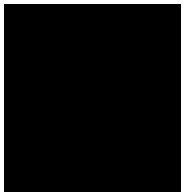


CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE ELECTRÓNICA DEL ESTADO SÓLIDO

"Modelo no-lineal de Memristores basado en red neuronal artificial"

T E S I S

Que presenta:

ING. MARIO ALBERTO GUTIERREZ MONDRAGON

Para obtener el grado de:

MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE INGENIERÍA ELÉCTRICA

Directores de Tesis:

DR. FELIPE GÓMEZ CASTAÑEDA
DR. JOSÉ ANTONIO MORENO CADENAS

Ciudad de México Diciembre, 2016

Dedicatoria:

Este trabajo de maestría está dedicado a la señora Eufrosina Cedillo Medina, abuela, madre y principal motivación para el cumplimiento de este trabajo; tus consejos, cuidados y guía han sido siempre el principal incentivo para mi desarrollo personal y profesional, gracias por tu cariño, apoyo y por siempre confiar en mí sin escatimar ningún esfuerzo.

Gracias también a mis tíos, Isaías Gutiérrez y María Concepción por todo el tiempo que han desgastado en convertirme en una persona de bien, lleno de entusiasmo en la búsqueda de un mejor desarrollo profesional, gracias por dedicar todos los recursos emocionales y económicos posibles, por su paciencia, cuidados y sobre todo por cada una de sus enseñanzas.

Agradecimientos:

A mis familiares y amigos: Gracias a todos y cada uno de ustedes que me han impulsado y considerado siempre como un hermano e hijo más, gracias por el cariño y por hacerme parte de sus vidas, gracias por sus consejos, apoyo, y sobre todo por estar en todo momento.

A mi generación: Gracias SEES-2014, *Armando*, *Daniel*, *Víctor* y *Rodrigo*, un placer conocerlos y compartir estos dos años de esfuerzo y dedicación, gracias por su amistad y por cada una de las experiencias que nos tocó vivir; por los cumpleaños que pasé con ustedes, gracias, me quedo con una enseñanza de cada uno.

A mis Asesores de tesis: Al *Dr. Felipe Gómez Castañeda* y *Dr. J. Antonio Moreno Cadenas*, gracias principalmente por su paciencia y por dedicar el tiempo para transmitirme cada uno de los conocimientos y experiencias que han adquirido en los años de enseñanza, gracias sobre todo por ser la guía en el desarrollo de este proyecto.

A mis Revisores y Sinodales: Gracias al *Dr. Alfredo Reyes Barranca* y al *Dr. Alejandro Ávila* por el tiempo dedicado a la revisión de este trabajo, gracias por la paciencia, por los consejos y las ideas que fungen como base en la mejora del trabajo.

A los auxiliares del laboratorio de VLSI: Gracias al *Dr. Oliverio Arellano Cárdenas* por su apoyo y consejos en el desarrollo de este proyecto, gracias también al *Ing. Emilio Espinoza García* por sus palabras, el apoyo y por su gran labor en el laboratorio.

Al personal administrativo: *Yesenia Cervantes Aguirre*, *Erika Diana Serrano Andrade* y *Angélica*, gracias por su orientación y consejos; principalmente les agradezco por cada palabra de impulso y por la paciencia cada vez que requerí orientación.

A los profesores de la SEES: Gracias a cada uno de los profesores que ofrecieron su tiempo y dedicación en la enseñanza de cada uno de los temas que fundamentan mi desarrollo educativo.

Reconocimientos:

Un especial agradecimiento por el apoyo económico proporcionado por el Consejo Nacional de Ciencia y Tecnología (**CONACYT**) para el desarrollo de este trabajo.

Quiero reconocer también a mi compañero y amigo *M. en C. Gerardo Tornez Xavier*, por su apoyo y por su participación en la fundamentación de este proyecto, gracias por ser guía y consejero durante la estancia en el laboratorio.

Principalmente, quiero hacer un especial reconocimiento al *M. en C. Luis Martín Flores Nava*, gracias profesor por sus consejos, por sus ideas y por la dedicación en el desarrollo de este proyecto, gracias por tener la paciencia y ofrecerme las mejores palabras que me sirvieron como aliciente; de manera personal, es una guía y un ejemplo a seguir en el quehacer profesional.

Contenido

Resumen

Este trabajo se enfoca en el análisis del comportamiento del dispositivo de dos terminales descrito por el profesor de la Universidad de California, Dr. Leon Chua, el Memristor. La evolución de este dispositivo ha surgido en los últimos años, gracias al modelo propuesto por la compañía Hewlett Packard, llamando la atención de los investigadores por el comportamiento no lineal del dispositivo y las múltiples aplicaciones que presenta.

Para efectuar la simulación del dispositivo en sistemas más complejos que contengan más componentes o elementos memristivos, es necesario el desarrollo de modelos que sean capaces de dar una respuesta que asemeje la conducta del Memristor. De esta forma, mediante el potencial de aprendizaje que presentan las redes neuronales artificiales, es posible describir la metodología necesaria en favor del desarrollo de un modelo de red neuronal artificial que aproxime la respuesta *corriente-voltaje* del dispositivo.

Asimismo, el perfil de entrenamiento será representado en un sistema digital, específicamente en un dispositivo lógico programable, FPGA. Así, se producirá un *bloque digital* que contenga la arquitectura de red que es capaz de aproximar la respuesta esperada para un Memristor.

Abstract

This work addresses the behavior analysis of the two terminals device predicted by professor Leon Chua from the Berkeley University in California, the Memristor. Its evolution has been growing up in the last years, due to the model proposed by the company Hewlett Packard. This has called the attention of the researchers due to the non-linear behavior of the device and the applications that may show.

To make the simulation of the device in more complex systems that contain more components or memristive elements, is through the development of models that may demonstrate a similar answer regarding the behavior of the Memristor. This way, through the potential that offer the artificial neural networks, it is possible to develop a methodology providing a way for the development of a network architecture that could approximate the *voltage-current* behavior of this device, specifically using a recurrent architecture network trained through the Levenberg-Marquardt algorithm with supervised training. Also, the training profile will be represented in a digital system, specifically in a programmable logic device, FPGA, that allows having a block containing the network architecture that will approximate the expected response.

Introducción

En 1971 el profesor de la Universidad de California, Dr. Leon Chua postuló una teoría acerca de la existencia de un cuarto elemento eléctrico fundamental de dos terminales, el Memristor. Este dispositivo se define matemáticamente con base en el vínculo entre la Carga y el Flujo Magnético y su comportamiento se describe de acuerdo con una resistencia dinámica con memoria.

Además, en favor del desarrollo de un marco de referencia que describiera las características de un dispositivo que mostrara un comportamiento similar al Memristor, Chua postuló junto con su alumno Kang, la teoría de los "*sistemas memristivos"* con base en dos ecuaciones principales; la primera expresión descrita de acuerdo con la ley de Ohm y la segunda definida en términos de la evolución del sistema. Asimismo, profundizaron, en el análisis del comportamiento *i-v*, presentando principalmente el comportamiento de la gráfica, similar al de un *"moño"* y manifestando, además, que la frecuencia afectaría al dispositivo.

Posteriormente en 2008, un grupo de investigadores de la empresa Helwett Packard, descubriría un comportamiento *i-v* similar al descrito por Chua en las estructuras tipo MIM *(Metal-Insulator-Metal)* que habían desarrollado para aplicación en memorias no volátiles. Mediante el trabajo de Chua y con base en el análisis y la observación postularon el descubrimiento del Memristor, y definieron las expresiones que modelaban su comportamiento. El modelo de Memristor de HP® llamó la atención de los investigadores para explotar el potencial del dispositivo; por consecuencia, la evolución del modelado del Memristor se volvió indispensable, a fin de obtener características más *reales* en la respuesta.

Una forma práctica para modelar el comportamiento del Memristor, es utilizando redes neuronales artificiales, es decir, mediante una arquitectura que sea capaz de equiparar la respuesta no lineal del dispositivo a través del proceso de aprendizaje en función del entrenamiento. Para la realización de este trabajo, se utilizan redes recurrentes o dinámicas, específicamente la arquitectura de red tipo NARX, la cual centra su funcionamiento en el uso del lazo de retroalimentación en la salida, generando la respuesta de los datos de entrada a través de los eventos pasados almacenados en las líneas de retardo.

Con base en el análisis de la teoría, será posible definir la metodología que permita efectuar el proceso de entrenamiento de la red, útil en la obtención de la respuesta del dispositivo; de manera general, el método consiste en el pre procesamiento de los datos, la elección de la arquitectura de red y del algoritmo de entrenamiento, para finalmente efectuar la evaluación de la red.

Organización de la Tesis

Este trabajo de maestría aborda 4 capítulos principales que fundamentan el desarrollo del proyecto.

En el primer capítulo se efectúa una revisión de los aspectos generales del dispositivo propuesto por el Dr. Leon O. Chua, el Memristor. Además, se da una introducción acerca del potencial de las redes neuronales, haciendo énfasis en la forma en que la red adquiere el conocimiento a través del proceso de aprendizaje.

En el segundo capítulo se describe el análisis general del modelo propuesto por HP®, puntualizando en los aspectos fundamentales del comportamiento de la estructura desarrollada por el equipo de investigadores. Asimismo, se efectúa un análisis general del comportamiento de la estructura en los bordes, tomando en cuenta, las diferentes funciones de ventana que muestran comportamientos no lineales en los extremos del dispositivo. Finalmente, con base en las expresiones de HP®, se describe el desarrollo del modelo de Memristor mediante el entorno de trabajo de Simulink®.

El tercer capítulo se enfoca en el uso del sistema a bloques construido en Simulink®, con el principal objetivo de probar su funcionamiento en un sistema más complejo que contemple otros elementos electrónicos. Principalmente, se lleva a cabo el análisis de un sistema oscilador sin reactancia, aprovechando la característica no lineal en la resistencia del Memristor para asemejar la carga y descarga de un elemento capacitivo que generaría la oscilación en el sistema.

Finalmente, en el cuarto capítulo se desarrolla la metodología utilizada para la construcción de la arquitectura de red neuronal artificial. Además, con el objetivo de implementar la arquitectura en tecnología digital, FPGA, se desarrolla el análisis de un sistema a bloques que represente la arquitectura de red con la precisión con la que los datos van a ser introducidos en el dispositivo lógico programable; realizado esto, se describen cada uno de los bloques involucrados en la construcción de la red en el lenguaje de descripción de hardware VHDL, para finalmente implementar el código en un FPGA Spartan 6, extraer los datos y llevar a cabo el análisis de la respuesta.

Objetivos

Objetivo General

- Desarrollo de una metodología de modelado de Memristores haciendo uso de redes neuronales artificiales.

Objetivos Específicos

- Efectuar un estudio general del modelo de Memristor y sus variantes a través de funciones de *"ventana"*, capaces de aproximar determinados comportamientos no lineales del dispositivo.

- Llevar a cabo la representación de las ecuaciones que definen el comportamiento del modelo de Memristor mediante un sistema a bloques en Simulink®; además, efectuar el análisis de su respuesta aplicado a un módulo electrónico que contenga más elementos.

- Elaborar una arquitectura de red neuronal artificial mediante Matlab® que sea capaz de aproximar la respuesta del Memristor.

- Realizar la implementación de la arquitectura de red en tecnología digital, específicamente, en un dispositivo lógico programable (FPGA).

Capítulo 1 Marco Teórico.

1.1. Introducción.

Con base en la descripción de este capítulo se presentará un marco de referencia que permita interpretar los resultados del trabajo de investigación. Particularmente, se manifiestan los aspectos principales acerca de la teoría del nuevo elemento pasivo de dos terminales propuesto por el Dr. Leon Chua, el Memristor.

De esta forma, a partir de la definición generalizada del Memristor y de los sistemas memristivos, será posible la comprensión de los resultados que se presenten en función del funcionamiento universal del dispositivo.

Asimismo, en favor del entendimiento teórico acerca de las redes neuronales artificiales, se presenta la descripción de las partes principales que definen una estructura neuronal y la forma como se desarrollan los modelos matemáticos que pretenden aproximar el procesamiento que efectúa el cerebro humano.

1.2. El cuarto elemento fundamental: El Memristor.

Con base en aspectos de simetría, Leon Chua, profesor de la Universidad de California, en Berkeley, propuso en 1971 la existencia de un cuarto elemento fundamental en la teoría de circuitos. A partir de la definición axiomática de los elementos básicos (resistor, capacitor e inductor), postuló la falta de un enlace entre dos de las cuatro variables de circuitos fundamentales (voltaje, corriente, carga y flujo), generando la concepción de un cuarto elemento de dos terminales, el Memristor [1].

Mediante el modelo de Aristóteles, acerca de la teoría de la materia, donde se observa la relación existente entre las variables principales: tierra, agua, aire y fuego y cómo exhibían dos de cuatro propiedades fundamentales (humedad, sequedad, frío, calor), Chua argumentó a través de un modelo similar (**Figura 1.1**) la existencia de un enlace faltante en la relación de las variables de circuitos, específicamente el vínculo entre la Carga y el Flujo Magnético.

De acuerdo con las relaciones matemáticas básicas entre las variables de circuitos, Chua notó la existencia de seis diferentes pares que podrían ser formados. Una de estas relaciones, define a la corriente eléctrica como la derivada en el tiempo de la carga $\left(i = \frac{dq}{dt}\right)$ y, de acuerdo a la ley de Faraday, el voltaje puede determinarse como la derivada del flujo con respecto al tiempo $\left(V = \frac{d\varphi}{dt}\right)$.

Las relaciones restantes, involucran a los tres elementos básicos en la teoría de circuitos: la resistencia, como la relación entre el voltaje y la corriente $\left(R = \frac{dv}{di}\right)$; el capacitor como la razón de cambio de la carga a través del voltaje $\left(C = \frac{dq}{dv}\right)$; finalmente, el inductor en función del flujo y la corriente $\left(L = \frac{d\varphi}{di}\right)$, [2].

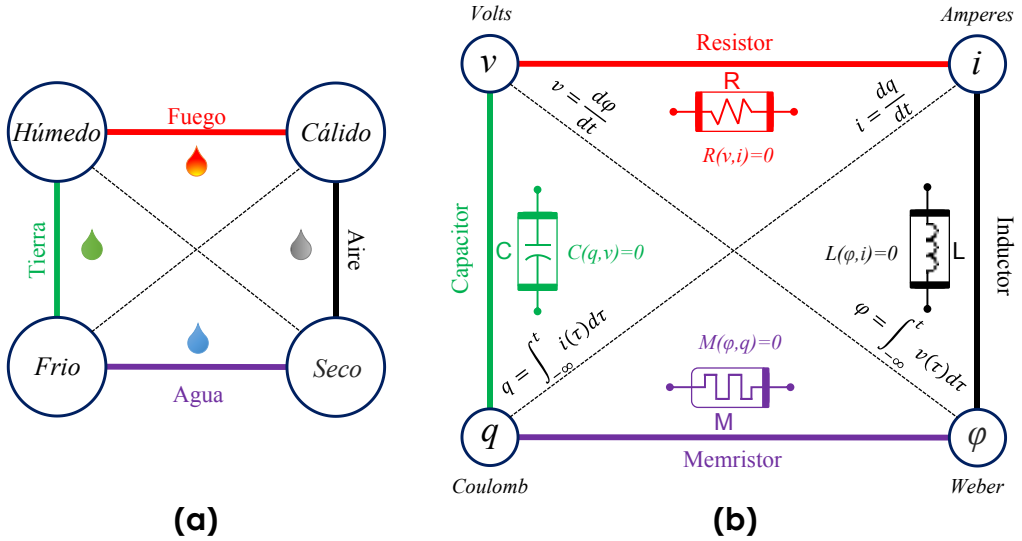


Figura 1.1 *a) Modelo de la teoría de la materia de Aristóteles. b) Modelo de la relación entre las cuatro variables fundamentales en la teoría de circuitos.*

De esta manera, justificó la falta de un vínculo entre las variables de carga y flujo, manifestando la existencia de un cuarto elemento, el Memristor, definido como un dispositivo de dos terminales caracterizado por una relación nombrada *"constitutiva"*: ($f_M(\varphi, q)=0$), demostrando de esta forma, una conexión bidireccional entre las variables matemáticas de carga y flujo [3], (Ver **Figura 1.3**).

Esta asociación permite, mediante los procesos de diferenciación e integración la obtención de una expresión para la resistencia o la conductancia del dispositivo en función del mecanismo de control (flujo o carga), además de la obtención analítica de la variable de salida (corriente o voltaje).

La expresión para la resistencia (o conductancia) del Memristor se define en función de las expresiones matemáticas que describen al flujo y la carga, es decir:

$$q(t) \triangleq \int_{-\infty}^{t} i(\tau)d\tau \qquad \textbf{(1.1)}$$

$$\varphi(t) \triangleq \int_{-\infty}^{t} v(\tau)d\tau \qquad \textbf{(1.2)}$$

Chua describe a un Memristor controlado por carga (flujo) si su relación puede ser expresada a través de una función explicita del flujo *"φ"* (o la carga *"q"*), mostrando la "*relación constitutiva"* del elemento en función del mecanismo de control:

$$\varphi = \hat{\varphi}(q). \tag{1.3}$$

$$q = \hat{q}(\varphi) \tag{1.4}$$

Diferenciando en el tiempo la expresión **(1.3)** para el caso de un Memristor controlado por carga, se tiene:

$$\frac{d\varphi}{dt} = \frac{d\hat{\varphi}(q)}{dq}\frac{dq}{dt} \tag{1.5}$$

De acuerdo con esta expresión y considerando las relaciones matemáticas para la definición de la carga y el flujo, la expresión **(1.5)** puede describirse como:

$$v = \frac{d\hat{\varphi}(q)}{dq} i \tag{1.6}$$

A través de esta expresión es factible determinar analíticamente la resistencia del dispositivo como:

$$R_M(q) \triangleq \frac{d\hat{\varphi}(q)}{dq} \tag{1.7}$$

El termino $R_M(q)$, se refiere a la *"memristencia incremental"* del dispositivo, "*M(q)"*, medida en Ohms (Ω). Así, es posible definir al Memristor como un elemento que impone una relación entre la carga y el flujo, específicamente, la pendiente en la curva *φ-q* en cualquier punto de operación $Q(q_a, \varphi_a)$ determinará la memristencia [4], tal como se ilustra en el ejemplo de la **Figura 1.2**.

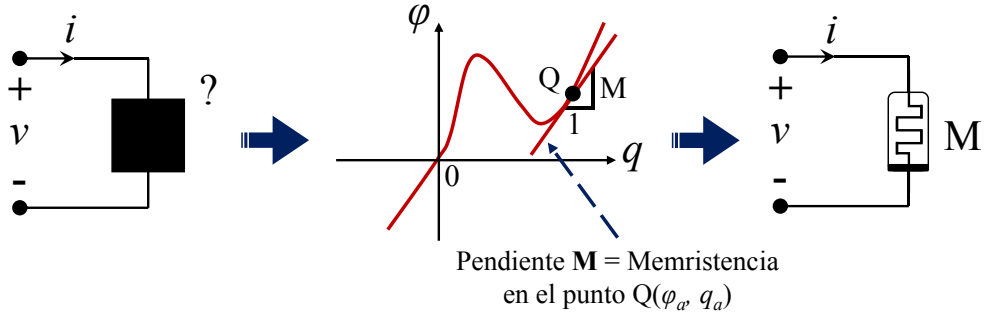


***Figura 1.2**. Ejemplo ilustrativo para la definición de la resistencia del Memristor.*

De forma similar, diferenciando la expresión **(1.4)** para el caso de un Memristor controlado por flujo:

$$\frac{dq}{dt} = \frac{d\hat{q}(\varphi)}{d\varphi}\frac{d\varphi}{dt} \tag{1.8}$$

$$\text{Donde: } \frac{d\hat{q}(\varphi)}{d\varphi} \triangleq G(\varphi) \tag{1.9}$$

Expresando en este caso, la conductancia del dispositivo: *"memductancia incremental"*, medida en Siemens (S). Tanto la expresión **(1.7)** como **(1.8)** pueden ser interpretadas a través de un punto de vista generalizado de la ley de Ohm, resultando en:

$$v = M(q)i \tag{1.10}$$

$$i = G(\varphi)v \tag{1.11}$$

De estas expresiones (**(1.10)** y **(1.11)**), se tiene la manifestación de un comportamiento en un plano *i-v*, pero aplicado a un tipo de resistencia dinámica y (totalmente) no lineal; su estado actual depende de la historia previa de la entrada, puesto que la memristencia o memductancia se determinan de acuerdo con las integrales en el tiempo del voltaje y la corriente, respectivamente.

De la **Figura 1.3** puede notarse cómo el proceso de integración tiene conexión con el efecto de memoria del elemento, es decir, dependiendo del total de carga pasando a través del Memristor, el efecto del flujo es acumulado (integrado), lo que puede interpretarse como un cambio gradual del área bajo la curva de memristencia del dispositivo (de forma similar ocurre en la curva de conductancia) [3]. Con base en la observación de este fenómeno, Chua estableció tal nombre al dispositivo: Memristor = *"Memory-Resistor"* o *resistencia con memoria*, refiriéndose precisamente a la propiedad resistiva dinámica con memoria.

Retomando las relaciones en las ecuaciones **(1.3)** y **(1.4)**, puede demostrarse la equivalencia entre las variables a través de los procesos de diferenciación e integración [5]. Por ejemplo, con base en demostrar la relación descrita a través de la expresión **(1.3)**, es posible integrar **(1.6)** para el caso de un Memristor controlado por carga, es decir:

$$\begin{aligned}
\varphi \triangleq \int_{-\infty}^{t} v(\tau)d\tau &= \int_{-\infty}^{t} R\left(q(\tau)\right) i(\tau)d\tau \\
&= \int_{-\infty}^{t} R\left(q(\tau)\right)\frac{dq(\tau)}{d\tau}d\tau \\
&= \int_{q(-\infty)}^{q(t)} R\left(q(\tau)\right)dq(\tau) \\
&= \int_{q(-\infty)}^{q(t)} R\left(q\right)dq \\
&= \hat{\varphi}(q)
\end{aligned} \tag{1.12}$$

De esta manera, se demuestra mediante el proceso de integración, la posibilidad de exhibir una expresión de flujo equivalente, resultando en una ecuación definida por la integral de la resistencia del dispositivo en función de la carga. Por lo tanto, el flujo a través del dispositivo puede indicarse también como:

$$\hat{\varphi}(q) = \int_{q(-\infty)}^{q(t)} R_M\,(q)dq \qquad \textbf{(1.13)}$$

De manera análoga, para un Memristor controlado por flujo, la carga puede evidenciarse a través de la integral de la conductancia en función del flujo, resultando:

$$\hat{q}(\varphi) = \int_{\varphi(-\infty)}^{\varphi(t)} G_M\,(\varphi)d\varphi \qquad \textbf{(1.14)}$$

De manera general, la relación *"carga-flujo"* del dispositivo, manifiesta un comportamiento que *une* a las variables, es decir, mediante el proceso de diferenciación, la expresión para la resistencia (conductancia) del Memristor puede ser obtenida, asimismo, ésta puede ser de utilidad para regresar la expresión del flujo (o la carga) en función de la carga (o el flujo) mediante integración. Además, utilizando la relación inversa, la conductancia (resistencia) puede ser obtenida.

Gráficamente puede notarse este proceso en la **Figura 1.3 (**adaptada de [3]), donde además, es posible visualizar analíticamente la obtención de la variable de salida en función de la señal de entrada y del mecanismo de control.

Además de la teoría descrita para el Memristor, Chua, mediante el uso de redes compuestas en función de elementos activos fue capaz de simular la respuesta particular *"φ-q",* presentando un comportamiento característico *"monótono creciente"* [6].

De esta manera, con base en proposiciones teóricas y mediante simulaciones que permitieron comprobar su teoría, estableció los fundamentos que sustentaron el comportamiento del elemento propuesto. No obstante, Chua no descartó la posibilidad de que cualquier dispositivo de dos terminales que exhibiera un comportamiento similar cuando se polariza con cualquier señal de excitación periódica, podría presentar una conducta memristiva [7], por consiguiente, estableció un marco matemático de referencia en favor de englobar las caracteristicas particulares del Memristor, con el objetivo de entablar la creación de modelos mas realistas y que permitiera la generalizacion de la noción de los Memristores.

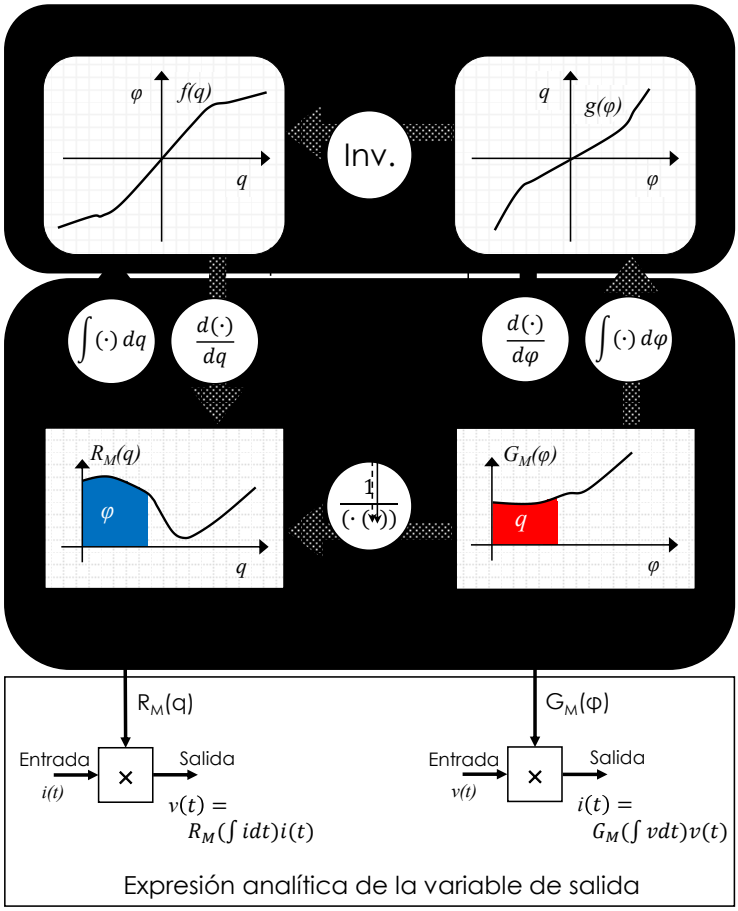


Figura 1.3 *Relaciones características para un Memristor controlado por carga (lado izquierdo) y para uno controlado por flujo (lado derecho), mostrando además la asociación y la equivalencia entre las representaciones que permiten conocer la expresión del voltaje(corriente) a partir de la corriente(voltaje).* [3]

Finalmente, con relación a la existencia del concepto de resistor con memoria, es importante mencionar que se presentó incluso antes de la postulación del trabajo de Chua. Fue en 1960 cuando el profesor de la Universidad de Stanford, Bernard Widrow, creó un elemento nombrado *"Memistor"*. Su funcionamiento se fundamentó en un dispositivo de tres terminales, donde la conductancia en el dispositivo era controlada por la integral en el tiempo de la corriente de la tercera terminal, de manera que su resistencia se controlaba por medio de la carga. Sin embargo, efectuando un análisis posterior, postuló que se trataba de un dispositivo diferente, puesto que se conformaba a partir de una red tipo ADALINE [8], por lo que se le consideró más como un "*gadget"* (artefacto) y no como un elemento en la teoría de circuitos en sí.

1.2.1. Sistemas Memristivos.

Hablando en términos de dispositivos no lineales, se debe tener claro que los elementos *"Resistor, Inductor y Capacitor"* se asocia a las propiedades intrínsecas "*Resistencia, Capacitancia e Inductancia",* es decir, la *"Memristencia"* puede ser considerada como una propiedad intrínseca de ciertos dispositivos. Esto hace pensar en qué variedad de dispositivos contemplan propiedades y comportamientos similares, por consecuencia, y a falta de una definición más correcta del Memristor, Chua y su estudiante Kang propusieron un marco teórico que se encargaría de generalizar la noción de la descripción de los sistemas dinámicos, nombrado *"Sistemas Memristivos"* [9]; principalmente, presentan un entorno general de las propiedades básicas que debería cumplir un elemento cuyo comportamiento se exhibe de forma *"no-lineal"*, asimismo, describen las expresiones básicas que definen tal conducta.

Particularmente, Chua propuso un sistema de dos ecuaciones; ambas definen la base matemática que describe el comportamiento de un Memristor. La primera expresión se encarga de representar la evolución del sistema, a través de una variable general en función de la variación en la entrada; y la segunda expresión, representa la salida del sistema, esto es:

$$\dot{x} = f(x, u, t) \tag{1.15}$$

$$y = g(x, u, t)u \tag{1.16}$$

De estas expresiones, "u" y "y" describen la entrada y la salida del sistema, respectivamente; la variable *x* especifica el estado del sistema, la función *"f"* se refiere a una función vectorial de "n" dimensiones y "g" es una función escalar continua. Por ejemplo, si "u" es la corriente y "y" el voltaje, se genera la definición de un sistema memristivo en términos de la memristencia del dispositivo, "g"; además, la función *"f"* involucraría la forma en que la variable de estado cambia en función del tiempo y la variación de corriente.

En relación con un sistema memristivo de *orden-n* controlado por corriente, las relaciones particulares pueden definirse a partir de las siguientes expresiones:

$$\dot{x} = f(x, i, t) \tag{1.17}$$

$$v = R(x, i, t)i \tag{1.18}$$

La ecuación ***(1.8*(1.18)** está descrita de acuerdo con la ley de Ohm, y la ecuación **(1.17)**, es una expresión dinámica que describe el cambio en la variable de estado en función de la corriente y el tiempo.

De forma similar, para la descripción de un sistema memristivo controlado por voltaje se tiene:

$$\dot{x} = f(x, v, t) \quad \textbf{(1.19)}$$

$$i = G(x, v, t)v \quad \textbf{(1.20)}$$

Las variables *"R"* y *"G"* de las expresiones **(1.18)** y **(1.20)**, describen la memristencia y la memductancia del sistema, respectivamente, y *"f"* la evolución de la variable de estado en función del tiempo y la variación en la entrada del sistema.

Mediante el uso de este sistema de ecuaciones, Chua y Kang entablaron un marco de referencia que especifica la relación de la entrada con la evolución del sistema a partir de una variable general *"x",* en favor de la mejor interpretación del comportamiento resistivo debido al flujo o la carga, facilitando el modelado del sistema.

La principal característica de los llamados sistemas memristivos, es que la respuesta del sistema será cero sin importar el valor de la variable de estado, siempre y cuando la entrada tenga un valor igual a cero, es decir, de acuerdo con las expresiones que describen a un sistema memristivo, la salida (y) es producto de la entrada (u) por la función "g".

De esta manera, considerando cualquier señal periódica de excitación (*corriente o voltaje*), se generaría siempre un cruce por cero en el plano *entrada-salida* del sistema, formando una figura "*Lissajous"*, con un *"pellizco"* en el origen, como se muestra en el ejemplo de la **Figura 1.4**.

Además, se debe tener presente que la respuesta de histéresis puede ser obtenida bajo polarización de cualquier forma de onda periódica. De manera que bajo cualquier señal y forma de excitación se produciría una figura nombrada *"lazo de histéresis pellizcado"*; centrada en el origen y doblemente valuada en función de la entrada, hablando en términos del comportamiento *corriente-voltaje*.

Figura 1.4 *Figura de Lissajous, presentando un ejemplo del comportamiento memristivo.*

Así, el lazo de histéresis se convierte en una propiedad distintiva de un sistema que presente una conducta memristiva, además, tomando en cuenta un punto de vista idealizado del comportamiento del Memristor, la respuesta *i-v* debería ser simétrica, formando dos lóbulos de las mismas dimensiones.

Sin embargo, de acuerdo con la teoría, un Memristor pasivo excitado bajo una señal de onda periódica, se degenera en una resistencia totalmente lineal conforme la frecuencia tiende a infinito, es decir, idealmente un Memristor bajo polarización en corriente directa *"DC"* debería eventualmente recaer en uno de sus valores límites de resistencia, dependiendo de la polaridad de la señal de excitación. El comportamiento de histéresis hipotético para un Memristor excitado bajo una señal de corriente a diferentes frecuencias se muestra en la **Figura 1.5**.

De manera general, Chua presenta un dispositivo que fundamenta su comportamiento resistivo en la no linealidad, en función de la carga y el flujo, a diferencia de una resistencia cuyo comportamiento es totalmente lineal; esto corresponde, según la teoría, a la pendiente en la curva *"q-φ"* (**Figura 1.2**). Un comportamiento constante en la memristencia correspondería a una respuesta totalmente lineal de la curva *"q-φ"*.

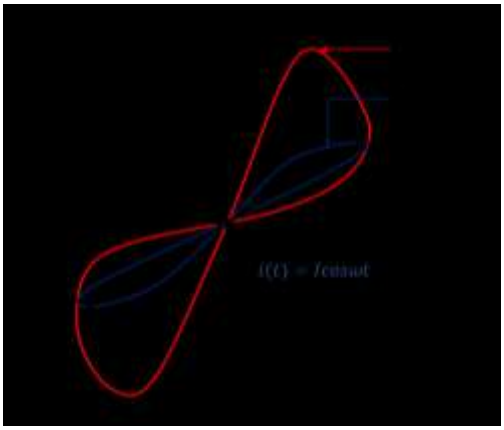


Figura 1.5 *Característica de la respuesta corriente-voltaje de un sistema memristivo.*

1.2.2. Mecanismos de conmutación.

En dispositivos que presentan efectos de conmutación resistiva existen diferentes mecanismos, comúnmente conformados a partir de una combinación de efectos físicos y químicos; de esta forma, se generan diferentes categorías que posibilitan el entendimiento del cambio de estado en la resistencia del dispositivo. Rainer Waser y Masakazu Aono destacan principalmente dos modos de comutacion con base en la polaridad eléctrica que es necesaria para que se efectúe el cambio de estado [10], y los cuales están presentes usualmente en estructuras tipo MIM (*Metal-Insulator-Metal*), utilizadas en el desarrollo de memorias *no-volátiles*.

Según la polaridad necesaria para que se efectué el mecanismo, se habla de una conmutación llamada bipolar si manifiesta una dependencia en el cambio de la polaridad de la señal aplicada (**Figura 1.6b**), mientras que, para el caso de conmutación unipolar (**Figura 1.6a**), se manifiesta dependencia en la amplitud del voltaje aplicado [11], por consiguiente, la polaridad en el estado de encendido y apagado es la misma, denominándosele también conmutación simétrica.

Para el caso de conmutación unipolar, es necesario, para llevar al dispositivo a un estado encendido o "*SET" (de un estado de alta resistencia OFF a un estado resistivo bajo ON)* la aplicación de un voltaje de umbral, donde la corriente se ve limitada por el "*Compliance Current (CC)*[1]*"* de la señal de control. Asimismo, el cambio de un estado encendido a uno apagado, "*RESET" (de baja resistencia ON a alta resistencia OFF)* tiene lugar a un voltaje menor que en el proceso de encendido, pero a una mayor corriente que la corriente de conmutación *(CC)* en el proceso de encendido.

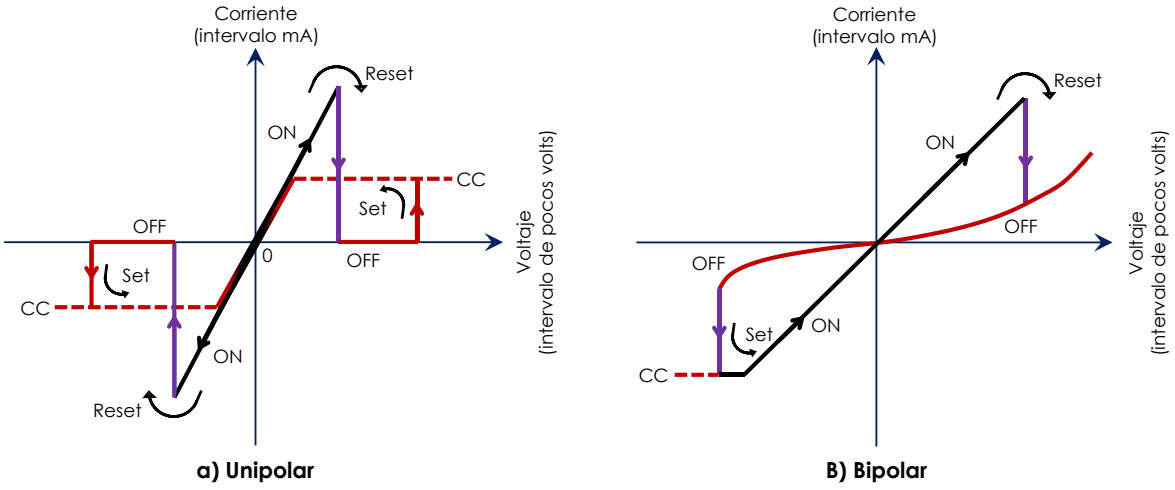


Figura 1.6 *Curvas características de los modos de conmutación básicos basados en la polaridad eléctrica.*

En el proceso de conmutación bipolar, el estado de encendido se genera también con la aplicación de un voltaje de umbral, sin embargo, la fase de apagado se produce cuando la polaridad de la señal es opuesta, denotando independencia en la amplitud de la señal de control, además, dependiendo de la aplicación, el *"CC"* puede ser omitido. Asimismo, la teoría sugiere que, para la generación de este mecanismo de comportamiento, la estructura debería tener alguna asimetría, por ejemplo, la utilización de diferentes materiales en la conformación de los electrodos.

De manera general, existe un número grande de mecanismos físicos conocidos involucrados en cualquiera de los dos modos de conmutación resistiva mencionados, los cuales pueden describirse en

[1]El término *"Compliance Current"* se refiere al intervalo de corriente sobre el cual la regulación de carga está dentro de los límites específicos.

función de algún proceso que sugiera una contribución dominante, proveniente de algún efecto térmico, electrónico/electrostático o iónico.

Finalmente, en la **Figura 1.7**, se presenta una clasificación de los modos de conmutación sugerida por la teoría, para los diferentes mecanismos de conmutación resistiva [12]. Resultando en una generalización de los mecanismos de conmutación, que toma en cuenta los materiales que constituyen el dispositivo, el modo de conmutación o el surgimiento del mecanismo físico, principalmente.

