
| 1 | 00000001 |
| … | … |
| 180 | 10110100 |

- Minutos: de 0' a 60'. Son necesarios 6 bits.

$2^6 = 64$

Se representará el número de minutos con su equivalente binario.

| Minutos | $B_{11} - B_6$ |
|---|---|
| 0 | 000000 |
| 1 | 000001 |
| … | … |
| 60 | 111100 |

- Segundos: de 0'' a 60''. Son necesarios 6 bits.

$2^6 = 64$

Se representará el número de minutos con su equivalente binario.

| Segundos | $B_5 - B_0$ |
|---|---|
| 0 | 000000 |
| 1 | 000001 |
| … | … |
| 60 | 111100 |

### *7.4.2.3 Cantidad de bits total*

Número total de bits a utilizar.

| | Signo | Grados | Minutos | Segundos | Nº bits total |
|---|---|---|---|---|---|
| Latitud | 1 ( $A_{19}$ ) | 7 ( $A_{18} - A_{12}$ ) | 6 ( $A_{11} - A_6$ ) | 6 ( $A_5 - A_0$ ) | 20 |
| Longitud | 1 ( $B_{20}$ ) | 8 ( $B_{19} - B_{12}$ ) | 6 ( $B_{11} - B_6$ ) | 6 ( $B_5 - B_0$ ) | 21 |

Para el envío de una posición son necesarios 41 bits de datos separados en dos palabras (A corresponde a Latitud y B a Longitud) con tres campos cada una (grados, minutos y segundos).

### 7.4.3 Encriptación o cifrado

Para aumentar la seguridad de los datos se utilizará un algoritmo de encriptación. A modo de demostración se implementará uno sencillo, pudiendo luego el operador aumentar la complejidad del mismo según sus requerimientos.

Se utilizará un algoritmo de cifrado de César o por desplazamiento. Este tipo de cifrado es del tipo "por sustitución" y consiste en cambiar cada campo, en nuestro caso los grados por ejemplo, por otro valor desplazado $n$ veces. Para declarar cuantas posiciones se va a desplazar el campo, se utilizará una llave, la cual será utilizada para encriptar y desencriptar la información.

Por simplicidad, en nuestro caso, es condición que la llave tenga tres o seis letras ya que la información está compuesta por seis campos (si la llave es de tres letras, se utiliza para cifrar Latitud y Longitud; si tiene seis letras, las primeras tres cifran Latitud y le segunda terna, Longitud). La idea es tomar el código ASCII de cada letra y sumar dicho valor al campo que corresponda desplazándolo. En el receptor, se restará el valor ASCII como corresponda para recuperar la información.

Demostración:

Denotamos al campo que se cifre como $C_x$ (ejemplo: $C_x \equiv G_{Lat} = +56°$, corresponde a 56° Latitud Sur) y denotamos como $K_x$ al valor en decimal del código ASCII de la letra de la llave que se corresponda con el campo de la coordenada que se está cifrando. Para encriptar cada campo realiza la siguiente operación a cada campo de las coordenadas según corresponda:

$$C_{x.enc} = (C_x + K_x) \bmod p$$

Donde: $C_{x.enc}$ es el valor del campo desplazado; $p$ puede tomar el valor de 90 (si se cifra el campo grados de Latitud), 180 (si se cifra el campo grados de Longitud) o 60 (si se cifra el campo minutos o segundos de cualquier coordenada).

*Nota*: si el valor del campo de coordenada correspondiente a los Grados sumado al valor del código ASCII de la primera letra de la llave supera $n$ veces 90 (para Latitud) o 180 (para Longitud), se deberá cambiar el signo correspondiente al indicador del hemisferio multiplicando el mismo por $(-1)^n$.

Se aplica $\bmod p$ para que el desplazamiento caiga dentro de algún valor existente para dicho campo.

En el receptor, se debe aplicar el proceso inverso:

$$K_x' = (-K_x) \bmod p$$

$$C_{x.rec} = C_{x.enc} - K_x{}'$$

*Nota*: si el valor desencriptado es negativo, se debe sumar dicho valor a $p$ ($C_x = p + C_{x.rec}$).

Al igual que para el proceso de encriptación, se debe realizar el proceso inverso a cada campo de ambas coordenadas.

### 7.4.4 Ordenamiento de los bits

Se ubican los bits menos significativos, o sea, los correspondientes a la información de los segundos al comienzo y a final del paquete, ya que es la parte del mensaje con más probabilidad de ser perdida, pudiendo comenzar a muestrear el audio tardíamente o pudiendo no terminar de grabarlo por completo.

Este tiene un total de 41 bits formado por dos palabras, una correspondiente a la Latitud (A) y otra a la Longitud (B). Los bits se alternan, ya que de esta manera es menos probable que se pierda más información en una palabra que en la otra. A continuación se presenta la secuencia de bits que contiene la información a enviar (Figura 7.1).

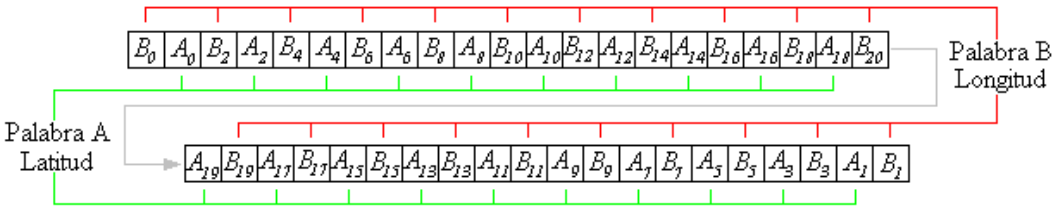



Figura 7.1.

## 7.5 Ejemplo

Este ejemplo sirve para afianzar lo explicado en este capítulo. Se toman valores de coordenadas al azar y se desarrolla el proceso completo hasta obtener la palabra binaria a enviar ya cifrada.

Los valores escogidos son los siguientes:

- Latitud: +57° 34’ 12’’
- Longitud: -125° 21’ 47’’

Utilizaremos la siguiente llave: IUA

Se escogió una llave de tres letras, la cual debe cifrar los campos correspondientes a Latitud y Longitud.

A la llave elegida se le toma el código ASCII de cada letra en decimal y se suma a cada campo de las coordenadas.

| Letra | ASCII decimal |
|---|---|
| I | 73 |
| U | 85 |
| A | 65 |

Lo que se hace es sumar el código ASCII en decimal de cada letra de la llave a cada campo de la coordenada, tanto para Latitud como Longitud, de la siguiente manera:

- Latitud: $\pm G_{Lat}{}^{o} M_{Lat}{}' S_{Lat}{}''$

Valor original: +57° 34’ 12’’

$$G_{Lat} = 57$$
$$M_{Lat} = 34$$
$$S_{Lat} = 12$$

Cálculo de los nuevos valores:

$$\pm G_{Lat}{}^{o} M_{Lat}{}' S_{Lat}{}''$$
$$\updownarrow \quad \updownarrow \quad \updownarrow$$
$$I \quad U \quad A$$

$$G_{Lat.enc} = (57 + 73(I)) \bmod 90 = 40$$
$$M_{Lat.enc} = (34 + 85(U)) \bmod 60 = 59$$
$$S_{Lat.enc} = (12 + 65(A)) \bmod 60 = 17$$

*Nota*: si el valor del campo de coordenada correspondiente a los Grados sumado al valor del código ASCII de la primera letra de la llave supera $n$ veces 90, se debe cambiar el signo correspondiente al indicador del hemisferio multiplicando el mismo por $(-1)^n$.

Valor de Latitud encriptado: -40º 59’ 17’’.

- Longitud: $\pm G_{Lon}{}^{o} M_{Lon}{}' S_{Lon}{}''$

Valor original: -125º 21’ 17’’

$$G_{Lon} = 125$$
$$M_{Lon} = 21$$
$$S_{Lon} = 47$$

Cálculo de los nuevos valores:

$$\pm G_{Lon}{}^{o} M_{Lon}{}' S_{Lon}{}''$$
$$\updownarrow \quad \updownarrow \quad \updownarrow$$
$$I \quad U \quad A$$

$$G_{Lon.enc} = (125 + 73(I)) \bmod 180 = 18$$
$$M_{Lon.enc} = (21 + 85(U)) \bmod 60 = 46$$
$$S_{Lon.enc} = (47 + 65(A)) \bmod 60 = 52$$

*Nota*: si el valor del campo de coordenada correspondiente a los Grados sumado al valor del código ASCII de la primera letra de la llave supera $n$ veces 180, se debe cambiar el signo correspondiente al indicador del hemisferio multiplicando el mismo por $(-1)^n$.