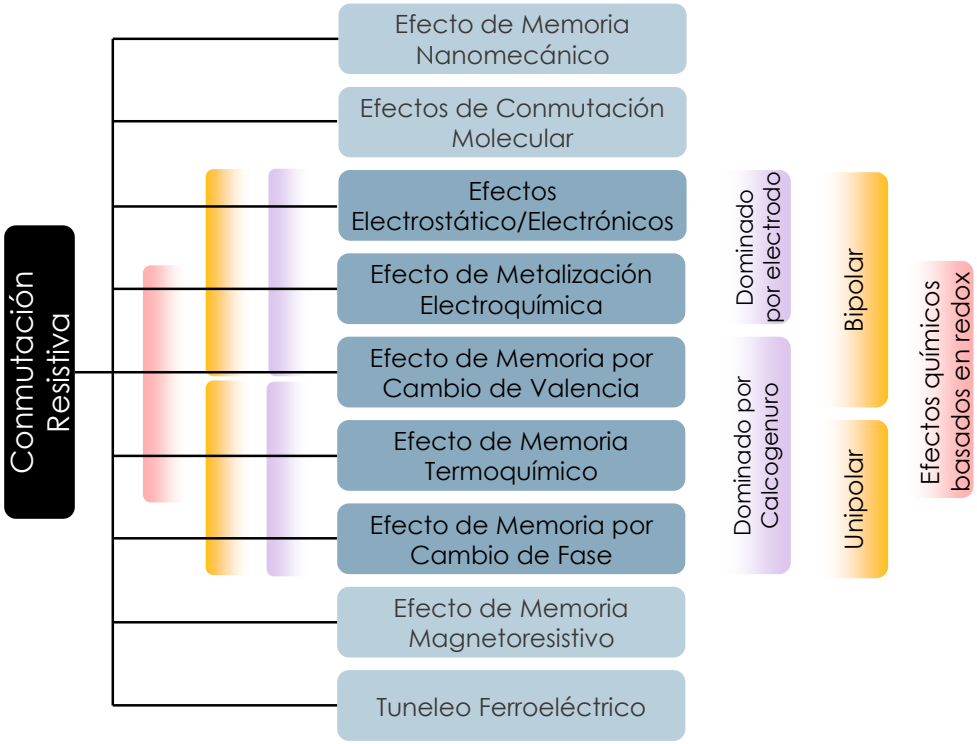


Figura 1.7 *Diagrama propuesto para la clasificación de los diferentes mecanismos de conmutación.*

1.2.3. Aplicaciones del Memristor.

Desde el establecimiento de la teoría propuesta por Leon Chua hasta el surgimiento de la estructura desarrollada por HP® (*estudiado en el capítulo posterior*), se ha manifestado un gran interés por parte de los investigadores para aprovechar las características del dispositivo, y se han reportado una amplia variedad de investigaciones enfocadas principalmente en 3 clases de aplicaciones: la primera en relación con el diseño de memorias no volátiles, la segunda para el desarrollo de sistemas digitales y analógicos y por último, la utilización del dispositivo en sistemas neuromórficos.

Con base en el reporte presentado en la referencia [13], es posible presentar una categorización para las aplicaciones del Memristor, tomando en cuenta tanto el campo de los arreglos de barras transversales (*crossbar*) como el enfoque en una aplicación de tipo discreta. El diagrama de la **Figura 1.8** presenta los posibles circuitos de aplicación utilizando un dispositivo memristivo.

Hablando en términos de un arreglo discreto, la utilización del Memristor pretende conformar un sistema que contenga diferentes elementos, ya sean pasivos o activos, en favor de aprovechar el comportamiento no lineal que manifiesta para mejorar el sistema. En los arreglos de barras transversales, los cuales consisten básicamente en la conexión cruzada de dos conjuntos de alambres en paralelo, donde cada intersección contiene un elemento de conmutación, lo que se pretende es reducir el área geométrica de la estructura, además de beneficiar el consumo de energía.

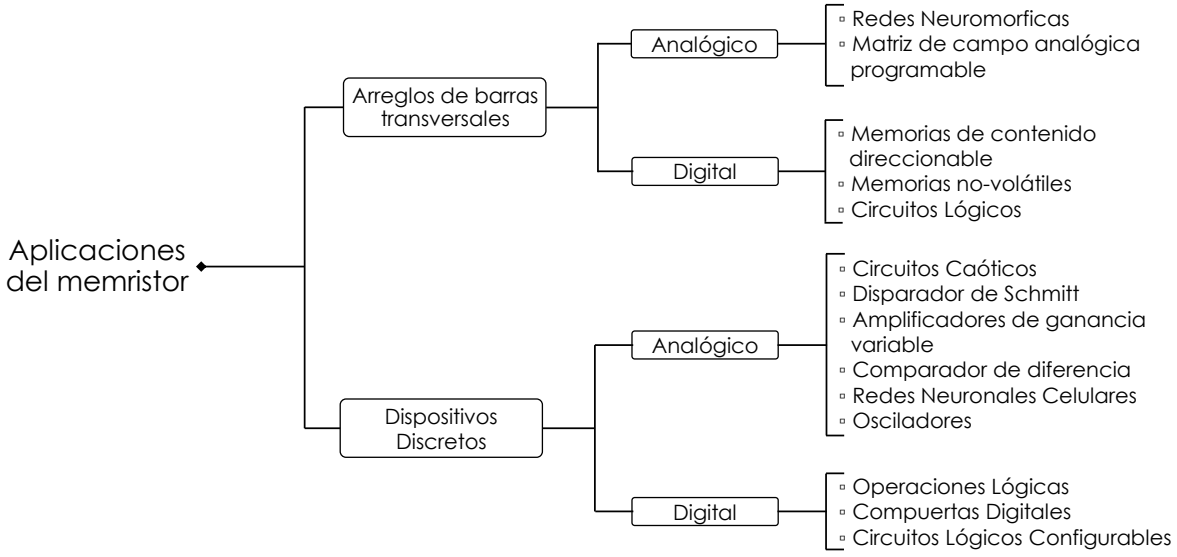


Figura 1.8 *Diagrama jerárquico de las diferentes aplicaciones del Memristor.*

1.3. Redes Neuronales.

Amplia variedad de problemas pueden ser resueltos a través del uso de algoritmos que permitan a un sistema computacional conocer y llevar a cabo cálculos necesarios para generar una respuesta; sin embargo, existen situaciones donde el sistema no es capaz de tomar alguna decisión o realizar el procesamiento para producir la solución esperada, consecuencia de la complejidad, la velocidad del procesamiento y probablemente, debido a la falta de una estructura de código más robusta.

De esta manera, se describe la importancia de las redes neuronales, puesto que el cerebro es capaz de efectuar infinidad de cálculos, consecuencia de la interconexión de aproximadamente *10¨billones"* de elementos llamados "*neuronas",* procesando la información en forma "*paralela"* en la mayor parte del cerebro, mejorando de esta forma, la velocidad del procesamiento [14].

En resumen, el cerebro es capaz de memorizar múltiples eventos y aprender a través de la experiencia, permitiendo a los seres humanos el procesamiento de múltiples tareas y el conocimiento de la respuesta de un sistema con el que interactúa por primera vez. La investigación en redes neuronales artificiales, se enfoca precisamente en un intento de modelar el procesamiento de la información del sistema nervioso, para el aprovechamiento en diferentes campos de aplicación [15]. De esta manera, el estudio en redes neuronales se describe de acuerdo con la aproximación de las diferentes características que permiten el procesamiento en el cerebro.

Una de las características principales del cerebro es la capacidad de ***"re-organizarse"***, es decir, mediante el ***"aprendizaje"*** es posible la formación de un tipo de error que las neuronas contemplan en favor de mejorar el procesamiento y por lo tanto la respuesta esperada.

Otra característica importante de la interconexión entre neuronas es concretamente la capacidad de ***"generalizar"*** y ***"asociar datos"***, es decir, el cerebro tiene la capacidad de encontrar solución a problemas que se han presentado antes, aprovechando el aprendizaje para dar respuesta a un evento del cual no ha recibido alguna clase de entrenamiento; esto puede interpretarse como un alto grado de ***"tolerancia a errores"***.

De esta manera, las neuronas aprovechan la capacidad de continuamente *re-organizarse*, es decir, el cerebro no pierde capacidad en la respuesta, puesto que es capaz de generar conexiones nuevas si fuese necesario, debido a deficiencias en la red en función de factores ajenos.

1.3.1. Procesamiento del cerebro humano.

En favor de realizar una aproximación computacional del procesamiento que efectúa el cerebro, es de vital importancia comprender el funcionamiento del elemento principal, la neurona y la conformación de la red neuronal a través de la interconexión entre neuronas.

De esta manera, debe considerarse en primer lugar, la determinación de una metodología útil que permita aproximar el procesamiento que efectúan las neuronas a través de un modelo básico, en función de expresiones matemáticas o componentes electrónicos. Asimismo, es importante mencionar que la red neuronal le permite al cerebro desarrollar una tarea particular, en función de la experiencia a través de un proceso de aprendizaje.

El diagrama a bloques de la **Figura 1.9** describe de manera global, la forma del procesamiento que se efectúa en el sistema nervioso [16]. En este caso, el bloque principal es el cerebro, representado mediante una red neuronal; los *"receptores"*, se encargan de transportar impulsos eléctricos que contienen la información proveniente de estímulos externos y la cual deberá ser procesada por la red; posteriormente, los *"efectores"*, producen la respuesta del estímulo que se generó, presentando la solución que se distingue como la salida del sistema. De manera general, la red neuronal está recibiendo información, procesándola y es capaz de tomar decisiones a través de reglas que él mismo genera en favor de presentar una respuesta.

Asimismo, de acuerdo con la forma en cómo se efectúa la transmisión de la información, puede mencionarse que se presenta un procesamiento tanto hacia adelante (*"forward"*) como efectuando también un tratamiento de retroalimentación o hacia atrás (*"feedback"*).

Finalmente, es importante mencionar que las estructuras neuronales encargadas de proporcionar la respuesta ante los estímulos, se definen a través del tiempo, con la edad mediante el aprendizaje, principalmente. En las diferentes etapas de la vida de una persona, las estructuras neuronales cambian y son capaces de adaptarse ante el desgaste y las situaciones desconocidas.

Figura 1.9 *Diagrama a bloques para la aproximación de la respuesta del sistema nervioso*

1.3.2. Modelo de neurona biológica.

Con base en la aproximación de un modelo *"matemático/computacional"* que permita efectuar un procesamiento similar al del cerebro, es importante presentar un entorno acerca de los principales elementos de los que consta la entidad fundamental: la neurona, encargada de conformar la arquitectura cerebral. En el diagrama de la **Figura 1.10**, se presentan los componentes básicos que definen su comportamiento.

En el diagrama, se observa un tipo de filamentos en los extremos, similares a la ramificación de un árbol. Estos componentes llamados *"dendritas"*, se encargan del transporte de las señales eléctricas al cuerpo celular, son además un tipo de zona receptiva que permite la conexión (sinapsis) con otras neuronas.

El cuerpo celular es el elemento principal, descrito con base en el procesamiento de las señales eléctricas generadas mediante diferentes fuentes, por ejemplo, un impulso originado en el cuerpo humano. El resultado del procesamiento es transportado a otras entidades o *"fuentes neuronales"* a través del axón, el cual, a diferencia de las dendritas, se conforma de pocas ramificaciones, pero puede llegar a ser mucho más grande.

Tanto el axón como las dendritas, permiten la interconexión e interacción de las múltiples neuronas que conforman la arquitectura cerebral a través de la sinapsis. Para comprender el rol tan importante que tiene la sinapsis, es importante considerar que la información en el cerebro es transmitida por impulsos eléctricos, estos viajan entre neuronas ayudadas por señales químicas, generadas por un proceso que libera una substancia transmisora que se difunde entre la unión del axón y la dendrita, esta substancia "*neurotransmisora*", inhibirá o dejará pasar las diferentes señales al cuerpo celular. Asimismo, con base en la combinación de las señales transmitidas a la célula, se genera cierto nivel de activación que permite efectuar la transmisión del procesamiento a través del axón a una determinada frecuencia, estableciendo, de esta manera, el proceso de aprendizaje.

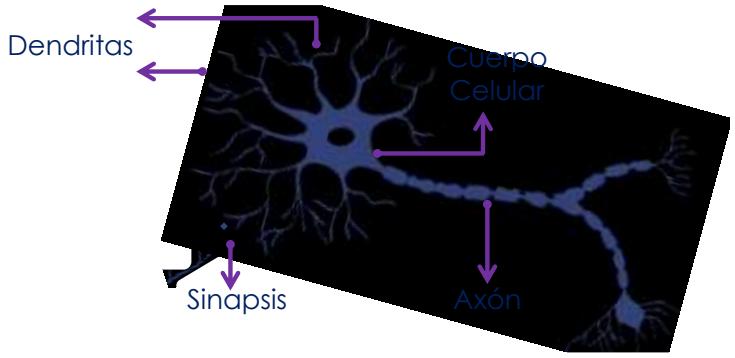


Figura 1.10 *Modelo de neurona biológica básico.*

1.3.3. Modelo de neurona artificial.

Warren McCulloch y Walter Pitts, presentaron en 1943 uno de los primeros modelos de neurona artificial, el cual establecía, principalmente, el cálculo de la salida, comparando la suma ponderada de las señales de entrada con una función de activación o umbral. De manera general, si la suma presentaba un valor más grande o igual que el umbral, la salida se definía con un valor de 1, de forma contraria se establecía un valor en 0.

Diferente del modelo de McCulloch y Pitts, donde los parámetros debían ajustarse de forma manual, dificultando la convergencia, Frank Rosenblatt, propuso en 1958 una regla de aprendizaje para el entrenamiento de su neurona artificial, el "*perceptron*" [17], que permitiera definir la forma en que los pesos sinápticos son ajustados, en favor de presentar una mejor convergencia en la respuesta esperada.

De esta manera, el perceptron de Rosenblatt, definió un conjunto de reglas que permitirían dar solución a un problema planteado. Esta metodología de entrenamiento, describió la forma en que el aprendizaje tiene lugar. Asimismo, pueden describirse diferentes categorías en las reglas de aprendizaje. Principalmente, pueden considerarse tres clases importantes: aprendizaje supervisado, no supervisado y aprendizaje reforzado.

Además, es importante mencionar que el "perceptron" considera una sola neurona, dificultando la aproximación de funciones complejas. Sin embargo, este modelo de neurona presentó la base para el desarrollo tanto de redes multicapa, como para el mejoramiento del algoritmo de entrenamiento.

Por ejemplo, una de las reglas de aprendizaje actualmente utilizadas, se describe de acuerdo con el algoritmo de *"backpropagation"* [18], basado en el aprendizaje supervisado. Este algoritmo está definido en función del algoritmo LMS (Least Mean Square, *mínimos cuadrados*), utilizado por Widrow en 1960 para su red tipo ADALINE, similar al perceptron, sin embargo, la función de transferencia es lineal y la regla de aprendizaje está descrita con base en el error, el cual actúa como un mecanismo de control en favor de corregir el ajuste de los pesos sinápticos. De esta manera, cuando la función de rendimiento, por ejemplo, un promedio de errores, alcance su valor mínimo, la señal de salida será más cercana a la esperada.

1.3.3.1. Entrenamiento a través del aprendizaje supervisado

El desarrollo de este trabajo hace énfasis en el aprendizaje supervisado, utilizando la corrección de errores como regla de aprendizaje, por consecuencia, es importante presentar un entorno general acerca del funcionamiento de este tipo de entrenamiento.

En este caso, el aprendizaje se efectúa asistido de un "*maestro*", que tiene conocimiento del entorno *(representando un conjunto de entradas-salidas)*, de manera que posee completo conocimiento de la plantilla de entrenamiento, por lo que la red podría aprender y ajustar la respuesta que se espera haciendo uso del conocimiento que pueda adquirir del "*maestro*", si ambos se sometieran a un mismo ambiente mediante un "*vector de entrenamiento*", es decir, un conjunto de ejemplos o "conjunto de entrenamiento" que le permita a la red ajustar sus parámetros, considerando la desviación entre la respuesta deseada y la respuesta actual de la red (error), tal como lo ejemplifica la **Figura 1.11**.

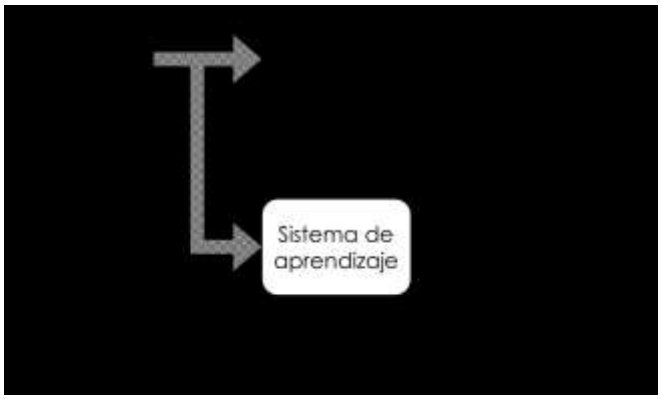


Figura 1.11 *Diagrama de la metodología del aprendizaje supervisado.*

La red mejora la respuesta conforme las iteraciones aumentan hasta el momento en donde el entrenamiento le permite prescindir del *maestro* para tratar por sí sola con la plantilla del entorno. Del diagrama anterior se nota este proceso y cómo la red revisa el error y lo corrige hasta que la función de rendimiento *(en este caso el error cuadrático medio)* es el óptimo. Esto se logra localizando los mínimos locales o globales de la gráfica de superficie de errores, haciendo uso de la regla del *"gradiente descendente"* [19]; la supervisión del maestro puede verse como un punto sobre la gráfica de superficie del error, por consecuencia, los parámetros de la red se van ajustando conforme el gradiente se acerque al mínimo.

Además, deberá tomarse en cuenta que existen otras reglas de aprendizaje que permiten la adquisición del conocimiento, tales como la regla centrada en memoria, aprendizaje Hebbiano, competitivo y la regla de Boltzmann, presentando una serie de normas que se involucran directamente en la forma en que la red es diseñada. Finalmente, se debe mencionar que algunos de estos algoritmos utilizan aprendizaje supervisado y otros no.

1.3.3.2. Bloque de neurona artificial.

En favor de interpretar el comportamiento del "perceptron", es de utilidad el modelo a bloques mostrado en el diagrama de la **Figura 1.12** [20]. El modelo está descrito en función de una sola entrada *"p"* que se enlaza a un bloque sumador a través de un "*peso sináptico (w)"*, generando la ponderación de la entrada. Esta señal se presenta en el bloque sumador, aplica un valor de polarización ("*bias*", "*b*") y genera la respuesta "*neta* (*n*)". Finalmente, la salida del sistema se presenta de acuerdo con la función de activación.

La inclusión de la señal externa aplicada o polarización en el bloque sumador, tiene efecto en la entrada *"neta"* de la función de activación ya sea disminuyéndola o incrementándola, dependiendo si es negativa o positiva. El objetivo principal de esta variable es lograr una convergencia más rápida de la red, sin embargo, su uso puede ser omitido dependiendo de la aplicación.

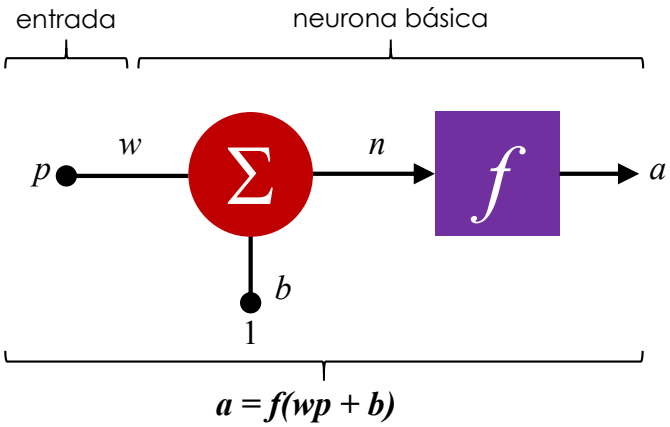


Figura 1.12 *Modelo de neurona de una sola entrada.*

Además, es importante mencionar, en relación con el modelo biológico, que ambos elementos: sumador y función activación, conforman el cuerpo celular; la conexión entre la entrada y el peso sináptico, puede interpretarse como la fuerza de la conexión que genera la sinapsis y la salida "*a"* del diagrama, puede identificarse como el axón, encomendado al transporte de la respuesta a otras entidades.

Finalmente, debe considerarse la variable de peso y de polarización como parámetros ajustables por la neurona, sin embargo, la función de transferencia (o activación) puede ser descrita de acuerdo con la respuesta deseada.

1.3.3.3. Función de Activación.

Una función de activación describe la salida de la neurona en función de la entrada *"neta"*, estableciendo la solución al problema planteado. De manera general, existen tres funciones básicas usadas comúnmente en el tratamiento de redes neuronales artificiales. Respecto de la aplicación, puede considerarse cualquiera de ellas.

La primera función está definida en función del "perceptron", utilizada ampliamente para poder clasificar patrones, puesto que entrega un valor 0 si el argumento de la función es 0, o 1 si el argumento es mayor o igual a 0. Mientras que otra función reportada en la teoría tiene el objetivo de entregar el valor de entrada en la salida. Finalmente, la tercera función de activación y más comúnmente utilizada en redes neuronales, es la función logarítmica sigmoidal, capaz de tomar el valor de la entrada *"neta"* y ajustarlo dentro de un intervalo de 0 y 1 y se puede expresar de la forma:

$$a = \frac{1}{1 + e^{-n}} \tag{1.21}$$

Esta función está definida con relación a un intervalo de 0 a 1, y su comportamiento asume una forma asimétrica respecto al origen. Es posible redefinir el comportamiento de ésta para presentar un intervalo determinado, acotando la entrada de la función de transferencia a un intervalo de ±1. Esta función es referida comúnmente como función sigmoidal y su forma puede ser aproximada de forma sencilla utilizando la función tangente hiperbólica mostrada en la **Figura 1.13**, definida mediante la siguiente expresión:

$$f(n) = tanh(n) \tag{1.22}$$

Del diagrama, se tiene una función creciente, la cual sigue una conducta no lineal, donde es posible asumir valores negativos en la entrada *"neta"*, además, esta función es continua y diferenciable en cualquier valor numérico real [±∞].

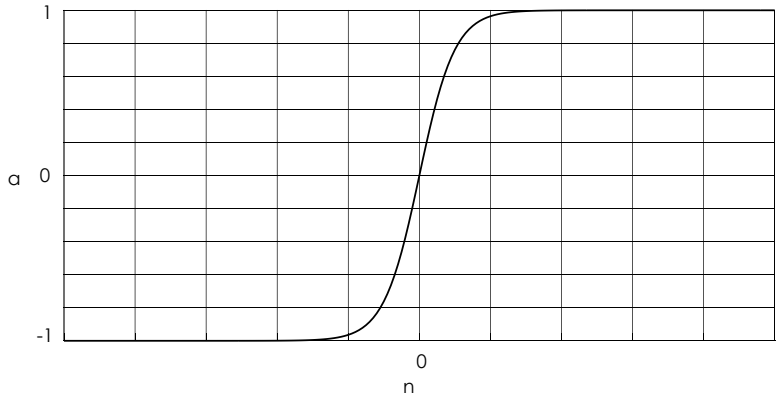


Figura 1.13 *Función tangente sigmoidal hiperbólica.*

1.3.3.4. Arquitectura de red

El funcionamiento del cerebro, puede definirse como una gran estructura capaz de llevar a cabo múltiples tareas a través del aprendizaje, utilizando el entrenamiento. La forma en que la red es entrenada mediante el algoritmo está relacionada comúnmente con la forma en que se estructura. El objetivo de una arquitectura de red, es precisamente desarrollar una metodología de funcionamiento que aproxime parte de la función de aprendizaje del cerebro, mediante algoritmos que definan la forma en que una red neuronal artificial efectúa los cálculos que le permitan aproximar la respuesta esperada.

De forma similar al cerebro, una sola neurona no es capaz de dar una respuesta demasiado compleja, la interconexión de millones de neuronas es el punto clave en el procesamiento del sistema nervioso. De igual manera, en redes artificiales es necesario el uso de varias neuronas interconectadas que en conjunto formen una *"capa"*, dando lugar a la primera clasificación de las arquitecturas de red. En este caso se tienen varias entradas conectadas con su respectivo bloque de procesamiento *(sumador y función de activación),* cada una con un peso y polarización específica, generando una nombrada red de *"capa simple"*, **Figura 1. 14**. En ésta, se presenta una capa de entrada que se *"computa"* o *"proyecta"* estrictamente hacia los nodos sumadores que generan el cálculo de la red, por lo que se le considera una red cuyo algoritmo se enfoca en el flujo de información hacia adelante *(feedforward)*.

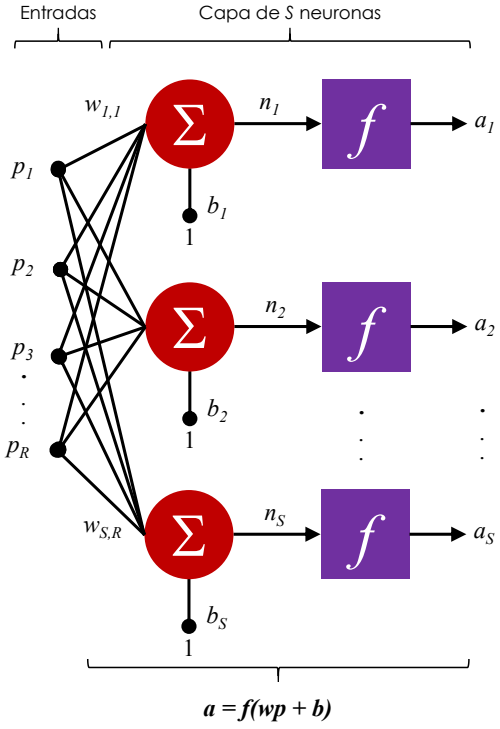


Figura 1. 14. *Red neuronal de capa simple.*

Otro tipo de arquitectura se conforma en la construcción de *"múltiples capas"* o también nombradas *"capas escondidas"*, tal como en el ejemplo de la **Figura 1. 15**, el cual presenta dos capas escondidas y una capa de salida. De esta manera se pretende mejorar en gran medida la respuesta deseada, consecuencia de ser una forma de englobar una cantidad grande de neuronas en diferentes capas que trabajen independientemente, pero de forma paralela, posibilitando de esta manera, un procesamiento más complejo para aplicaciones más robustas.

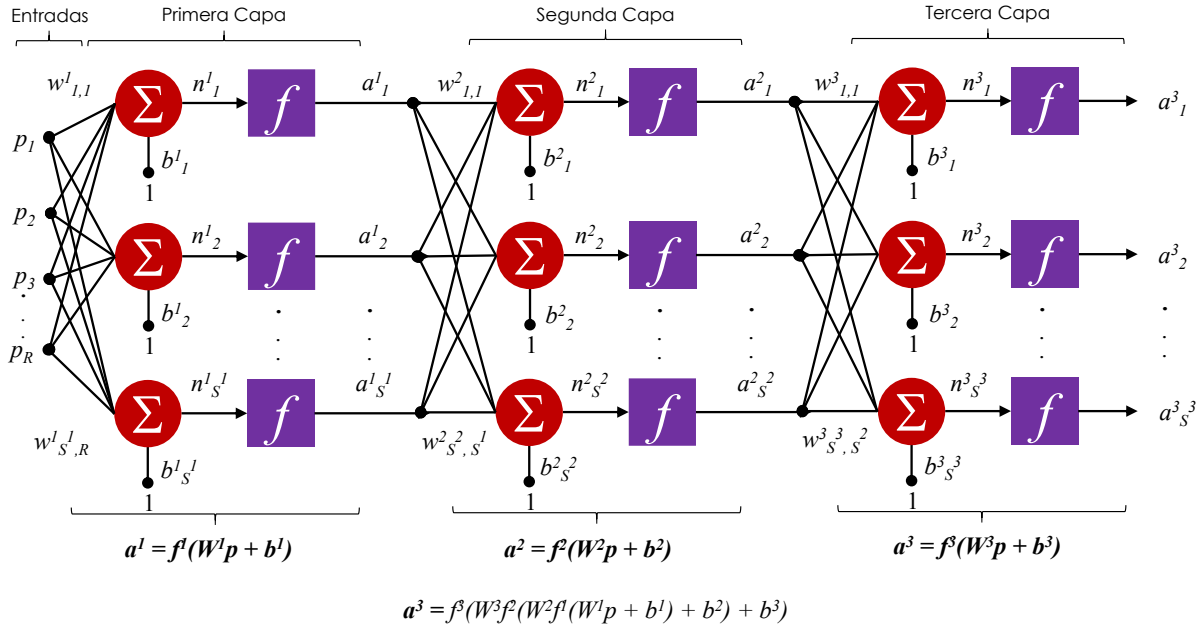


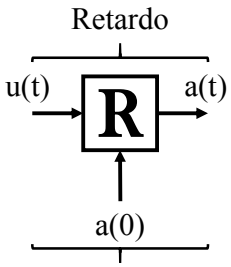
Figura 1. 15. *Red neuronal multicapa.*

En resumen, la salida de cada capa se convierte en la entrada de las capas subsiguientes, presentando la ventaja de que en cada capa se puede tener una función diferente, por ejemplo, en la capa escondida una sigmoidal, y en la capa de salida una lineal, facilitando el entrenamiento para la aproximación de más funciones.

Finalmente, es importante mencionar que, tanto la red de capa simple con la red multicapa, definen sus cálculos usando *"feedforward"*, cuya característica principal es que el flujo de la información sigue siempre una ruta hacia adelante, es decir, las conexiones de las entradas siempre provienen de la salida de las capas anteriores, a diferencia de las redes recurrentes, en las cuales se hace uso de un lazo de retroalimentación (*"feedback"*), donde la salida total de la red (la última capa), se puede convertir en la entrada. Asimismo, el uso de retroalimentación involucra la utilización de bloques o unidades de retardo, resultando en un comportamiento dinámico y dependiente del tiempo.

1.3.3.4.1. Redes Recurrentes

El desarrollo de este trabajo se enfoca al uso de redes recurrentes, por consiguiente, es necesario describir los aspectos básicos de su funcionamiento. En la **Figura 1.16** se muestra el bloque de retardo, útil en el procesamiento de la información e imprescindible en este tipo de arquitectura de red.



$$a(t) = u(t-1)$$

Figura 1.16 *Bloque básico para representar las unidades de retardo.*

La unidad mostrada modela el comportamiento básico del bloque de retardo; la salida (*a(t)*) se calcula a través de la entrada (*u(t)*). De la ecuación en el diagrama, se observa que la salida es función de la entrada demorada por una unidad de tiempo. Se debe tomar en cuenta además la inicialización del bloque (*a(0)*) en el tiempo *t = 0*. Este bloque de retardo puede encontrarse también denotado como z^{-1}.

Una red recurrente centra su funcionamiento en el uso de un lazo de retroalimentación, tal como lo muestra el diagrama de la **Figura 1.17**. En este tipo de redes la salida estará definida por la entrada a través de una función del tiempo, y en el ejemplo del diagrama, la salida de la red se calcula a través de los valores de salida previos, a diferencia de una red tipo *"feedforward"* donde la salida es constante debido a que la entrada se mantiene estática y la respuesta queda en función de la entrada.

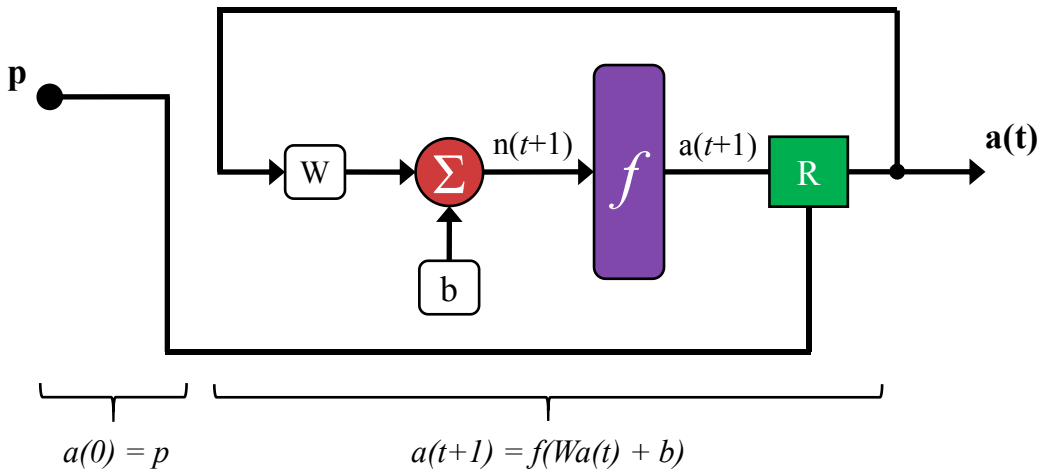


Figura 1.17 *Ejemplo de estructura de red recurrente.*

1.3.3.4.2. Red NARX

Una arquitectura de red comúnmente utilizada para llevar a cabo predicción y útil en la búsqueda de aproximar la respuesta del Memristor es la *red neuronal no-lineal auto-regresiva con entradas exógenas* o más comúnmente nombrada NARX, mostrada en la **Figura 1.18**. La nomenclatura en los bloques denota el tamaño del vector; por ejemplo, en el bloque que almacena los pesos de entrada (IW) se tiene un vector SxR (S neuronas por R elementos, dependiendo del tamaño del bloque de retardo) y el superíndice indica la capa a la que pertenece.

Esta red hace uso del lazo de retroalimentación comúnmente empleado en redes recurrentes o dinámicas y la principal ventaja que presenta es que puede ser entrenada haciendo uso de un algoritmo completamente estático, como si se tratase de una red tipo *"feedforward"*, donde la salida de la red no dependa de la serie de tiempo, sino directamente de la entrada. En el modelo de esta red se tiene una entrada simple "*p"* que se aplica a la línea de retardo, la salida se retroalimenta a la primera capa a través de otra línea de retardo; las líneas de retardo tienen el papel de hacer fluir los datos hacia los bloques de procesamiento, por lo que se interpretan como entradas de determinada longitud con pesos respectivos.

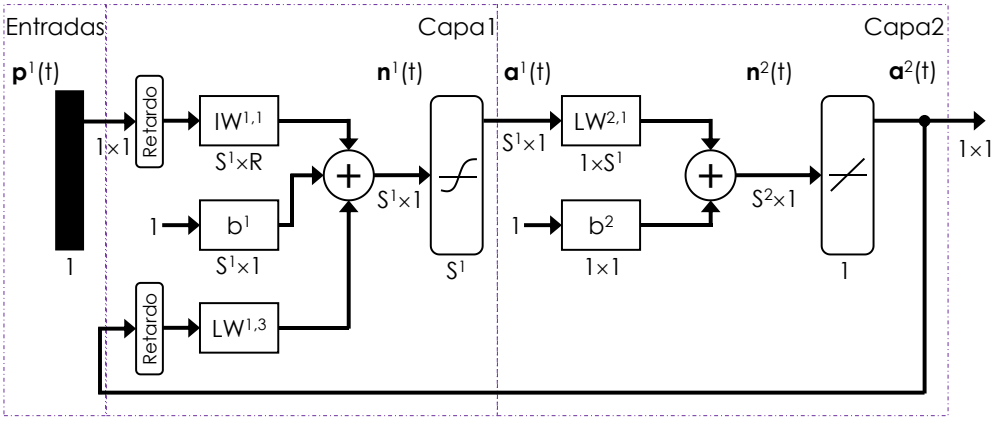


Figura 1.18 Modelo de red neuronal tipo NARX. En este caso se tiene una entrada *"p"* conectada a un bloque de retardo que envía la serie de datos ponderada al sumador.

Para definir la conducta del modelo NARX se debe tener en cuenta la forma en que los retardos devuelven los datos, es decir, el valor de salida de la serie de tiempo denotado como *y(t)* es *"retrocedido"* sobre los valores previos de la señal de salida y de los valores previos de una señal de entrada independiente originada afuera de la red ("*exógena"*) "*u(t)",*por lo que la red es capaz de llevar a cabo predicción del valor siguiente de entrada y la expresión para describir este comportamiento puede definirse como:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u)) \qquad \textbf{(1.23)}$$

De manera general, una arquitectura de red tipo NARX contempla una red multicapa como bloque principal de procesamiento; la aplicación del lazo de retroalimentación puede seguir diferentes caminos dependiendo de la aplicación. El lazo puede estar conectado directamente de la salida hacia a la entrada de la red, considerándosele un enlace global, sin embargo, es posible hacer uso también de enlaces locales ya sea que se tengan más capas escondidas o intermedias; éstas son las capas que se encuentran entre la entrada y la capa de salida, permitiendo obtener una arquitectura de red más robusta.

Para facilitar el cálculo de la respuesta al algoritmo de entrenamiento de la red, puede utilizarse una configuración en lazo abierto como lo muestra el diagrama de la **Figura 1.19a**, generando una red de tipo multicapa, y facilitando el aprendizaje mediante la utilización de un algoritmo común de retro-propagación o *"backpropagation"* [19]. La finalidad del método de *"backpropagation"* se enfoca en la modificación de los pesos sinápticos de la arquitectura de red para alcanzar el objetivo deseado, haciendo uso de la corrección en el error como regla de aprendizaje. El conocimiento o aprendizaje es almacenado en la interconexión de las neuronas con las entradas, a través de los pesos sinápticos.

El objetivo de utilizar una arquitectura en lazo abierto es, en concreto, presentar el perfil *"verdadero"* que se pretende aproximar, es decir, se posibilita revisar la entrada en función de ir ajustando los pesos sinápticos. Esto tiene la ventaja, además de facilitar el aprendizaje, de que el perfil que se está buscando se adecúe mejor, puesto que se está observando la respuesta deseada. Posteriormente, para conocer la solución común que presenta la arquitectura NARX, puede cerrarse el lazo (**Figura 1.19b**), con lo que se efectúa el proceso de predicción de la respuesta. En lazo cerrado, la predicción se realiza tomando en cuenta los pesos que se obtuvieron en el entrenamiento a lazo abierto y calcula los valores de salida a través de la serie de tiempo, contemplando los valores pasados y tomando en cuenta a ese instante, una sola entrada.

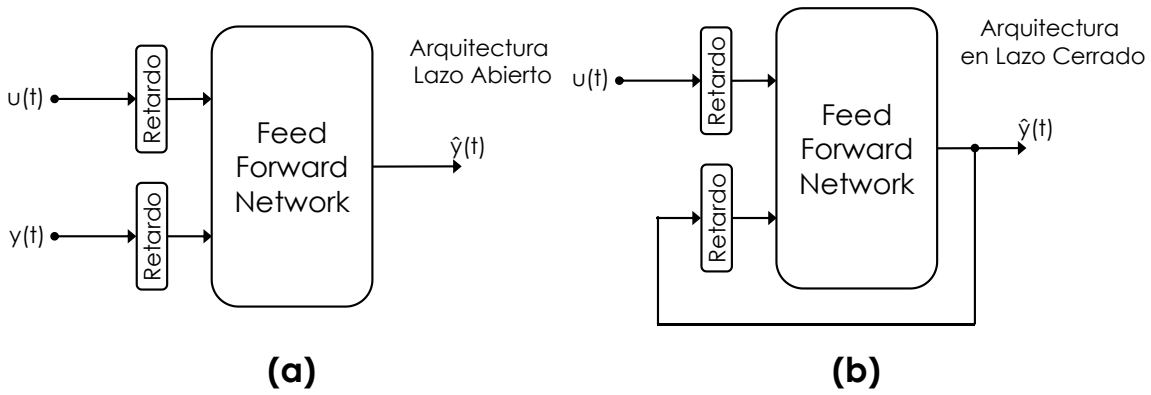


Figura 1.19 *a) Configuración en lazo abierto, permite simplificar el entrenamiento de la red. b) Configuración en lazo cerrado, posibilitando el uso de la predicción, consecuencia del lazo de retroalimentación.*

1.4.Conclusiones

A través de lo descrito en este capítulo, fue posible describir con base en el Memristor y las redes neuronales, los aspectos teóricos fundamentales que posibilitarán una mejor interpretación de los resultados obtenidos en las simulaciones que se efectúen.

Específicamente, se llevó a cabo el análisis de la teoría de acuerdo con los aspectos primordiales que definen el comportamiento del Memristor. Además, se describen los principales mecanismos involucrados en la conmutación de su resistencia y las posibles aplicaciones donde pueda emplearse.

Asimismo, se efectúo el estudio de la teoría establecida para el campo de las redes neuronales artificiales, haciendo énfasis en la forma en que adquieren conocimiento del entorno, a través del proceso de aprendizaje. Particularmente, se realizó el análisis de los aspectos principales acerca de las redes recurrentes, de forma específica, para la arquitectura de red tipo NARX, la cual, es fundamental en el desarrollo del proyecto, puesto que permitirá aproximar la respuesta *i-v* deseada para el Memristor.

Referencias

[1] L. O. Chua, "Memristor-The missing Circuit Element", *IEEE Transactions on Circuit Theory,* vol. 18, pp. 507-519, 1971.

[2] A. G. Radwan and M. E. Fouda, On the Mathematical Modeling of Memristor, Memcapacitor, and Meminductor, Springer, 2015.

[3] Z. Biolek, D. Biolek and V. Biolkova, "Analytical Solution of Circuits Employing Voltage- and Current-Excited Memristors", *IEEE Transactions on Circuits and Systems I: Regular Papers ,* vol. 59, no. 11, pp. 2619-2628, 2012.

[4] L. Chua, «Vignettes from Memristor», de *Memristor and Memristive Systems*, New York, Springer, pp. 33-90, 2014.

[5] L. Chua, "Resistence Switching Memories are Memristors", *Applied Physics A Materials Science & Processing,* pp. 765-783, 2011.

[6] A. Adamatzky and L. Chua, Memristor Networks, Springer International Publishing, 2014.

[7] L. Chua, "If it's pinched it's a memristor", *Semiconductor Science and Technology,* vol. 29, no. 10, pp. 1-43, 2014.

[8] B. Widrow, "An Adaptive "ADALINE" Neuron Using Chemical Memristors", Stanford Electronics Laboratories , Stanford, 1960.

[9] L. O. Chua and S. M. Kang, "Memristive devices and systems", *Proceedings of the IEEE,* vol. 64, no. 2, pp. 209-223, 1976.

[10] M. A. Rainer Waser, "Nanoionics-based resistive switching", *Nature Materials 6,* pp. 833-840, 2007.

[10] S. Zou, P. Xu and M. C. Hamilton, "Resistive switching characteristics in printed Cu/CuO/(AgO)/Ag memristors", *Electronics Letters,* vol. 49, no. 13, pp. 829-830, 2013.

[12] R. Waser, R. Dittmann, G. Staikov and K. Szot, "Redox-Based Resistive Switching Memories – Nanoionic Mechanisms, Prospects, and Challenges", *Advanced Materials,* vol. 21, no. 25-26, p. 2632–2663, 2009.

[13] P. Mazumder, S. M. Kang and R. Waser, "Memristors: Devices, Models, and Applications", *Proceedings of the IEEE,* vol. 100, no. 6, pp. 1911-1919, 2012.

[14] D. Kriesel, "dkriesel.com", [En linea].: http://www.dkriesel.com/en/science/neural_networks.

[15] R. Rojas, Neural Networks A Systematic Introduction, Berlin: Springer, 1996.

[16] S. Haykin, Neural Networks A comprehensive Foundation, New Jersey: Prentice Hall, 1999.

[17] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review,* vol. 65, no. 6, pp. 386-408, 1958.

[18] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 1986.

[19] R. Battiti, "First and Second Order Methods for Learning: Between Steepest Descent and Newton´s Method", *Neural Computation,* vol. 4, no. 2, pp. 141-166, 1992.

[20] M. T. Hagan, H. B. Demuth, M. H. Beale and O. d. Jesus, Neural Network Design (2nd Edition), Martin Hagan, 2014.

Capítulo 2 Modelado del Memristor.

La metodología utilizada en búsqueda de un modelo útil que describa el comportamiento de un sistema puede ser una tarea compleja y requiere de un análisis teórico particular, por consecuencia, es importante entender la esencia del dispositivo que se pretende aproximar. Además, es necesario tener en cuenta un grupo de datos basados en la experimentación, prueba e hipótesis, a fin de generar un modelo de acuerdo con representaciones matemáticas que sean capaces de reproducir las mediciones obtenidas. Un modelo que presente resultados correctos no debe ser necesariamente robusto, menos aún perfecto, puesto que comúnmente éstos se describen con base en aproximaciones, por consiguiente, deberá tenerse en cuenta la simplicidad para producir resultados satisfactorios. Asimismo, el modelo deberá ser fácil de comprender y sencillo de modificar, en caso de que las mediciones no se ajusten con exactitud.

2.1 Introducción.

La complejidad de tratar de modelar un dispositivo que presenta un comportamiento *no-lineal*, radica en pretender aproximar la respuesta caótica o impredecible del sistema. El encontrar la solución a ecuaciones que describen un comportamiento diverso y por tanto difícil de entender y resolver, puede ser una tarea complicada debido a la cantidad de suposiciones que deberán establecerse para la solución del sistema [1]. Chua describió cómo simplificar esta tarea a través de uno de dos procedimientos o la combinación de ambos; el primero consiste en un enfoque físico del dispositivo, asumiendo que se tiene completo conocimiento de los mecanismos de operación, lo que conlleva a encontrar un modelo a través del análisis de la información conocida acerca del dispositivo, desarrollar una ecuación física, simplificarla y darle solución. El segundo, tomando en cuenta una perspectiva a través de una *"caja negra",* donde, por el contrario, no se tiene conocimiento acerca de la física del dispositivo y por lo tanto deberán realizarse observaciones experimentales, desarrollar un modelo matemático y validarlo para que sea capaz de reproducir las mediciones obtenidas en la experimentación [2].

Tomando en cuenta estas consideraciones y mediante el análisis y la observación, Chua y Kang establecieron la base de los llamados *Sistemas Memristivos* [3], generando un marco *"teórico-matemático"* que facilitara la interpretación de las principales características de dispositivos que manifiestan este tipo de comportamientos no lineales. De esta forma, definen dos ecuaciones principales para tal sistema:

$$\dot{x} = f(x, u, t) \tag{2.1}$$

$$y = g(x, u, t)u \tag{2.2}$$

Estas expresiones, describen un sistema memristivo que puede contener cualquier tipo de señal de *entrada/salida* y su relación con el estado del sistema en función de las variables involucradas.

2.2 Modelo de Memristor de Hewlett-Packard®.

Hace aproximadamente 20 años, la compañía Hewlett-Packard reunió un grupo de investigadores dirigido por Stanley Williams en favor de cambiar de dirección a la compañía, retomando la investigación en dispositivos y tecnologías que revolucionaran la industria, y que redituara a la empresa. Williams, inmediatamente se dedicó al desarrollo de un tema de trabajo interesante y que presentara resultados útiles en los años siguientes. Así, el grupo de investigadores se enfocó en la fabricación y diseño de estructuras en escala nanométrica, donde incluso las dimensiones de los dispositivos no fueran una limitante.

En aquel tiempo, el tamaño característico de un transistor era aproximadamente de 350 nanómetros; este grupo de investigación tenía en cuenta la predicción descrita por Moore, acerca de cómo la capacidad de integración se duplicaría conforme las tecnologías de fabricación evolucionaran. De esta manera, se enfocaron en el desarrollo de soluciones innovadoras para la restricción de la reducción de los dispositivos electrónicos, puesto que en algún momento ésta se vería limitada por la física de los dispositivos.

Así, tras varios años de investigación, hallaron en sus estructuras, un comportamiento difícil de comprender; fue hasta el hallazgo del trabajo realizado por Leon Chua acerca del Memristor, que lograron descifrar y explicar los resultados obtenidos en sus mediciones experimentales.

Principalmente, fue de su interés una gráfica que mostraba el comportamiento *i-v* característico del Memristor. Esta gráfica, presentaba una *"especie"* de comportamiento de histéresis [4], y su propiedad fundamental era el denominado *"cruce por cero",* además, el área del "*lóbulo"* del lazo de histéresis se reducía uniformemente de acuerdo al incremento en la frecuencia de la señal de excitación y conforme ésta tendía a infinito, el ciclo de histéresis se aproximaba a una línea recta (**Figura 2.1a**), es decir, la variación de frecuencia en la señal, afectaba el valor de resistencia, resultando en un comportamiento *no-lineal* y variante en el tiempo.

En relación con este comportamiento, el equipo de HP® comprendió el funcionamiento de la estructura que había desarrollado. Notaron que el comportamiento de la corriente a través del voltaje en su dispositivo era similar; primero el voltaje incrementaba desde cero hasta un valor máximo, en ese punto disminuía a un valor mínimo negativo, para finalmente regresar a cero [5], mostrando un lazo de histéresis similar al descrito por Chua (**Figura 2.1b**).

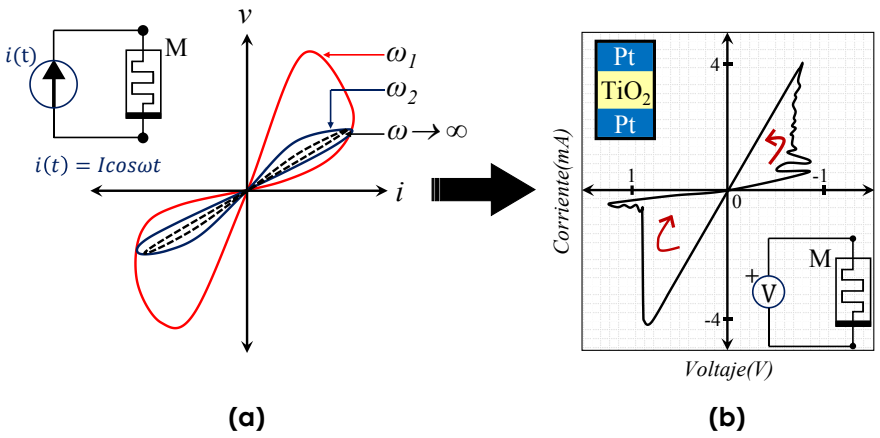


Figura 2.1 *a) Curva corriente-voltaje hipotética descrita por Leon Chua para un Memristor.* [3] *b) Gráfica de la característica corriente-voltaje obtenida del Memristor de HP® controlado por voltaje.* [6]

Además, el cambio en la resistencia obtenida experimentalmente dependía también de cuánto tiempo se mantenía aplicado el voltaje. Observaron que la resistencia era menor cuanto más tiempo se aplicaba un voltaje positivo, hasta llegar a un valor mínimo de resistencia. Por el contrario, cuanto más tiempo se aplicaba un voltaje negativo, mayor era la resistencia, hasta alcanzar un valor máximo. Incluso, dejando de aplicar la señal de excitación, la resistencia del dispositivo se "*congelaba*" hasta que la polarización se restablecía, es decir, el dispositivo era capaz de recordar la última carga que tuvo [7].

De esta forma, a través del compendio de características obtenidas experimentalmente y con base en el análisis y la observación, Williams, Strukov, Snider y Stewart validaron que el funcionamiento de su dispositivo era el de un Memristor y propusieron un modelo básico de su funcionamiento, con base en que un dispositivo de este tipo, presentará histéresis si existe alguna clase de reordenamiento en el material.

La descripción del modelo propuesto por el grupo de investigadores, se define en términos de la interpretación general de los mecanismos físicos ocurridos en la estructura. Además, en favor de simplificar la descripción de las expresiones que definen al modelo, se considera un caso simple de conductividad óhmica y arrastre lineal de los iones de impureza a través de un campo eléctrico uniforme [6].

El dispositivo descrito por HP® consiste en una capa de película delgada de óxido de titanio (TiO_2) de espesor *"D"*, intercalado entre dos contactos de platino. La película semiconductora se encuentra dividida en dos regiones, una región con espesor *"w"* y altamente dopada con vacancias de oxígeno (TiO_{2-x}), y otra región de espesor "*D-w",* no dopada o con baja concentración de vacancias, tal como se muestra en la **Figura 2.2**a.

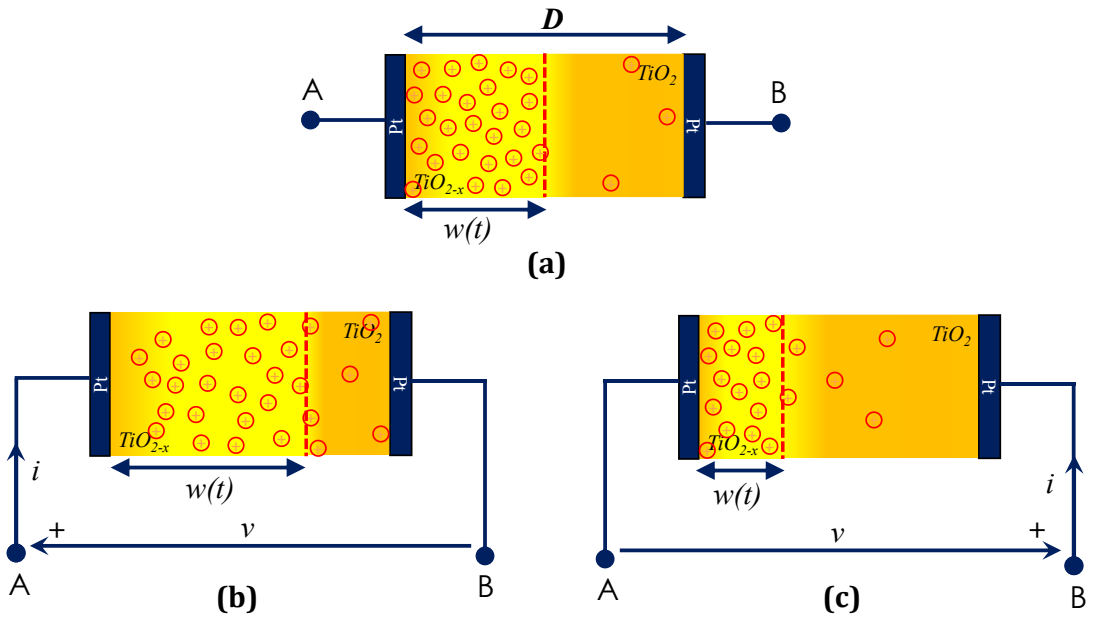


Figura 2.2 a) *Ilustración gráfica de la estructura del Memristor de Pt/TiO_2/Pt de HP® **b)** Caso 1: Si la corriente fluye de izquierda a derecha, el ancho de la región dopada aumenta debido a que las vacancias son empujadas hacia el contacto "B" debido al campo eléctrico **c)** Caso2: Efecto contrario cuando la polarización es negativa, las vacancias son atraídas al contacto "A".*

Básicamente, si se aplica un voltaje positivo en el electrodo de la izquierda del dispositivo *(A)*, éste repelerá las vacancias de oxígeno de la capa dopada (TiO_{2-x}) hacia la capa no dopada (TiO_2), es decir, el movimiento de las vacancias modulará el ancho de las regiones, incrementando la capa dopada y disminuyendo la no dopada (**Figura 2.2b**). Asimismo, se presenta un efecto contrario para voltajes negativos aplicados al dispositivo, es decir, las vacancias son atraídas al electrodo haciendo que la capa sin dopar se vuelva más ancha y, por tanto, más pequeña la región dopada (**Figura 2.2c**). En general el ancho de las regiones dependerá de la cantidad de carga que atraviesa al dispositivo.

De esta manera, la resistencia entre los dos electrodos muestra un comportamiento variante, en función del movimiento de la frontera entre regiones, presentando así, una forma general para describir cómo trabaja la memristencia provocando un comportamiento no lineal en el dispositivo.

El modelo propuesto considera que la resistencia total del Memristor puede determinarse a través de dos resistencias en serie, como lo muestra la **Figura 2.3**. Una de las regiones considera un valor resistivo bajo *"R_{ON}"* a raíz de que está altamente dopada, y la otra de alto valor resistivo *"R_{OFF}"*, consecuencia de la baja concentración (idealmente cero).

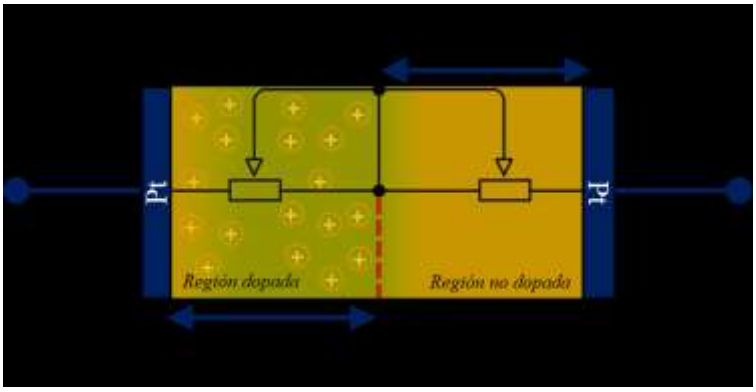


Figura 2.3 *Esquema simplificado de los estados resistivos del dispositivo a través de un diagrama equivalente de resistencias en serie.*

La resistencia del dispositivo es capaz de conmutar reversiblemente entre un estado menos conductivo (R_{OFF}) y uno más conductivo (R_{ON}) de acuerdo con la posición de la frontera entre regiones, por lo que la suma de éstas determinará la resistencia total del Memristor; esto es:

$$R_{MEM}(x) = R_{ON}x + R_{OFF}(1-x) \tag{2.3}$$

$$\text{Donde: } x = \frac{w(t)}{D} \in [0,1] \tag{2.4}$$

De esta expresión, la variable *"x"* se refiere a la posición normalizada de la frontera entre regiones, con respecto a la longitud total del dispositivo. Analíticamente, ante la aplicación de un voltaje positivo en la estructura, la respuesta de la expresión **(2.3)** es un estado resistivo bajo *"R_{on}"*, es decir, cuando la frontera entre regiones está al final de la región dopada, *w(t)= D*; asimismo, al final de la región no dopada y empalmando al contacto de platino, *w(t) = 0*, la respuesta es un valor de resistencia alto *"R_{off}"*. De esta manera, se cumple con una correcta aproximación del comportamiento de la resistencia en el dispositivo.

Respecto a la posición de la frontera, ésta puede modelarse en términos de la expresión del estado del sistema **(2.1)**, es decir:

$$\frac{dw(t)}{dt} = \mu_V \frac{R_{ON}}{D} i(t) \tag{2.5}$$

Así, es posible interpretar la evolución de la resistencia en función de la corriente *"i"*, y la movilidad iónica promedio *"μ_v"*, mediante una variable de estado general "*w(t)"*, que describe la rapidez de cambio de la frontera, diferente de la variable *"x(t)"* de los sistemas memristivos, en términos de la carga a través del Memristor.

Respecto a la relación *corriente–voltaje* del Memristor, puede expresarse fácilmente tomando en cuenta la ley de Ohm mediante la siguiente expresión:

$$v(t) = \left(R_{ON} \frac{w(t)}{D} + R_{OFF} \left(1 - \frac{w(t)}{D} \right) \right) i(t) \qquad \textbf{(2.6)}$$

A partir de esta expresión y en función del estado del sistema, puede determinarse una expresión para la memristencia del dispositivo, como:

$$M(q) = R_{OFF} \left(1 - \frac{\mu_V R_{ON}}{D^2} q(t) \right) \qquad \textbf{(2.7)}$$

De esta expresión, puede observarse que el valor de memristencia depende directamente de la movilidad de las impurezas y de la cantidad de carga que atraviesa al dispositivo. Además, tomando en cuenta la proporción inversa que tiene el espesor al cuadrado de la película con la memristencia, puede notarse cómo ésta tiene un papel primordial hablando en términos de reducción de circuitos electrónicos y cómo este fenómeno se hace más notable en dispositivos de apenas algunos nanómetros.

El modelo de Memristor de HP® presenta una primera interpretación física, correcta y sencilla para la aproximación del comportamiento descrito por Chua para este dispositivo. Así, la respuesta en las expresiones indicadas por el modelo, fundó la base en el desarrollo de modelos en favor del entendimiento de los fenómenos ocurridos en la estructura del Memristor. Además, con base en este primer acercamiento ha sido posible la evolución del modelado del dispositivo, enfatizando en el desarrollo de modelos que contemplen efectos físicos *"reales"* ocurridos en la estructura.

Finalmente, para llevar a cabo la simulación de la respuesta del Memristor, existen diferentes puntos de vista y métodos que permiten resolver las expresiones, por ejemplo, puede deducirse una expresión para la corriente a través del dispositivo [8], utilizando la ecuación para la variable de estado **(2.5)**. Sin embargo, para el caso de este trabajo y con el fin de simplificar, se toma en cuenta un punto de vista totalmente electrónico a través de la ley de Ohm, donde la corriente del Memristor puede conocerse con base en la solución de la ecuación de estado **(2.5)** mediante software [9], utilizando la herramienta Matlab®; los resultados pueden ser comprobados a través de las diferentes publicaciones, tomadas en cuenta a lo largo de este capítulo.

2.3 Modelo Lineal.

El descubrimiento del Memristor en 2008 por parte de la empresa Hewlett-Packard®, despertó el interés de los investigadores para aprovechar las características que presenta. De forma que se volvió necesario el desarrollo del modelo propuesto, con el principal objetivo de introducir fenómenos físicos *reales* ocurridos en el dispositivo. El modelo *ideal* de HP® asume que las vacancias tienen la libertad de moverse a través de todo el ancho del dispositivo y no considera comportamientos particulares que se presentan en el transporte de portadores en dispositivos nanométricos, tales como el Memristor. Por esta razón, se considera que este modelo toma en cuenta un punto de vista lineal, puesto que desestima las condiciones que se generan cerca de los bordes del dispositivo.

Con la finalidad de simular y observar este comportamiento lineal, es de utilidad la solución propuesta en la referencia [9], el método descrito permitirá analizar la gráfica *i-v* ideal del modelo propuesto por HP®. El diagrama de la **Figura 2.4** describe los pasos que permiten la solución de las expresiones del modelo.

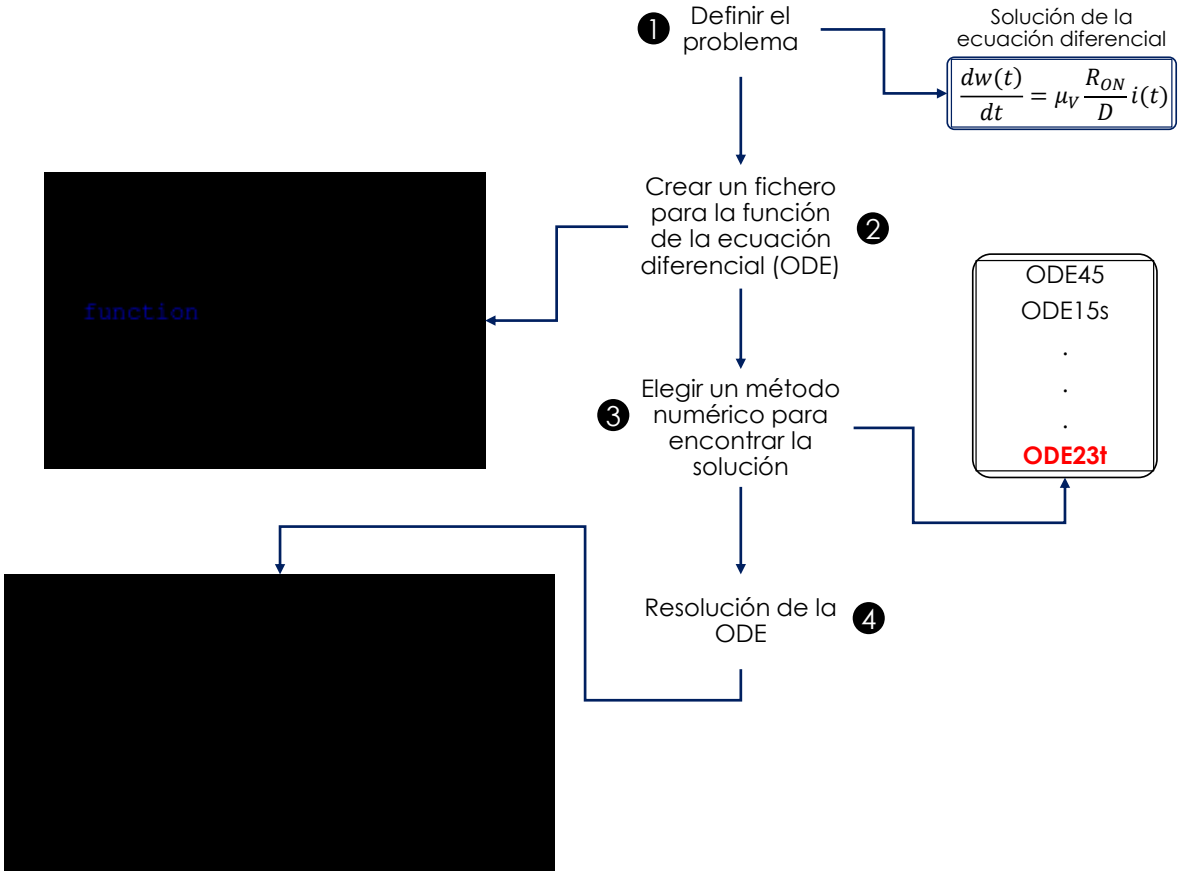


Figura 2.4. *Metodología empleada para llevar a cabo la simulación de la respuesta del modelo de HP®.*

De manera general, el método consiste en la solución de la ecuación diferencial **(2.5)**, en favor de conocer la *"posición"* de la frontera en términos del valor de la variable de estado (*w(t)*), por consiguiente, es posible determinar la resistencia del dispositivo, puesto que la normalización de la variable *"w(t)"* se involucra en la expresión para la resistencia del Memristor. De esta forma y despejando de la expresión para el voltaje **(2.6)** puede determinarse el valor de la corriente.

La solución de la ecuación de estado se efectúa mediante Matlab®, utilizando el procedimiento descrito para la solución de ecuaciones diferenciales ordinarias [10], haciendo uso de la herramienta ODE (*Ordinary Differential Equation*). Para obtener el resultado de esta ecuación, Matlab® presenta diferentes funciones ODE (**Tabla 2.1**) que resuelven el problema a través de métodos numéricos, los cuales varían dependiendo de la complejidad del problema. Una vez seleccionado el solucionador ODE, comúnmente éste divide el intervalo de tiempo en subintervalos, comenzando con un valor inicial para la variable de estado al que el integrador dará solución, *"w_0"*; la condición inicial puede calcularse despejando de la ecuación de resistencia del Memristor **(2.3)**, resultando en *0.3145*. Posteriormente y en función del método ODE, el fichero que almacena la ecuación diferencial calculará los valores siguientes en cada subintervalo.

Así, mediante el entorno de trabajo de Matlab®, es posible generar líneas de código que permitan la solución de las expresiones del modelo lineal de HP®. El código en términos del método descrito en la **Figura 2.4**, puede consultarse en el **Apéndice A**.

Función	Descripción
ODE5	Obtiene la solución en un solo paso, ideal para intentar obtener una primera aproximación. Se basa en el método Runge-Kutta.
ODE23	Más rápido, pero menos preciso, también se basa en el método Runge-Kutta y obtiene la solución en un solo paso.
ODE113	Obtiene la solución en múltiples pasos. Se utiliza en ecuaciones no demasiado complejas.
ODE15s	Obtiene la solución en varios pasos. Se utiliza cuando ODE45 no puede otorgar la solución.
ODE23s	Toma en cuenta ecuaciones complejas, obtiene la solución en un paso.
ODE23t	Resuelve ecuaciones de dificultad media. Puede resolver ecuaciones diferenciales algebraicas.
ODE23tb	Obtiene la solución de ecuaciones complejas. A veces más eficiente que ODE15s.

Tabla 2.1 *Funciones en Matlab® para la resolución de ODEs.*

Para la simulación de las ecuaciones, se toma en cuenta la aplicación de un voltaje senoidal de la forma "*V*sin (ω*t)"* y se utilizan los parámetros del modelo de TiO_2 para el Memristor de HP®, con R_{on}, R_{off}, D y μ_v a 100Ω, $16k\Omega$, $10nm$ y $10^{-10}cm^2s^{-1}v^{-1}$, respectivamente.

En la **Figura 2.5** se observa el resultado de la simulación variando la frecuencia en la señal, comprobando la condición de que el comportamiento de un Memristor tiende a ser el de una resistencia lineal conforme la frecuencia tiende a infinito.

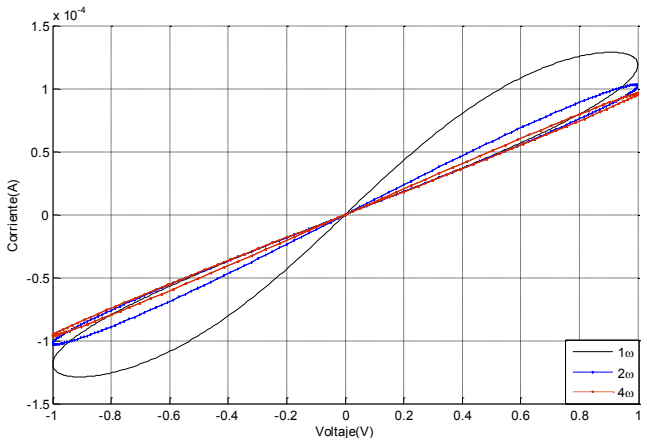


Figura 2.5 *Respuesta del comportamiento I-V de la simulación variando la frecuencia de trabajo.*

Finalmente, hablando en términos del comportamiento ideal de un Memristor, éste presenta un comportamiento simétrico de histéresis en la respuesta *i-v*. Esto significa que el cambio en la pendiente de los lóbulos de histéresis, demuestra que existe un fenómeno de conmutación entre diferentes estados resistivos, siendo una forma de identificar el comportamiento de un Memristor. De la gráfica, se observa que las expresiones propuestas para el modelo de Memristor de HP®, cumplen las características descritas por Chua, además de ser un modelo sencillo que permite entender el comportamiento de estos dispositivos.

2.4 Aproximación de la conducta no lineal mediante funciones de ventana.

El modelo descrito por HP® presenta algunas características particulares del Memristor y contempla valores de la posición de la frontera dentro del intervalo: $w(t) \in [0, D]$; sin embargo, este modelo no toma en cuenta las condiciones en los límites de tal intervalo. Básicamente en dispositivos nanométricos, pequeños voltajes pueden traer consigo campos eléctricos grandes, lo que produce un efecto no lineal en el arrastre de portadores. Además, considerando que la frontera entre regiones se mueve con base en la movilidad μ_v de los portadores, conforme se aproximan a los bordes, la velocidad tiende a reducirse [11], esto puede interpretarse como una disminución en el valor de la variable de estado. Asimismo, si se presentan condiciones donde se apliquen excursiones de voltaje grandes o que la velocidad de la señal sea tal que la frontera alcance saturación y se *"congele"*, el modelo no sería capaz de aproximar la respuesta, puesto que no contempla estas condiciones.

Para incluir este tipo de comportamientos, es de utilidad una familia de *"funciones de ventana"* [12], que permitan limitar la rapidez de cambio de la frontera cerca de los extremos del dispositivo.

De esta manera, multiplicando el lado derecho de la derivada de la variable de estado por una función de esta familia, es posible obtener una expresión que describa tales efectos, incluyendo las zonas cerca de los bordes, esto es:

$$\frac{dw(t)}{dt} = \mu_V \frac{R_{ON}}{D} i(t) \boldsymbol{f}(\boldsymbol{z}) \qquad \textbf{(2.8)}$$

De esta expresión, el término ***f(z)*** modela el fenómeno en los bordes, disminuyendo el valor de la variable de estado conforme tiende a los bordes. La respuesta de esta función normaliza el estado de la posición de la frontera y presenta una gráfica semejante a una forma parabólica [13], como lo muestra la **Figura 2.6**.

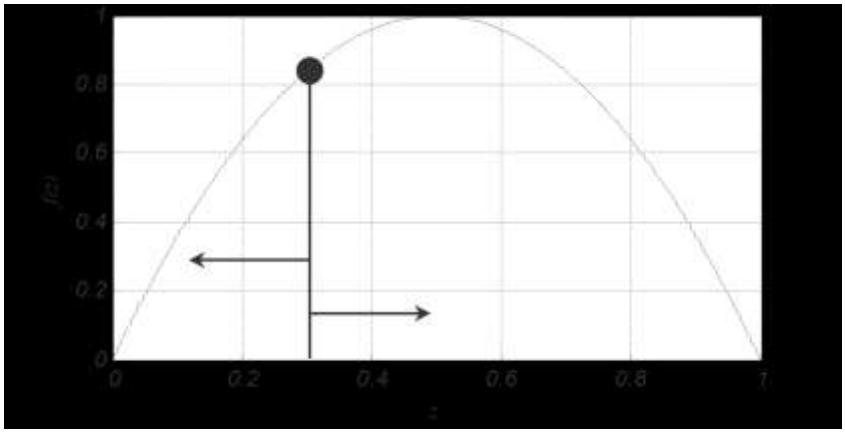


Figura 2.6 *Función cóncava normalizada, con dos raíces en z = 0 y z=1.*

En la gráfica es posible observar las características básicas que debe cumplir una función de ventana; es decir, en favor de modelar correctamente las condiciones en los bordes, la función debe estar descrita dentro del intervalo: $z \in [0, D]$, además, debe hacer que la derivada en el tiempo de la variable de estado tienda a cero cuando ésta se encuentre cerca de los límites de tal intervalo [14].

De forma particular en la referencia [15], además de proponer una función de ventana, se menciona una serie de condiciones adicionales a tomar en cuenta para el cumplimiento del ajuste de las diferentes funciones, esto es:

1) Deberá tomar en cuenta las condiciones en los bordes del dispositivo: $f(0) = f(1) = 0$.
2) Deberá ser capaz de imponer un arrastre *no-lineal* sobre el dispositivo: $f(0 < x < 1) \approx 1$.
3) Deberá proveer un enlace hacia el modelo de arrastre lineal.
4) Deberá ser escalable, $0 \leq f_{max}(w) \leq 1$.
5) Deberá poseer un parámetro de control que permita flexibilidad.

Mediante lo descrito en los siguientes segmentos, se analizarán las funciones de ventana propuestas por diferentes autores [6] [15] [16] [17], y se presentará un estudio breve acerca de las características de cada una. Cabe mencionar que, una función de ventana no contempla necesaria y totalmente un punto de vista no lineal, sin embargo, para el propósito de este trabajo, es de interés observar la respuesta del Memristor a través de estas funciones, puesto que describen el enfoque básico del comportamiento del dispositivo.

2.4.1 Strukov.

La primera función de ventana fue propuesta por Strukov [4], quien formaba parte del equipo de HP®. Strukov propuso una función parabólica simétrica con valor máximo al centro del dispositivo y que tiende a disminuir el valor de la variable de estado conforme se aproxima a los extremos.

$$f(w) = \frac{w(t)(D - w(t))}{D^2} \tag{2.9}$$

Esta función puede ser reescrita en términos de la normalización de la variable de estado, recordando que $x = w(t)/D$, resultando en:

$$f(x) = x - x^2 \tag{2.10}$$

De esta expresión, si se evalúa respecto a los bordes ($x \to 0$, $x \to D$), su valor en la frontera se vuelve cero ($f(0) = f(1) = 0$); además, considera el intervalo: $0 < x < D$, cumpliendo con la *no linealidad* del dispositivo, (**Figura 2.7**).

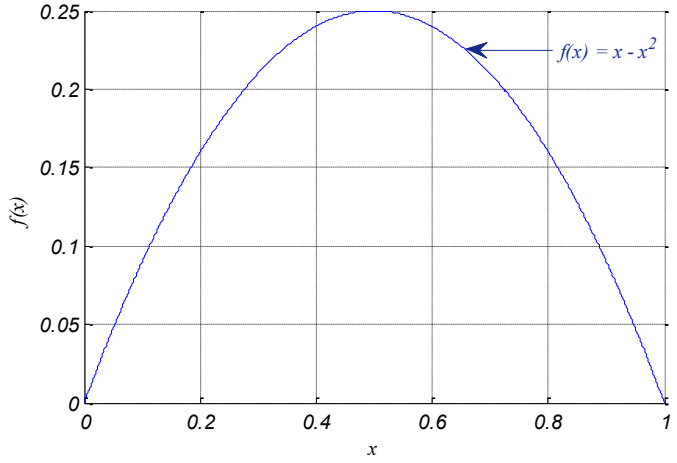


Figura 2.7 *Gráfica del comportamiento de la función propuesta por Strukov.*

El principal problema de esta función, se encuentra precisamente en los bordes, nombrado problema de *"estado terminal"*, es decir, cuando la frontera entre regiones tiende a los bordes del dispositivo, la ecuación de estado se vuelve cero, es decir:

$$\begin{cases} x \to 0 \\ x \to D \end{cases} \to \frac{dw(t)}{dt} = 0 \tag{2.11}$$

Este comportamiento, puede interpretarse como la ausencia de campo magnético que permita cambiar el valor de la variable de estado, truncando la posición de la frontera e impidiendo la conmutación en la resistencia del Memristor. A pesar de ello, y aunado a la falta de un parámetro de control que establezca el enlace con la aproximación del modelo lineal, ésta funcion presenta la base para el desarrollo de funciones de ventana subsiguientes mas completas.