Valor de Longitud encriptado: +18º 46’ 52’’.

Palabra A:

Valor de Latitud encriptado: -40º 59’ 17’’.

Pasamos a binario la información de cada campo:

- $- \rightarrow 1$
  $40 \rightarrow 0101000$
  $59 \rightarrow 111011$
  $17 \rightarrow 010001$

| $A_{19}$ | $A_{18}$ | $A_{17}$ | $A_{16}$ | $A_{15}$ | $A_{14}$ | $A_{13}$ | $A_{12}$ | $A_{11}$ | $A_{10}$ | $A_9$ | $A_8$ | $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

$\pm$ $G_{Lat.enc}$ $M_{Lat.enc}$ $S_{Lat.enc}$

Palabra B:

Valor de Longitud encriptado: +18º 46’ 52’’.

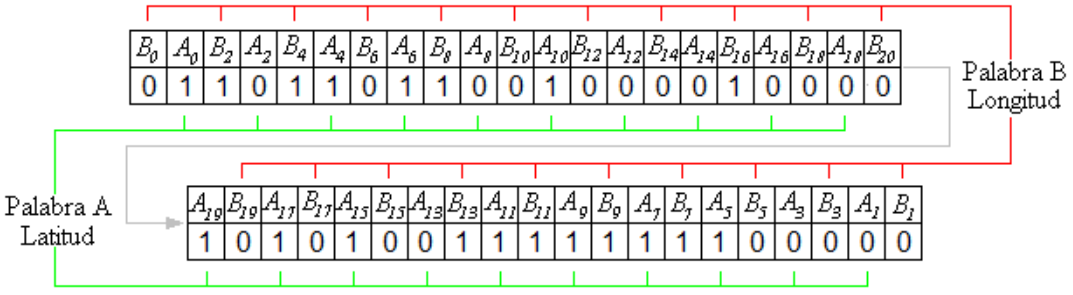
Pasamos a binario la información de cada campo:

- $+ \rightarrow 0$
- $18 \rightarrow 00010010$
- $46 \rightarrow 101110$
- $52 \rightarrow 110100$

| $B_{20}$ | $B_{19}$ | $B_{18}$ | $B_{17}$ | $B_{16}$ | $B_{15}$ | $B_{14}$ | $B_{13}$ | $B_{12}$ | $B_{11}$ | $B_{10}$ | $B_9$ | $B_8$ | $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

$\pm$ $G_{Lon.enc}$ $M_{Lon.enc}$ $S_{Lon.enc}$

Secuencia de bits a enviar.



Una vez codificada la información, la señal generada según el patrón de bits a enviar es sumada al archivo de audio y enviada.

En el receptor se lleva a cabo un procedimiento inverso al que se utilizó para generar el mensaje binario luego de haber extraído la marca de agua digital utilizando un sistema de dos estados basado en resonancia estocástica. Los bits son reordenados, desencriptados utilizando la misma llave que se usó para encriptar, e interpretados por el usuario.

# CONCLUSIONES

# 8 CONCLUSIONES

El análisis y demostración de la existencia del fenómeno de resonancia estocástica es de gran interés en la actualidad, y su aplicación no se limita sólo al ámbito de la Ingeniería, sino también a la Física, Química y Biología. Lo esencial, y a la vez llamativo, del concepto de resonancia estocástica consiste en la posibilidad de mejorar la relación señal-ruido de un sistema mediante el agregado del ruido apropiado al mismo. En efecto, la adición de "la cantidad justa" de ruido al sistema, permite que se genere un efecto cooperativo entre el ruido y la señal, cuyo resultado consiste en un aumento de la relación señal-ruido de salida del sistema. La noción precisa de "cantidad justa" de ruido agregado al sistema fue convenientemente analizada en el desarrollo del presente TFG.

La descripción matemática del fenómeno de resonancia estocástica que se efectuó para el modelo físico de un sistema de dos estados, permitió establecer las relaciones, entre los parámetros del mismo, necesarias para poder configurarlo y adaptarlo a una aplicación concreta. Asimismo, esto hizo posible demostrar su viabilidad de construcción para una implementación en una plataforma digital, describiendo dicho sistema en lenguaje VHDL.

Por otro lado, la posibilidad de detectar señales débiles en medios altamente ruidosos sugirió la aplicación escogida, la cual se basa en el envío de información oculta en archivos de audio. Esto representa una aplicación altamente concurrente con el área de Telecomunicaciones aplicada a la Defensa. Se demostró que la combinación del método de resonancia estocástica con un algoritmo de encriptación apropiado permite alcanzar un alto nivel de seguridad en la transmisión de datos. Concretamente, la aplicación consistió en un detector de marcas de agua digitales, el cual trabaja como un filtro altamente selectivo en frecuencia. Su selectividad y sencillez a la hora de implementarlo, son dos ventajas muy importantes. La primera es importante ya que, al conocerse la frecuencia de la señal que se espera filtrar (se conoce la composición espectral de la señal que representa la marca de agua digital), permite sintonizar el receptor a dicha frecuencia con la finalidad de obtener una señal de salida con una menor cantidad de ruido. Esto permite trabajar con valores más reducidos de relación señal-ruido de entrada al sistema, lo cual es importante para enmascarar una señal con ruido. La segunda ventaja de esta aplicación es también de mucha importancia ya que la hace competente frente a sistemas existentes, como por ejemplo, los filtros digitales.

Por otro lado, existe la desventaja que es un sistema limitado, ya que trabaja a muy baja frecuencia. No obstante, esta desventaja no representó grandes complicaciones para el desarrollo del receptor y detector de marcas de agua digitales en archivos de audio. Teniendo en cuenta esto último, se describieron los componentes de dicho receptor en lenguaje VHDL, con la finalidad de analizar su implementación en una plataforma digital. A pesar de que el software desarrollado tuvo problemas de compatibilidad con el hardware disponible en la facultad imposibilitando embeber dichos códigos en la plataforma digital, se comprobó que la

sintaxis de los mismos es correcta, ya que se probó su buen desempeño a través de simulaciones en Simulink (el cual es una herramienta de software de Matlab); esto demuestra la viabilidad de construcción del receptor.

Demostrado lo anterior, se le buscó una utilización concreta para el sistema desarrollado. Esta consta en el envío de coordenadas del plano terrestre, las cuales, tanto la Longitud como la Latitud, se representaron con una palabra binaria (la cual conforma la marca de agua) a la cual se le aplicó un algoritmo de encriptación sencillo (su complejidad se puede incrementar según los requerimientos del usuario). Esto permitió demostrar que se puede alcanzar un alto nivel de seguridad para la transmisión de datos gracias a la combinación de ambas técnicas.

Finalmente, podemos entonces confirmar la alta viabilidad de utilizar el método de resonancia estocástica como una herramienta para la detección y el filtrado de señales en medios ruidosos, siendo esto de gran importancia en diversos tópicos relacionados al área de las Telecomunicaciones.

Queda signada para trabajos futuros la adaptación de los códigos VHDL, desarrollados a lo largo de este trabajo final, a una plataforma digital (FPGA); como así también la obtención de una expresión formal para la constante de ajuste $k$ de entrada al receptor basado en resonancia estocástica mencionada en el Capítulo 3.

# BIBLIOGRAFÍA

[1] R. Benzi, A. Sutera y A. Vulpiani. “The mechanism of stochastic resonance”, J. Phys. A: Math. Gen. 14 (1981) L453-L457.

[2] B. McNamra y K. Wiesenfeld. “Theory of stochastic resonance” ”, Phys. Rev. A 39 (1989) 9, 4854-4869.

[3] L. Gammaitoni, P. Häggi, P. Jung y F. Marchesoni. “Stochastic resonance”, Rev. of Moth. Phys. 70 (1998) 1, 223-287.

[4] S. Sun, M. Jiang, X. Liu, J. Wan y S. Zheng. “A Blind Audio Watermarking Based on Stochastic Resonance Signal Processor”. IEEE. 2008.

[5] S. Sun, S. Kwong, B. Lei y S. Zheng. “Digital watermarking based on stochastic resonance signal processor”. IEEE. 2007.

[6] H. Chen, P. Varshney, S. Kay y J. Muchels. “Theory of the stochastic resonance effect in signal detection: Part I—Fixed Detectors”. IEEE. 2007.

[7] H. Chen y P. Varshney. “Theory of the stochastic resonance effect in signal detection—Part II: Variable Detectors”. IEEE. 2008.

[8] Boyce & Di Prima. “Ecuaciones diferenciales y problemas con valores en la frontera”.

[9] Digilent Basys Board manual reference. By Digilent, http://www.digilent.com. 2006.

[10] Digilent PmodAD1 analog to digital module converter board manual refenrence. By Digilent, http://www.digilent.com. 2005

[11] Digilent PmodDA1 digital to analog module converter board manual refenrence. By Digilent, http://www.digilent.com. 2005.

[12] Volnei A. Pedroni. “Circuit design with VHDL”. Massachusetts Institute of Technology. 2004

# ANEXO

# ANEXO 1

En este anexo se incluye la demostración formal de la estabilidad de los puntos críticos de la ecuación (3.2) del Capítulo 3, Sección 3.2.1 "Análisis cualitativo del sistema en ausencia de señal y de ruido".

Obtengamos la solución de la ecuación (3.2). Obtenida la solución, se analizará formalmente la estabilidad de cada punto crítico tomando las condiciones iniciales necesarias y luego obteniendo el límite de dicha solución cuando $t \to \infty$. De esta forma se obtendrá el retrato de fase del sistema y se observará si las curvas $x(t)$ encontradas escapan (punto inestable) o se hacen asintóticas (punto estable) a los puntos críticos.

A continuación se busca una expresión para la solución de la ecuación diferencial (3.2). Comenzamos factorizando el polinomio en $x$:

$$\frac{dx}{dt} = ax - bx^3$$
$$\Rightarrow \frac{dx}{dt} = xb\left(\frac{a}{b} - x^2\right)$$
$$\Rightarrow \frac{dx}{dt} = xb\left(\sqrt{\frac{a}{b}} - x\right)\left(\sqrt{\frac{a}{b}} + x\right)$$

Por inspección, observamos que es posible encontrar las soluciones para esta ecuación diferencial por el método de variables separables, por lo tanto separamos variables:

$$\frac{dx}{xb\left(\sqrt{\frac{a}{b}} - x\right)\left(\sqrt{\frac{a}{b}} + x\right)} = dt \qquad (3.4)$$

$$\left(\frac{B_1}{xb} + \frac{B_2}{\left(\sqrt{\frac{a}{b}} - x\right)} + \frac{B_3}{\left(\sqrt{\frac{a}{b}} + x\right)}\right) dx = dt \qquad (3.5)$$

La ecuación (3.4) nos muestra la separación de variables. Como paso siguiente, se buscan los valores para las constantes $B_i$:

Para simplificar la notación, en adelante llamaremos:

$$\frac{1}{xb\left(\sqrt{\frac{a}{b}}-x\right)\left(\sqrt{\frac{a}{b}}+x\right)} \equiv g(x)$$

Reemplazamos en (3.4):

$$g(x)dx = dt$$

Ahora buscamos las constantes $B_i$:

$$B_1 = \lim_{x\to 0}\left[g(x)x\right]_{x=0}$$
$$B_1 = \frac{b}{a}$$
$$B_2 = \lim_{x\to \sqrt{\frac{a}{b}}}\left[g(x)x\right]_{x=\sqrt{\frac{a}{b}}}$$
$$B_2 = \frac{1}{2a}$$
$$B_3 = \lim_{x\to -\sqrt{\frac{a}{b}}}\left[g(x)x\right]_{x=-\sqrt{\frac{a}{b}}}$$
$$B_3 = -\frac{1}{2a}$$

Obtenidos los valores de las constantes, reemplazamos los mismos en (3.5):

$$\left(\frac{1/a}{x}+\frac{1/2a}{\left(\sqrt{\frac{a}{b}}-x\right)}+\frac{-1/2a}{\left(\sqrt{\frac{a}{b}}+x\right)}\right)dx = dt \qquad (3.6)$$

Resolviendo (3.5) obtenemos:

$$\frac{1}{a}\ln|x| - \frac{1}{2a}\ln\left|x-\sqrt{\frac{a}{b}}\right| - \frac{1}{2a}\ln\left|x+\sqrt{\frac{a}{b}}\right| = t + C$$

$$\frac{1}{a}\ln\left(\frac{|x|}{\left|x-\sqrt{\frac{a}{b}}\right|^{1/2}\left|x+\sqrt{\frac{a}{b}}\right|^{1/2}}\right) = t + C$$

$$\frac{1}{a}\ln\left(\frac{|x|}{\left|x^2-\frac{a}{b}\right|^{1/2}}\right) = t + C$$

$$\frac{|x|}{\left|x^2-\frac{a}{b}\right|^{1/2}} = e^C e^{at}$$

$$|x|^2 = \left|x^2-\frac{a}{b}\right| e^{2C} e^{2at}$$

Donde:

$$E = e^{2aC}$$

Por lo tanto nos queda la ecuación de la siguiente forma.

$$|x|^2 = \left|x^2-\frac{a}{b}\right| E e^{2at} \qquad (3.7)$$

A partir de la ecuación (3.7), analizamos los cuatro casos posibles planteados en el análisis preliminar:

Caso 1: $x > \sqrt{\frac{a}{b}}$

$$x^2 = \left( x^2 - \frac{a}{b} \right) E e^{2at}$$

$$\Rightarrow x^2 = \frac{a}{b} \left( \frac{E e^{2at}}{E e^{2at} - 1} \right)$$

$$\Rightarrow x = \sqrt{\frac{a}{b}} \sqrt{\frac{E e^{2at}}{E e^{2at} - 1}}$$

Si $t = 0$, $x(0) = x_0$ sabiendo que $x_0 > \sqrt{\frac{a}{b}}$, entonces podemos despejar $E$, resultando:

$$E = \frac{{x_0}^2}{{x_0}^2 - \frac{a}{b}}$$

Entonces:

$$x(t) = \sqrt{\frac{a}{b}} \sqrt{\frac{\left( \frac{{x_0}^2}{{x_0}^2 - \frac{a}{b}} \right) e^{2at}}{\left( \frac{{x_0}^2}{{x_0}^2 - \frac{a}{b}} \right) e^{2at} - 1}}$$

Lo cual satisface:

$$\lim_{t \to \infty} x(t) = \sqrt{\frac{a}{b}} \; ; \left( x_0 > \sqrt{\frac{a}{b}} \right)$$

Caso 2: $x < -\sqrt{\frac{a}{b}}$

$$x^2 = \left( x^2 - \frac{a}{b} \right) E e^{2at}$$

$$\Rightarrow x^2 = \frac{a}{b} \left( \frac{E e^{2at}}{E e^{2at} - 1} \right)$$

$$\Rightarrow x = -\sqrt{\frac{a}{b}} \sqrt{\frac{E e^{2at}}{E e^{2at} - 1}}$$

Si $t = 0$, $x(0) = x_0$ sabiendo que $x_0 < -\sqrt{\frac{a}{b}}$, entonces podemos despejar $E$, resultando:

$$E = \frac{{x_0}^2}{{x_0}^2 - \frac{a}{b}}$$

Entonces:

$$x(t) = -\sqrt{\frac{a}{b}} \sqrt{\frac{\left( \frac{{x_0}^2}{{x_0}^2 - \frac{a}{b}} \right) e^{2at}}{\left( \frac{{x_0}^2}{{x_0}^2 - \frac{a}{b}} \right) e^{2at} - 1}}$$

Lo cual satisface:

$$\lim_{t \to \infty} x(t) = -\sqrt{\frac{a}{b}} \; ; \; \left( x_0 < -\sqrt{\frac{a}{b}} \right)$$

Caso 3: $0 < x < \sqrt{\frac{a}{b}}$

$$x^2 = \left( \frac{a}{b} - x^2 \right) E e^{2at}$$

$$\Rightarrow x^2 = \frac{a}{b} \left( \frac{E e^{2at}}{E e^{2at} + 1} \right)$$

$$\Rightarrow x = \sqrt{\frac{a}{b}} \sqrt{\frac{E e^{2at}}{E e^{2at} + 1}}$$

Si $t = 0$, $x(0) = x_0$ sabiendo que $0 < x_0 < \sqrt{\frac{a}{b}}$, entonces podemos despejar $E$, resultando:

$$E = \frac{{x_0}^2}{\frac{a}{b} - {x_0}^2}$$

Entonces:

$$x(t) = \sqrt{\frac{a}{b}} \sqrt{\frac{\left( \frac{{x_0}^2}{\frac{a}{b} - {x_0}^2} \right) e^{2at}}{\left( \frac{{x_0}^2}{\frac{a}{b} - {x_0}^2} \right) e^{2at} + 1}}$$