2.4.2 Joglekar.

Joglekar y Wolf [16], propusieron una función de ventana considerando la expresión descrita por Strukov. Notaron que ésta podía ser modificada a través de un parámetro de control que permitiera establecer un vínculo con la linealidad y la no linealidad del dispositivo, generando una familia de curvas parametrizadas a través de la siguiente expresión:

$$f(x) = 1 - (2x - 1)^{2p} \tag{2.12}$$

De esta función, *"p"* es el parámetro de control, definido en relación con números positivos, y el término "*x"* se refiere a la variable de estado normalizada, además, cumple con la condición en los bordes: $f(0) = f(1) = 0$.

En la **Figura 2.8,** puede notarse que la función propuesta por Joglekar, es una versión escalada de la función de Strukov multiplicada por un factor de 4, cuando $p = 1$. Evaluada al centro del dispositivo ($x = 0.5$), presenta un valor igual a uno, por consecuencia, y de acuerdo con el incremento en la variable de control, la función tiende a formar un *"rectángulo"*, por lo tanto, el fenómeno de comportamiento *no-lineal* disminuye, es decir, conforme $p \rightarrow \infty$, el enfoque es el de un comportamiento totalmente lineal en el dispositivo, $f(0 < x < D) \approx 1$, presentando la descripción *ideal* del modelo de Memristor.

De esta manera se resuelve el problema de la falta del enlace entre aproximaciones de la función de Strukov, no obstante, se mantiene el inconveniente del *"estancamiento"* en los bordes, es decir, la imposibilidad de cambiar el valor de la variable de estado a través de cualquier excitación externa. En favor de solucionar este problema, algunos investigadores consideraron valores distintos de cero en la función de ventana de Joglekar, planteando una suposición de que la frontera no está exactamente empalmando los bordes.

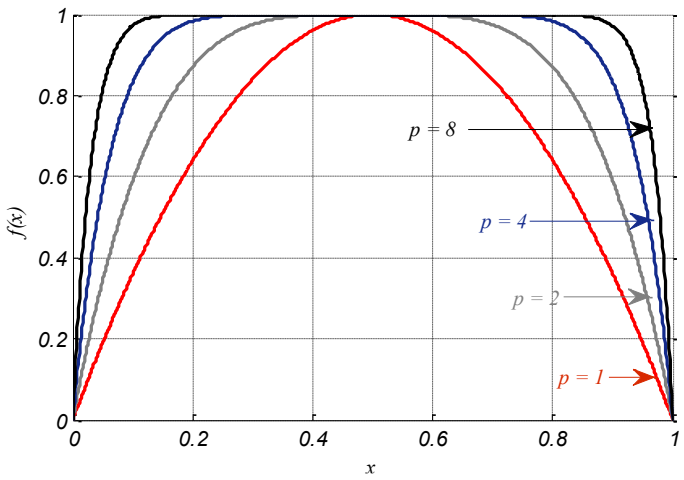


Figura 2.8 *Función de ventana propuesta por Joglekar, considerando diferentes valores de p.*

2.4.3 Biolek.

Biolek y su grupo de trabajo desarrollaron una función de ventana enfocados en la solución del problema del *estancamiento* de las funciones anteriores. Éstas, no son capaces de regresar otro valor en la variable de estado cuando se evalúan en los bordes, provocando que la frontera *permanezca* en alguno de los extremos; por consecuencia, se pierde el efecto memristivo, debido a que la resistencia se vuelve lineal, manteniendo uno de los valores límite, R_{on} o R_{off}.

La propuesta de Biolek se enfoca principalmente en que, para regresar la variable de estado a su valor inicial, puede considerarse el cambio en la dirección de la señal de excitación [17]. Por lo tanto, el equipo de trabajo de Biolek modificó la propuesta de Joglekar, introduciendo un parámetro adicional con el cual manipular el movimiento de la frontera en los bordes.

Así, en función del signo de la corriente que pasa a través del Memristor y considerando que ésta es positiva si incrementa el ancho de la región dopada, es posible obtener una serie de curvas que presenten diferentes valores para aproximarse y regresar de los bordes, es decir:

$$f(x) = 1 - (x - sgn(-i))^{2p} \tag{2.13}$$

$$sgn(i) = \begin{matrix} 1 \rightarrow i \geq 0 \\ 0 \rightarrow i < 0 \end{matrix} \tag{2.14}$$

De la misma forma que en la función de Joglekar, *"p"* es el parámetro de control y conforme éste tiende a infinito, la consideración es la aproximación del modelo lineal. Además, de esta expresión, el término *"sgn"* se refiere al signo de la señal de corriente *"i"*, designando un valor de 1, mientras la señal sea positiva; por el contrario, cuando la polaridad de la señal sea negativa, asignará un valor 0. De esta manera, la solución de la función evaluada en los bordes es diferente de 0, (*f (0) = f (1) = 1*), por consecuencia, la función es capaz de regresar otro valor en la variable de estado.

Este comportamiento puede interpretarse en función de dos curvas, consecuencia del cambio de signo en la señal, es decir, cuando la frontera entre regiones está en alguno de los bordes, ocurre un cambio de estado y la frontera regresa a su posición inicial a través de la segunda curva, **Figura 2.9**. De esta forma, se resuelve hasta cierto punto el problema de estado terminal, puesto que la función muestra discontinuidades, es decir, la solución propuesta por Biolek se traduce en comportamientos diferentes en el dispositivo: uno cuando la frontera va hacia el extremo y otro cuando regresa, afectando directamente el comportamiento de la memristencia.

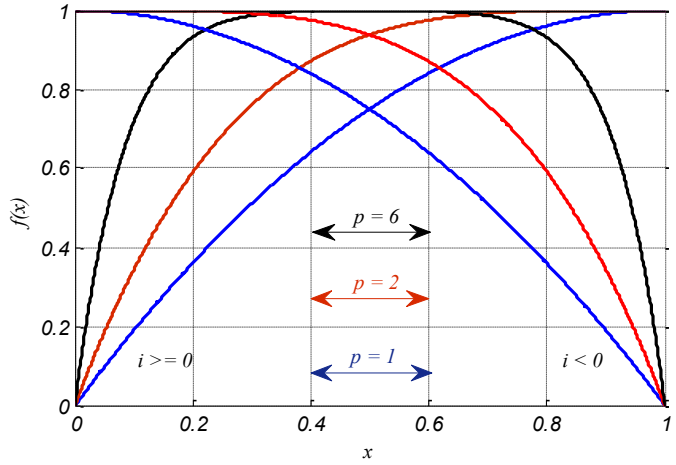


Figura 2.9 *Comportamiento de la función propuesta por Biolek para diferentes valores de p.*

A pesar de ello, esta función de ventana cumple con una correcta aproximación de las condiciones mostradas por el Memristor. Además, la propuesta del signo de la corriente, presenta una metodología a tomar en cuenta en el modelado del dispositivo. El análisis descrito por Biolek, sugiere el considerar factores, tales como la carga necesaria para que el dispositivo conmute; así, puede mencionarse que el análisis del comportamiento del Memristor puede seguir una perspectiva diferente, tomando en cuenta, por ejemplo, el campo eléctrico, generando comportamientos altamente no lineales.

2.4.4 Prodromakis.

Una característica que continuaba sin tratarse en las funciones de ventana descritas, era la escalabilidad, es decir, el valor máximo en la función es uno y no es posible obtener valores por encima o por debajo ($0 \leq f_{max}(w) \leq 1$); lo que en determinados casos es útil, puesto que permite conocer distintos comportamientos que podrían ajustar mejor los datos experimentales. A raíz de esto, Prodromakis, propuso una expresión descrita en términos de la función de Strukov, como:

$$f(x) = 1 - [(x - 0.5)^2 + 0.75]^p \qquad \textbf{(2.15)}$$

Esta expresión, presenta propiedades similares a las funciones anteriores, sin embargo, el parámetro *"p"* se describe con base en números reales, diferente del parámetro de Joglekar y Biolek, en función de números enteros positivos, mostrando *flexibilidad* en el valor de la variable de control.

En la **Figura 2.10** se presenta la solución de esta función para diferentes valores de *"p"*, puede notarse que, cuando *p=1*, el resultado es la curva de la función de Strukov. Además, para valores de *p=1, 4* y *8*, el enfoque, es el del comportamiento de arrastre *no lineal*, mientras que para valores arriba de *40*, la suposición es la del comportamiento lineal sobre toda el área del dispositivo, presentando una mejor adecuación en la variable de control.

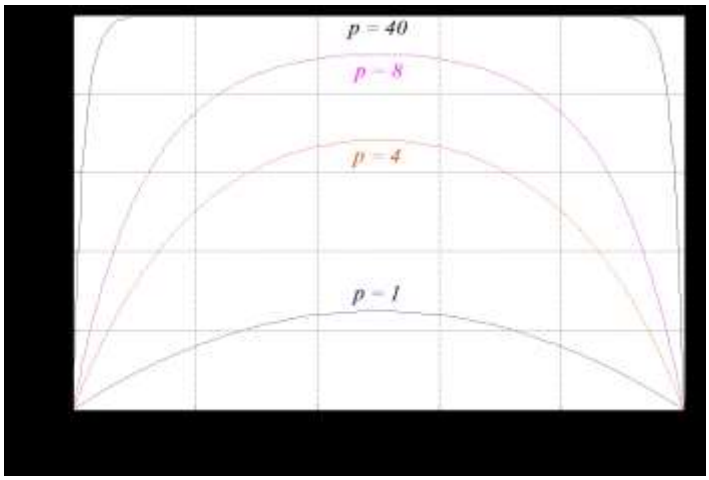


Figura 2.10 *Gráfica de la función propuesta por Prodromakis, presentando flexibilidad en el valor del parámetro de control "p".*

Respecto a la *escalabilidad*, la solución que permite presentar valores en $f_{max} \geq 1$, es a través de un tercer parámetro *"j"*, así, la función de ventana descrita por Prodromakis, puede expresarse como:

$$f(x) = j(1 - [(x - 0.5)^2 + 0.75]^p) \qquad \textbf{(2.16)}$$

A partir de esta expresión, es posible la formación de una familia de curvas en función de la definición previa del parámetro *"p"*. En la **Figura 2.11**, se muestra el comportamiento que presenta la función con *p=10* y variando el valor del parámetro *"j"*. De la gráfica, se observa la flexibilidad, además del alto grado de escalabilidad que demuestra el uso de los parámetros de control de la función de Prodromakis, con lo que se facilita el análisis del comportamiento no lineal del Memristor.

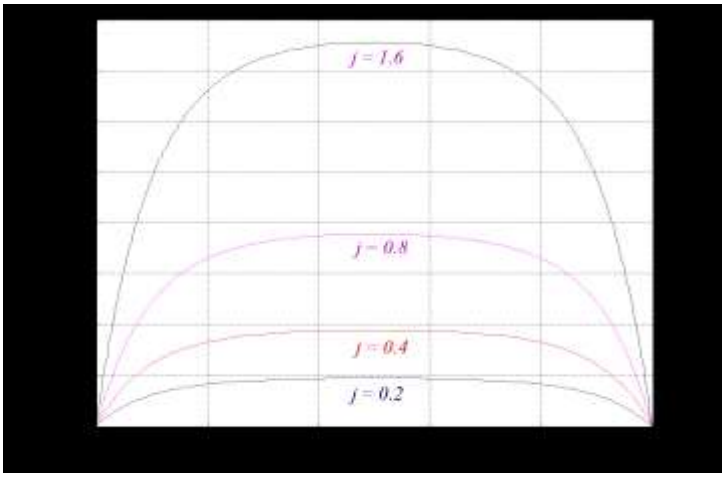


Figura 2.11 *Función de ventana con base en la escalabilidad, considerando p=10 y variando j.*

2.4.5 Comparación y Discusión.

Las *funciones de ventana* analizadas, contemplan ciertas particularidades que permiten modelar de mejor manera el comportamiento del Memristor. A través de la **Tabla 2.2**, se describe un entorno general de las características principales con las que cumplen las funciones tratadas [14].

Función de ventana *f(x)*	**Strukov**	**Joglekar**	**Biolek**	**Prodromakis**
Contempla simetría	☑	☑	☑	☑
Resuelve las condiciones de frontera	Parcialmente	Parcialmente	Discontinuidades	Parcialmente
Vínculo con el modelo de arrastre lineal	☒	☑	☑	☑
Escalable $0 \leq fmax(x) \leq 1$	☒	☒	☒	☑
Flexibilidad (parámetro de control)	☒	☑	☑	☑

Tabla 2.2 *Tabla comparativa de las características de las funciones de ventana.*

En general, las funciones de ventana descritas presentan dos problemas básicos: el primero, el nombrado "*problema de estado terminal"*, que congela el valor de la resistencia del Memristor cuando la frontera entre regiones está en los bordes y se genera en cualquiera de las funciones de ventana. Por ejemplo, la función no será capaz de regresar otro valor si el dispositivo es llevado a saturación, al no considerar correctamente los efectos en los bordes.

Biolek tomó en cuenta este problema y presentó un análisis general del porqué ocurre esta situación en saturación, postulando de esta forma, un segundo problema a ser tomado en cuenta en el modelado de Memristores: la "*acumulación de carga"*, es decir, el inconveniente radica en que el modelo toma en cuenta únicamente la variable de estado "*w(t)"* como parámetro para modelar la evolución del dispositivo, implicando que el Memristor es capaz de recordar la cantidad de carga que pasa a través de él. La solución descrita por Biolek, considera el sentido de la señal de corriente, resolviendo parcialmente esta condición, puesto que se generan discontinuidades que afectan directamente la memristencia del dispositivo.

Finalmente, es importante mencionar que el modelado correcto del Memristor involucra el desarrollo de una estrategia que solvente, en primera instancia, estos dos problemas. Sin embargo la dificultad principal radica en la variedad de mecanismos de conmutación que exhibe el comportamiento memristivo, dificultando el desarrollo de un modelo universal que ajuste los mecanismos físicos de trabajo de la memristencia. Así, la continua evolución del modelado del Memristor juega un papel importante hasta que se desarrollen y estandaricen las tecnologías de fabricación.

2.5 Modelo de Memristor a través de Simulink®.

Mediante la función de Matlab® para la solución de ecuaciones diferenciales ordinarias, y utilizando el entorno de trabajo de Simulink®, es posible desarrollar un esquema electrónico descrito en función de las expresiones matemáticas que definen al modelo de Memristor. De esta manera, se presenta una herramienta útil para efectuar el análisis del dispositivo, además, de acuerdo con la respuesta que se origine, se posibilitará la utilización de este *bloque* en un sistema que contenga más elementos electrónicos.

Biolek, describió dos perspectivas diferentes a tomar en cuenta de acuerdo con el desarrollo de la metodología para modelar el comportamiento del Memristor [18]. La primera, con base en las características sustantivas del dispositivo, es decir, el modelado se describe en relación con la aproximación de la parte *esencial* del Memristor; la segunda, en función de propiedades más *reales*, es decir, contemplando un sistema más robusto y por lo tanto difícil de implementar.

De esta manera, considerando la base del modelo de PSPICE descrito por Biolek [17] y en función del método descrito en la referencia [7], es posible implementar un modelo con base en bloques, que permitan aproximar la respuesta del dispositivo y que considere las condiciones en los bordes a través de las diferentes funciones de ventana, además, deberá tomar en cuenta la polaridad del dispositivo.

El desarrollo del modelo propuesto se efectúa desde el punto de vista *idealizado* del comportamiento del Memristor, es decir, el principio físico que describe la memristencia en el dispositivo, toma en cuenta la posición de la frontera entre regiones para determinar el valor de resistencia, por consiguiente, la descripción del modelo se efectúa de acuerdo con la solución de la ecuación de estado **(2.5)**.

La ecuación de estado, presenta una expresión dependiente del tiempo, como lo es la corriente que pasa a través del Memristor. Aprovechando esta situación, es posible describir una expresión general en función del tiempo que calcule el valor de la variable de estado. Así, integrando ambos lados de la ecuación, se tiene:

$$x(t) = \int_0^t ki(t) \rightarrow k = \frac{\mu_v R_{ON}}{D^2} \tag{2.17}$$

La solución de esta integral, permite conocer el valor de resistencia en el Memristor, puesto que se involucra en la expresión de resistencia, como:

$$R_{MEM}(x) = R_{ON}x + R_{OFF}(1 - x) \tag{2.18}$$

Finalmente, la corriente a través del dispositivo puede determinarse con base en el análisis de la ley de Ohm, mediante:

$$i(t) = \frac{v(t)}{R_{MEM}(x)} \tag{2.19}$$

El comportamiento de estas expresiones, puede describirse en función del esquema eléctrico mostrado en la **Figura 2.12**, donde se observa, además, que el comportamiento no lineal del dispositivo puede ser ajustado a través de un bloque de retroalimentación *f(x),* permitiendo utilizar las diferentes funciones de ventana.

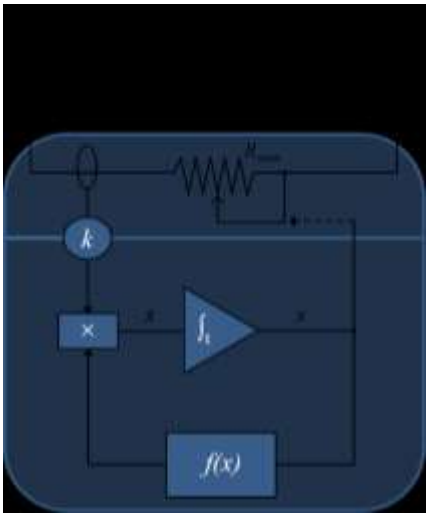


Figura 2.12 *Esquema Eléctrico del Memristor.* [17]

El diagrama mostrado, interpreta el comportamiento memristivo en función de una resistencia variable, de la cual, es necesario determinar el primer valor a partir de una condición inicial en la variable de estado, a fin de que el sistema *inicie.* De esta forma, se obtiene un primer valor de corriente que el sistema *censa* para posteriormente continuar con la determinación de los valores siguientes en términos de la ley de Ohm, y ante la aplicación de una señal de voltaje senoidal.

A partir de lo descrito y con base en el entorno de trabajo de Simulink®, es posible desarrollar un sistema similar al esquema eléctrico, capaz de resolver las expresiones propuestas para el Memristor. De esta forma, se proveerá un sistema en forma de *bloque* que facilite el análisis del comportamiento *corriente-voltaje* del Memristor, asimismo, facilitará su uso en sistemas que contengan más elementos. El esquema en Simulink® que permite interpretar la respuesta del Memristor se muestra en **Figura 2.13.**

Para que el integrador genere la solución, Matlab® presenta distintos métodos en función de la complejidad del problema, por lo que dependerá de éste el definir el tipo de función. Anteriormente, en la **Tabla 2.1** se mostró un listado de los tipos de solucionadores ODE que presenta la herramienta. Para el diseño de este modelo, se empleó el solucionador *ODE23t* y se considera un valor igual a *0.015* en la muestra del paso máximo de la configuración global de la simulación [9].

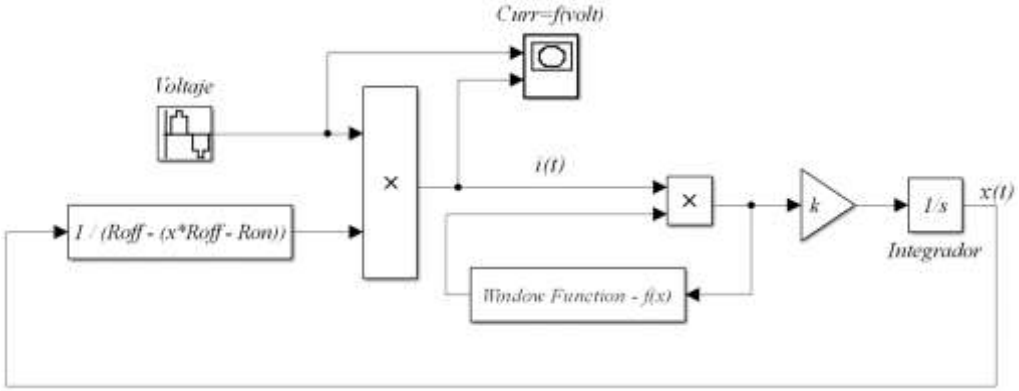


Figura 2.13 *Modelo del comportamiento del Memristor a través de Simulink®.*

El bloque integrador requiere que se presente la condición inicial en la variable de estado, x_0, en favor de que el sistema *inicie*. Esta condición, puede obtenerse a partir de la ecuación de la memristencia del dispositivo **(2.3)**. Por consiguiente, la expresión que define el primer valor al que el integrador dará solución se expresa como:

$$x_0 = \frac{R_{off} - R_{init}}{R_{off} - R_{on}} \qquad \textbf{(2.20)}$$

Respecto a las condiciones en los bordes, es de utilidad el desarrollo de un *subsistema*: *"Window Function f(x)",* que permita interpretar las expresiones descritas por las diferentes funciones de ventana, asimismo, que considere el ajuste de los parámetros de control *"p"* o *"j"*. El esquema propuesto para este bloque se muestra en la **Figura 2.14**.

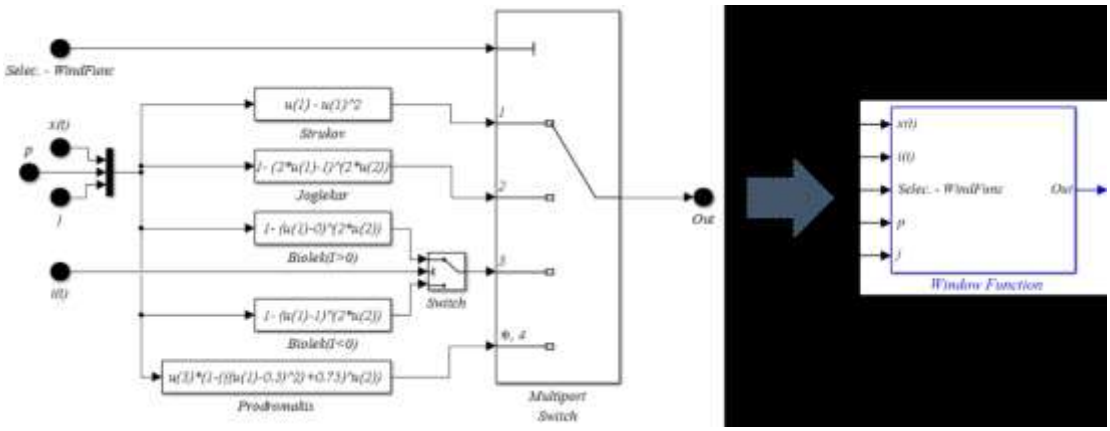


Figura 2.14 *Esquema a bloques para la aproximación de la respuesta de las funciones de ventana.*

De forma general, el esquema presentado en la **Figura 2.13** es capaz de resolver las expresiones matemáticas propuestas para el modelado del Memristor, sin embargo, es necesario construir un sistema electrónico que considere el funcionamiento electrónico descrito hipotéticamente en el esquema de la **Figura 2.12**.

A fin de construir este sistema, es de utilidad la librería Simscape® [19] de Simulink®, que contiene un conjunto de herramientas que permiten la simulación de sistemas *físicos*, ya sea que se desee aproximar la respuesta de sistemas hidráulicos, mecánicos, electrónicos, etc. Por ejemplo, en la **Figura 2.15** se muestran algunos de los bloques útiles para desarrollar un circuito electrónico.

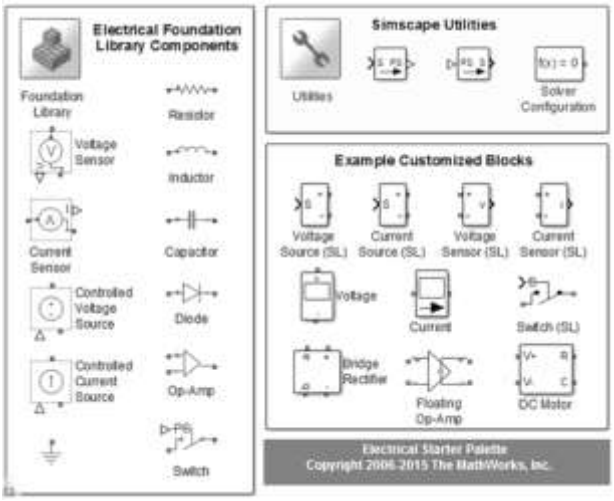


Figura 2.15 *Librería Simscape® Electronics de Simulink® para la aproximación de modelos físicos.* [20]

De esta manera, a través de Simscape® es posible desarrollar un diagrama en función de la descripción de un sistema electrónico, que tome en cuenta el flujo de corriente a través de una resistencia *variable*, excitada mediante una señal de voltaje senoidal, tal como se muestra en la **Figura 2.16.**

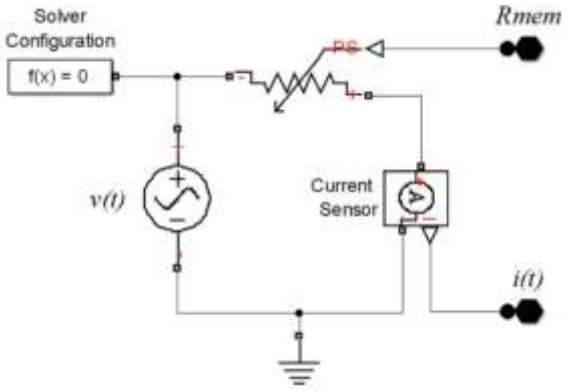


Figura 2.16 *Esquema electrónico que permite interpretar el principio de funcionamiento de la resistencia del Memristor.*

Es importante mencionar que cada red de circuito representada por un diagrama a bloques en Simscape®, requiere parámetros de configuración local que permitan la solución de la simulación [21]. Esto se efectúa mediante el bloque denotado como *"configuration solver"*, especificando los parámetros necesarios para que la simulación comience. En este caso, se consideran los valores predeterminados *(default)* del solucionador.

Finalmente, para concluir con el desarrollo del modelo propuesto en función del diagrama *matemático* de la **Figura 2.13** y el diagrama *electrónico* de **Figura 2.15**, es necesario, el uso de bloques que permiten la interacción entre un sistema de Simscape® y uno de Simulink® (PSS).

Asimismo, se debe considerar el uso de algún bloque que permita conocer la respuesta de corriente o voltaje que presentan los elementos del diagrama electrónico, es decir, el entorno de Simscape® requiere de *elementos* que *censen* el voltaje o corriente, en favor de presentar la respuesta.

Esta librería (Simscape®) contiene bloques que permiten el *censado* de manera sencilla (*"Current Sensor" o "Voltage Sensor"*). De esta forma, mediante la utilización de estos bloques, se podrá obtener el valor de corriente en función de la variación de la resistencia en el Memristor a fin de obtener el valor de la variable de estado, mediante la solución de la integral.

Finalmente, el esquema completo del modelo *matemático/electrónico* propuesto del Memristor se describe en la **Figura 2.17**.

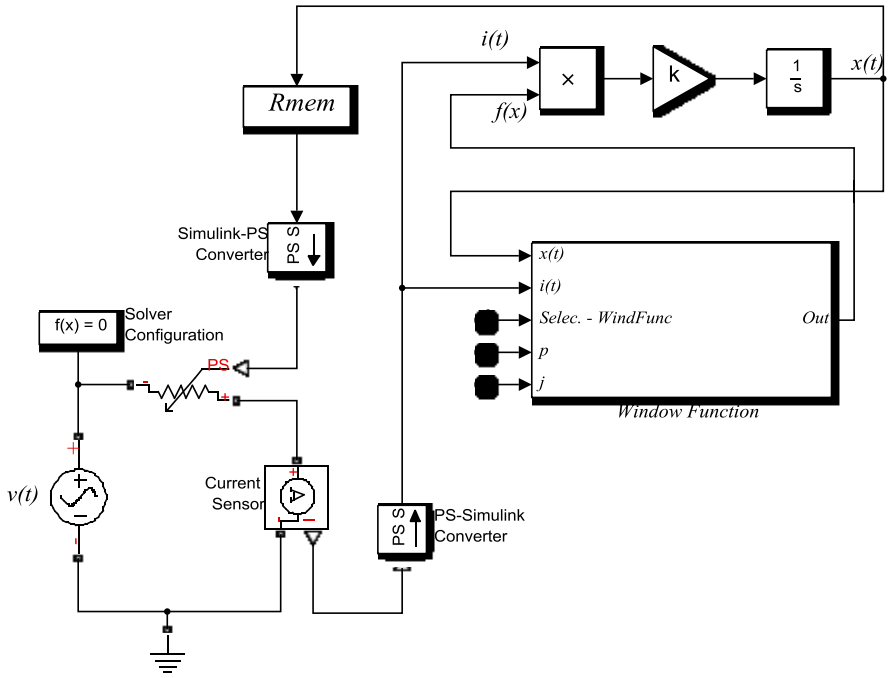


Figura 2.17 *Diagrama físico del Memristor a través de Simulink® y Simscape®.*

A partir del modelo descrito en Simscape® y Simulink® es posible desarrollar un *subsistema* que contenga la parte electrónica y matemática que permiten aproximar la respuesta del Memristor. De esta forma, se genera un dispositivo de dos terminales en función de un bloque que actúa como *"caja negra"*. El bloque descrito se muestra en la **Figura 2.18**, y presenta la ventaja de permitir visualizar los diferentes comportamientos que se generan en los bordes, utilizando las funciones de ventana.

Además, el modelo es conmutable, es decir, la polaridad del Memristor puede ser cambiada, esto es importante, puesto que la conducta de la resistencia del dispositivo incrementa o disminuye según la polaridad de la señal aplicada [22]. Respecto con el símbolo propuesto por la teoría de Chua, se presenta un modelo de Memristor, donde el valor de resistencia disminuye si la corriente fluye de la barra negra hacia abajo, de forma contraria, aumenta.

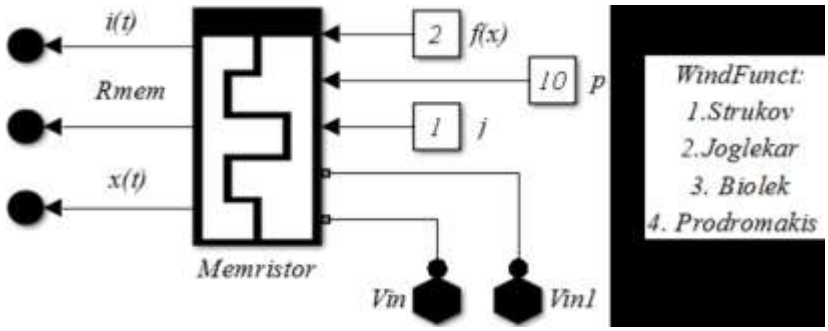


Figura 2.18 *Modelo de Memristor como dispositivo de 2 terminales en Simulink®.*

La respuesta de la simulación de este *subsistema* se efectúa aplicando una señal senoidal de 1 volt de amplitud y 1 Hz en frecuencia. Además, utilizando la función de ventana de Joglekar con $p = 10$ y considerando los parámetros del modelo de Memristor de HP®. (Ver **Figura 2.19**)

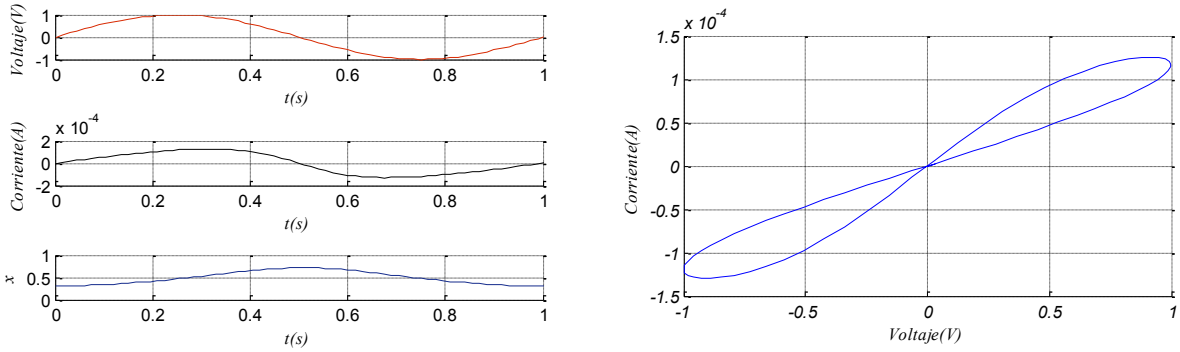


Figura 2.19 *Análisis transitorio de las señales de salida y respuesta corriente – voltaje en el dispositivo.*

Finalmente, es de mencionarse que se efectuó la simulación del modelo propuesto en Simulink®, utilizando diferentes solucionadores ODE, sin embargo, se optó por utilizar el solucionador ODE23t, puesto que genera una respuesta más exacta y resultados que ajustan mejor el comportamiento del Memristor [23].

Los resultados obtenidos de las diferentes simulaciones, son comparables con los resultados generados por el modelo de PSPICE descrito por Biolek en la referencia [17].

2.6 Conclusiones

Mediante la descripción de este capítulo, fue posible el análisis de la teoría que permite interpretar el comportamiento del modelo de Memristor propuesto por la compañía HP®. Además, con el objetivo de conocer la respuesta en función de las condiciones en los bordes de la estructura, se efectuó el análisis general de las diferentes funciones de ventana.

Respecto a la simulación de la respuesta del modelo de HP®, ésta se realizó con base en lo descrito en la teoría y de acuerdo con las diferentes herramientas de trabajo de Matlab®, describiendo la respuesta del Memristor en relación con líneas de código y a través de un sistema a bloques mediante Simulink® y Simscape®. De esta manera, fue posible el análisis y la exploración de herramientas que muestran ambientes de desarrollo sencillos de manipular y que adecuan correctamente la respuesta de sistemas complejos.

Finalmente, es importante mencionar que, aprovechando el ambiente de integración de Matlab®, es posible manipular de manera sencilla los datos obtenidos del modelo de Memristor descrito en Simulink®, es decir, el desarrollar tanto el modelo de Memristor como la arquitectura de red neuronal en el mismo entorno de trabajo, facilita y agiliza el trabajo de investigación.

Referencias

[1] R. S. Williams and M. D. Pickett, "The art and Science of Constructing a Memristor Model", de *Memristor and Memristive Systems*, New York, Springer, 2014, pp. 93-104.

[2] L. Chua, «Device modeling via nonlinear circuit elements», *IEEE Transactions on Circuits and Systems,* vol. 27, nº 11, pp. 1014-1044, 1980.

[3] L. O. Chua and S. M. Kang, "Memristive devices and systems", *Proceedings of the IEEE,* vol. 64, no. 2, pp. 209-223, 1976.

[4] S. P. Adhikari, M. P. Sah, H. Kim and L. O. Chua, "Three Fingerprints of Memristor", *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 60, no. 11, pp. 3008 - 3021, 2013.

[5] R. S. Williams, «How We Found The Missing Memristor», *IEEE Spectrum,* vol. 45, nº 12, pp. 28-35, 2008.

[6] D. B. Strukov, G. S. Snider, D. R. Stewart y R. S. Williams, «The missing memristor found», *Nature,* vol. 453, pp. 80-83, 2088.

[7] A. Adamatzky y L. Chua, Memristor Networks, Springer International Publishing, 2014.

[8] O. Kavehei, Y.-S. Kim, A. Iqbal, K. Eshraghian, S. F. Al-Sarawi and D. Abbott, "The fourth element: Insights into the memristor," *Communications, Circuits and Systems, 2009. ICCCAS 2009. International Conference on.,* pp. 921-927, 2009.

[9] K. Zaplatilek, «Memristor modeling in MATLAB & Simulink,» *ECC'11 Proceedings of the 5th European Conference on Computing Systems,* pp. 62-67, 2011.

[10] A. Gilat, «Matlab Una introducción con ejemplos prácticos,» Editorial Reverté, S. A, 2005, pp. 251-270.

[11] O. Kavehei, A. Iqbal, Y. S. Kim, K. Eshraghian, S. F. Al-Sarawi y D. Abbott, «The Fourth Element: Characteristics, Modelling, and Electromagnetic Theory of the Memristor,» *Proceedings of The Royal Society,* 2010.

[12] H. Elgabra, I. A. H. Farhat, A. S. A. Hosani, D. Homouz y B. Mohammad, «Mathematical modeling of a memristor device,» *Innovations in Information Technology,* pp. 156-161, 2012.

[13] P. S. Georgiou, S. N. Yaliraki, M. Drakakis y M. Barahona, «Window functions and sigmoidal behaviour of memristive systems,» *International Journal of Circuit Theory and Applications,* 2016.

[14] A. G. Radwan y M. E. Fouda, On the Mathematical Modeling of Memristor, Memcapacitor, and Meminductor, Springer, 2015.

[15] T. Prodromakis, B. P. Peh, C. Papavassiliou y C. Toumazou, «A Versatile Memristor Model With Nonlinear Dopant Kinetics,» *IEEE Transactions on Electron Devices,* vol. 58, nº 9, pp. 3099-3105, 2011.

[16] Y. N. Joglekar y S. J. Wolf, «The elusive memristor: properties of basic electrical circuits,» *European Journal of Physics,* vol. 30, pp. 661-675, 2009.

[17] Z. Biolek, D. Biolek y V. Biolková, «Spice Model of Memristor With Nonlinear Dopant Drift,» *Radioengineering,* vol. 18, pp. 210-214, 2009.

[18] D. Biolek and Z. Biolek, "Fourth Fundamental Circuit Element: SPICE Modeling and Simulation", en *Memristors and Memristive Systems*, New York, Springer, 2014, pp. 105-162.

[19] [En línea]: http://www.mathworks.com/help/physmod/simscape/.

[20] [En línea]: http://www.mathworks.com/products/simelectronics/features.html#key-features.

[21] [En línea]: http://www.mathworks.com/help/physmod/simscape/ref/solverconfiguration.html.

[22] H. Kim, M. P. Sah, C. Yang, S. Cho y L. O. Chua, «Memristor Emulator for Memristor Circuit Applications,» *IEEE Transactions on Circuits and Systems I: Regular Papers ,* vol. 59, nº 10, pp. 2422-2431, 2012

[23] Mathworks, Matlab&Simulink: Simscape User's Guide, 2016.