Lo cual satisface:

$$\lim_{t \to \infty} x(t) = \sqrt{\frac{a}{b}} \; ; \left( 0 < x_0 < \sqrt{\frac{a}{b}} \right)$$

Caso 4: $0 > x > -\sqrt{\frac{a}{b}}$

$$x^2 = \left(\frac{a}{b} - x^2\right) Ee^{2at}$$

$$\Rightarrow x^2 = \frac{a}{b}\left(\frac{Ee^{2at}}{Ee^{2at}+1}\right)$$

$$\Rightarrow x = -\sqrt{\frac{a}{b}}\sqrt{\frac{Ee^{2at}}{Ee^{2at}+1}}$$

Si $t = 0$, $x(0) = x_0$ sabiendo que $0 > x_0 > -\sqrt{\frac{a}{b}}$, entonces podemos despejar $E$, resultando:

$$E = \frac{{x_0}^2}{\frac{a}{b} - {x_0}^2}$$

Entonces:

$$x(t) = -\sqrt{\frac{a}{b}}\sqrt{\frac{\left(\frac{{x_0}^2}{\frac{a}{b} - {x_0}^2}\right)e^{2at}}{\left(\frac{{x_0}^2}{\frac{a}{b} - {x_0}^2}\right)e^{2at}+1}}$$

Lo cual satisface:

$$\lim_{t\to\infty} x(t) = -\sqrt{\frac{a}{b}}\ ;\ \left(0 > x_0 > -\sqrt{\frac{a}{b}}\right)$$

Del anterior análisis de los casos 3 y 4 se concluye que $x = 0$ es un punto crítico inestable, ya que estableciendo como condición inicial $x_0$, tal que $x_0 = \varepsilon$, donde $0 < |\varepsilon| \ll 1$, al tomar límite para $t \to \infty$, entonces la solución tenderá a $\sqrt{\frac{a}{b}}$ cuando $\varepsilon > 0$, y a $-\sqrt{\frac{a}{b}}$ cuando $\varepsilon < 0$.

Además, el análisis conjunto de los casos 1 y 2 permite afirmar que estableciendo como condición inicial cualquier punto $x_0$, tal que $\left|x_0 - \sqrt{\frac{a}{b}}\right| = \varepsilon$ donde $0 < |\varepsilon| \ll 1$, entonces el límite $t \to \infty$ de la solución es igual a $\sqrt{\frac{a}{b}}$. Análogamente, los casos 2 y 4 permiten afirmar que estableciendo como condición inicial cualquier punto $x_0$, tal que $\left|x_0 + \sqrt{\frac{a}{b}}\right| = \varepsilon$ donde $0 < |\varepsilon| \ll 1$, entonces el límite $t \to \infty$ de la solución es igual a $-\sqrt{\frac{a}{b}}$. Esto confirma el análisis cualitativo realizado previamente, en base al signo y valor de $\dot{x}$ y $x$.

# ANEXO 2

En el presente Anexo se añaden los códigos fuente en lenguaje VHDL del análisis de la implementación.

Código VHDL conversor analógico a digital. Archivo *AD1RefComp.vhd*.

```
----------------------------------------------------------------------------------
--    AD1RefComp.VHD
----------------------------------------------------------------------------------
--    Author          : Ioana Dabacan
-- CopyRight 2008 Digilent Ro.
----------------------------------------------------------------------------------
--    Description : This file is the VHDL code for a PMOD-AD1 controller.
--
----------------------------------------------------------------------------------
--    Revision History:
-- Feb/29/2008 Created    (Ioana Dabacan)
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

----------------------------------------------------------------------------------
--Title          : AD1 Reference Component
--
--    Inputs          : 5
--    Outputs         : 5
--
--    Description: This is the AD1 Reference Component entity. The input
--                        ports are a 50 MHz clock and an asynchronous reset
--                      button along with the data from the ADC7476 that
```

```
--                          is serially shifted in on each clock
cycle(SDATA1 and
--                       SDATA2). The outputs are the SCLK signal
which clocks
--                        the PMOD-AD1 board at 12.5 MHz and a
chip select
--                         signal (nCS) that enables the ADC7476
chips on the
--                        PMOD-AD1 board as well as two 12-bit
output
--                      vectors labeled DATA1 and DATA2 which can
be used by
--                     any external components. The START is used
to tell
--                      the component when to start a conversion.
After a
--                      conversion is done the component activates
the DONE
--                     signal.
--
----------------------------------------------------------------
--------------------------------

entity AD1RefComp is
  Port    (
  --General usage
    CLK       : in std_logic;
    RST       : in std_logic;

  --Pmod interface signals
    SDATA1    : in std_logic;
    SCLK      : out std_logic;
    nCS       : out std_logic;

    --User interface signals
    DATA1     : out std_logic_vector(11 downto 0);
    START     : in std_logic;
    DONE      : out std_logic
             );
```

```
end AD1RefComp ;

architecture AD1 of AD1RefComp is

-------------------------------------------------------------------------------
-- Title             : Local signal assignments
--
-- Description       : The following signals will be used to drive the
--                      processes of this VHDL file.
--
--     current_state : This signal will be the pointer that will point at the
--                     current state of the Finite State Machine of the
--                     controller.
--     next_state    : This signal will be the pointer that will point at the
--                     current state of the Finite State Machine of the
--                     controller.
--     temp1         : This is a 16-bit vector that will store the 16-bits of data
--                         that are serially shifted-in form the first ADC7476 chip
--                     inside the PMOD-AD1 board.
--     temp2         : This is a 16-bit vector that will store the 16-bits of data
--                         that are serially shifted-in form the second ADC7476 chip
--                     inside the PMOD-AD1 board.
--     clk_div       : This will be the divided 12.5 MHz clock signal that will
--                     clock the PMOD-AD1 board
--     clk_counter   : This counter will be used to create a divided clock signal.
--
--     shiftCounter  : This counter will be used to count the shifted data from
```

```
--                          the ADC7476 chip inside the PMOD-AD1
board.
--      enShiftCounter: This signal will be used to enable the
counter for the
--                              shifted data from the ADC7476 chip
inside the PMOD-AD1.
--    enParalelLoad : This signal will be used to enable the load
the shifted
--                          data in a register.
-------------------------------------------------------------------
-------------------------------------

type states is (Idle,
                ShiftIn,
                SyncData);
          signal current_state : states;
          signal next_state    : states;

          signal temp1            : std_logic_vector(15 downto
0);
          signal clk_div       : std_logic;
          signal clk_counter   : std_logic_vector(1 downto 0);
          signal shiftCounter   : std_logic_vector(3 downto 0)
:= x"0";
          signal enShiftCounter: std_logic;
          signal enParalelLoad : std_logic;


begin

-------------------------------------------------------------------
-------------------------------------
-- Title           :      clock divider process
--
-- Description     : This is the process that will divide the 50
MHz clock
--                          down to a clock speed of 12.5 MHz to
drive the ADC7476 chip.
-------------------------------------------------------------------
-------------------------------------
```

```
        clock_divide : process(rst,clk)
        begin
            if rst = '1' then
                clk_counter <= "00";
            elsif (clk = '1' and clk'event) then
                clk_counter <= clk_counter + '1';
            end if;
        end process;

        clk_div <= clk_counter(1);
        SCLK <=  not clk_counter(1);

------------------------------------------------------------
------------------------------------------------------------
--
-- Title      : counter
--
-- Description: This is the process were the converted data will
be colected and
--                output.When the enShiftCounter is activated,
the 16-bits of data
--               from the ADC7476 chips will be shifted inside
the temporary
--                registers. A 4-bit counter is used to keep
shifting the data
--                inside temp1 and temp2 for 16 clock cycles.
When the enParalelLoad
--               signal is generated inside the SyncData state,
the converted data
--               in the temporary shift registers will be placed
on the outputs
--              DATA1 and DATA2.
--
------------------------------------------------------------
------------------------------------------------------------

counter : process(clk_div, enParalelLoad, enShiftCounter)
        begin
            if (clk_div = '1' and clk_div'event) then
```

```
                if (enShiftCounter = '1') then
                    temp1 <= temp1(14 downto 0) & SDATA1;
                     shiftCounter <= shiftCounter + '1';
                elsif (enParalelLoad = '1') then
                    shiftCounter <= "0000";
                    DATA1 <= temp1(11 downto 0);
                end if;
            end if;
        end process;

-------------------------------------------------------------------------------
--
-- Title       : Finite State Machine
--
-- Description: This 3 processes represent the FSM that contains three states.
--                  The first state is the Idle state in which a temporary registers
--                  are assigned the updated value of the input "DATA1" and "DATA2".
--                 The next state is the ShiftIn state where the 16-bits of data
--                  from each of the ADCS7476 chips are left shifted in the temp1
--                  and temp2 shift registers. The third state, SyncData drives the
--                  output signal nCS high for 1 clock period maintainig nCS high
--                also in the Idle state telling the ADCS7476 to mark the end of
--                the conversion.
-- Notes:             The data will change on the lower edge of the clock signal. There
--                         is also an asynchronous reset that will reset all signals to
--                        their original state.
--
-------------------------------------------------------------------------------
```

```
------------------------------------------------------------
------------------------------------------------------------
--
-- Title       : SYNC_PROC
--
-- Description: This is the process were the states are changed
synchronously. At
--                reset the current state becomes Idle state.
--
------------------------------------------------------------
------------------------------------------------------------
SYNC_PROC: process (clk_div, rst)
   begin
      if (clk_div'event and clk_div = '1') then
         if (rst = '1') then
            current_state <= Idle;
         else
            current_state <= next_state;
         end if;
      end if;
   end process;

------------------------------------------------------------
------------------------------------------------------------
--
-- Title       : OUTPUT_DECODE
--
-- Description: This is the process were the output signals are
generated
--                    unsynchronously based on the state only
(Moore State Machine).
--
------------------------------------------------------------
------------------------------------------------------------
OUTPUT_DECODE: process (current_state)
   begin
      if current_state = Idle then
            enShiftCounter <='0';
            DONE <='1';
```

```
            nCS <='1';
            enParalelLoad <= '0';
        elsif current_state = ShiftIn then
            enShiftCounter <='1';
            DONE <='0';
            nCS <='0';
            enParalelLoad <= '0';
        else --if current_state = SyncData then
            enShiftCounter <='0';
            DONE <='0';
            nCS <='1';
            enParalelLoad <= '1';
        end if;
    end process;

-------------------------------------------------------------
-------------------------------------
--
-- Title      : NEXT_STATE_DECODE
--
-- Description: This is the process were the next state logic is
generated
--                depending on the current state and the input
signals.
--
-------------------------------------------------------------
-------------------------------------
    NEXT_STATE_DECODE:    process    (current_state,    START,
shiftCounter)
    begin

       next_state <= current_state;   -- default is to stay in
current state

       case (current_state) is
          when Idle =>
             if START = '1' then
                next_state <= ShiftIn;
             end if;
          when ShiftIn =>
```

```
            if shiftCounter = x"F" then
                next_state <= SyncData;
            end if;
         when SyncData =>
            if START = '0' then
            next_state <= Idle;
            end if;
         when others =>
            next_state <= Idle;
      end case;
   end process;


end AD1;
```

Código VHDL conversor digital a analógico. Archivo *DA2RefComp.vhd*.

```
----------------------------------------------------------------------------------
--    DA2 Reference Component
----------------------------------------------------------------------------------
--    Author    :    Ioana Dabacan
--    CopyRight 2008 Digilent Ro.
----------------------------------------------------------------------------------
--     Desription     :     This file is the VHDL code for a
PMOD-DA2 controller.
--
----------------------------------------------------------------------------------
--    Revision History:
--    Feb/29/2008 (Created)    Ioana Dabacan
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
use IEEE.NUMERIC_STD.ALL;

----------------------------------------------------------------------------------
--
-- Title       : DA1 controller entity
--
-- Inputs      : 5
-- Outputs     : 5
--
-- Description: This is the DA2 Reference Component entity. The input ports are
--                a 50MHz clock and and an asynchronous reset button along with the
--                     data to be serially shifted in the 2 DAC121S101 chips on a DA2
--                      Pmod on each clock cycle.There is also a signal to start a
--               conversion.
--                The outputs of this entity are: a output clock signal, two serial
--                  output signals D1 and D2, a sync signal to synchronize the data
--                in the DAC121S101 chip, a done signal to tell that the chip is
--                 done converting the data and another set of data can be sent.
--
----------------------------------------------------------------------------------

entity DA2RefComp is
    Port (

     --General usage
       CLK       : in std_logic;     -- System Clock (50MHz)
       RST       : in std_logic;

     --Pmod interface signals
       DOUT        : out std_logic;
```

```
        CLK_OUT  : out std_logic;
        nSYNC    : out std_logic;

      --User interface signals
        DATA2    : in std_logic_vector(11 downto 0);
        START2   : in std_logic;
        DONE2    : out std_logic

         );
end DA2RefComp ;

architecture DA2 of DA2RefComp is


--       control constant: Normal Operation
          constant control      : std_logic_vector(3 downto 0)
:= "0000";
-------------------------------------------------------------
-------------------------------------------------------------
--     Title       : signal assignments
--
-- Description: The following signals are enumerated signals for
the
--                finite state machine,2 temporary vectors to be
shifted out to the
--                 DAC121S101 chips, a divided clock signal to
drive the DAC121S101 chips,
--                a counter to divide the internal 50 MHz clock
signal,
--                 a 4-bit counter to be used to shift out the
16-bit register,
--                 and 2 enable signals for the paralel load and
shift of the
--               shift register.
--
-------------------------------------------------------------
-------------------------------------------------------------
          type states is (Idle,
                          ShiftOut,
                          SyncData);
```

```
            signal current_state : states;
            signal next_state    : states;

            signal temp1            : std_logic_vector(15 downto
0);
            signal clk_div        : std_logic;
            signal clk_counter    : std_logic_vector(27 downto 0);
            signal shiftCounter   : std_logic_vector(3 downto 0);
            signal enShiftCounter: std_logic;
            signal enParalelLoad : std_logic;



begin



------------------------------------------------------------
------------------------------------------------------------
--
-- Title      : Clock Divider
--
-- Description: The following process takes a 50 MHz clock and
divides it down to a
--                  25 MHz clock signal by assigning the signals
clk_out and clk_div
--                    to the 2nd bit of the clk_counter vector.
clk_div is used by
--                  the Finite State Machine and clk_out is used
by the DAC121S101 chips.
--
------------------------------------------------------------
------------------------------------------------------------

         clock_divide : process(rst,clk)
         begin
              if rst = '1' then
                   clk_counter                                    <=
"00000000000000000000000000";
              elsif (clk = '1' and clk'event) then
```

```
                clk_counter <= clk_counter + '1';
            end if;
        end process;

        clk_div <= clk_counter(0);
        clk_out <= clk_counter(0);


------------------------------------------------------------
------------------------------------------------------------
--
-- Title      : counter
--
-- Description: This is the process were the teporary registers
will be loaded and
--                      shifted.When the enParalelLoad signal is
generated inside the state
--                  the temp1 and temp2 registers will be loaded
with the 8 bits of control
--                  concatenated with the 8 bits of data. When
the enShiftCounter is
--                    activated, the 16-bits of data inside the
temporary registers will be
--                    shifted. A 4-bit counter is used to keep
shifting the data
--                inside temp1 and temp 2 for 16 clock cycles.
--
------------------------------------------------------------
------------------------------------------------------------

counter : process(clk_div, enParalelLoad, enShiftCounter)
        begin
            if (clk_div = '1' and clk_div'event) then
                if enParalelLoad = '1' then
                    shiftCounter <= "0000";
                    temp1 <= control & DATA2;
                elsif (enShiftCounter = '1') then
                    temp1 <= temp1(14 downto 0)&temp1(15);
                    shiftCounter <= shiftCounter + '1';
                end if;
```

```
            end if;
        end process;

                    DOUT <= temp1(15);


-------------------------------------------------------------------------------
--
-- Title      : Finite State Machine
--
-- Description: This 3 processes represent the FSM that contains three states.
--                    First one is the Idle state in which the temporary registers are
--                    assigned the updated value of the input "DATA1" and "DATA2".
--                The next state is the ShiftOut state which is the state where the
--                16-bits of temporary registers are shifted out left from the MSB
--                    to the two serial outputs, D1 and D2. Immediately following the
--                second state is the third state SyncData. This state drives the
--                    output signal sync high for2 clock signals telling the DAC121S101
--                to latch the 16-bit data it just recieved in the previous state.
-- Notes:          The data will change on the upper edge of the clock signal. Their
--                is also an asynchronous reset that will reset all signals to their
--                original state.
--
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
```

```
--
-- Title       : SYNC_PROC
--
-- Description: This is the process were the states are changed
synchronously. At
--                reset the current state becomes Idle state.
--
-------------------------------------------------------------
-------------------------------------------------------------
SYNC_PROC: process (clk_div, rst)
   begin
      if (clk_div'event and clk_div = '1') then
         if (rst = '1') then
            current_state <= Idle;
         else
            current_state <= next_state;
         end if;
      end if;
   end process;

-------------------------------------------------------------
-------------------------------------------------------------
--
-- Title       : OUTPUT_DECODE
--
-- Description: This is the process were the output signals are
generated
--                    unsynchronously based on the state only
(Moore State Machine).
--
-------------------------------------------------------------
-------------------------------------------------------------
OUTPUT_DECODE: process (current_state)
   begin
      if current_state = Idle then
            enShiftCounter <='0';
            DONE2 <='1';
            nSYNC <='1';
            enParalelLoad <= '1';
        elsif current_state = ShiftOut then
```

```
            enShiftCounter <='1';
            DONE2 <='0';
            nSYNC <='0';
            enParalelLoad <= '0';
        else --if current_state = SyncData then
            enShiftCounter <='0';
            DONE2 <='0';
            nSYNC <='1';
            enParalelLoad <= '0';
        end if;
    end process;

-------------------------------------------------------------
-------------------------------------------------------------
--
-- Title      : NEXT_STATE_DECODE
--
-- Description: This is the process were the next state logic is
generated
--              depending on the current state and the input
signals.
--
-------------------------------------------------------------
-------------------------------------------------------------
    NEXT_STATE_DECODE: process (current_state, START2,
shiftCounter)
    begin

      next_state <= current_state;  --default is to stay in
current state

      case (current_state) is
         when Idle =>
            if START2 = '1' then
               next_state <= ShiftOut;
            end if;
         when ShiftOut =>
            if shiftCounter = x"F" then
               next_state <= SyncData;
            end if;
```

```
            when SyncData =>
               if START2 = '0' then
               next_state <= Idle;
               end if;
            when others =>
               next_state <= Idle;
         end case;
      end process;


end DA2;
```