Capítulo 3 Aplicación del Memristor.

El rápido crecimiento de la industria tecnológica ha traído grandes cambios en la vida de las personas, facilitando muchas de las actividades realizadas día con día. Por consecuencia, en favor de que los dispositivos electrónicos realicen tareas más específicas, es indispensable la constante evolución de estos, enfatizando tanto en el desarrollo de nuevas tecnologías de fabricación, como en las técnicas de diseño electrónico asociadas. Esto implica nuevos retos, tales como buscar soluciones innovadoras para que los materiales sean capaces de mantener sus características conforme la tendencia de reducción se mantenga. Es por esto que el Memristor siendo un nuevo elemento eléctrico, juega un papel primordial en esta revolución tecnológica, puesto que, por ejemplo, manteniendo las dimensiones de un transistor combinado con un Memristor se podría innovar radicalmente el desempeño de los circuitos digitales [1], permitiendo además un área de diseño reducida.

3.1.Introducción.

Una posible aplicación para el Memristor de acuerdo con su comportamiento no lineal, es para la creación de circuitos osciladores [2] [3] [4] [5], imprescindibles en sistemas electrónicos para generar señales de frecuencia y forma de onda determinadas, para aplicaciones, por ejemplo, de prueba y medición. El incremento y decremento en la resistencia del Memristor de acuerdo con el voltaje aplicado, se asemeja a la carga y descarga de un elemento reactivo, por lo que es posible utilizarlo en el diseño de osciladores para producir, por ejemplo, ondas cuadradas, indispensables en sistemas digitales; la frecuencia y periodo de un oscilador depende comúnmente de la carga y descarga de elementos reactivos (capacitores o inductores). En este caso, el retraso inherente en la respuesta del Memristor es ideal para realizar la función de oscilación, además, la propiedad del almacenamiento de la resistencia del Memristor elimina la necesidad de utilizar dichos elementos reactivos, los cuales, además, incrementarían el área de diseño.

El oscilador, puede conformarse a partir de un Memristor en serie con una resistencia; ésta puede ser variable para adecuar la frecuencia del oscilador. Asimismo, en favor de generar la señal de salida, es necesario un circuito de control ($F(V_i)$) que permitirá ajustar el voltaje en el Memristor a través de una función de retroalimentación. Las posibles configuraciones se muestran en la **Figura 3.1**.

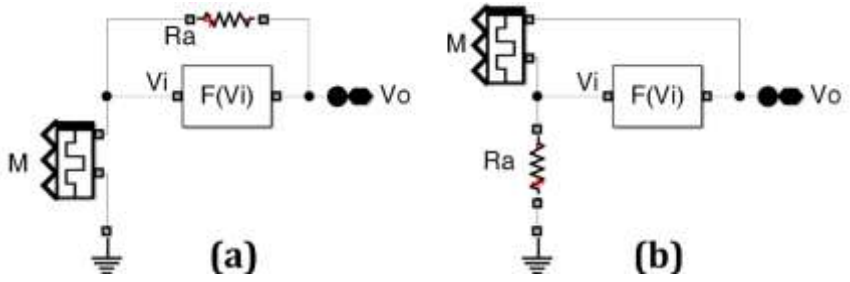


Figura 3.1 *a) Oscilador Resistor-Memristor (R-M) b) Oscilador Memristor-Resistor (M-R).*[6]

3.2. Oscilador Resistor-Memristor sin reactancia.

En este capítulo se efectuará el análisis del oscilador sin reactancia con base en un Memristor, presentado en la referencia [2]. De acuerdo con el diagrama anterior, el circuito simula la respuesta de carga y descarga de un circuito RC a través de un divisor de voltaje conformado a partir de una resistencia y un Memristor en serie. Además, el módulo de control puede desarrollarse en función de dos comparadores en paralelo (**Figura 3.2a**), donde, la salida de cada uno se convierte en el control de entrada para una compuerta AND encargada de generar la oscilación.

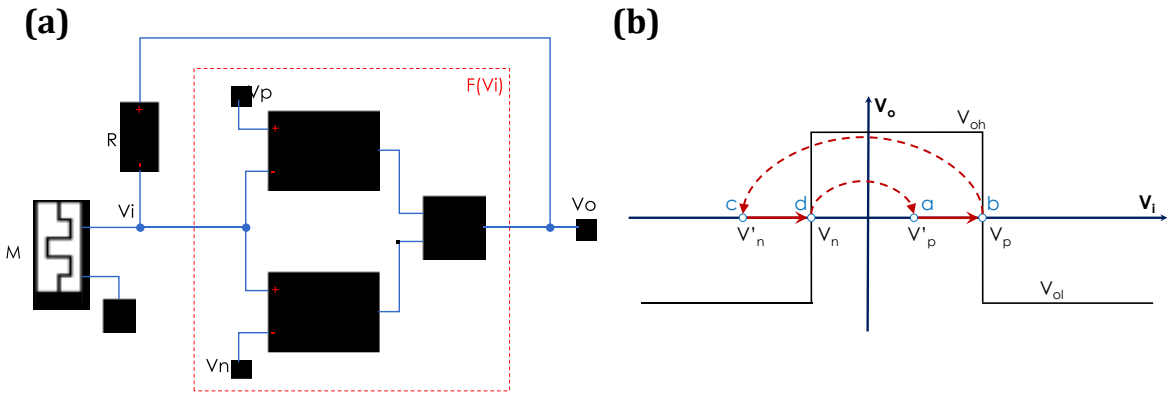


Figura 3.2 *a) Oscilador sin reactancia con base en un Memristor. b) Función de transferencia F(V_i) mostrando la transición entre los diferentes puntos de operación.*

Con base en la correcta oscilación del circuito y con base en la operación de los comparadores, los voltajes de polarización, deberán definirse tal que: $V_p > V_n$. Asimismo, los voltajes en alto y bajo de la compuerta se describen a través de la condición: $V_{OH} > V_{OL}$.

Además, el Memristor está polarizado tal que su resistencia (R_M) incrementa si el voltaje de retroalimentación V_O es más positivo respecto a tierra. De esta manera, asumiendo el inicio de operación en el punto *'a'* de la **Figura 3.2b**, puede trazarse el funcionamiento del oscilador como:

a) La operación del circuito comienza asumiendo un voltaje de salida positivo ($V_O=V_{OH}$), por consecuencia, la resistencia del Memristor incrementará y por lo tanto el voltaje en el nodo V_i. La salida del circuito se conserva en alto, mientras: $V_p' \leq V_i < V_p$, puesto que la condición de salida en los comparadores define un valor en 0 si $V_p \leq V_i$ y en 1 si $V_i \geq V_n$, [7].

b) En el punto 'b', el voltaje en el nodo V_i es equivalente a V_p, consecuencia de que la resistencia R_M presenta su valor máximo y, considerando el principio de operación de los comparadores, la salida del circuito conmutará al voltaje en bajo de la compuerta ($V_O=V_{OL}$).

c) En este punto, se presenta un voltaje de la misma magnitud que V_p, pero de sentido contrario ($V_i=V_n'$) y la salida del circuito se mantiene en bajo, mientras: $V_n' \leq V_i < V_n$.

d) A partir de este punto, la resistencia R_M es mínima y el voltaje en el nodo $V_i=V_n$, además, $V_i<V_p$, por consecuencia, el voltaje de salida del circuito conmutará al voltaje en alto de la compuerta ($V_o=V_{OH}$); regresando al punto de operación 'a'.

Mediante la operación descrita y tomando en cuenta el modelo de Memristor de HP®, es posible desarrollar las expresiones necesarias para encontrar la frecuencia de trabajo del circuito de acuerdo con el método descrito en la referencia [8]. Además, se efectuará el análisis considerando voltaje de referencia y sin considerar la referencia de voltaje en el Memristor.

El Memristor puede referenciarse a voltaje en favor de que, a través de éste, sea posible ajustar la frecuencia de oscilación. De esta forma, es posible el desarrollo de un modulador digital/analógico en función de la señal de entrada que se aplique. El diagrama de la **Figura 3.3** muestra la configuración descrita del circuito.

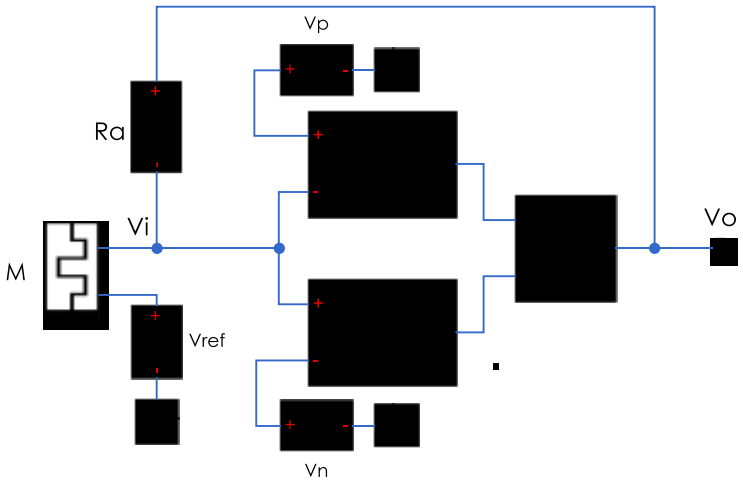


Figura 3.3 *Oscilador R-M sin reactancia.*

La función de oscilación se efectúa con base en el voltaje en el nodo "*Vi*", sin embargo, para facilitar el análisis se considera el voltaje en el Memristor, el cual puede describirse como:

$$V_M(t) = (V_o(t) - V_{ref})\frac{R_m}{R_m + R_a} \qquad \textbf{(3.1)}$$

Este voltaje se determina en relación con el valor de retroalimentación "$V_o(t)$", puesto que define si la resistencia en el Memristor R_M aumenta o disminuye. Además, la condición de salida del módulo de control ($F(V_i)$), puede definirse de acuerdo a la lógica descrita por la compuerta AND y en función de la operación de los comparadores, es decir:

$$F(V_i) = \begin{matrix} \boldsymbol{V_{OH}} \rightarrow V_p' \leq V_i < V_p \\ de\ otra\ manera\ \rightarrow \boldsymbol{V_{OL}} \end{matrix} \qquad \textbf{(3.2)}$$

De esta expresión el voltaje V_p', se refiere al voltaje en relación con la conmutación en alto de la compuerta, puesto que R_M es mínima y $V_i=V_n$. Asimismo, cuando $V_i=V_p$ con R_M máxima, se presentará el voltaje V_n' de acuerdo con la salida en bajo de la compuerta.

Respecto a la variación en el valor de memristencia R_M, ésta se define con base en los voltajes de polarización de los comparadores, "V_p" y "V_n", presentando un valor de resistencia máximo en "R_{Mp}" y mínimo en "R_{Mn}", respectivamente. Estos valores pueden determinarse a partir de **(3.1)** como:

$$R_{Mp} = R_a \frac{V_p - V_{ref}}{V_{oh} - V_p}; R_{Mn} = R_a \frac{V_n - V_{ref}}{V_{ol} - V_n} \quad \textbf{(3.3)}$$

Los voltajes de polarización en los comparadores "V_p" y "V_n", son de utilidad en el funcionamiento adecuado del oscilador, puesto que evitan que "R_M" se mantenga constante, a raíz de que la frontera entre regiones del Memristor alcanzó alguno de los bordes. De esta forma, R_{Mp} y R_{Mn} deberán definirse tal que la frontera no alcance saturación, es decir:

$$R_{on} < R_{Mn} < R_{Mp} < R_{off} \quad \textbf{(3.4)}$$

Asimismo, hablando en términos de parámetros de diseño, de acuerdo con el valor de "R_a", la condición necesaria para la correcta oscilación del circuito puede obtenerse sustituyendo **(3.3)** en **(3.4)**, resultando en:

$$R_{on}\left(\frac{V_{ol} - V_n}{V_n - V_{ref}}\right) < R_a < R_{off}\left(\frac{V_{oh} - V_p}{V_p - V_{ref}}\right) \quad \textbf{(3.5)}$$

Además, el tiempo requerido por el circuito para que cambie su estado del punto 'a' al punto 'b' está descrito en función del tiempo necesario para que el Memristor cambie su resistencia de *"R_{Mn}"* a *"R_{Mp}"*. Así, el tiempo para obtener el semiciclo positivo de la señal puede describirse como:

$$T_H = Tiempo(R_{Mn} \rightarrow R_{Mp}) \quad \textbf{(3.6)}$$

Respecto a la variación de resistencia en el Memristor, se considera el modelo de HP® [9], donde se describe el movimiento de la frontera entre la región dopada y la no dopada, mediante la ecuación de estado en función del parámetro *"w(t)"*, es decir:

$$\frac{dw(t)}{dt} = k\, i(t)\, f(x) \rightarrow k = \frac{\mu_v R_{on}}{D^2} \quad \textbf{(3.7)}$$

Para el análisis del oscilador se hace referencia a la función de ventana descrita por Joglekar [10], es decir: $f(x) = 1 - (2x - 1)^{2p}$; evaluada en los bordes, resulta en: $f(0) = f(1) = 0$; cumpliendo la no linealidad del dispositivo. Además, se considera $p = 1$, puesto que a un valor mayor en p, la función se vuelve lineal, presentando un efecto no deseado en el análisis de este circuito.

Para facilitar el desarrollo de las expresiones [11], la función de ventana puede describirse como:

$$f(x) = 4x(1-x). \tag{3.8}$$

De la misma forma, considerando el modelo de HP®, la resistencia del Memristor se define de acuerdo con la suma de las resistencias en serie respecto a la región dopada y no dopada; es decir: $R_{MEM}(x) = R_{ON}\,x + R_{OFF}\,(1-x)$. Simplificando en términos de x, la resistencia del Memristor puede definirse como:

$$R_{MEM}(x) = R_{OFF} - x(R_{OFF} - R_{ON}) \tag{3.9}$$

Además, del diagrama de la **Figura 3.3**, puede determinarse analíticamente la corriente que circulará a través del Memristor, es decir:

$$i_m(t) = \frac{V_o(t) - V_{ref}}{R_a + R_{MEM}} \tag{3.10}$$

A través de esta expresión y de acuerdo a la función de ventana de Joglekar, es posible obtener una versión de la ecuación de estado que permitirá obtener una expresión útil para el cálculo de la frecuencia. Por lo tanto, la ecuación para definir el estado del sistema puede expresarse como:

$$\frac{dx}{dt} = \frac{k(V_O(t) - V_{ref})(4x(1-x))}{R_a + R_{OFF} - x(R_{OFF} - R_{ON})} \tag{3.11}$$

Considerando esta expresión, es posible obtener los tiempos en los que el oscilador mostrará una salida en estado alto, T_H, y en estado bajo, T_L. Así, integrando desde x_{Mp} a x_{Mn} y en función del tiempo T_H, considerando $V_O(t) = V_{OH}$, se tiene:

$$\frac{1}{4k(V_{OH} - V_{ref})}\int_{x_{Mn}}^{x_{Mp}} \left(\frac{R_{OFF} + R_a}{x} + \frac{R_{on} + R_a}{1-x}\right) dx = \int_0^{T_H} dt \tag{3.12}$$

De esta expresión x_{Mp} y x_{Mn} corresponden al valor de la frontera cuando el valor de resistencia es "R_{mn}" y "R_{mp}", respectivamente. Estas variables se determinan a través de la ecuación de la resistencia del Memristor **(3.9)**, despejando la variable de estado, resultando en:

$$x_{mp} = \frac{R_{mp} - R_{off}}{R_{off} - R_{on}} \qquad x_{mn} = \frac{R_{mn} - R_{off}}{R_{off} - R_{on}} \tag{3.13}$$

Resolviendo la integral **(3.12)** y realizando las sustituciones correspondientes, es posible obtener una ecuación a través de la cual pueda determinarse el valor del semiciclo positivo de la señal T_H, es decir:

$$T_H = \frac{\left(R_a + R_{off}\right)\ln\left(\frac{R_{Mn} - R_{off}}{R_{Mp} - R_{off}}\right) + (R_{on} + R_a)\ln\left(\frac{R_{on} - R_{Mp}}{R_{on} - R_{Mn}}\right)}{4k(V_{OH} - V_{ref})} \tag{3.14}$$

Asimismo, para conocer el semiciclo negativo de la señal, se integra la ecuación de estado **(3.11)** desde x_{Mn} a x_{Mp} en función de T_L y con $V_O(t) = V_{OL}$, resultando en:

$$T_L = \frac{(R_a + R_{off}) \ln\left(\frac{R_{Mp} - R_{off}}{R_{Mn} - R_{off}}\right) + (R_{on} + R_a) \ln\left(\frac{R_{on} - R_{Mn}}{R_{on} - R_{Mp}}\right)}{4k(V_{OL} - V_{ref})} \quad \textbf{(3.15)}$$

Tomando en cuenta la relación del tiempo en que la señal se encuentra activa (T_H) y su periodo (T), se puede obtener la expresión para el ciclo de trabajo del oscilador como:

$$D = \frac{T_H}{T} \quad \textbf{(3.16)}$$

De esta expresión, el periodo "T" se determina a partir de la suma del tiempo en alto y bajo de la señal, es decir: $T = T_H + T_L$. De esta forma, a través del recíproco del período de la señal, puede determinarse la frecuencia de trabajo del oscilador como:

$$f = \frac{1}{T_H + T_L} \quad \textbf{(3.17)}$$

Con base en las expresiones descritas en **(3.14)** y **(3.15)**, pude notarse que la frecuencia de oscilación del circuito puede controlarse mediante diferentes mecanismos. Por ejemplo, puede depender del parámetro k del Memristor **(3.7)**, también de los valores de polarización en los comparadores. En general, la manera más sencilla de controlar la frecuencia de oscilación, es mediante la resistencia R_a, colocada en serie con el Memristor, puesto que es un parámetro accesible y sencillo de adecuar.

Además, considerando el voltaje de referencia, éste podría también modular la frecuencia, sin embargo, en este caso, fue de utilidad principalmente para ajustar los voltajes de salida en la compuerta, puesto que el simulador no permite valores negativos, tales como los utilizados en [2]. Por lo tanto, para realizar la simulación del circuito, se ajustan tanto los parámetros de voltaje en la compuerta como los de polarización en los comparadores, considerando voltaje de referencia $V_{ref}=1V$.

Considerando este valor y sustituyendo en **(3.3)**, la variación de resistencia esperada en el Memristor se presenta a través de un intervalo en "R_M" mínima equivalente a $R_{mn} = 3k\Omega$ y "R_M" máxima igual a $R_{Mp} = 9K\Omega$; cumpliendo la condición descrita en **(3.4)**.

Además, para la simulación de este circuito se utilizó la herramienta Simulink® de Matlab®, tomando en cuenta el modelo de Memristor desarrollado con la misma herramienta con R_{on}, R_{off}, D y μ_v a 100Ω, $38k\Omega$, $10nm$ y $10^{-10}cm^2s^{-1}v^{-1}$ respectivamente; los parámetros del circuito fueron extraídos y ajustados de [2], con V_{OL}, V_{OH}, V_n y V_p a $0V$, $2V$, $0.5V$ y $1.75V$ respectivamente, además de $R_a=3k\Omega$.

En la **Figura 3.4** se muestran los resultados de la simulación. Puede notarse el intervalo de oscilación para la memristencia y la variación del voltaje en el Memristor. Además, se cumple la condición descrita en **(3.2)**, es decir, la conmutación tiene lugar cuando el voltaje de entrada V_i es equivalente a V_p o V_n. Finalmente, se muestra también la señal esperada a la salida; asimismo, cabe mencionar que el ciclo de utilidad del circuito es del 50%, es decir, el tiempo en que la señal está en el semiciclo positivo es el mismo en que se encuentra en el semiciclo negativo.

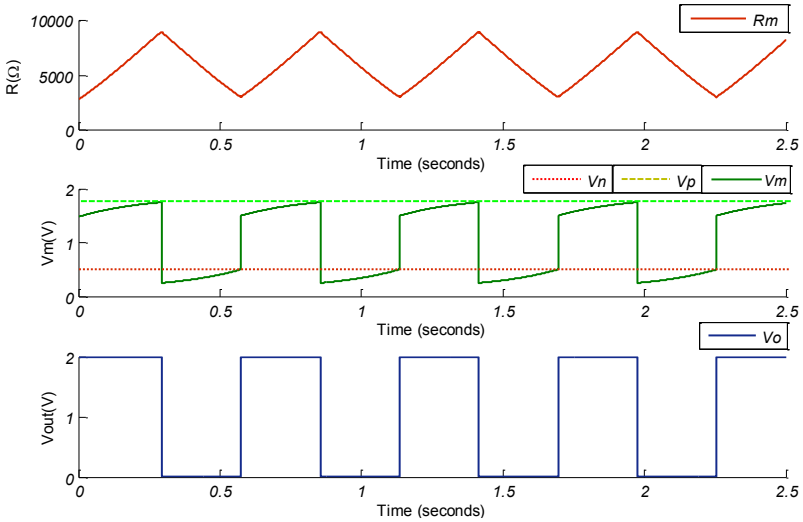


Figura 3.4 5 *Resultados de la simulación para la graficas de resistencia y voltaje: R_m, V_i y V_o; considerando, $R_a = 3k\Omega$, $V_{oh} = 2V$, $V_{ol} = 0V$, $V_p = 1.75$ y $V_n = 0.5V$.*

Como se mencionó, la frecuencia del circuito puede ser controlada a través de la resistencia R_a, por lo tanto, se efectuó un barrido desde 500Ω hasta $12k\Omega$ en función de comprobar el análisis presentado. En la **Figura 3.6** se muestra una gráfica del comportamiento de la frecuencia en función de la variación en la resistencia R_a.

De esta gráfica, puede observarse que el comportamiento de R_a es inversamente proporcional a la frecuencia de trabajo del circuito, es decir, conforme R_a aumenta, la frecuencia del circuito tiende a cero, por el contrario, cuando R_a tiende a disminuir la frecuencia aumenta.

Además, la variación en la frecuencia de oscilación obtenida va desde $f=11.09Hz$ a $f=0.14Hz$ con R_a mínima y máxima respectivamente. Por lo tanto, y de acuerdo con el valor de frecuencia de oscilación, es de notarse que el resultado es el de un oscilador de baja frecuencia.

Es importante mencionar también que de acuerdo con la condición en **(3.5)**, no es posible realizar un análisis con valores más pequeños de resistencia, por consiguiente, la frecuencia máxima de trabajo del oscilador es con $R_a = 500\Omega$.

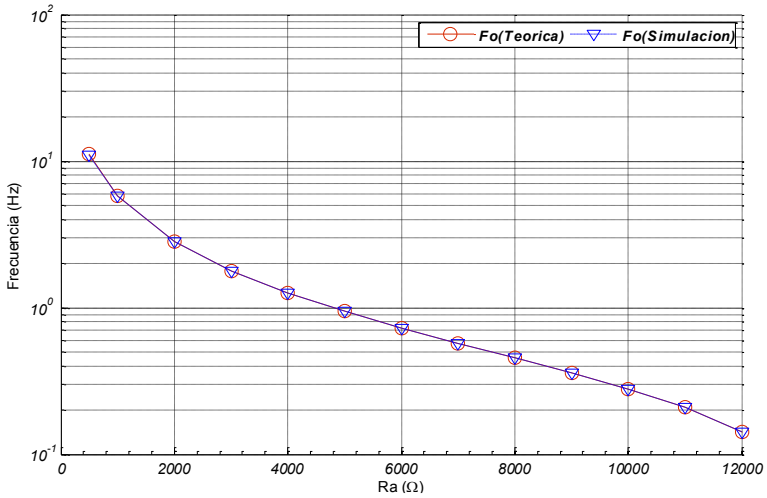


Figura 3.6 *Frecuencia de oscilación para diferentes valores en R_a.*

En la **Tabla 3.1** se muestra una comparación con los datos obtenidos de la simulación y los obtenidos teóricamente. Además, se muestra el porcentaje de error, mostrando un valor por debajo del 1%, por lo que puede mencionarse que la desviación de los datos obtenidos en la simulación con los esperados a través de las expresiones matemáticas es aceptable.

Ra(Ω)	F_O(Hz) (Teórica)	F_O(Hz) (Simulación)	Error (%) ((F_OT - F_OS / F_OT) *100)
500	11.1591	11.0497	0.9804
1000	5.7899	5.7636	0.4542
2000	2.8205	2.8129	0.2695
3000	1.7879	1.7841	0.2125
4000	1.2626	1.2602	0.1901
5000	0.9436	0.9425	0.1166
6000	0.7286	0.7280	0.0823
7000	0.5730	0.5727	0.0524
8000	0.4543	0.4542	0.0220
9000	0.3595	0.3593	0.0556
10000	0.2805	0.2804	0.0357
11000	0.2108	0.2107	0.0474
12000	0.1412	0.1412	0.0000

Tabla 3.1 *Tabla comparativa de los resultados en la simulación y el porcentaje de error respecto de la teoría.*

Finalmente, de acuerdo con el análisis sin considerar el voltaje de referencia, es decir, para el caso donde $V_{ref} = 0$, los tiempos en los que la señal se encuentra en el semiciclo positivo y negativo quedan expresados de la siguiente forma:

$$T_H = \frac{(R_a + R_{off})\ln\left(\frac{R_{mn} - R_{off}}{R_{mp} - R_{off}}\right) + (R_{on} + R_a)\ln\left(\frac{R_{on} - R_{mp}}{R_{on} - R_{mn}}\right)}{4kV_{oh}}$$

$$T_L = \frac{(R_a + R_{off})\ln\left(\frac{R_{mp} - R_{off}}{R_{mn} - R_{off}}\right) + (R_{on} + R_a)\ln\left(\frac{R_{on} - R_{mn}}{R_{on} - R_{mp}}\right)}{4kV_{ol}} \quad \textbf{(3.18)}$$

El ciclo de utilidad de la señal y la frecuencia se calculan de la misma forma que en **(3.16)** y **(3.17)**.

Respecto a la simulación con voltajes negativos, ésta puede efectuarse mediante el módulo mostrado en la **Figura 3.7**. De manera general, a través de dos fuentes de voltaje en paralelo controladas mediante un interruptor, es posible elegir entre voltaje positivo o negativo a la salida del circuito de control en función de la compuerta AND.

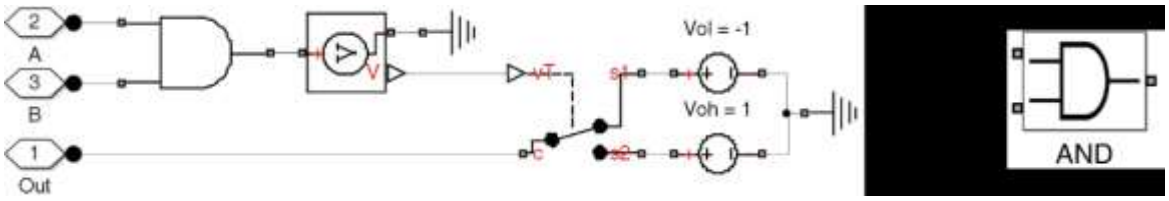


Figura 3.7 *Diagrama electrónico para la simulación de valores negativos.*

Para realizar la simulación se utilizan los parámetros del circuito sin tomar en cuenta V_{ref}, por lo que V_{OL}, V_{OH}, V_n, V_p, y R_a se definen tal como en la referencia original [2]. Por lo tanto, y puesto que el ajuste es proporcional, se obtuvo la misma variación en la resistencia del Memristor, $R_{mn} = 3k\Omega$ y $R_{mp} = 9k$. Además, considerando $Ra=3k\Omega$, la frecuencia de oscilación es equivalente a $f=1.788Hz$. Los resultados de la simulación se muestran en la **Figura 3.8**.

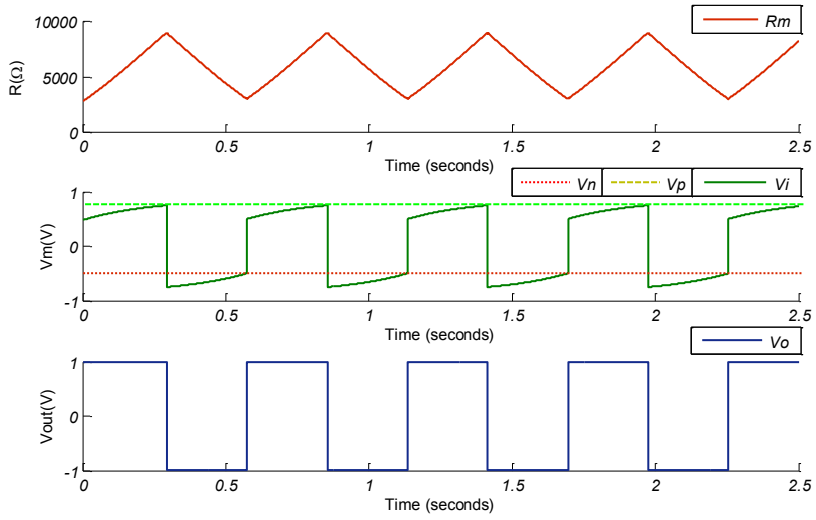


Figura 3.8 *Resultados de la simulación para la graficas de resistencia y voltaje: R_m, V_i y V_o; considerando, $R_a = 3k\Omega$, $V_{oh} = 1V$, $V_{ol} = -1V$, $V_p = 0.75$ y $V_n = -0.5V$.*

3.3.Oscilador Memristor-Resistor sin reactancia.

Un arreglo diferente para el desarrollo del oscilador, es intercambiando la posición del Memristor [7], como lo muestra la **Figura 3.9a**. En este caso, los dos comparadores del circuito de control ($F(V_i)$) definen las entradas de una compuerta tipo NAND, puesto que, es necesaria una condición contraria a la salida de la compuerta de acuerdo a las condiciones presentadas tanto por los comparadores como por los parámetros del circuito.

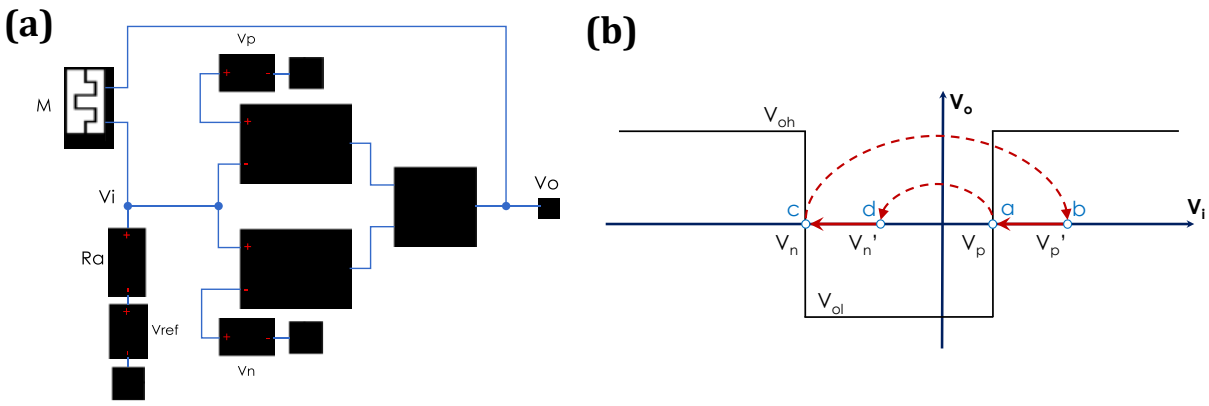


Figura 3.9 *Oscilador M-R sin reactancia.*

El comportamiento del módulo de control es similar al del caso anterior, sin embargo, la operación del circuito se traza a partir del punto 'd' de la **Figura 3.9b** y asumiendo la salida en estado bajo "V_{OL}", por consecuencia, la resistencia del Memristor disminuirá y por tanto el voltaje en el nodo V_i mientras: $V_n \leq V_i \leq V_n$'. En el momento que $V_i < V_n$, la salida de la compuerta conmutará a estado alto manteniendo ese estado mientras: $V_p \leq V_i \leq V_p$'. Finalmente, la salida del circuito regresará al comienzo de la operación ($V_O = V_{OL}$) en el momento que: $V_i < V_p$.

La condición de salida del módulo de control ($F(V_i)$), se define respecto a la lógica de la compuerta NAND en función de los comparadores, es decir:

$$F(V_i) = \begin{matrix} \boldsymbol{V_{OL}} \rightarrow V_n i \leq V_i \leq V_n' \\ de\ otra\ manera \rightarrow \boldsymbol{V_{OH}} \end{matrix} \qquad \textbf{(3.19)}$$

El análisis para obtener las expresiones generales de oscilación y la frecuencia de trabajo, se efectúa de forma similar al método descrito en el análisis del oscilador *Resistor-Memristor* de la sección anterior. De esta manera, de acuerdo a la posición del Memristor el voltaje en el nodo V_i puede describirse como:

$$V_i(t) = (V_o(t) - V_{ref}) \frac{R_a}{R_a + R_m} \qquad \textbf{(3.20)}$$

En el momento que el voltaje en V_i es equivalente a V_p o V_n, el valor de memristencia está definido por R_{Mp} y R_{Mn}, respectivamente, descritos con base en la expresión del divisor de voltaje, es decir:

$$R_{mp} = R_a \frac{V_{OH} - V_p - V_{ref}}{V_p}; R_{mn} = R_a \frac{V_{OL-} V_n - V_{ref}}{V_n} \quad \textbf{(3.21)}$$

Asimismo, es posible determinar la condición necesaria para que ocurra la oscilación del circuito en términos de R_a. Sustituyendo **(3.21)** en **(3.4)** y **(3.5)**, se obtiene:

$$R_{on}\left(\frac{V_p}{V_{OH} - V_p - V_{ref}}\right) < R_a < R_{off}\left(\frac{V_n}{V_{OL-} V_n - V_{ref}}\right) \quad \textbf{(3.22)}$$

A partir de este punto, respecto a la determinación de la frecuencia del circuito, ésta se efectúa a través de las mismas consideraciones que en la configuración anterior, es decir, tomando en cuenta el modelo de Hp® y la función de ventana de Joglekar.

Además, es importante enfatizar el hecho de que la corriente que circula a través del Memristor se define de la misma forma que en **(3.10)**, puesto que la configuración del circuito está definida en función de un arreglo en serie. Por lo tanto, la ecuación de estado puede expresarse de forma equivalente a la configuración *R-M*, expresión **(3.11)**.

De esta forma, el tiempo en que el oscilador se encuentra en estado alto, "T_H", puede obtenerse como:

$$T_H = \frac{\left(R_a + R_{off}\right) \ln\left(\frac{R_{mn} - R_{off}}{R_{mp} - R_{off}}\right) + \left(R_{on} + R_a\right) \ln\left(\frac{R_{on} - R_{mp}}{R_{on} - R_{mn}}\right)}{4k(V_{oh} - V_{ref})} \quad \textbf{(3.23)}$$

Asimismo, el semiciclo negativo de la señal queda expresado como:

$$T_L = \frac{\left(R_a + R_{off}\right) \ln\left(\frac{R_{mp} - R_{off}}{R_{mn} - R_{off}}\right) + \left(R_{on} + R_a\right) \ln\left(\frac{R_{on} - R_{mn}}{R_{on} - R_{mp}}\right)}{4k(V_{ol} - V_{ref})} \quad \textbf{(3.24)}$$

Finalmente, considerando estas expresiones, es posible el cálculo de la frecuencia de oscilación de manera similar que en el arreglo anterior, es decir, considerando la expresión **(3.17)**.

Respecto a la simulación de esta configuración, se determinan los parámetros de voltaje respecto a [2] y en función de $V_{ref}=1V$. De esta manera, V_{ol}, V_{oh}, V_n y V_p se definen considerando $0V$, $2V$, $0.25V$ y $1.5V$, respectivamente. Asimismo, con base en estos parámetros y de acuerdo a la condición **(3.21)**, se esperaría un intervalo de oscilación en la resistencia del Memristor entre: $R_{Mn} = 1k\Omega$ y $R_{Mp} = 3k\Omega$.

Los resultados de la simulación se presentan en la **Figura 3.10**, considerando R_a=*3kΩ*. En el diagrama, se muestra la señal cuadrada esperada a la salida, en función de la variación en la resistencia del Memristor (R_M), puesto que adecua el voltaje en el nodo V_i y por tanto la frecuencia en la señal de salida.

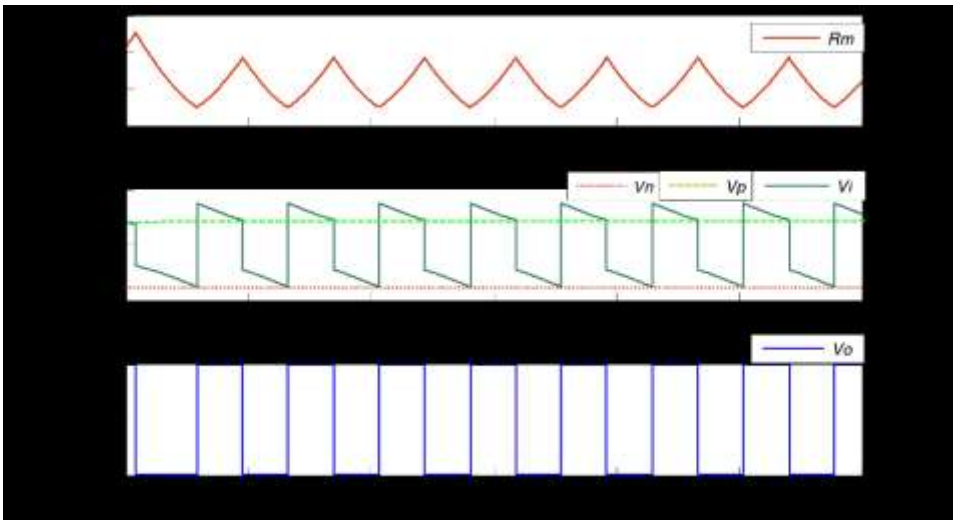


Figura 3.10 *Resultados de simulación para R_m, V_i, y V_o, con R_a = 3kΩ, V_{oh} = 2V, V_{ol} = 0V, V_p = 1.5 y V_n = 0.25V.*

Además, se llevó a cabo un barrido en el valor de la resistencia R_a de acuerdo con la condición descrita en **(3.22)**. En la **Figura 3.11**, se muestra la gráfica del comportamiento de la frecuencia respecto a la variación en la resistencia R_a, donde es de notarse un pequeño aumento en la frecuencia de oscilación, es decir, el valor de frecuencia con R_{min} es equivalente a f *=14.12Hz,* distinto de la configuración *R-M* con f*=11.09Hz*.