Código VHDL del detector basado en resonancia estocástica. Archivo *fpga_detector_basadosr_clk6250ns_06_cw*.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use work.conv_pkg.all;
-- synopsys translate_off
library unisim;
use unisim.vcomponents.all;
-- synopsys translate_on
entity xlclockdriver is
  generic (
    period: integer := 2;
    log_2_period: integer := 0;
    pipeline_regs: integer := 5;
    use_bufg: integer := 0
  );
  port (
    sysclk: in std_logic;
    sysclr: in std_logic;
    sysce: in std_logic;
    clk: out std_logic;
    clr: out std_logic;
    ce: out std_logic
  );
end xlclockdriver;
architecture behavior of xlclockdriver is
  component bufg
```

```
    port (
      i: in std_logic;
      o: out std_logic
    );
  end component;
  component synth_reg_w_init
    generic (
      width: integer;
      init_index: integer;
      init_value: bit_vector;
      latency: integer
    );
    port (
      i: in std_logic_vector(width - 1 downto 0);
      ce: in std_logic;
      clr: in std_logic;
      clk: in std_logic;
      o: out std_logic_vector(width - 1 downto 0)
    );
  end component;
  function size_of_uint(inp: integer; power_of_2: boolean)
    return integer
  is
    constant inp_vec: std_logic_vector(31 downto 0) :=
      integer_to_std_logic_vector(inp,32, xlUnsigned);
    variable result: integer;
  begin
    result := 32;
    for i in 0 to 31 loop
      if inp_vec(i) = '1' then
        result := i;
      end if;
    end loop;
    if power_of_2 then
      return result;
    else
      return result+1;
    end if;
  end;
  function is_power_of_2(inp: std_logic_vector)
```

```
    return boolean
is
    constant width: integer := inp'length;
    variable vec: std_logic_vector(width - 1 downto 0);
    variable single_bit_set: boolean;
    variable more_than_one_bit_set: boolean;
    variable result: boolean;
begin
    vec := inp;
    single_bit_set := false;
    more_than_one_bit_set := false;
    -- synopsys translate_off
    if (is_XorU(vec)) then
      return false;
    end if;
     -- synopsys translate_on
    if width > 0 then
      for i in 0 to width - 1 loop
        if vec(i) = '1' then
          if single_bit_set then
            more_than_one_bit_set := true;
          end if;
          single_bit_set := true;
        end if;
      end loop;
    end if;
    if (single_bit_set and not(more_than_one_bit_set)) then
      result := true;
    else
      result := false;
    end if;
    return result;
end;
function ce_reg_init_val(index, period : integer)
    return integer
is
     variable result: integer;
  begin
      result := 0;
      if ((index mod period) = 0) then
```

```
            result := 1;
        end if;
        return result;
    end;
    function remaining_pipe_regs(num_pipeline_regs, period :
integer)
      return integer
    is
       variable factor, result: integer;
    begin
        factor := (num_pipeline_regs / period);
        result := num_pipeline_regs - (period * factor) + 1;
        return result;
    end;

    function sg_min(L, R: INTEGER) return INTEGER is
    begin
        if L < R then
                return L;
        else
                return R;
        end if;
    end;
    constant max_pipeline_regs : integer := 8;
    constant pipe_regs : integer := 5;
    constant num_pipeline_regs : integer := sg_min(pipeline_regs,
max_pipeline_regs);
    constant rem_pipeline_regs : integer :=
remaining_pipe_regs(num_pipeline_regs,period);
    constant period_floor: integer := max(2, period);
    constant power_of_2_counter: boolean :=
      is_power_of_2(integer_to_std_logic_vector(period_floor,32,
xIUnsigned));
    constant cnt_width: integer :=
      size_of_uint(period_floor, power_of_2_counter);
    constant clk_for_ce_pulse_minus1: std_logic_vector(cnt_width
- 1 downto 0) :=
      integer_to_std_logic_vector((period_floor - 2),cnt_width,
xIUnsigned);
```

```
  constant clk_for_ce_pulse_minus2: std_logic_vector(cnt_width
- 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period - 3),cnt_width,
xlUnsigned);
  constant clk_for_ce_pulse_minus_regs:
std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period -
rem_pipeline_regs),cnt_width, xlUnsigned);
  signal clk_num: unsigned(cnt_width - 1 downto 0) := (others
=> '0');
  signal ce_vec : std_logic_vector(num_pipeline_regs downto 0);
  attribute MAX_FANOUT : string;
  attribute MAX_FANOUT of ce_vec:signal is "REDUCE";
  signal internal_ce: std_logic_vector(0 downto 0);
  signal cnt_clr, cnt_clr_dly: std_logic_vector (0 downto 0);
begin
  clk <= sysclk;
  clr <= sysclr;
  cntr_gen: process(sysclk)
  begin
    if sysclk'event and sysclk = '1' then
      if (sysce = '1') then
        if ((cnt_clr_dly(0) = '1') or (sysclr = '1')) then
          clk_num <= (others => '0');
        else
          clk_num <= clk_num + 1;
        end if;
    end if;
    end if;
  end process;
  clr_gen: process(clk_num, sysclr)
  begin
    if power_of_2_counter then
      cnt_clr(0) <= sysclr;
    else
      if (unsigned_to_std_logic_vector(clk_num) =
clk_for_ce_pulse_minus1
          or sysclr = '1') then
        cnt_clr(0) <= '1';
      else
```

```
          cnt_clr(0) <= '0';
        end if;
      end if;
    end process;
    clr_reg: synth_reg_w_init
      generic map (
        width => 1,
        init_index => 0,
        init_value => b"0000",
        latency => 1
      )
      port map (
        i => cnt_clr,
        ce => sysce,
        clr => sysclr,
        clk => sysclk,
        o => cnt_clr_dly
      );
    pipelined_ce : if period > 1 generate
        ce_gen: process(clk_num)
        begin
            if      unsigned_to_std_logic_vector(clk_num)       =
clk_for_ce_pulse_minus_regs then
                ce_vec(num_pipeline_regs) <= '1';
            else
                ce_vec(num_pipeline_regs) <= '0';
            end if;
        end process;
        ce_pipeline:  for  index  in  num_pipeline_regs  downto  1
generate
            ce_reg : synth_reg_w_init
                generic map (
                    width => 1,
                    init_index => ce_reg_init_val(index, period),
                    init_value => b"0000",
                    latency => 1
                    )
                port map (
                    i => ce_vec(index downto index),
                    ce => sysce,
```