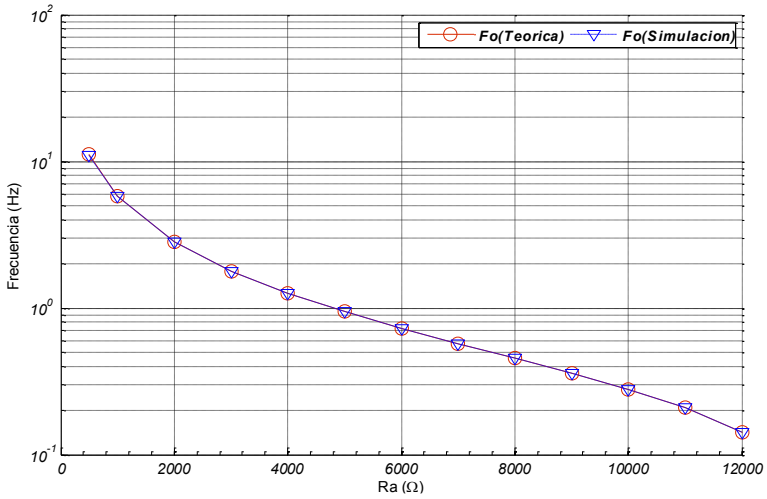


Figura 3.11 *Frecuencia de oscilación para diferentes valores en R_a.*

En favor de comprobar el análisis teórico respecto a los datos obtenidos de la simulación, es de utilidad la **Tabla 3.2**, puesto que describe el porcentaje de error que se presentó. De acuerdo con los datos presentados en la tabla, puede describirse una pérdida aceptable y por consecuencia, un ajuste correcto de acuerdo con lo descrito en la teoría del oscilador.

Ra	F_O(Hz) (Teórico)	F_O(Hz) (Simulación)	Error (%) ((F_OT - F_OS / F_OT) *100)
500	14.1266	13.9860	0.9953
1000	9.1686	9.0909	0.8475
2000	5.0058	4.9875	0.3656
3000	3.3866	3.3783	0.2451
4000	2.5344	2.5252	0.3630
5000	2.0094	2.0080	0.0697
6000	1.6537	1.6515	0.1330
7000	1.3967	1.3947	0.1432
8000	1.2024	1.2004	0.1663
9000	1.0503	1.0493	0.0952
10000	0.9279	0.9267	0.1293
11000	0.8273	0.8267	0.0725
12000	0.7432	0.7429	0.0404

Tabla 3.2 *Tabla comparativa con los resultados obtenidos en la simulación y el porcentaje de error respecto a la teoría.*

3.4. Conclusiones.

A través del desarrollo del presente capítulo, fue posible demostrar el funcionamiento del modelo de Memristor desarrollado en Simscape® y Simulink®, siendo parte de un sistema más complejo; concretamente, en un circuito oscilador. De manera específica, se efectuó el análisis de las expresiones que permiten determinar la frecuencia de oscilación del circuito en función del Modelo de Hp®.

La particularidad principal que presenta el uso del Memristor en el oscilador, es precisamente, el área de diseño reducida que permite, puesto que, para la obtención de un valor de baja frecuencia como el descrito en las simulaciones, sería necesario el uso de elementos de alto valor capacitivo o inductivo; penalizando el área de integración.

También, se debe hacer énfasis en que la configuración *Memristor-Resistor* osciló un poco más rápido que el arreglo *Resistor-Memristor*. Esto se atribuiría, a que el intervalo en la variación de la resistencia del Memristor es más pequeño y por tanto la conmutación es más rápida, generando valores de frecuencia más grandes. Además, existe la posibilidad de generar otras configuraciones tales como: usar un arreglo de tipo *Memristor-Memristor* o realizar el análisis del comportamiento si se cambia la polaridad del Memristor.

En general, se obtuvieron resultados de simulación que se ajustan de manera correcta con la respuesta teórica del oscilador. Es decir, el modelo a bloques del Memristor desarrollado en Simulink® es capaz de presentar una aproximación correcta del elemento "*físico*", permitiendo su utilización en otras aplicaciones, en favor de conocer la respuesta y ventajas que presentaría.

Referencias

[1] R. Tetzlaff, Memristor and Memristive Systems, New York: Springer, 2014.

[2] M. A. Zidan, H. Orman, A. G. Radwan y K. N. Salama, «Memristor-Based Reactance-Less Oscillator», *Electronic Letters,* vol. 47, pp. 1220-1221, 2011.

[3] A. Talukdara, A. Radwan y K. Salamaa, «Generalized model for memristor-based Wien family oscillators», *Microelectronics Journal,* vol. 42, nº 9, pp. 1032-1038, 2011.

[4] M. E. Fouda, A. G. Mosad, K. N. Salama y A. G. Radwan, «Memristor-based relaxation oscillators using digital gates», *Computer Engineering & Systems(ICCES),* pp. 98-102, 2012.

[5] A. I. Hussein y M. E. Fouda, «A simple MOS realization of current controlled memristor emulator», *2013 25th International Conference on Microelectronics (ICM),* pp. 1-4, 2013.

[6] M. A. Zidan, H. Omran, G. Radwan y K. N. Salama.United States Patente US 2015/0008988 A1, 2015.

[7] R. L. Boylestad y L. Nashelsky, «Capítulo 13: Circuitos integrados anaógicos-digitales», de *Electronica: Teoría de Circuitos y Dispositivos Electrónicos*, México, Pearson Educación, 2009, pp. 712-718.

[8] M. E. Fouda y A. G. Radwan, «Memristor-based voltage-controlled relaxation», *International Journal of Circuit Theory and Applications,* vol. 42, pp. 1092-1102, 2014.

[9] D. Strukov, G. Snider, D. Stewart y R. Williams, «The missing memristor found», *Nature,* vol. 453, pp. 80-83, 2008.

[10] Y. N. Joglekar, «The elusive memristor: properties of basic electrical circuits», *European Journal of Physics,* vol. 30, pp. 661-675, 2009.

[11] A. G. Radwan y M. E. Fouda, On the Mathematical Modeling of Memristor, Memcapacitor, and Meminductor, Springer, 2015.

Capítulo 4 Implementación de un modelo de Memristor mediante RNA

En el área de la Ingeniería, el uso de redes neuronales artificiales (RNA) en favor del desarrollo tecnológico, se ha convertido en una alternativa de bajo costo y alto desempeño frente a soluciones tradicionales, puesto que ha permitido, a través de sistemas de control inteligente la aproximación de funciones complejas.

Además, es importante enfatizar la importancia del proceso de aprendizaje que presenta un sistema RNA, es decir, una red neuronal mediante reglas que desarrolla en función del entrenamiento, es capaz de tomar decisiones, no al grado en el que lo hace una persona, pero sí en un punto donde puede efectuar tareas muy complejas, tales como el procesamiento de imágenes, reconocimiento de patrones, filtros e incluso predicción. Esto, en tiempos menores en los que lo haría un sistema más complejo, hablando en términos de la cantidad de elementos electrónicos y la dificultad para efectuar la programación e implementación.

4.1Introducción.

A través de este capítulo se describe la metodología utilizada para el desarrollo de una red neural artificial, capaz de aproximar la respuesta de un sistema no lineal, tal como el Memristor. De esta manera, considerando que el dispositivo se encuentra en un punto de desarrollo, se posibilita el contar con un prototipo que aproxime su respuesta, permitiendo conocer la solución que presentaría siendo parte de sistemas más complejos.

De forma particular, la utilización de redes neuronales dinámicas es una forma ideal para aproximar la respuesta del Memristor, puesto que su comportamiento, se enfoca en la predicción de la respuesta de sistemas altamente no lineales a través de elementos de retroalimentación y retardos, es decir, a través de una serie de tiempo, donde los valores previos son de utilidad para calcular los posteriores. Específicamente, se considera la arquitectura de red NARX, puesto que es un tipo de red popular que permite tanto el análisis de series de tiempo, como el modelado de sistemas dinámicos.

En resumen, la construcción de la red se efectuará mediante Matlab® y será capaz de aproximar el perfil de corriente en función del perfil de voltaje, presentando una gráfica del comportamiento de histéresis; posteriormente para conocer la respuesta de acuerdo con la implementación en tecnología FPGA, se llevará a cabo el análisis del comportamiento de la red en Simulink®.

4.2Implementación de la RNA en Matlab®.

La implementación de una red neuronal artificial, conlleva una serie de pasos que permiten facilitar el proceso de diseño de la red[1]. De esta forma, considerando que las redes neuronales "*dinámicas"* pueden describirse en función de una metodología similar de implementación que las redes "*estáticas"*, puede considerarse el diagrama de la **Figura 4.1** para el proceso de desarrollo de la red.

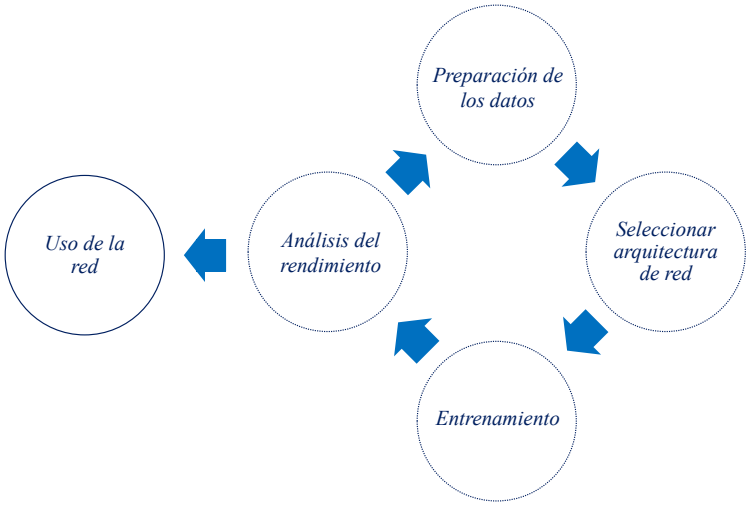


Figura 4.1 *Diagrama de flujo del proceso de implementación de la red neural.*

De manera general, la metodología descrita en el diagrama, es útil en la construcción de la arquitectura de red, sin embargo, es importante señalar que no existe una configuración ideal que aproxime correctamente la respuesta, consecuencia de que el entrenamiento es un proceso iterativo, enfocado en el proceso de: *"prueba y error"*.

4.2.1 Preparación de los datos.

El diseño de la red comienza con la preparación de los datos, es decir, en relación con una correcta aproximación de la respuesta deseada, deberá considerarse la recolección de una base de datos que contemple puntos suficientes. De esta forma, si la red conoce más puntos del perfil de entrenamiento, el proceso de aprendizaje mejora y por consecuencia, ésta sería capaz de generar una respuesta más cercana ante datos que no se presentaron en el entrenamiento.

Asimismo, el *pre-procesamiento* de los datos es un punto importante y considera una serie de factores útiles previo al comienzo del entrenamiento de la red [2]. Por ejemplo, si se efectúa la normalización de la base de datos, se facilita el cálculo de la respuesta y por consecuencia el trabajo de la red neuronal se vuelve más eficiente. De la misma forma, si se considera una base de datos grande, la respuesta debería ser más sencilla de ajustar, sin embargo, esto depende de la complejidad del sistema y se refleja, además, en el número de neuronas necesarias para presentar la respuesta.

4.2.1.1 Recolección.

En primer lugar, para efectuar el entrenamiento de la red neuronal artificial, es necesario tener claro el sistema que se desea aproximar, es decir, debe considerarse la base de datos que contiene el perfil de la respuesta buscada. Por lo tanto, es indispensable llevar a cabo la recolección de datos a través de modelos matemáticos o a partir de simulaciones experimentales.

En este caso, la base de datos se genera a través del modelo de Memristor de Simscape®; particularmente, la entrada que se le presentará al sistema, considera una señal de voltaje senoidal de *1V* de amplitud y *1Hz* en frecuencia, tal como se muestra **Figura 4.2**. De manera general, la base de datos se define en función de una matriz de [500x2], es decir, la primera columna, describe 500 puntos como la entrada de voltaje, y la segunda, 500 puntos para el perfil de corriente.

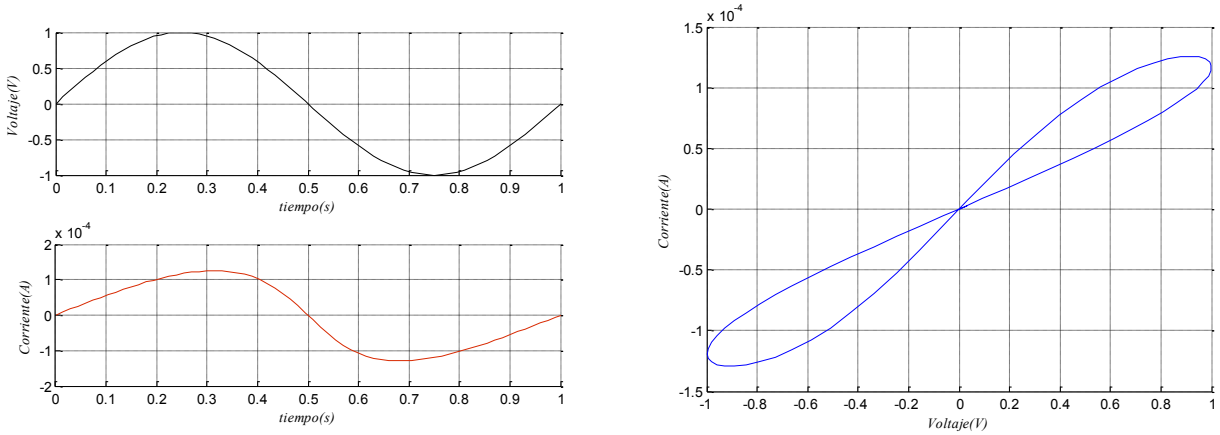


Figura 4.2 *Perfil de entrenamiento.*

4.2.1.2 Pre-procesamiento.

El entrenamiento es más eficiente si se considera en primer lugar, la normalización de la base de datos, puesto que evita que la función de transferencia se sature y por consecuencia, la red no sea capaz de presentar la salida del sistema. De forma general, la normalización se efectúa tanto en los vectores de entrada como en los vectores objetivo del conjunto de datos y se especifica el valor máximo y mínimo de la base de datos, presentando un intervalo definido, en este caso ±1.

Asimismo, considerando que la arquitectura de red NARX procesa la información dato por dato, es necesario presentar los datos de manera secuencial, en favor de que la red calcule la salida a través de una serie de tiempo en función de los eventos previos. Tanto la normalización como la conversión de la base de datos pueden efectuarse a través de la sintaxis mostrada en la **Figura 4.3**.

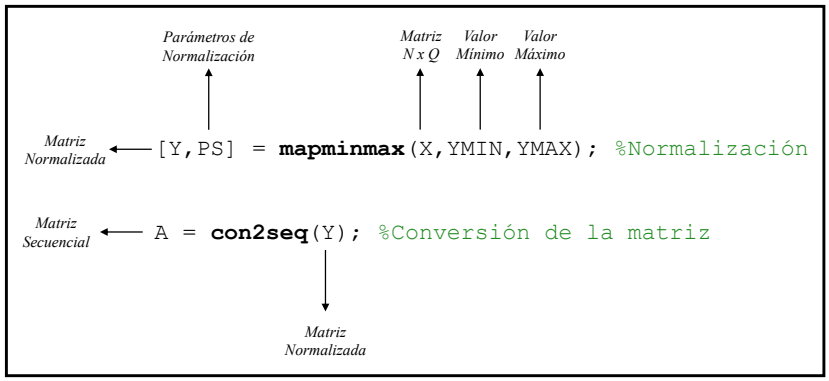


Figura 4.3 *Sintaxis para la normalización y conversión de la matriz de datos.*

Además, en redes neuronales dinámicas, el procesamiento de los datos implica también considerar el estado inicial en favor de que comience el proceso de predicción. Es decir, cuando se trabaja en lazo cerrado, es necesario ajustar los datos en función de las líneas de retardo, por consecuencia, la base de datos deberá desplazarse de acuerdo con la cantidad de retardos, las veces que sea necesario.

Matlab® contiene una función a través de la cual pueden adecuarse los datos. Esta función devuelve la base de datos con las condiciones necesarias de acuerdo con las líneas de retardo de la red, modificando de esta manera, tanto la secuencia de entrada como la de objetivo. La sintaxis de esta función se muestra en la **Figura 4.4**.

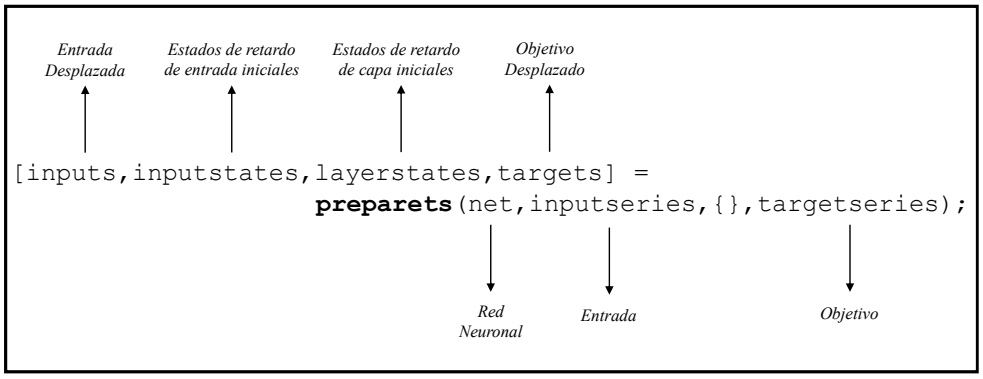


Figura 4.4 *Sintaxis de la función para la preparación de datos.*

Una vez presentada la base de datos con el formato adecuado, se debe determinar la forma en que se realizará el entrenamiento con el fin de mejorar la capacidad de la red para generalizar y asociar datos. Este proceso permite ajustar el comportamiento de la red ante situaciones ajenas a las que conoce, es decir, que su conducta es errática ante datos que no se tomaron en cuenta durante el proceso de entrenamiento. De forma general, la mejora de la generalización se puede realizar tanto a través del método de detención temprana como mediante el método de regularización.

Además, es importante mencionar que el uso de alguno de los dos métodos, depende de factores tales como la aplicación que se le dé a la red, el análisis de la respuesta, y también de la base de datos que se utilice. Para el desarrollo de la red neuronal que aproximaría la respuesta del Memristor fue posible desarrollar una configuración adecuada que originaría una buena generalización de la red en función de los dos métodos y a través del análisis y el estudio de las necesidades de la aplicación.

4.2.1.2.1 Método de detención temprana (Early Stopping).

Un método útil y comúnmente utilizado en la mejora de la generalización, es el nombrado método de detención temprana (*early stopping*). Este consiste en la división de los datos en tres subconjuntos: entrenamiento, validación y prueba.

El primer subconjunto, se refiere a los datos que deberán *pre-procesarse*, puesto que contienen gran parte del perfil que la red aproximará. Además, la red utiliza este subconjunto para calcular el gradiente, el cual, es un algoritmo de optimización, cuyo objetivo es encontrar el mínimo local de una función. A través de este protocolo de entrenamiento se realiza la aproximación de la respuesta en función de la actualización de los pesos y polarizaciones.

El segundo segmento, se refiere al conjunto de datos para la validación, en él se monitorea el error durante el proceso de entrenamiento y contribuye también a evitar el "*sobre-entrenamiento*", es decir, por cada iteración que realiza el algoritmo, el error de validación y de entrenamiento disminuye, por consecuencia, los datos se ajustan mejor. Sin embargo, si las iteraciones son demasiadas, como consecuencia de que el algoritmo trabajó por demasiado tiempo, puede surgir el sobre ajuste de los datos, causando que el error en los datos de validación aumente y, de acuerdo a un número determinado de iteraciones conforme el error aumentó, el proceso de entrenamiento se detiene, presentando los últimos valores de pesos y polarizaciones que obtuvo.

Por último, se encuentra el subconjunto de prueba, útil al final del entrenamiento; este grupo permite conocer el comportamiento global de la red y presenta una *medida* de la generalización. De esta forma, si el error sobre el conjunto de prueba es bajo, significa que la red será capaz de ajustar correctamente el comportamiento ante datos que no se presentaron durante la fase de entrenamiento.

De acuerdo con lo descrito, para efectuar una buena segmentación de los datos, se debe considerar una base suficientemente grande, es decir, es importante contar con puntos suficientes para el grupo de validación, puesto que a través de ellos se calcula el error y se detiene el entrenamiento. Comúnmente, se define el 70% para el conjunto de entrenamiento, 15% para validación y 15% para prueba.

Sin embargo, esto no corresponde a una configuración ideal, puesto que dependerá del análisis que se efectúe a través de las gráficas de regresión si es necesario realizar modificaciones en la configuración. Además, existen comandos adicionales que permiten presentar la base de datos de forma aleatoria, describir un arreglo en forma de bloques o incluso por índices, en favor de mejorar el cálculo que efectuará la red.

4.2.1.2.2 Regularización.

El método de regularización [1] está descrito con base en la modificación de la función de rendimiento de la red. Comúnmente, esta función se describe a través del error cuadrático medio (*mse*), es decir, la diferencia cuadrática promedio entre la salida que presenta la red y el objetivo deseado, y puede representarse mediante la siguiente expresión:

$$mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - \alpha_i)^2 \qquad \textbf{(4.1)}$$

De esta expresión, "*N*" se refiere al total de muestras, mientras que "t_i" se describe como el objetivo de la red y "α_i" al valor de salida de la red. De manera general, el método de regularización consiste en ajustar esta expresión considerando el error cuadrático a través de los pesos de la red, es decir:

$$msw = \frac{1}{N}\sum_{j=1}^{N} {w_j}^2 \qquad \textbf{(4.2)}$$

A través de estas expresiones **(4.1)** y **(4.2)**, es posible determinar una expresión útil en la generalización de la red, es decir, adicionando un factor que ajuste el cálculo de los errores, es posible generar valores más pequeños de pesos y polarización, obligando a la red a presentar una respuesta más "*suave*" en el proceso de búsqueda del mínimo local o global de la función. De esta forma, la expresión que describe el método de regularización se define como:

$$msereg = \gamma mse + (1 - \gamma)\, msw \qquad \textbf{(4.3)}$$

De esta expresión, la variable "γ" se refiere al factor de rendimiento y comúnmente se define con un valor de *0.5*, sin embargo, el problema con el método de regularización es precisamente la dificultad para determinar el valor óptimo, puesto que, si éste es demasiado grande, podría causar sobreajuste; asimismo, un valor pequeño podría no ajustar de manera correcta la respuesta.

Con base en la solución a este problema, puede considerarse la utilización de otro algoritmo para efectuar el entrenamiento [2], puesto que el algoritmo podría definir de forma automática el valor óptimo en el coeficiente de rendimiento "γ", sin embargo, la utilización de otro algoritmo de entrenamiento dificulta la implementación.

4.2.2 Arquitectura de red.

La arquitectura de red ideal para efectuar el proceso de predicción de la respuesta, enfoca su funcionamiento en líneas de retardo, puesto que permiten el almacenamiento de los valores previos, en función de la determinación de los valores de salida del sistema. Este tipo de redes se conoce comúnmente como redes neuronales enfocadas en retrasos de tiempo (*Focused Time-Delay Neural Network, FTDNN)* y son una clase general de redes dinámicas, descritas con base en una red de tipo *feedforward* o de cálculo hacia adelante, donde las líneas de retardo se presentan solamente en la capa de entrada de la red, tal como se muestra en la **Figura 4.5.**

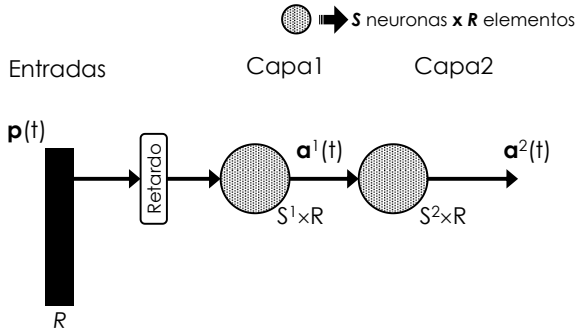


Figura 4.5 *Arquitectura de red tipo FTDNN de dos capas.*

Con en base en el principio de funcionamiento de este tipo de redes se describe la arquitectura de red NARX, utilizada para la aproximación de la respuesta del Memristor. La predicción del valor de salida de la red se efectúa de acuerdo con los valores previos definidos tanto en la entrada como en la salida y almacenados en las líneas de retardo (**Figura 4.6)**.

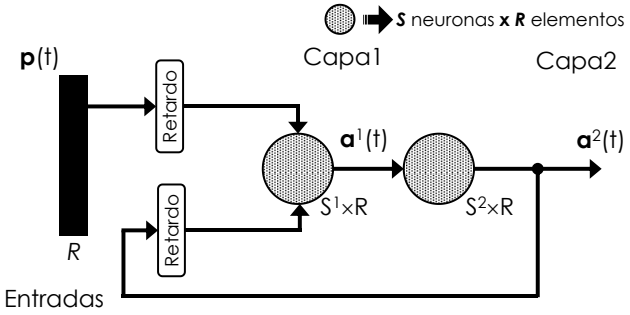


Figura 4.6 *Arquitectura de red tipo NARX.*

Es importante mencionar que en el proceso de entrenamiento puede no considerarse el lazo de retroalimentación de la arquitectura, puesto que permitirá llevar a cabo un entrenamiento común a través del algoritmo de *"backpropagation"* o *retro-propagación*, es decir, considerando una red común multicapa y facilitando de esta manera, la implementación y el ajuste de los datos. De otra forma, sería necesario la utilización de un algoritmo *"dinámico de retro-propagación"* [3].

4.2.2.1 Configuración de la arquitectura de red.

Una vez definida la arquitectura que se va a utilizar, es necesario configurar los parámetros de la red neuronal, por ejemplo, definir tanto el número de capas como de neuronas. Además, considerando la arquitectura, es importante determinar la cantidad de bloques de retardo.

Asimismo, de acuerdo con la arquitectura de este tipo de red, es importante considerar el número de entradas de acuerdo a la cantidad de retardos. En este caso, se considera una entrada externa de los datos a la red y otra respecto al lazo de retroalimentación de la salida a la entrada, cada una, con 2 bloques de retardo.

De manera general, la determinación de los parámetros de la red dependerá tanto del perfil de entrada que se le presente como de la complejidad en general del sistema. La sintaxis de la función que permite la creación y configuración de la red se muestra en **Figura 4.7**.

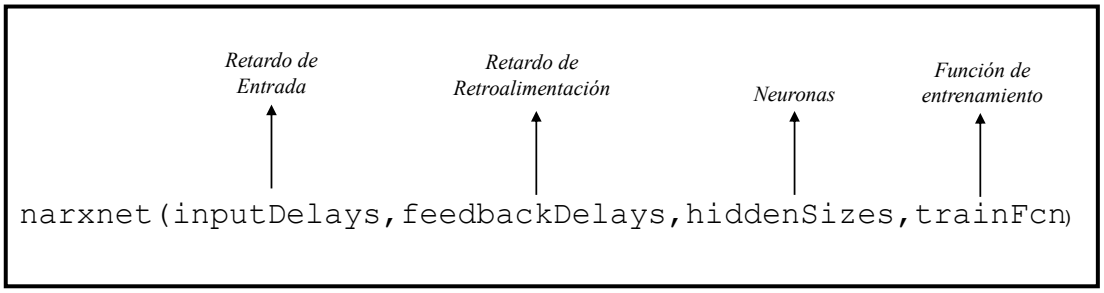


Figura 4.7 *Sintaxis para la creación de la red.*

La arquitectura de red propuesta para el desarrollo de este trabajo, contempla una capa oculta y una capa de salida, conformadas por tres y una neurona respectivamente. Asimismo, se contemplan las 2 líneas de retardo en cada entrada. De esta forma, y de acuerdo con la sintaxis de la **Figura 4.7**, el diagrama que presenta la función se muestra en la **Figura 4.8**.

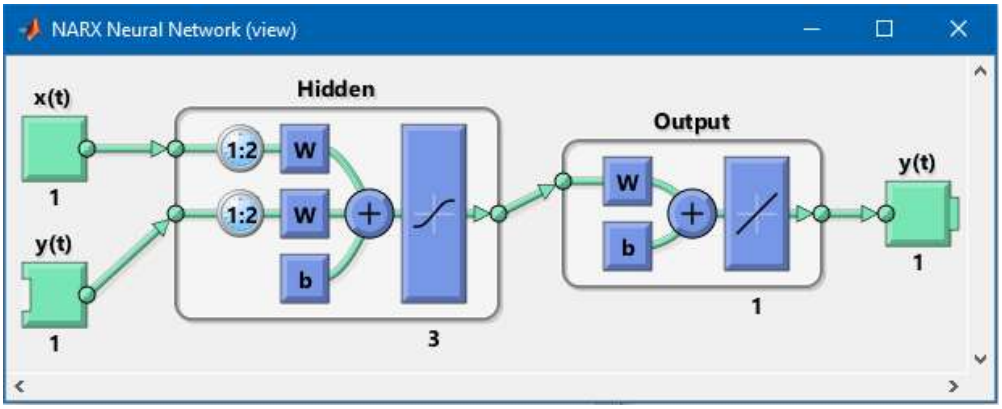


Figura 4.8 *Configuración red neuronal tipo NARX.*

4.2.3 Entrenamiento de la red.

Una vez que se configuró la base de datos y se determinó la arquitectura de red, puede llevarse a cabo el entrenamiento. Este proceso considera previamente la inicialización de los pesos y polarizaciones, además de la determinación tanto del algoritmo de entrenamiento como del criterio de detención. También, es importante considerar la función que definirá el índice de rendimiento de la red, puesto que presentará una medida del ajuste de la respuesta.

En resumen, puede mencionarse que el proceso de entrenamiento es repetitivo y requiere efectuar varios entrenamientos para obtener resultados correctos. Además, es necesario intentar diferentes configuraciones, por ejemplo, utilizar un algoritmo de entrenamiento diferente, asimismo, determinar qué criterio de detención presenta mejores resultados y particularmente, probar con el número de neuronas y retardos.

4.2.3.1 Inicialización de los pesos.

Previo al comienzo del entrenamiento, comúnmente se realiza la inicialización de los pesos, puesto que, a través de estos valores, se determina la rapidez de aprendizaje de las neuronas, traducido en la capacidad de la red para aproximar la respuesta deseada. Típicamente los valores iniciales se definen de acuerdo con valores muy pequeños, distribuidos entre ±0.5 y, hablando en términos de una base normalizada, entre ±1.

Para el desarrollo de este trabajo, la red se describe de acuerdo con pesos aleatorios, es decir, cada entrenamiento presenta diferentes resultados de peso, en favor de determinar los valores que presentan mejores resultados. Sin embargo, es importante mencionar que, en caso de ser necesario, Matlab® presenta una función que permite comenzar el entrenamiento con los últimos pesos que obtuvo, es decir, considerando estos valores como una condición inicial.

4.2.3.2 Algoritmo de entrenamiento.

Existen diferentes algoritmos de optimización útiles en el entrenamiento de la red descritos con base en métodos numéricos. Principalmente dos son los métodos de optimización más utilizados: el primero se define de acuerdo con el cálculo del descenso del gradiente y, el segundo con base en el cálculo de la matriz Jacobiana de los errores respecto a los pesos.

Ambos métodos utilizan el algoritmo de *retro-propagación* ("*Backpropagation*"). Este algoritmo, propaga capa a capa una excitación hasta generar un cálculo final en la neurona de salida; este resultado se compara con el objetivo de la red y determina el error, inmediatamente el resultado se retransmite hacia atrás, es decir, en las neuronas de las capas anteriores.

De esta manera, el algoritmo calcula el error de cada neurona en las diferentes capas y genera el error cuadrático medio con el objetivo de conocer si es necesario reajustar los pesos.

Además, existen variantes de este algoritmo, por ejemplo, el algoritmo Levenberg Marquardt o el algoritmo de regularización Bayasiana. Este tipo de algoritmos permiten acelerar la convergencia de la red a través de métodos que se describen con base en la *conjugación del gradiente* o de acuerdo con aproximaciones que permitan determinar mínimos o máximos locales de una función mediante métodos numéricos de *Quasi-Newton* (a través del cálculo de la matriz *Jacobiana* o *Hessiana*). Ambos métodos se describen de acuerdo al proceso que efectúa el método de *retro-propagación,* es decir, las derivadas son procesadas desde la última capa hacia la primera.

Para el desarrollo de este trabajo, el entrenamiento se efectúa a través del algoritmo *Levenberg Marquardt* [4], descrito de acuerdo con el cálculo de la matriz *Jacobiana*, puesto que permite conocer el gradiente de la función. Este método es una de las técnicas de optimización más utilizadas en el entrenamiento de redes neuronales y se define respecto a la técnica de *retro-propagación*, además, es considerado más eficiente de acuerdo con redes neuronales donde el tamaño de ésta no es demasiado grande, es decir, no involucra demasiados cálculos y por consecuencia, ocupa menos requerimientos en memoria y almacenamiento.

4.2.3.3 Criterios de detención.

Durante el proceso de entrenamiento, idealmente lo que se pretende es que el error de entrenamiento converja en cero. Sin embargo, en redes neuronales esto no sucede, por consecuencia, existen diferentes técnicas que permiten detener el proceso cuando el ajuste de la respuesta es el más cercano. Por ejemplo, el entrenamiento podría detenerse si el error alcanza un valor límite específico, sin embargo, es poco probable determinar de manera eficaz cuál es el nivel de error más óptimo.

De esta manera, uno de los criterios que puede utilizarse para detener el entrenamiento es a través del error en el conjunto de datos para la validación, es decir, determinando un número de iteraciones que detengan el proceso si el error en el grupo de datos se mantuvo incrementando. Este método es un estándar utilizado regularmente para efectuar la detención del entrenamiento, puesto que permite evitar el sobre ajuste de los datos.

También, puede detenerse el entrenamiento si el índice de rendimiento se redujo por cada iteración efectuada. El índice de rendimiento se refiere a la suma de los errores al cuadrado y genera una idea de la convergencia de la red.

Sin embargo, el problema que presenta este criterio, es que la detención del entrenamiento puede darse muy rápido, puesto que el error pudo mantenerse casi constante en un número determinado de iteraciones, consecuentemente, el ajuste de la red podría ser errático.

Asimismo, considerando que el gradiente debería ser cero en el mínimo de la función de rendimiento, el entrenamiento podría detenerse cuando éste se acerque al valor mínimo o más próximo.

Finalmente, otro criterio comúnmente utilizado en la detención del entrenamiento, es a través del número de iteraciones, es decir, definir un número limitado de repeticiones para que el proceso trabaje. El problema con este método, es precisamente determinar el número de iteraciones, sin embargo, normalmente se define un valor alto y, si los pesos no convergen al terminar el proceso, puede reiniciarse el entrenamiento en función, por ejemplo, de los pesos finales obtenidos.

Además de los métodos descritos, existen otras técnicas para detener el entrenamiento. De forma general, en la **Tabla 4.1** se muestran las funciones que Matlab® presenta para la detención del entrenamiento.

Parámetro	Descripción
min_grad	*Magnitud mínima del gradiente.*
max_fail	*El número máximo de iteraciones en el segmento de validación incremento.*
time	*Intervalo de tiempo máximo para el entrenamiento.*
goal	*Valor mínimo en la función de rendimiento.*
epochs	*Número máximo de iteraciones.*

Tabla 4.1 *Criterios de detención del entrenamiento de la red en Matlab®.*

Es importante mencionar que la elección del criterio de detención, dependerá principalmente del análisis que se efectúe después del entrenamiento, es decir, a través de las gráficas de rendimiento, regresión, estado del entrenamiento, etc., puesto que permitirá tener una idea más clara de la convergencia de la red y si aproximó correctamente la respuesta.

En este caso, para la aproximación de la respuesta del Memristor, el criterio de detención que mejor resultado presentó en el proceso de entrenamiento, fue mediante el número de iteraciones, particularmente, utilizando 30 iteraciones (**Figura 4.9**).

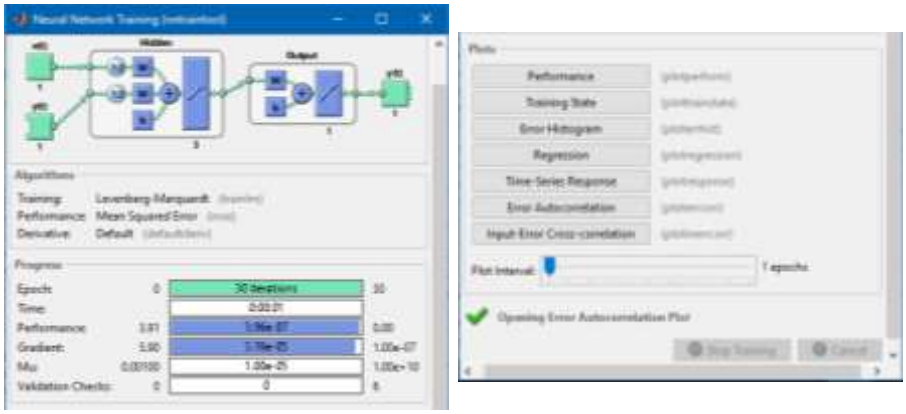


Figura 4.9 *Herramienta de entrenamiento de Matlab®.*

4.2.4 Evaluación de la red.

Una vez terminado el entrenamiento, es preciso llevar a cabo un análisis del ajuste de la red para determinar si es necesario realizar cambios en la configuración del entrenamiento, la arquitectura de red o en el conjunto de datos. De esta forma, en primer lugar, puede utilizarse el comando "*tr*" de Matlab® para obtener una descripción general acerca de los parámetros de rendimiento.

Asimismo, en favor de facilitar el análisis de la respuesta de la red, es posible efectuar la evaluación mediante un análisis estadístico en función de la respuesta gráficamente, es decir, Matlab® a través de la herramienta para el entrenamiento (**Figura 4.9**) presenta algunas funciones graficas que permiten analizar la respuesta.

En general, dos son las gráficas comúnmente utilizadas para evaluar la red: la primera, la gráfica de rendimiento, que permite conocer si el error cuadrático medio es el óptimo; también, puede utilizarse la gráfica de regresión, donde se muestra la correlación existente entre los datos obtenidos del entrenamiento con respecto a los datos esperados de salida.

Además de estas gráficas, puede considerarse la gráfica del histograma de error, la respuesta de la red en función de una gráfica de tiempo y, para el caso de predicción, puede tomarse en cuenta la gráfica de auto-correlación en el error. Asimismo, el probar diferentes perfiles de entrenamiento es otra forma útil de conocer el ajuste de la respuesta, es decir, el determinar el perfil que la red aproxima mejor si se cierra el lazo, puesto que en ocasiones las gráficas de desempeño no presentan el ajuste esperado.

Particularmente, es importante mencionar que, si el ajuste es adecuado, la configuración utilizada en el entrenamiento, presentará la ventaja de ser una red particularmente pequeña, es decir, la arquitectura se conforma de una capa escondida y una capa de salida, con 3 y 1 neuronas, respectivamente, por consecuencia, los recursos y la complejidad de implementación son mínimos.

4.2.4.1 Gráfica del rendimiento de la red

El criterio más común para conocer si el ajuste de la red es el óptimo, es a través del error cuadrático medio, puesto que presenta una medida de la desviación de los datos de acuerdo a la diferencia al cuadrado de los datos de salida esperados y el resultado de la red, es decir:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \qquad \textbf{(4.4)}$$

De esta expresión, "*N*" se refiere al total de muestras, "t_i" y "a_i" a los datos obtenidos y esperados, respectivamente.

Mediante este cálculo, se presenta una idea general de la optimización de la red, es decir, mientras la desviación en los datos se encuentre más próxima a cero, el ajuste es mejor. En la **Figura 4.10,** puede notarse el valor en el error conforme las repeticiones del proceso aumentan hasta que finalmente el entrenamiento se detiene. En este caso, en favor del ajuste óptimo, la red neuronal presentó un valor aproximado en el error cuadrático medio de 1.3654×10^{-6}.

Figura 4.10 *Gráfica del rendimiento de la red.*

4.2.4.2 Gráfica de regresión.

Mediante la gráfica de regresión, es posible conocer la relación entre la salida de la red y el objetivo, es decir, mediante una función lineal simple, puede calcularse la correlación existente entre la variable dependiente y la variable independiente. De esta forma, la herramienta de entrenamiento presenta la gráfica de regresión en función de la siguiente expresión:

$$a_q = mt_q + c + \varepsilon_q \qquad \textbf{(4.5)}$$

De esta expresión, "a_q" se refiere a la salida de la red entrenada, "m" a la pendiente, "t_q" al objetivo de la red, "c" al offset o el punto donde se intercepta "a_q" y finalmente, "ε_q" se refiere al error en la regresión.

Asimismo, los términos en la expresión para la regresión se calculan a través de las siguientes expresiones:

$$\hat{m} = \frac{\sum_{q=1}^{Q}(t_q - \bar{t})(a_q - \bar{a})}{\sum_{q=1}^{Q}(t_q - \bar{t})^2} \quad \textbf{(4.6)}$$

$$\hat{c} = \bar{a} - \hat{m}\bar{t} \quad \textbf{(4.7)}$$

Donde:

$$\bar{a} = \frac{1}{Q}\sum_{q=1}^{Q} a_q, \bar{t} = \frac{1}{Q}\sum_{q=1}^{Q} t_q \quad \textbf{(4.8)}$$

En resumen, a través de estas expresiones pueden determinarse los puntos que se alejen de la línea de regresión, es decir, los valores atípicos, puesto permiten conocer si es necesario reajustar el entrenamiento.

La gráfica que se obtuvo del entrenamiento se muestra a continuación en la **Figura 4.11**. Mediante este diagrama es posible conocer el grado de correlación existente entre las dos variables (los datos esperados y los obtenidos).

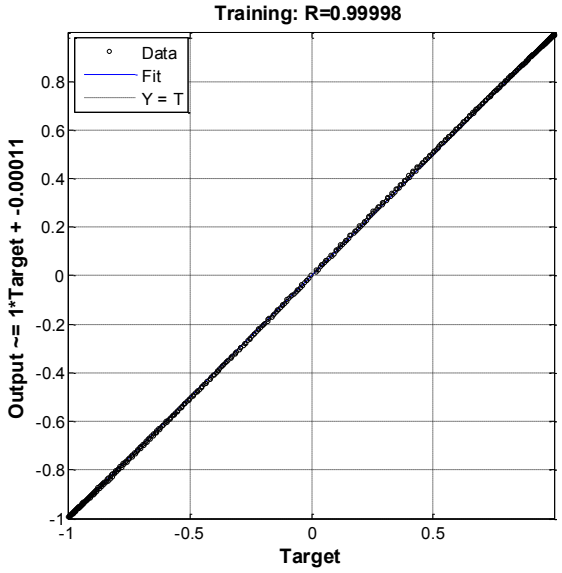


Figura 4.11 *Gráfica de regresión.*

Asimismo, es posible conocer de forma numérica el ajuste que se obtuvo, es decir, la correlación entre las dos variables. El coeficiente numérico comúnmente utilizado para medir el grado de correlación, es el coeficiente de Pearson, y puede calcularse a través de la siguiente expresión:

$$R = \frac{\sum_{q=1}^{Q}(t_q - \bar{t})(a_q - \bar{a})}{(Q-1)s_t s_a} \tag{4.9}$$

Donde "s_t" y "s_a", se refieren a los coeficientes de Pearson y pueden calcularse como:

$$s_t = \sqrt{\frac{1}{(Q-1)}\sum_{q=1}^{Q}(t_q - \bar{t})} \; ; \; s_a = \sqrt{\frac{1}{(Q-1)}\sum_{q=1}^{Q}(a_q - \bar{a})} \tag{4.10}$$

En general, mediante el coeficiente de Pearson es posible conocer el grado de variación de las dos variables, además, presenta una idea del sentido de la correlación entre ellas, directa si es positiva e inversa si es negativa.

Finalmente, de acuerdo a la variación en las variables de la **Figura 4.11** puede mencionarse que el ajuste es el óptimo, puesto que los datos que presentó el entrenamiento se ajustan de forma correcta a los esperados, de acuerdo al valor obtenido en el coeficiente de correlación *R* (próximo a 1). Sin embargo, esto no quiere decir que al hacer uso de la red, ésta presentará la respuesta esperada, puesto que debe considerarse que en lazo cerrado la salida no depende de los valores de entrada, sino de los valores que se almacenan en las líneas de retardo a través de una serie de tiempo.

4.2.4.3 Gráfica de la respuesta en series de tiempo.

El análisis previo proporciona una idea general del rendimiento en el entrenamiento, es decir, como pudo observarse, la función de regresión es útil para analizar los datos del entrenamiento, indicando si la respuesta de la red es la óptima. Sin embargo, es útil revisar también otra de las funciones que presenta Matlab® en favor de completar el análisis y, de esta manera, decidir si hacer uso de la red o intentar otro entrenamiento.

La gráfica de la respuesta en series de tiempo permite visualizar el comportamiento de la red en función del cálculo en el error entre el objetivo y la salida esperada a través del muestreo de los datos. La gráfica obtenida del mejor entrenamiento para la aproximación del perfil de corriente del Memristor, se muestra en la **Figura 4.12**.

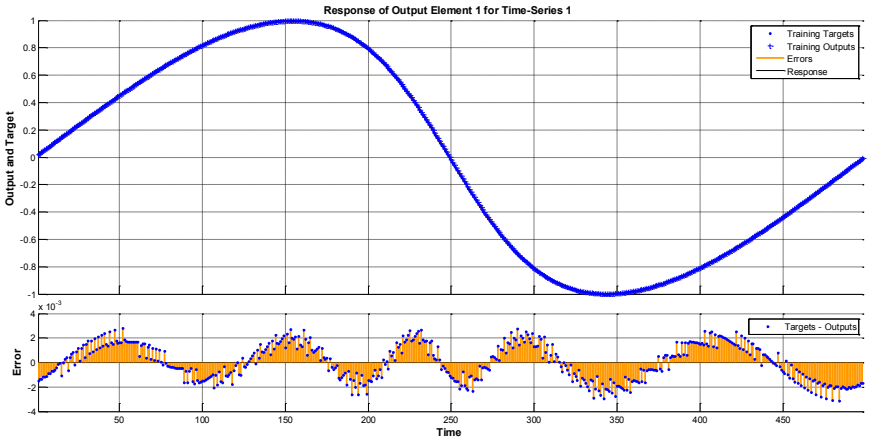


Figura 4.12 *Respuesta de la red dinámica a través de una serie de tiempo.*

De la gráfica se puede observar visualmente que el perfil se ajusta de manera adecuada, sin embargo, la variación en el error podría mejorar en favor de obtener el error óptimo y, por consecuencia, el proceso de entrenamiento debería repetirse. Sin embargo, es importante mencionar que esta configuración de entrenamiento fue la que presentó mejores resultados en las gráficas del análisis del desempeño de la red.

4.2.5 Uso de la red

El término *"uso de la red"* en redes multicapa se refiere a verificar el comportamiento de la arquitectura una vez entrenada y validada, es decir, el proceso de diseño ha finalizado y es posible utilizar la red para la resolución del problema. Sin embargo, para este caso, el uso de la red tiene como objetivo calcular la respuesta de salida en función de los bloques de retardo y, de acuerdo a la retroalimentación de la arquitectura NARX, es decir, la red presentará un comportamiento como predictor.

En general, el proceso de diseño de la arquitectura, para que se efectúe el proceso de predicción, requiere definir tanto la base de datos como el perfil utilizado en el entrenamiento. Asimismo, es necesaria la creación del lazo de retroalimentación, es decir, a través de la función *"closeloop"*, es posible la transformación de la red del entrenamiento en una configuración de lazo cerrado (**Figura 4.13)**.

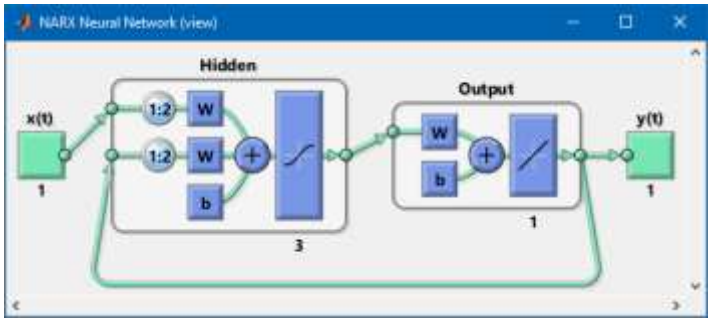


Figura 4.13 *Configuración lazo cerrado red neuronal tipo NARX.*

Además, debe considerarse el ajuste de los datos de acuerdo a las condiciones iniciales descritas por los bloques de retardo (a través del comando *"preparets"*). En este caso, se toman en cuenta los dos primeros valores, tanto de corriente como de voltaje a fin de utilizarlos como condiciones iniciales.

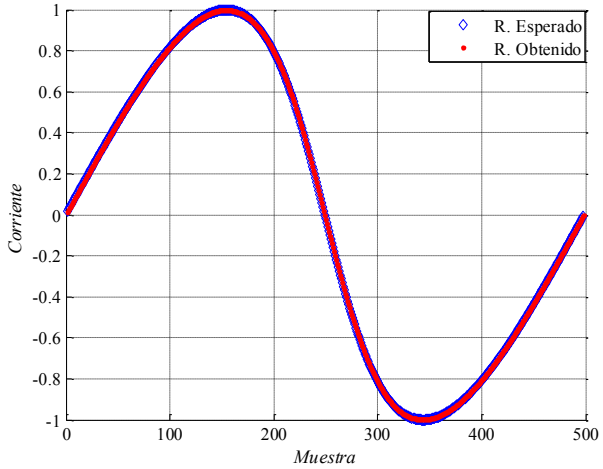


Figura 4.14 *Gráfica del perfil de corriente obtenido contra el esperado.*

Una vez configurados los parámetros de la red, es posible mostrar la respuesta de salida. En este caso el perfil de corriente, se muestra en la **Figura 4.14**, donde puede observarse que el perfil se ajusta correctamente, es decir, la red fue capaz de calcular la salida de los datos de corriente para cada punto de voltaje a través de la retroalimentacion en función de las líneas de retardo.

De esta forma, graficando la respuesta de corriente de la red a través de los datos de voltaje, se presenta el perfil de histéresis de la simulacion realizada con el modelo de Simscape®, tal como se muestra en la **Figura 4.15**.

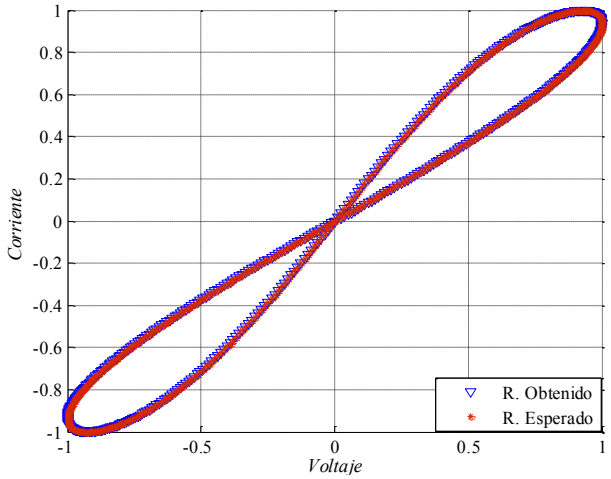


Figura 4.15 *Perfil de histéresis obtenido (Azul) contra el perfil esperado (Rojo).*

En general, el perfil mostrado presenta un ajuste correcto de la respuesta, demostrando que a través del uso de redes recurrentes, es posible llevar a cabo la estimación de la respuesta de sistemas que presentan un comportamiento no lineal, tal como el Memristor.

Finalmente, el código completo de la implementación de la red neuronal artificial puede consultarse en el **Apéndice B,** en la sección de anexos.

4.3Representación de la RNA en Simulink®.

En favor de comprobar el comportamiento de la red previo a la implementación en el dispositivo FPGA, se elaborará un modelo a bloques de la arquitectura de red a través de Simulink® [5], puesto que presenta un entorno que permite incluir los requisitos de la implementación en hardware.

Particularmente, el simulador permite la selección del tipo de dato para representar los valores, tal como el punto fijo (*Fixed Point)*. Esta representación es de gran utilidad para simbolizar valores fraccionarios y es útil en ocasiones para reducir la cantidad de almacenamiento en memoria.

De esta manera, a través de la interpretación de los valores en la arquitectura, se posibilita el entendimiento de su comportamiento en el momento que se efectúe la conversión de los datos de un formato decimal a binario.

Para este caso, se utiliza una palabra con una precisión de 18 bits, es decir, 1 bit de signo, 4 para la parte entera y los 13 restantes para la parte fraccionaria. El tamaño de palabra está determinado con base en el multiplicador integrado en el FPGA, puesto que efectúa la multiplicación de palabras de 18×18 bits, suficientes para ajustar correctamente la interpretación numérica de las variables.

De manera general, la implementación de la arquitectura en Simulink® permitirá evaluar el desempeño numérico de la arquitectura, posibilitando, además, percibir la dificultad del código necesario para efectuar la implementación en el dispositivo FPGA y los inconvenientes que se presentarían de acuerdo con el recorte de los decimales utilizados para representar los datos.

4.3.1 Proceso de diseño.

Considerando el ambiente de diseño en Simulink®, es posible la construcción de múltiples arquitecturas de red en función de bloques sumadores, multiplicadores, e incluso bloques que representen la respuesta de la función de transferencia. Asimismo, el software presenta también bloques de configuración sencilla que permiten simular las líneas de retardo.

La metodología para llevar a cabo la síntesis de la arquitectura de red propuesta, comienza por realizar la captura del esquemático de una neurona digital básica, puesto que es el bloque principal, descrito de acuerdo con el procesamiento de los datos de entrada, posteriormente, se llevará a cabo la conformación de la arquitectura, con base en subsistemas. Asimismo, se incluirán las líneas de retardo y el lazo de retroalimentación en favor de completar la arquitectura de red NARX.

Finalmente, es importante mencionar la configuración que deberá efectuarse en los parámetros de cada uno de los bloques, es decir, presentar la representación numérica de los datos de acuerdo con el tamaño de palabra.

4.3.2 Modelo de Neurona Digital.

El bloque fundamental para la construcción de una arquitectura de red, es la neurona simple de una sola entrada. De esta manera, considerando la arquitectura de red mostrada en la **Figura 4.13**, se presenta una red neuronal artificial con base en dos retardos; por consecuencia el vector de entrada deberá modificarse, tal como lo muestra la **Figura 4.16**.

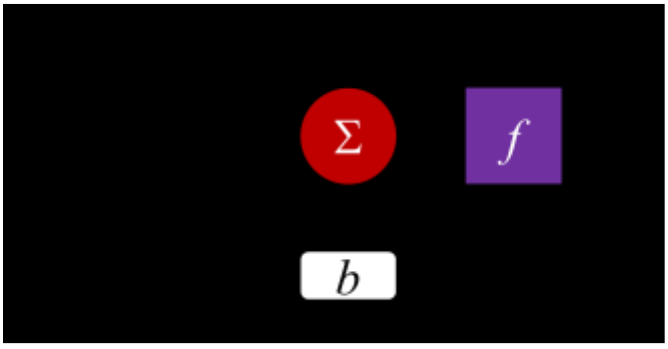


Figura 4.16 *Diagrama del modelo de neurona con múltiples entradas.*

En resumen, el bloque sumador calculará la salida neta de la red en función de la ponderacion de las entradas y considerando el valor de polarizacion. Así, se presentará la respuesta del vector de entrada en la funcion de activacion, en favor de presentar la respuesta de la neurona. Particularmente, se utiliza la función hiperbólica sigmoidal, puesto que permite considerar el resultado del sumador dentro de un intervalo ±1.

De esta forma, considerando lo descrito, el bloque que representa el comportamiento de una neurona a través de Simulink® se muestra en la **Figura 4.17**. Es importante además, presentar pesos y polarizaciones tanto para las 3 neuronas en la capa de entrada como para la neurona de salida.

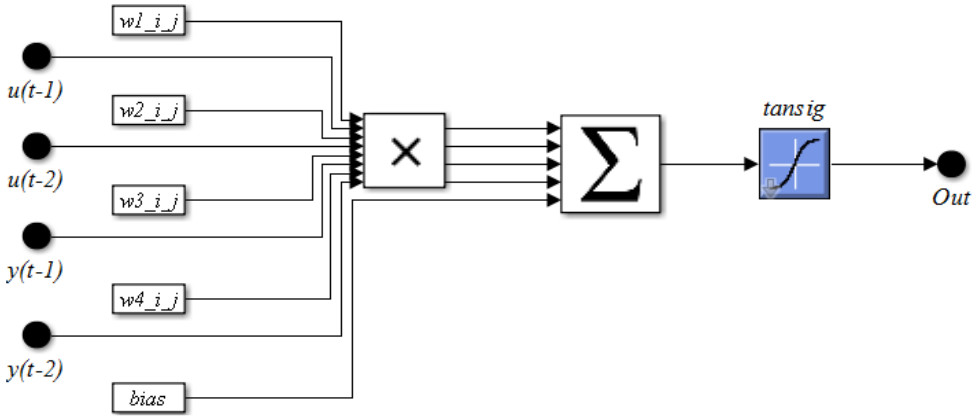


Figura 4.17 *Neurona digital representada en Simulink®.*

De esta manera, es necesario extraer los valores de peso y polarización del entrenamiento en Matlab®, utilizando las funciones *net.IW* y *net.B*. El resultado de estos comandos se muestra en la **Tabla 4.2** y **Tabla 4.3**, para la capa 1 y la capa 2, respectivamente.

Capa 1					
Neurona 1	*W11_1*	*W12_1*	*W13_1*	*W14_1*	*Bias*
	-1.0212	-0.4886	0.4599	0.4891	*1.5448*
Neurona 2	*W21_1*	*W22_1*	*W23_1*	*W24_1*	*Bias*
	-0.5888	*0.6486*	*0.4211*	*0.0831*	*-0.7302*
Neurona 3	*W31_1*	*W32_1*	*W33_1*	*W34_1*	*Bias*
	1.4024	*-1.3911*	*0.1245*	*0.247*	*0.588*

Tabla 4.2 *Pesos y Polarizaciones alcanzadas en la capa 1.*

Capa 2				
Neurona 1	*W11_1*	*W12_1*	*W13_1*	*Bias*
	0.0954	*1.2056*	*2.1528*	*-0.48224*

Tabla 4.3 *Pesos y polarizaciones en la capa de salida.*

4.3.3 Arquitectura de Red Neuronal Artificial.

A partir del bloque de neurona simple, es posible desarrollar subsistemas para completar la arquitectura de red. En general, debe considerarse la entrada de la red, es decir, presentar la base de datos de voltaje y desarrollar bloques de 3 y 1 neuronas para la capa de entrada y de salida, respectivamente. Asimismo, deben tomarse en cuenta las líneas de retardo; éstas, pueden representarse a través de un bloque integrador. Sin embargo, en favor de facilitar la implementación, es de utilidad el bloque de retardo presentado por Simulink®, el cual *"atrasa"* la entrada por el período de tiempo que se le especifique.

De esta manera, el diagrama de la arquitectura conformado por los bloques de retardo, la cantidad de neuronas y la base de datos, se muestra en la **Figura 4.18**.

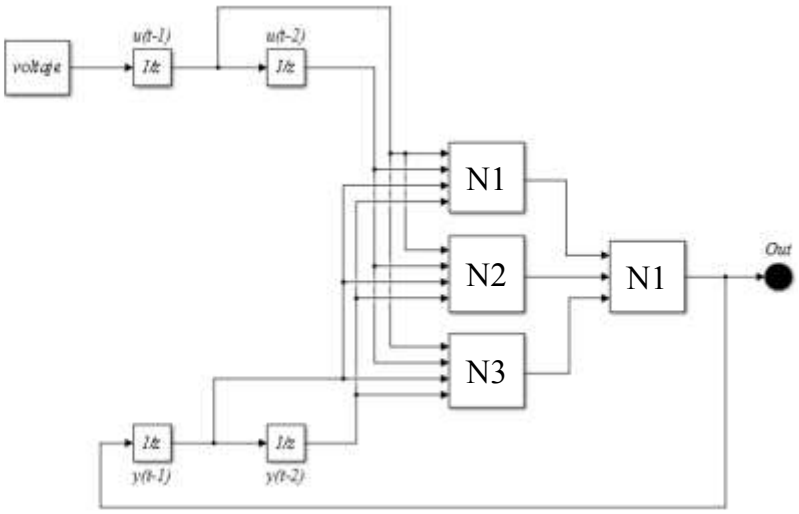


Figura 4.18 *Diagrama de arquitectura de red neuronal tipo NARX construida mediante Simulink®.*

Además, es importante mencionar que de acuerdo a la precisión de los datos, la cual comúnmente es de tipo *"doble"* por defecto en los modelos elaborados en Simulink®, se puede representar una amplia gama de valores numéricos en función de una cadena de hasta *"64 bits"*, debido al uso del *"punto flotante"*. Sin embargo, en este caso es necesario realizar la simulación de la arquitectura con la precisión requerida por la implementación digital en VHDL, es decir, de acuerdo a la precisión de los multiplicadores.

La precisión de los datos se especifica editando los parámetros de configuración de cada uno de los bloques que constituyen la arquitectura en general (multiplicadores, sumadores, etc.). Particularmente, se modifican los atributos de la señal, es decir, se define el tamaño de palabra y el tipo de dato. De esta manera, la exactitud de los datos se transforma de tipo *doble* a un formato en *punto fijo*, tal como lo muestra el bloque de la **Figura 4.19**.

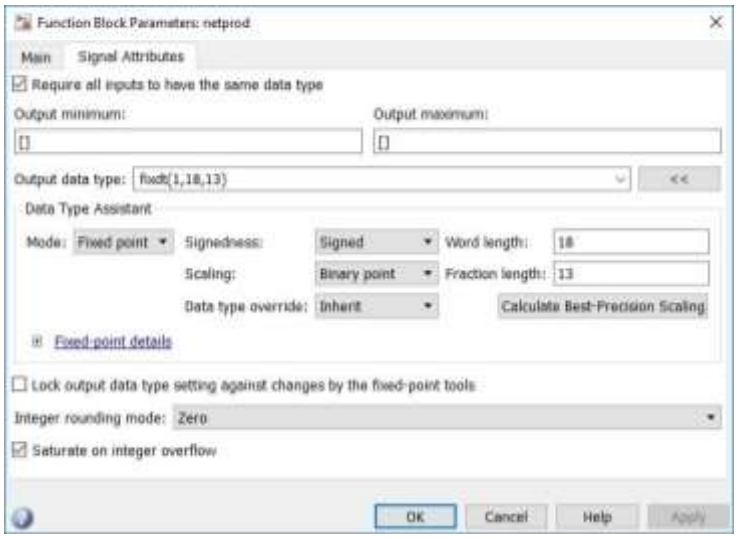


Figura 4.19 *Ventana de configuración de los atributos de los bloques en Simulink®.*

Así, es posible presentar finalmente, la respuesta de la simulación, tal como se muestra en la **Figura 4.20**, donde es importante mencionar que de acuerdo con el cálculo en función de una precisión menor en los valores, se observa un desfase en el valor máximo de la señal de corriente y, por consecuencia, el perfil de histéresis tampoco alcanza el máximo.

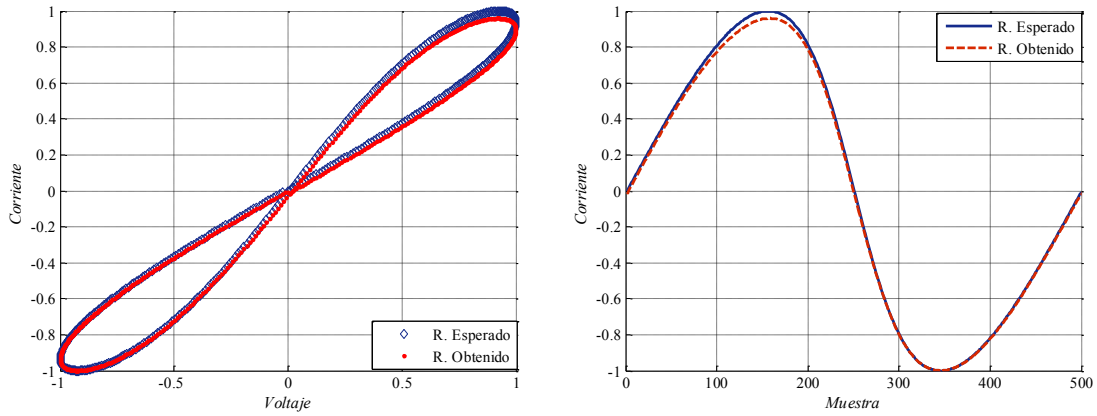


Figura 4.20 *Respuesta de la simulación efectuada en Simulink® contra la respuesta de la red implementada mediante las funciones de Matlab®.*

Sin embargo, de forma general, a pesar de que podría aumentarse el número de bits en la parte fraccionaria en favor de mejorar la aproximación de la respuesta, la pérdida se toma como aceptable, puesto que el ajuste es correcto y es importante considerar que la arquitectura no es razonablemente robusta; por consecuencia, la descripción de la estructura en el lenguaje VHDL no requeriría algún método de optimización, facilitando la implementación.

4.4Implementación de la RNA en VHDL.

El diseño digital ha evolucionado y permitido el desarrollo de sistemas complejos que implicarían costos elevados y dificultad en la implementación, debido a la robustez requerida y en algunos casos a los inconvenientes de fabricación.

Particularmente, la implementación digital del modelo de Memristor permitirá el desarrollo de un módulo que proporcione reproducibilidad en el comportamiento, puesto que existe variabilidad en los resultados obtenidos de las estructuras del dispositivo, consecuencia de la falta del perfeccionamiento de los métodos de fabricación y del insuficiente conocimiento de los mecanismos de comportamiento.

De esta manera, el presentar un sistema que sea capaz de aproximar la respuesta de histéresis, permitiría la exploración, investigación y el análisis de la respuesta de sistemas completos, en función de elementos no lineales, tales como el Memristor.

En general, para efectuar la implementación digital en VHDL de la arquitectura de red, es importante realizar la conversión de los datos en favor del desarrollo del sistema, puesto que facilita su manipulación en sistemas digitales [6]. En este caso, la conversión de los datos se realiza a través de líneas de código en Matlab®, transformando la cadena de datos de voltaje a un formato hexadecimal.

Una vez efectuada la conversión de los datos, es posible la descripción del sistema para la representación de la red neuronal en FPGA. En resumen, es necesario el desarrollo de un módulo de control que permita la sincronización del flujo de datos, es decir, que permita la comunicación entre las diferentes etapas del sistema. Asimismo, es indispensable la utilización de un bloque de memoria que almacene la base de datos de entrada; también, se deberá tomar en cuenta un bloque que realice el procesamiento que lleva a cabo la red neuronal y, finalmente, es importante desarrollar un bloque para la transmisión de los resultados.

La descripción de los bloques que conformarán el sistema se lleva a cabo a través de la herramienta de diseño digital VHDL, puesto que permite de manera simple, efectuar la simulación y sintetización de cualquier circuito lógico combinacional que se tenga en mente [7]. Particularmente, la descripción del sistema de red neuronal para la aproximación de la respuesta del Memristor y su implementación en un dispositivo lógico programable, FPGA.

4.4.1 Diseño estructural.

El lenguaje VHDL permite llevar a cabo una programación de tipo estructural [8], favoreciendo la claridad y el tiempo de desarrollo del programa a través de bloques desarrollados por el usuario o mediante modelos lógicos establecidos por el fabricante.

Particularmente, para la construcción del sistema que permita la aproximación de la respuesta del Memristor, es necesario identificar y desarrollar las etapas que conformarían el sistema, es decir, deberá presentarse la metodología del flujo de diseño.

De esta forma, a través del diagrama a bloques de la **Figura 4.21**, se presentan las etapas del sistema. En el diagrama se muestra el uso de un bloque de control, descrito en favor del desarrollo de las señales de reloj, útiles en la coordinación del flujo de datos entre las diferentes etapas del circuito. Además, este bloque presenta el control que permite al usuario interactuar con el sistema, en favor de llevar a cabo tareas básicas, por ejemplo, la inicialización del sistema o la extracción de los datos.

También, se presenta un bloque nombrado de “entrada”, descrito con base en el almacenamiento de los datos que presentar el perfil de corriente que se aproximó a través de la red neuronal artificial.

También, de acuerdo con el desarrollo de las operaciones que efectúa la red neuronal, se describe el bloque de "proceso" y, finalmente, en función del almacenamiento y transmisión de los resultados, se desarrolla el bloque de "salida".

En general, la descripción de cada uno de los bloques se efectúa en las secciones siguientes y, el código completo del sistema puede consultarse en el **Apéndice D**.

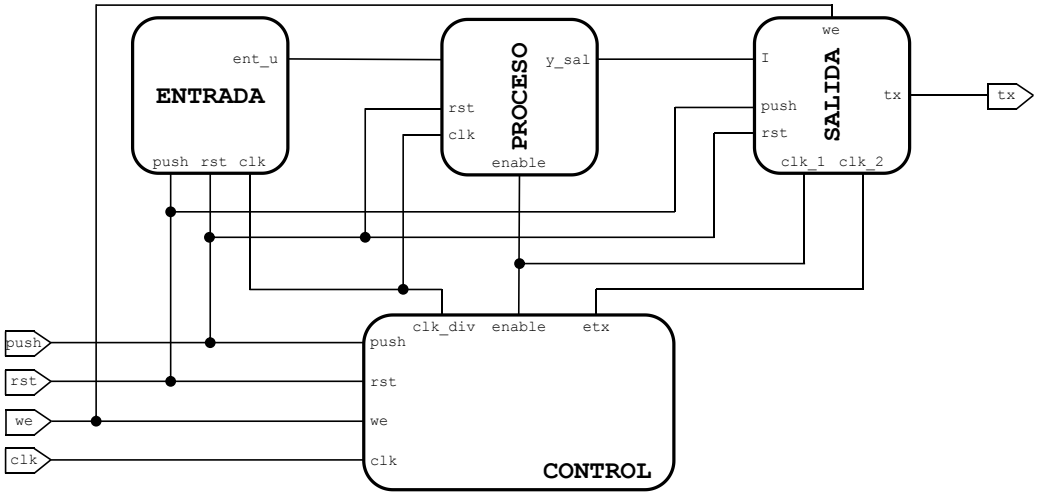


Figura 4.21 *Diagrama a bloques del diseño para la aproximación de la arquitectura de red.*

4.4.2 Módulo de entrada.

Una de las ventajas de la implementación del sistema mediante el leguaje VHDL, es que la estructura de la ejecución del código puede no seguir un orden específico, en función del estilo de diseño que se utilice [9].

De esta manera, considerando que el estilo estructural permite el desarrollo de los módulos en la jerarquía que se desee, el primer bloque a desarrollar es el de "entrada", puesto que presentará el almacenamiento de la base de datos útil para efectuar el procesamiento de la red neuronal.

El diagrama a bloques para la descripción de este sistema se muestra en la **Figura 4.22**. De manera general, este bloque se describe con base en el desarrollo de un módulo que permita la lectura de los datos de entrada, es decir, a través de un bloque de memoria constituida por 500 localidades.

En esta unidad de memoria se presenta la base de datos de voltaje como valores constantes, sin embargo, es necesario la descripción del control que permitirá presentar los datos en el módulo para la aproximación de la arquitectura, es decir, es necesaria la descripción de los procesos de lectura, asignación y cambio de dirección con base en la señal de reloj.

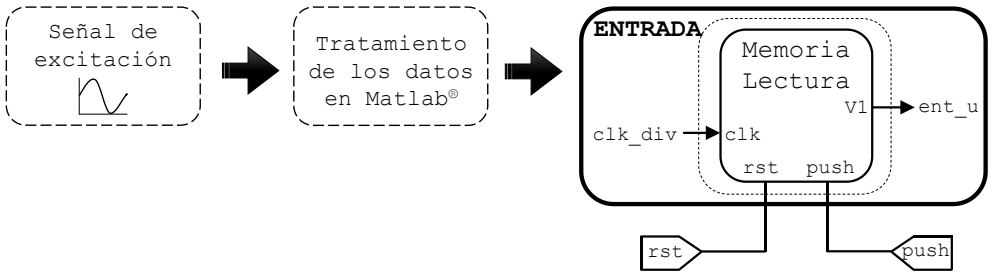


Figura 4.22 *Módulo para la descripción del almacenamiento de los datos de entrada.*

Finalmente, de acuerdo con la conversión de los datos, es importante mencionar que ésta se efectúa a través de líneas de código en el bloque "Tratamiento de los datos en Matlab®", mostrado en la **Figura 4.22**. Esta descripción de código, permite la conversión del formato decimal de la información a un sistema hexadecimal y, además, presenta los datos en el formato descrito por el lenguaje de programación VHDL.

4.4.2.1 Tratamiento de la base de datos.

Respecto al almacenamiento de los datos en el bloque de memoria, es importante describir la forma como se efectúa la simbolización de estos en sistemas digitales, es decir, en favor de representar la base de datos de forma que el sistema pueda procesarla, puede establecerse la correspondencia entre el sistema numérico utilizado por las personas y el de un sistema digital.

De forma general, el sistema tradicional utilizado en la representación de variables numéricas es el sistema decimal o de base 10, un sistema posicional, donde el valor final es el resultado de la suma ponderada de cada digito, diferente de un sistema electrónico digital, descrito con base en la representación de las señales de acuerdo con el uso de uno de dos valores o bits y, en favor de representar una sucesión de números o símbolos, se utiliza una cadena de bits. Este sistema numérico binario es también posicional, es decir, cada digito tiene un valor expresado a través de una potencia de 2, cuya suma expresará el valor decimal.

Asimismo, en una cadena de bits, el bit en el extremo izquierdo se le denomina bit más significativo (*Most Significant Bit*, MSB) y, el bit en el extremo derecho se describe como el menos significativo (*Least Significant Bit, LSB*). Además, de acuerdo con la representación de números negativos, se utiliza el sistema de complemento a dos [10] considerando el sistema de magnitud con signo, es decir, se dispone de un bit extra para representar la magnitud positiva o negativa del número; comúnmente el bit más significativo de la serie, es el denominado *"bit de signo"*, designando un cero lógico para representar el signo más y un uno lógico para el signo menos.

Los sistemas numéricos base 10 y base 2 son importantes tanto para realizar operaciones cotidianas como para el procesamiento digital, respectivamente. Sin embargo, existen sistemas numéricos de otras bases (*8* y *16*), que permiten codificar señales de múltiples bits. De esta forma, la representación de los datos es más sencilla, puesto que es más fácil representar una cadena grande de bits a través de grupos de 3 y 4 dígitos, es decir, en sistema octal y hexadecimal, respectivamente.

Asimismo, es importante considerar la manera más eficiente de representar numéricamente los datos, es decir, el formato en que el programa va a simbolizar los valores. Por ejemplo: *"punto fijo"* [11] o *"punto flotante"*, con base en la precisión y los recursos de hardware que se utilizarían [12].

Particularmente, de acuerdo con los multiplicadores del FPGA, la representación de los datos, en este caso, se efectúa en "*punto fijo*", describiendo los valores de voltaje en una palabra de dieciocho bits: un bit para la simbolización del signo, cuatro para representar la parte entera y los trece restantes para el segmento fraccionario, tal como lo muestra la **Figura 4.23**, que describe la constitución de la palabra que almacenará el dato en memoria.

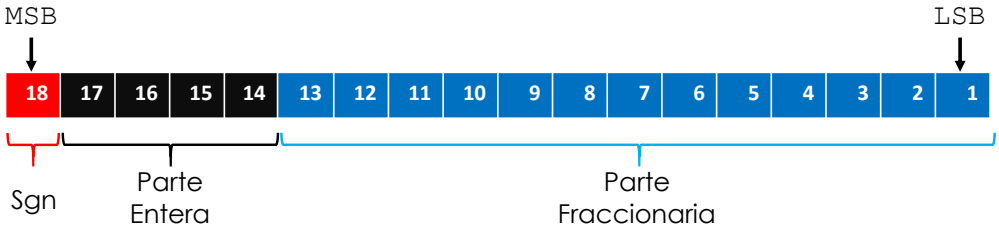


Figura 4.23 *Representación del tamaño de palabra.*

De manera general, la base de datos se almacena en memoria de acuerdo a la representación descrita, considerando el sistema hexadecimal en la cadena de bits, puesto que Matlab® presenta una función que realiza de forma automática la conversión de cadenas decimales a hexadecimales. Por lo tanto, a través de líneas de código se obtienen los datos de voltaje representados de acuerdo con el tamaño de palabra y con el formato descrito por VHDL. Asimismo, el código completo para la conversión de la base de datos puede consultarse en el **Apéndice C**.