```
                    clr => sysclr,
                    clk => sysclk,
                    o => ce_vec(index-1 downto index-1)
                    );
      end generate;
      internal_ce <= ce_vec(0 downto 0);
  end generate;
  use_bufg_true: if period > 1 and use_bufg = 1 generate
    ce_bufg_inst: bufg
      port map (
        i => internal_ce(0),
        o => ce
      );
  end generate;
  use_bufg_false: if period > 1 and (use_bufg = 0) generate
    ce <= internal_ce(0);
  end generate;
  generate_system_clk: if period = 1 generate
    ce <= sysce;
  end generate;
end architecture behavior;
library IEEE;
use IEEE.std_logic_1164.all;
use work.conv_pkg.all;

entity default_clock_driver is
  port (
    sysce: in std_logic;
    sysce_clr: in std_logic;
    sysclk: in std_logic;
    ce_1: out std_logic;
    clk_1: out std_logic
  );
end default_clock_driver;

architecture structural of default_clock_driver is
  attribute syn_noprune: boolean;
  attribute syn_noprune of structural : architecture is true;
  attribute optimize_primitives: boolean;
```

```
  attribute optimize_primitives of structural : architecture is
false;
  attribute dont_touch: boolean;
  attribute dont_touch of structural : architecture is true;

  signal sysce_clr_x0: std_logic;
  signal sysce_x0: std_logic;
  signal sysclk_x0: std_logic;
  signal xlclockdriver_1_ce: std_logic;
  signal xlclockdriver_1_clk: std_logic;

begin
  sysce_x0 <= sysce;
  sysce_clr_x0 <= sysce_clr;
  sysclk_x0 <= sysclk;
  ce_1 <= xlclockdriver_1_ce;
  clk_1 <= xlclockdriver_1_clk;

  xlclockdriver_1: entity work.xlclockdriver
    generic map (
      log_2_period => 1,
      period => 1,
      use_bufg => 0
    )
    port map (
      sysce => sysce_x0,
      sysclk => sysclk_x0,
      sysclr => sysce_clr_x0,
      ce => xlclockdriver_1_ce,
      clk => xlclockdriver_1_clk
    );

end structural;
library IEEE;
use IEEE.std_logic_1164.all;
use work.conv_pkg.all;

entity fpga_detector_basadosr_rx_clk6250ns_06_cw is
  port (
    ce: in std_logic := '1';
```

```
    clk: in std_logic; -- clock period = 6250.0 ns (0.16 Mhz)
    gateway_in: in std_logic_vector(11 downto 0);
    gateway_out: out std_logic_vector(11 downto 0)
  );
end fpga_detector_basadosr_rx_clk6250ns_06_cw;

architecture structural of
fpga_detector_basadosr_rx_clk6250ns_06_cw is
  component xlpersistentdff
    port (
      clk: in std_logic;
      d: in std_logic;
      q: out std_logic
    );
  end component;
  attribute syn_black_box: boolean;
  attribute syn_black_box of xlpersistentdff: component is true;
  attribute box_type: string;
  attribute box_type of xlpersistentdff: component is
"black_box";
  attribute syn_noprune: boolean;
  attribute optimize_primitives: boolean;
  attribute dont_touch: boolean;
  attribute syn_noprune of xlpersistentdff: component is true;
  attribute optimize_primitives of xlpersistentdff: component is
false;
  attribute dont_touch of xlpersistentdff: component is true;

  signal ce_1_sg_x0: std_logic;
  attribute MAX_FANOUT: string;
  attribute MAX_FANOUT of ce_1_sg_x0: signal is "REDUCE";
  signal clkNet: std_logic;
  signal clk_1_sg_x0: std_logic;
  signal gateway_in_net: std_logic_vector(11 downto 0);
  signal gateway_out_net: std_logic_vector(11 downto 0);
  signal persistentdff_inst_q: std_logic;
  attribute syn_keep: boolean;
  attribute syn_keep of persistentdff_inst_q: signal is true;
  attribute keep: boolean;
  attribute keep of persistentdff_inst_q: signal is true;
```

```
  attribute preserve_signal: boolean;
  attribute preserve_signal of persistentdff_inst_q: signal is true;

begin
  clkNet <= clk;
  gateway_in_net <= gateway_in;
  gateway_out <= gateway_out_net;

  default_clock_driver_x0: entity work.default_clock_driver
    port map (
      sysce => '1',
      sysce_clr => '0',
      sysclk => clkNet,
      ce_1 => ce_1_sg_x0,
      clk_1 => clk_1_sg_x0
    );

  fpga_detector_basadosr_rx_clk6250ns_06_x0: entity
work.fpga_detector_basadosr_rx_clk6250ns_06
    port map (
      ce_1 => ce_1_sg_x0,
      clk_1 => clk_1_sg_x0,
      gateway_in => gateway_in_net,
      gateway_out => gateway_out_net
    );

  persistentdff_inst: xlpersistentdff
    port map (
      clk => clkNet,
      d => persistentdff_inst_q,
      q => persistentdff_inst_q
    );

end structural;
```

A continuación se presentan los códigos VHDL de los componentes del detector.

Código VHDL del generador de reloj. Archivo *default_clock_driver_x0.vhd*.

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
use IEEE.numeric_std.all;
use work.conv_pkg.all;
-- synopsys translate_off
library unisim;
use unisim.vcomponents.all;
-- synopsys translate_on
entity xlclockdriver is
  generic (
    period: integer := 2;
    log_2_period: integer := 0;
    pipeline_regs: integer := 5;
    use_bufg: integer := 0
  );
  port (
    sysclk: in std_logic;
    sysclr: in std_logic;
    sysce: in std_logic;
    clk: out std_logic;
    clr: out std_logic;
    ce: out std_logic
  );
end xlclockdriver;
architecture behavior of xlclockdriver is
  component bufg
    port (
      i: in std_logic;
      o: out std_logic
    );
  end component;
  component synth_reg_w_init
    generic (
      width: integer;
      init_index: integer;
      init_value: bit_vector;
      latency: integer
    );
    port (
      i: in std_logic_vector(width - 1 downto 0);
      ce: in std_logic;
      clr: in std_logic;
```

```
      clk: in std_logic;
      o: out std_logic_vector(width - 1 downto 0)
    );
end component;
function size_of_uint(inp: integer; power_of_2: boolean)
  return integer
is
  constant inp_vec: std_logic_vector(31 downto 0) :=
    integer_to_std_logic_vector(inp,32, xIUnsigned);
  variable result: integer;
begin
  result := 32;
  for i in 0 to 31 loop
    if inp_vec(i) = '1' then
      result := i;
    end if;
  end loop;
  if power_of_2 then
    return result;
  else
    return result+1;
  end if;
end;
function is_power_of_2(inp: std_logic_vector)
  return boolean
is
  constant width: integer := inp'length;
  variable vec: std_logic_vector(width - 1 downto 0);
  variable single_bit_set: boolean;
  variable more_than_one_bit_set: boolean;
  variable result: boolean;
begin
  vec := inp;
  single_bit_set := false;
  more_than_one_bit_set := false;
  -- synopsys translate_off
  if (is_XorU(vec)) then
    return false;
  end if;
   -- synopsys translate_on
```

```
    if width > 0 then
      for i in 0 to width - 1 loop
        if vec(i) = '1' then
          if single_bit_set then
            more_than_one_bit_set := true;
          end if;
          single_bit_set := true;
        end if;
      end loop;
    end if;
    if (single_bit_set and not(more_than_one_bit_set)) then
      result := true;
    else
      result := false;
    end if;
    return result;
  end;
  function ce_reg_init_val(index, period : integer)
    return integer
  is
     variable result: integer;
   begin
      result := 0;
      if ((index mod period) = 0) then
          result := 1;
      end if;
      return result;
  end;
  function   remaining_pipe_regs(num_pipeline_regs,   period   :
integer)
    return integer
  is
     variable factor, result: integer;
  begin
      factor := (num_pipeline_regs / period);
      result := num_pipeline_regs - (period * factor) + 1;
      return result;
  end;

  function sg_min(L, R: INTEGER) return INTEGER is
```

```
  begin
      if L < R then
            return L;
      else
            return R;
      end if;
  end;
  constant max_pipeline_regs : integer := 8;
  constant pipe_regs : integer := 5;
  constant num_pipeline_regs : integer := sg_min(pipeline_regs,
max_pipeline_regs);
  constant rem_pipeline_regs : integer :=
remaining_pipe_regs(num_pipeline_regs,period);
  constant period_floor: integer := max(2, period);
  constant power_of_2_counter: boolean :=
    is_power_of_2(integer_to_std_logic_vector(period_floor,32,
xlUnsigned));
  constant cnt_width: integer :=
    size_of_uint(period_floor, power_of_2_counter);
  constant clk_for_ce_pulse_minus1: std_logic_vector(cnt_width
- 1 downto 0) :=
    integer_to_std_logic_vector((period_floor - 2),cnt_width,
xlUnsigned);
  constant clk_for_ce_pulse_minus2: std_logic_vector(cnt_width
- 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period - 3),cnt_width,
xlUnsigned);
  constant clk_for_ce_pulse_minus_regs:
std_logic_vector(cnt_width - 1 downto 0) :=
    integer_to_std_logic_vector(max(0,period -
rem_pipeline_regs),cnt_width, xlUnsigned);
  signal clk_num: unsigned(cnt_width - 1 downto 0) := (others
=> '0');
  signal ce_vec : std_logic_vector(num_pipeline_regs downto 0);
  attribute MAX_FANOUT : string;
  attribute MAX_FANOUT of ce_vec:signal is "REDUCE";
  signal internal_ce: std_logic_vector(0 downto 0);
  signal cnt_clr, cnt_clr_dly: std_logic_vector (0 downto 0);
begin
  clk <= sysclk;
```

```
  clr <= sysclr;
  cntr_gen: process(sysclk)
  begin
    if sysclk'event and sysclk = '1'  then
      if (sysce = '1') then
        if ((cnt_clr_dly(0) = '1') or (sysclr = '1')) then
          clk_num <= (others => '0');
        else
          clk_num <= clk_num + 1;
        end if;
    end if;
    end if;
  end process;
  clr_gen: process(clk_num, sysclr)
  begin
    if power_of_2_counter then
      cnt_clr(0) <= sysclr;
    else
      if        (unsigned_to_std_logic_vector(clk_num)          =
clk_for_ce_pulse_minus1
          or sysclr = '1') then
        cnt_clr(0) <= '1';
      else
        cnt_clr(0) <= '0';
      end if;
    end if;
  end process;
  clr_reg: synth_reg_w_init
    generic map (
      width => 1,
      init_index => 0,
      init_value => b"0000",
      latency => 1
    )
    port map (
      i => cnt_clr,
      ce => sysce,
      clr => sysclr,
      clk => sysclk,
      o => cnt_clr_dly
```

```
    );
  pipelined_ce : if period > 1 generate
      ce_gen: process(clk_num)
      begin
          if      unsigned_to_std_logic_vector(clk_num)      =
clk_for_ce_pulse_minus_regs then
              ce_vec(num_pipeline_regs) <= '1';
          else
              ce_vec(num_pipeline_regs) <= '0';
          end if;
      end process;
      ce_pipeline:  for  index  in  num_pipeline_regs  downto  1
generate
          ce_reg : synth_reg_w_init
              generic map (
                  width => 1,
                  init_index => ce_reg_init_val(index, period),
                  init_value => b"0000",
                  latency => 1
                  )
              port map (
                  i => ce_vec(index downto index),
                  ce => sysce,
                  clr => sysclr,
                  clk => sysclk,
                  o => ce_vec(index-1 downto index-1)
                  );
      end generate;
      internal_ce <= ce_vec(0 downto 0);
  end generate;
  use_bufg_true: if period > 1 and use_bufg = 1 generate
    ce_bufg_inst: bufg
      port map (
        i => internal_ce(0),
        o => ce
      );
  end generate;
  use_bufg_false: if period > 1 and (use_bufg = 0) generate
    ce <= internal_ce(0);
  end generate;
```

```
  generate_system_clk: if period = 1 generate
    ce <= sysce;
  end generate;
end architecture behavior;
library IEEE;
use IEEE.std_logic_1164.all;
use work.conv_pkg.all;

entity default_clock_driver is
  port (
    sysce: in std_logic;
    sysce_clr: in std_logic;
    sysclk: in std_logic;
    ce_1: out std_logic;
    clk_1: out std_logic
  );
end default_clock_driver;

architecture structural of default_clock_driver is
  attribute syn_noprune: boolean;
  attribute syn_noprune of structural : architecture is true;
  attribute optimize_primitives: boolean;
  attribute optimize_primitives of structural : architecture is
false;
  attribute dont_touch: boolean;
  attribute dont_touch of structural : architecture is true;

  signal sysce_clr_x0: std_logic;
  signal sysce_x0: std_logic;
  signal sysclk_x0: std_logic;
  signal xlclockdriver_1_ce: std_logic;
  signal xlclockdriver_1_clk: std_logic;

begin
  sysce_x0 <= sysce;
  sysce_clr_x0 <= sysce_clr;
  sysclk_x0 <= sysclk;
  ce_1 <= xlclockdriver_1_ce;
  clk_1 <= xlclockdriver_1_clk;
```

```
  xlclockdriver_1: entity work.xlclockdriver
    generic map (
      log_2_period => 1,
      period => 1,
      use_bufg => 0
    )
    port map (
      sysce => sysce_x0,
      sysclk => sysclk_x0,
      sysclr => sysce_clr_x0,
      ce => xlclockdriver_1_ce,
      clk => xlclockdriver_1_clk
    );

end structural;
library IEEE;
use IEEE.std_logic_1164.all;
use work.conv_pkg.all;

entity fpga_detector_basadosr_rx_clk6250ns_06_cw is
  port (
    ce: in std_logic := '1';
    clk: in std_logic; -- clock period = 6250.0 ns (0.16 Mhz)
    gateway_in: in std_logic_vector(11 downto 0);
    gateway_out: out std_logic_vector(11 downto 0)
  );
end fpga_detector_basadosr_rx_clk6250ns_06_cw;

architecture structural of
fpga_detector_basadosr_rx_clk6250ns_06_cw is
  component xlpersistentdff
    port (
      clk: in std_logic;
      d: in std_logic;
      q: out std_logic
    );
  end component;
  attribute syn_black_box: boolean;
  attribute syn_black_box of xlpersistentdff: component is true;
  attribute box_type: string;
```

```
  attribute box_type of xlpersistentdff: component is
"black_box";
  attribute syn_noprune: boolean;
  attribute optimize_primitives: boolean;
  attribute dont_touch: boolean;
  attribute syn_noprune of xlpersistentdff: component is true;
  attribute optimize_primitives of xlpersistentdff: component is
false;
  attribute dont_touch of xlpersistentdff: component is true;

  signal ce_1_sg_x0: std_logic;
  attribute MAX_FANOUT: string;
  attribute MAX_FANOUT of ce_1_sg_x0: signal is "REDUCE";
  signal clkNet: std_logic;
  signal clk_1_sg_x0: std_logic;
  signal gateway_in_net: std_logic_vector(11 downto 0);
  signal gateway_out_net: std_logic_vector(11 downto 0);
  signal persistentdff_inst_q: std_logic;
  attribute syn_keep: boolean;
  attribute syn_keep of persistentdff_inst_q: signal is true;
  attribute keep: boolean;
  attribute keep of persistentdff_inst_q: signal is true;
  attribute preserve_signal: boolean;
  attribute preserve_signal of persistentdff_inst_q: signal is true;

begin
  clkNet <= clk;
  gateway_in_net <= gateway_in;
  gateway_out <= gateway_out_net;

  default_clock_driver_x0: entity work.default_clock_driver
    port map (
      sysce => '1',
      sysce_clr => '0',
      sysclk => clkNet,
      ce_1 => ce_1_sg_x0,
      clk_1 => clk_1_sg_x0
    );
```

```
  fpga_detector_basadosr_rx_clk6250ns_06_x0: entity
work.fpga_detector_basadosr_rx_clk6250ns_06
    port map (
      ce_1 => ce_1_sg_x0,
      clk_1 => clk_1_sg_x0,
      gateway_in => gateway_in_net,
      gateway_out => gateway_out_net
    );

  persistentdff_inst: xlpersistentdff
    port map (
      clk => clkNet,
      d => persistentdff_inst_q,
      q => persistentdff_inst_q
    );

end structural;
```