4.4.3 Módulo de procesamiento.

El módulo de procesamiento se refiere al desarrollo de la arquitectura de red, es decir, a través de este bloque se describen las operaciones que realiza la entidad del procesamiento básico, la neurona.

El diagrama a bloques que describe la interpretación de este bloque de "proceso" se muestra en la **Figura 4.24**. De forma general, en el bloque "*Neurona Digital*", se efectúa la ponderación de las entradas de cada neurona y se aplica la función de transferencia con base en el control de la señal de reloj "enable". De esta manera, se genera la respuesta de los datos de entrada.

Asimismo, de acuerdo con la arquitectura de red, a través de registros, es posible describir las líneas de retardo y la generación del lazo de retroalimentación. El funcionamiento de este sistema de retardos se describe respecto a la señal de reloj "clk", es decir, a través de este control, es posible presentar los datos para su procesamiento dos unidades de tiempo atrás.

Además, las condiciones iniciales en la arquitectura de red se describen con base en un evento en la señal de control "*rst*", es decir, mediante la habilitación de esta señal, los dos primeros valores de la base de datos se presentan como valor inicial y es posible el comienzo de la operación del sistema.

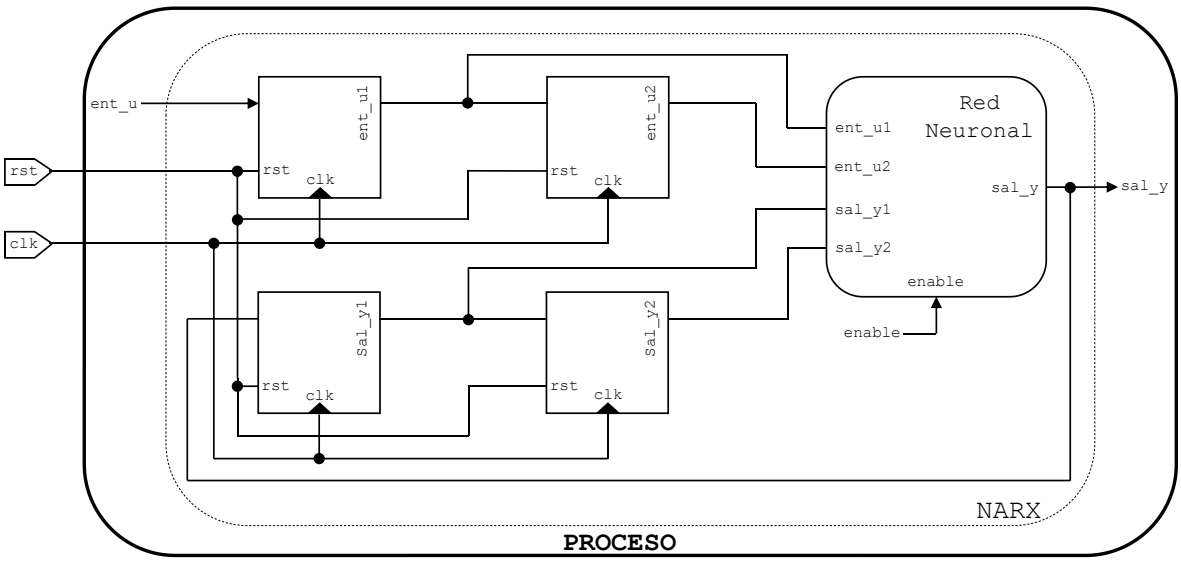


Figura 4.24 *Módulo de procesamiento descrito en función de la arquitectura de red NARX.*

4.4.3.1 Modelo de Neurona mediante VHDL.

De manera general, el bloque de neurona digital básico, indispensable para la unificación de la arquitectura de red, se muestra en la **Figura 4.25**. Este bloque se describe con base en un procesamiento de tipo paralelo, es decir, de acuerdo con el tamaño de la red, es posible llevar a cabo las sumas del bloque en líneas de transmisión diferentes, contribuyendo a la velocidad en la transmisión de los datos.

De esta manera, en función de la cantidad de multiplicadores en el FPGA, la optimización del código no es indispensable, puesto que los necesarios para la conformación de la arquitectura de red se ajustan de manera correcta.

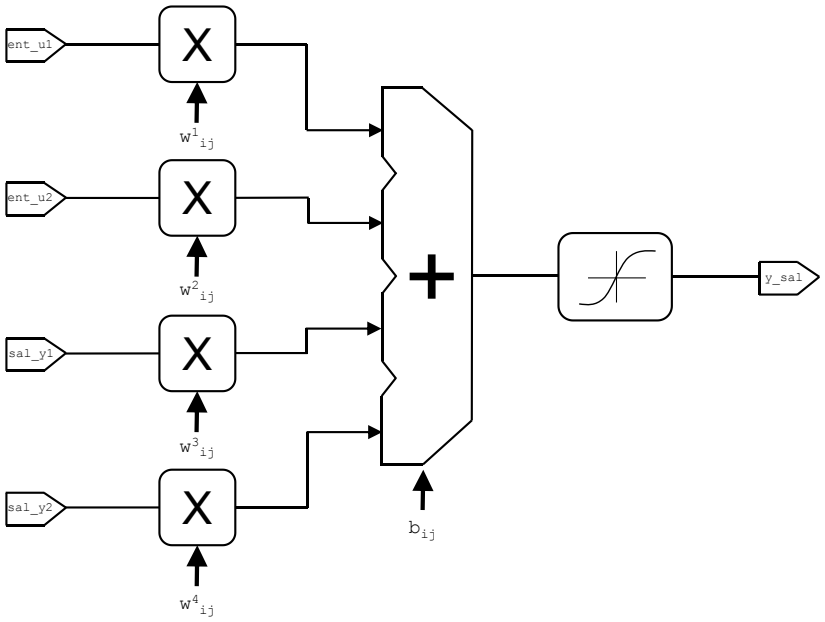


Figura 4.25 *Diagrama básico para la construcción del módulo de neurona básica de múltiples entradas.*

Además, en el diagrama se observan las cuatro entradas de acuerdo a las unidades de retardo. Éstas, se ponderan respecto a los pesos obtenidos en el entrenamiento y la nomenclatura "*i*, *j*", se refiere a la neurona y a la capa, respectivamente. Los pesos y las polarizaciones de cada neurona y capa se describen como valores constantes en la arquitectura del código, considerando los valores descritos en la **Tabla 4.2** y **Tabla 4.3**, realizando la conversión de la misma forma que con los datos de entrada.

Finalmente, es importante mencionar que, puesto que VHDL no presenta un módulo capaz de sintetizar la función de transferencia, será necesario efectuar el desarrollo de un bloque que aproxime la respuesta de esta función.

4.4.3.2 Aproximación de la función de transferencia.

La función sigmoide logarítmica es comúnmente utilizada para generar la activación de la respuesta de la neurona. Esta función es capaz de limitar la salida dentro de un intervalo de ±1, facilitando el procesamiento de los datos y la expresión para definir la función se describe como:

$$f(x) = \frac{e^{n}+e^{-n}}{e^{n}+e^{-n}} \qquad \textbf{(4.11)}$$

En favor de aproximar la respuesta de esta expresión, es necesario considerar algún método que permita presentar su comportamiento, puesto que no es posible la síntesis de la expresión de forma directa en el lenguaje de programación VHDL, es decir, el lenguaje no cuenta con módulos que permitan la convergencia, tanto del cálculo de la exponencial como de la operación de la división.

De acuerdo con lo descrito, puede considerarse alguno de los métodos reportados en la teoría [13] [14] [15], puesto que presentan una metodología útil en favor del desarrollo del bloque para la aproximación de la función de activación. De manera general, el uso específico del algún criterio requiere del análisis de los requerimientos, por ejemplo, si es necesario utilizar multiplicadores, deberá considerarse la cantidad requerida para su desarrollo.

Para el caso de este trabajo, se utiliza el método de segmentación. Este método, se describe con base en la aproximación de la sigmoide a través de la ecuación de la recta ($y = ax + b$); de esta forma, en función del intervalo de recta, es posible evaluar el valor de entrada.

De forma particular, en favor de la correcta aproximación de la sigmoide se utilizaron 16 segmentos de recta, considerando un intervalo en el eje de las abscisas de *0* a *3*. El cálculo de la pendiente y del valor de la ordenada al eje de las abscisas de cada segmento, se efectúa mediante líneas de código en Matlab®, utilizando la función "*polyfit*", y los valores obtenidos se presentan en la implementación del código como valores constantes, de la misma forma que los valores que conforman el eje horizontal.

Asimismo, es importante mencionar que, el análisis de la respuesta de este bloque se efectúa de acuerdo con el desarrollo de un trabajo de maestría anterior [16].

4.4.4 Módulo de salida.

Finalmente, una vez efectuado el procesamiento de los datos en función de la arquitectura de red, es necesario el almacenamiento de los resultados y la transmisión de los mismos para su análisis posterior.

De esta manera, el diagrama que constituye el bloque de salida se muestra en la **Figura 4.26**. En el diagrama se muestra la unidad que describe el almacenamiento de los resultados que presentó la red, es decir, el módulo de memoria, similar al módulo de entrada. Sin embargo, de acuerdo con la función de este bloque de "salida", es necesario llevar a cabo el control de escritura de los resultados que presentó la unidad de procesamiento.

De esta forma, este bloque requiere el uso de una señal de control externo que permita interactuar con el sistema para efectuar los procesos de escritura/lectura. De esta manera, a través de la habilitación del control "we", el proceso de escritura tiene lugar en función de la señal de reloj "*clk_1*". Asimismo, la lectura de los resultados en favor de su transmisión, se efectúa a través de la señal "*clk_2*", puesto que permite la transmisión de los datos de acuerdo con la velocidad descrita por el puerto *USB-UART*.

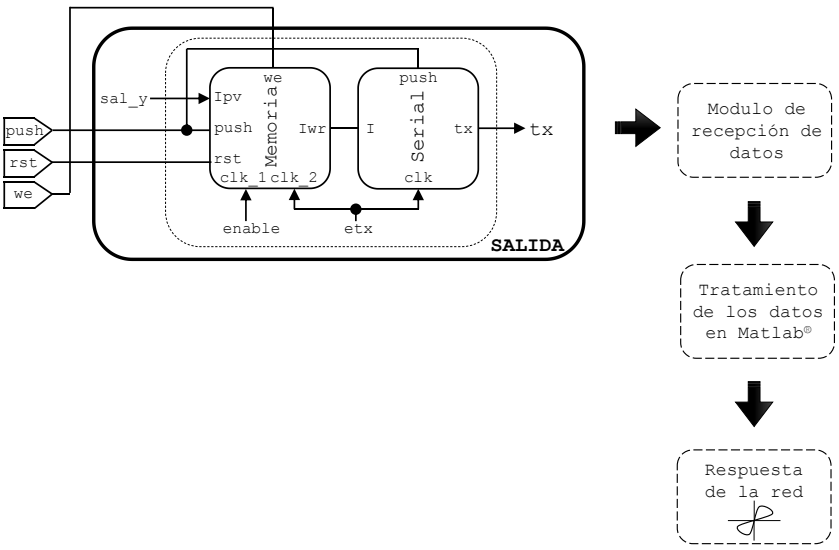


Figura 4.26 *Bloque de salida descrito con base en el almacenamiento y transmisión de los resultados de la red neuronal.*

4.4.4.1 Transmisión de datos.

Para llevar a cabo el proceso de transmisión/recepción, es necesario utilizar protocolos que presentan las reglas básicas para permitir el intercambio de datos. Además, es importante considerar la forma en que se transmitirá la información, puesto que permitirá conocer la velocidad de la transmisión en función de la cantidad de datos.

De manera general, puede efectuarse transmisión en paralelo, permitiendo una velocidad de intercambio de información alta, puesto que cada bit en la cadena se envía al mismo tiempo, sin embargo, penaliza el número de líneas para el envío y, por consecuencia, la implementación. Asimismo, puede utilizarse transmisión en serie, en función de utilizar una sola línea de transmisión, sin embargo, puesto que el envió se efectúa bit por bit, la velocidad de transmisión se reduce.

En este caso, la transmisión de los datos se efectúa de forma serial de acuerdo al funcionamiento del transmisor/receptor universal asíncrono (UART), un estándar comúnmente utilizado en un sistema de comunicaciones para establecer las condiciones de transmisión. Estas condiciones se describen de acuerdo a la norma RS-232 que designa la norma para el intercambio de datos entre un equipo de comunicación (*DCE*) y un equipo terminal (*DTE*).

De manera general, el protocolo de transmisión se muestra en la **Figura 4.27**. Este consiste en enviar, en primer lugar, un bit de arranque o inicio que le permita al receptor conocer en qué momento se efectuará el envío de la trama de datos; posteriormente, una vez completado el envió de la trama, se manda un bit de paro en función de indicar el fin de la transmisión, y este proceso se repite hasta que se complete la transmisión de toda la base de datos.

Asimismo, es importante considerar la velocidad de transmisión, es decir, el periodo de cada uno de los bits. Comúnmente la velocidad de conexión en un UART va desde 300 hasta 115200 baudios. Particularmente, puesto que la base de datos no es demasiado grande, es posible utilizar una velocidad para la transmisión de 9600 baudios. De esta forma, la base de datos de la respuesta de la red se genera en un tiempo aproximadamente de *1.75s*.

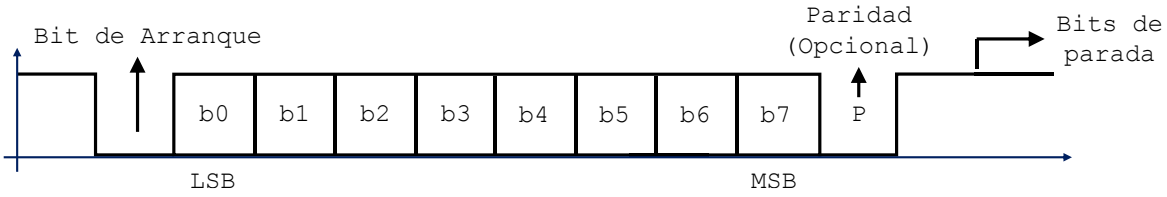


Figura 4.27 *Protocolo de comunicación asíncrona.*

Finalmente, es importante mencionar que en función del protocolo para la transmisión de los datos, deberá generase una cadena de 24 bits, es decir, 3 tramas de 8 bits, puesto que la precisión de los datos está descrita respecto a 18 bits, por lo tanto, la tercera trama será completada en ceros en favor de obtener el tamaño de palabra requerido.

4.4.5 Módulo de control.

El modulo de la **Figura 4.28** está descrito con base en el control de flujo de datos entre las entidades, es decir, en este bloque se definen las diferentes señales de reloj en función del módulo de 100MHz presente en el FPGA. Asimismo, este bloque presenta el control de la interaccion con el sistema, en favor de realizar acciones básicas, tales como el reset del sistema, la inicializacion de la transmision y la habilitacion de los procesos de escritura/lectura.

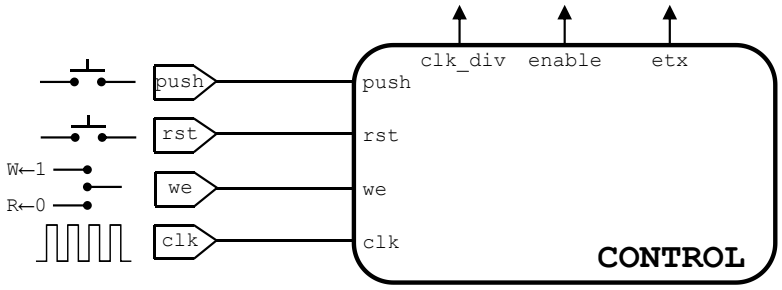


Figura 4.28 *Módulo de control definido en función de la sincronización de las etapas del sistema.*

Particularmente, el control *"rst"*, está descrito con base en la inicialización del sistema, es decir, la habilitación de esta señal al comienzo de la operación permitirá presentar las condiciones iniciales en los registros del bloque de procesamiento. Además, este control efectúa tareas tales como el reinicio en la dirección de las memorias (*entrada y salida*) y la inicialización de las señales de reloj que permiten la sincronización de los eventos en los bloques de entrada y procesamiento.

De la misma forma, el control externo *"push"*, se define con base en la habilitación del envió de los datos, es decir, la señal deberá activarse para permitir la transferencia de la información. De manera general, esta señal efectúa diferentes tareas en el sistema, tales como el restablecimiento de las señales de reloj involucradas en el protocolo de transmisión, la inicialización de las tramas a enviar y, de forma particular, permite efectuar un reset en la dirección de la memoria de salida en función de reiniciar la transmisión.

Finalmente, la tercera señal que permite interactuar con el sistema, "*we*", se define con base en los procesos de escritura/lectura en memoria de salida. Si este control permanece en su estado activo, permitirá la escritura del resultado de los datos en la memoria de salida, de forma contraria, en favor de iniciar la transmisión de los datos, deberá deshabilitarse puesto que permitirá efectuar el proceso de lectura de los resultados.

Asimismo, además de la interacción con el sistema, el bloque de control presenta el objetivo de generar señales de reloj que permitan la sincronización de los diferentes eventos presentes en el sistema. El diagrama de tiempo que describe cada una de las señales se muestra en la **Figura 4.29**.

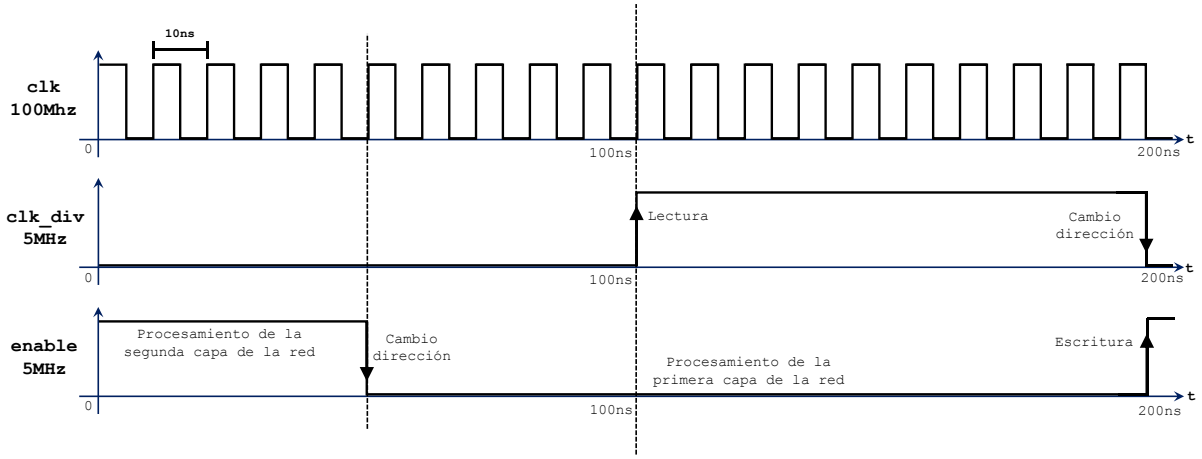


Figura 4.29 *Diagrama temporizado de las señales de control.*

Es importante mencionar que el diagrama considera únicamente las señales utilizadas en el bloque de entrada, procesamiento y escritura en memoria de salida, puesto que la velocidad de procesamiento en estos bloques puede describirse de forma similar. Estas señales se generan a partir del reloj principal de la tarjeta lógica (FPGA), el cual considera una frecuencia de 100Mhz, es decir, el periodo de la señal se define a través de una fase de 10ns, por lo tanto, con base en el periodo es posible la generación de señales de reloj que presenten tiempos suficientes para que se efectúe el procesamiento en la arquitectura de red.

De esta manera, la primera señal *"clk_div"*, se describe de acuerdo con una frecuencia de 5Mhz, es decir, considerando 20 ciclos del reloj principal. Esta señal permite efectuar la sincronización de los datos en los registros de acuerdo con los procesos de lectura y cambio de dirección en memoria del bloque de "entrada".

De forma similar, la señal "*enable"* se define con base en una frecuencia de 5MHz, sin embargo, el tiempo en estado activo es menor. Durante este periodo se realiza el procesamiento en la segunda capa de la red y tanto las sumas y multiplicaciones de la primera capa, como el procesamiento de la función de activación, se efectúa través de su estado inactivo.

Finalmente, en función de llevar a cabo la transmisión de los datos es necesario desarrollar una señal que permita la transferencia de la información a la velocidad descrita para el puerto UART (9600 baudios). Por lo tanto, considerando que el periodo de la señal de transmisión es de *0.10416ms*, puede describirse el envío de un bit de información en función de 10416 ciclos del reloj del FPGA.

De esta manera, la señal "*etx*" mostrada en la **Figura 4.30**, está descrita con base en 343,728 ciclos del reloj de 100Mhz de acuerdo con la transmisión de 3 tramas de 11 bits. Asimismo, esta señal permite llevar a cabo procesos de lectura y cambio de dirección en memoria de salida.

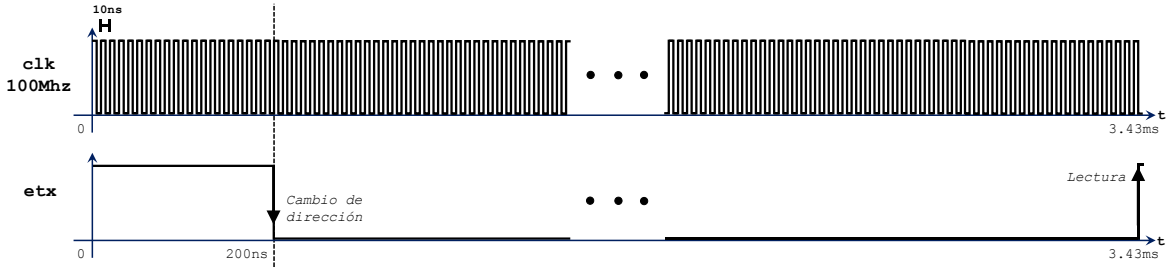


Figura 4.30 *Diagrama de tiempo para la creación de la señal encargada de la transmisión de los resultados.*

4.4.6 Resultados.

Una vez realizada la implementación de la red neuronal en el lenguaje de programación, se efectúa la transmisión de los resultados a través del puerto serial del FPGA. Sin embargo, es importante describir el procesamiento que se efectúa en los datos en favor de observar la respuesta.

De manera general, el procesamiento que se efectúa para presentar el análisis de los resultados de la transmisión, se muestra mediante la **Figura 4.31**.

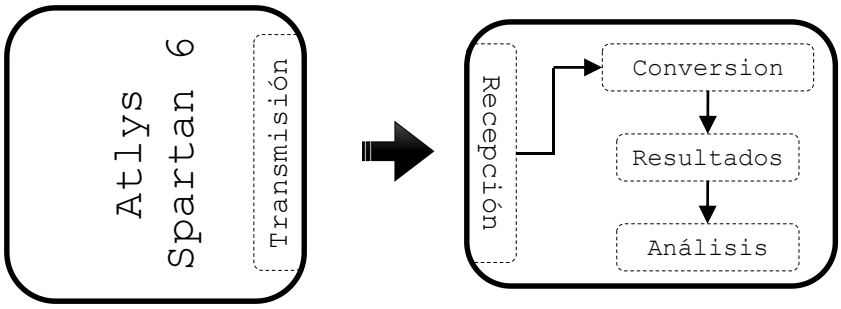


Figura 4.31 *Proceso del tratamiento de los datos arrojados por la red implementada en FPGA.*

En función de llevar a cabo el proceso de recepción de los resultados a través del puerto serie, deberá configurarse el puerto, especificar la velocidad de transmisión, abrir y cerrar el puerto, etc. Esto se realiza a través de líneas de código en Matlab® que permiten el funcionamiento del proceso de transmisión/recepción.

Sin embargo, es indispensable, además, llevar a cabo el proceso de reconversión de los datos. De esta forma, en favor de la conversión de la cadena de datos de formato binario a decimal, es de utilidad el código desarrollado en Matlab® del **Apéndice E**. En resumen, este código une las tramas recibidas y presenta una cadena de 18 bits en un proceso que desarrolla la conversión del formato binario a decimal.

De esta manera, se obtiene la base de datos de los resultados presentados por la arquitectura de red implementada en el FPGA, tal como se muestra en la gráfica de corriente de la **Figura 4.32**. En el diagrama se observa una comparación del ajuste de la respuesta en el FPGA de acuerdo con la respuesta presentada por el modelo de Memristor de HP®.

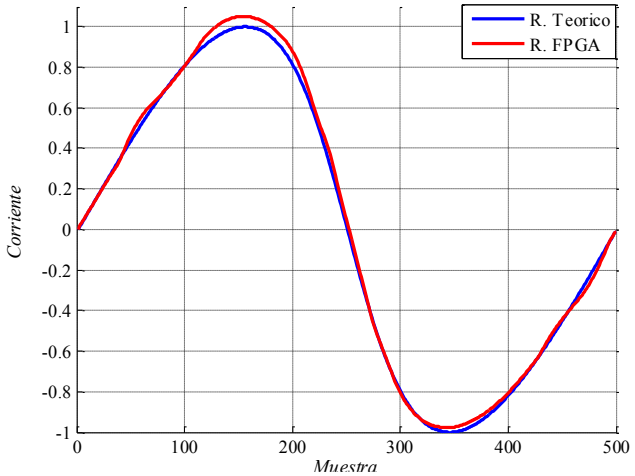


Figura 4.32 *Gráfica de la respuesta obtenida mediante el FPGA contra la respuesta de la red en Matlab®.*

Asimismo, en la gráfica de la **Figura 4.33** se observa la respuesta de histéresis obtenida mediante el FPGA en comparación con la respuesta teórica esperada.

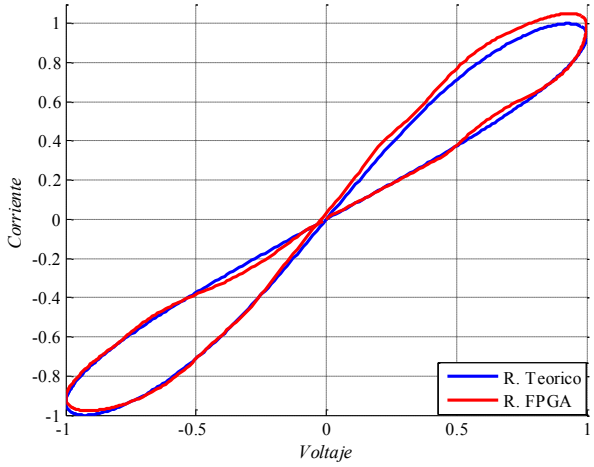


Figura 4.33 *Respuesta de histéresis obtenida del módulo de aproximación del Memristor.*

Finalmente, con base en lo mostrado en los diagramas de la respuesta, es importante mencionar que el desfase en el ajuste de la respuesta en FPGA se debe a la resolución de palabra presentada para la implementación, sin embargo, de acuerdo con el análisis de la implementación en Simulink®, es posible obtener un mejor ajuste aumentando la resolución decimal en la parte fraccionaria.

Además, en favor de conocer numéricamente el desajuste que se presenta, es posible utilizar el error cuadrático medio, puesto que describe la desviación de los datos. De esta forma, a través de la expresión **(4.4)**, se presenta un error de 9.1399×10^{-4}.

Asimismo, es posible el cálculo del coeficiente de correlación, permitiendo conocer la variación entre el resultado teórico y el obtenido. En este caso, la expresión **(4.9)** para la correlación, presentó un resultado de *0.9996.*

4.5Conclusiones

De acuerdo con lo descrito a través de éste capítulo, fue posible la definición de la metodología empleada en el desarrollo de la red neuronal que permite aproximar la respuesta del Memristor.

De manera general, la arquitectura de red se describe a través de líneas de código en Matlab®, efectuando el desarrollo de cada una de las etapas involucradas tanto en el entrenamiento como en el uso de la red, esto, de acuerdo con la arquitectura NARX. Además, en favor de conocer el ajuste de la red en el entrenamiento, se presentó el análisis de la respuesta a través de las gráficas de rendimiento.

Asimismo, en favor de conocer la respuesta de la red respecto a la implementación en VHDL se efectuó la descripción previa de la arquitectura en Simulink®, puesto que permitió conocer la convergencia de la respuesta ante la precisión descrita por los multiplicadores del FPGA, es decir, utilizando el formato en punto fijo en favor de presentar la descripción del tamaño de palabra.

Finalmente, con base en el desarrollo de la arquitectura mediante el lenguaje VHDL, se describió cada uno de los bloques que conforman el sistema para la aproximación de la respuesta del Memristor. De esta forma, mediante las simulaciones efectuadas y a través de los datos generados por el dispositivo lógico FPGA, puede mencionarse que la respuesta obtenida de la implementación se ajusta correctamente al modelo teórico de HP®.

Referencias

[1] M. T. Hagan, H. B. Demuth, M. H. Beale y O. D. Jesús, Neural Network Design (2nd Edition), Martin Hagan, 2014.

[2] M. H. Beale, M. T. Hagan y H. B. Demuth, Neural Network Toolbox Design Book, 2011.

[3] O. D. Jesus y M. T. Hagan, «Backpropagation Algorithms for a Broad Class of Dynamic Networks,» *IEEE Transactions on Neural Networks,* vol. 18, nº 1, pp. 14-27, 2007.

[4] M. T. Hagan y M. B. Menhaj, «Training feedforward networks with the Marquardt algorithm,» *IEEE Transactions on Neural Networks,* vol. 5, nº 6, pp. 989-993, 1994.

[5] «MathWorks,» [En línea]: http://www.mathworks.com/help/simulink/index.html.

[6] R. J. Tocci. y N. S. Widmer, Sistemas Digitales: Principios y Aplicaciones, Pearson Educación, 2002.

[7] V. A. Pedroni, Circuit Design with VHDL, Massachussets Institute of Technology, 2004.

[8] J. A. Boluda y F. Pardo, *VHDL: Lenguaje para síntesis y modelado de circuitos,* Alfaomega, 2000.

[9] M. A. R. Barranca, O. A. Cárdenas y L. M. F. Nava, *Curso Electronica Digital - Unidad 1: Descripción y Simulación de Circuitos Digitales Utilizando VHDL,* México: CINVESTAV, 2014.

[10] J. F. Wakerly, Diseño Digital: principios y prácticas, Mexico: Perason Eduacación, 2001.

[11] X. Li, M. Moussa y S. Areibi, «Arithmetic Formats for Implementing Artificial Neural Networks on FPGAs,» *Canadian Journal of Electrical and Computer Engineering,* vol. 31, nº 1, pp. 31-40, 2006.

[12] A. W. Savich, M. Moussa y S. Areibi, «The Impact of Arithmetic Representation on Implementing MLP-BP on FPGAs: A Study,» *IEEE Transactions on Neural Networks,* vol. 18, pp. 240-252, 2007.

[13] A. H., K. Curtis y B. Gill, «Piecewise Linear Approximation applied to Non-Linear Function of a Neural Network.,» *IEE Proceedings Circuits, Devices and Systems,* vol. 144, pp. 313-317, 1997.

[14] M. D.J. y R. Hutchinson, «Efficient Implementation of Piecewise Linear Activation Function for Digital VLSI Neural Networks,» *Electronics Letters,* vol. 25, p. 1662–1663, 1989.

[15] J. Zhu y P. Sutton, «FPGA Implementations of Neural Networks-A Survey of a Decade of Progress,» *Conference Field Programmable Logic,* pp. 1062-1066, 2003.

[16] G. M. Tornez Xavier, *Desarrollo en FPGA de un emulador de panel fotovoltaico,Tesis de Maestría en Ciencias*, México: CINVESTAV, 2014.

Conclusiones Generales

- La importancia del estudio del modelo de Memristor, se define con base en el análisis de la respuesta que se esperaría de un dispositivo físico, considerando además, el comportamiento que presentaría ante diferentes situaciones de trabajo. De esta forma, se justifica la necesidad de presentar un antecedente del análisis teórico del modelo propuesto por Hewlett Packard y la teoría propuesta por el Dr. Leon Chua para el Memristor.

- Asimismo, el análisis del modelo de HP® a través de "*funciones ventana"* permitió la descripción de los fenómenos que se presentan en los bordes del dispositivo. De esta manera, es posible desarrollar una perspectiva más amplia acerca del funcionamiento del Memristor.

- La comprobación de las ecuaciones del modelo de HP® se efectuó a través del desarrollo de un módulo a bloques en Simulink® que permitió reproducir el comportamiento de las expresiones que definen al dispositivo. Este modelo, presenta además, la descripción de las diferentes funciones de *"ventana"* y permite el cambio de polaridad en el dispositivo. Asimismo, fue posible la comprobación de su funcionamiento a través del desarrollo de un oscilador sin reactancia.

- La descripción de la metodología para la construcción de la red neuronal que permite aproximar la respuesta del Memristor, se fundamenta en la arquitectura tipo NARX, desarrollada mediante líneas de código en Matlab®. Además, es importante mencionar que la base de datos utilizada en el entrenamiento de la red, se generó mediante el modelo desarrollado en Simulink®, demostrando flexibilidad e integración de los entornos de trabajo que presenta Matlab®.

- Con base en la obtención de un sistema digital que permitiera aproximar la respuesta del Memristor, se llevó a cabo la implementación de la arquitectura de red neuronal en tecnología FPGA. Particularmente, la descripción del código se efectuó mediante VHDL, utilizando un estilo de diseño estructural. Asimismo, en favor de conocer la respuesta del sistema de acuerdo con la precisión descrita por los multiplicadores del FPGA, se desarrolló de forma previa un sistema a bloques en Simulink, que permitiera conocer la respuesta del sistema ante el recorte de los decimales en los valores.

- Finalmente, de acuerdo con el análisis de la respuesta obtenida a través de los diferentes entornos de trabajo fue posible determinar el desempeño de la red neuronal, demostrando el potencial que presenta de acuerdo con el desarrollo de sistemas que se definen con base en comportamientos no lineales.

Perspectivas: Trabajo Futuro

- Considerar el desarrollo del modelo en red neuronal en función de la relación constitutiva *carga-flujo*, presentando una ecuación de corriente en favor de conocer si la respuesta se ajusta mejor.

- Este trabajo presenta las bases fundamentales del modelo de Memristor, sin embargo, es posible realizar un estudio particular de los mecanismos que definen el comportamiento del dispositivo. De esta manera, se presentaría un mejor entendimiento de parámetros tales como la movilidad, corriente, el espesor de la película, etc., permitiendo mejorar la respuesta del modelo en red neuronal para la aproximación del comportamiento del Memristor.

- En función de un mejor ajuste en la respuesta de predicción de la red, es posible, efectuar la modificación de los parámetros de enteramiento. Por ejemplo, de acuerdo con la mejora del aprendizaje, presentar más bloques de retardo o utilizar un algoritmo de entrenamiento dinámico.

- Asimismo, con base en la modificación de los parámetros de la arquitectura de red, es necesario considerar la adecuación de la implementación en el FPGA, es decir, de acuerdo con la arquitectura de red, es indispensable tener en cuenta los recursos necesarios para el desarrollo del código, por ejemplo, en función del número de multiplicadores, determinar si es necesario efectuar técnicas de optimización, tales como el multiplexado.

ANEXOS

Apéndice A

Sección 1. *Código para la solución de la ecuación diferencial del Memristor.*

```
function Memristor_Matlab
Ron = 100; Roff = 16e3;
Rint = 11e3; DR = Roff-Ron;
tmin = 0; tmax = 1;                  %Intervalo de tiempo
N=500;                               %Numero de pasos
OsaT = tmin:(tmax-tmin)/N:tmax;      %Eje de Tiempo
x0 = (Roff-Rint)/DR;                 %Condición Inicial

[t,x] = ode23t(@ODE_Memri_A,OsaT,x0); %Llamado de la ODE
V = 1*sin(2*pi*t);                    %Entrada de Voltaje
I = V./(Roff-x*DR);                   %Corriente Obtenida

figure(1)
plot(V,I)               %Perfil de Histéresis
figure(2)
plot(OsaT,V)            %Perfil de Voltaje
figure(3)
plot(OsaT,I)            %Perfil de Corriente
figure(4)
plot(OsaT,x)            %Posición de la frontera


%--- Definición ODE ---
function dx=ODE_Memri_A(t,x)
    D = 10e-9; Ron = 100;
    Roff = 16e3; DR = Roff-Ron;
    uv = 1e-14;         %Movilidad
    k = uv*Ron/D^2; p = 10;

%--- Solución ODE ---
V = 1*sin(2*pi*1*t);    %Entrada de Voltaje
fx = (1 - (2*x-1)^(2*p));
dx = k*(V/(Roff-x*DR))*fx;
```

Apéndice B

Sección 1. *Código del entrenamiento de la red neuronal artificial.*

```
%Neural Network Nonlinear Autoregressive with Exogenous Inputs
%                          NARX
%%%%%%%%%%%%%%%%%%%%%%%%% ENTRENAMIENTO %%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clc
%%%%% Preparación de Datos
load DCV_sen.dat           %Perfil de voltaje senoidal 500 datos

vol = DCV_sen (:,1);
corr = DCV_sen (:,2);

[Y, PS] = mapminmax (corr');

v = con2seq (vol');
i = con2seq (Y);

%%%%% Configuración
net = narxnet (1:2,1:2,3);
net.divideFcn = '';
net.trainParam.epochs = 30;

[Xs, Xi, Ai, Ts] = preparets (net, v, {}, i);

%%%%% Entrenamiento de la Red
[net, tr] = train (net, Xs, Ts, i);
yp = sim (net, Xs, i);
view (net)
```

Sección 2*. Código de la simulación de la red neuronal artificial.*

```
%Neural Network Nonlinear Autoregressive with Exogenous Inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%% SIMULACION %%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clc
%%%%% Preparación de Datos
load Esenoidal10.mat            %Perfil de entrenamiento
load DCV_sen.dat                %Perfil de voltaje senoidal 500
datos
vol = DCV_sen (:,1);
corr = DCV_sen (:,2);

[Y, PS] = mapminmax (corr');

v = con2seq (vol');
i = con2seq(Y);

%%%%% Configuración

narx_net_closed = closeloop (net);
view (narx_net_closed)
```

```
[p1, Pi1, Ai1, t1] = preparets (narx_net_closed, v, {}, i);

%%%%% Respuesta
yp = narx_net_closed (p1, Pi1);
figure (1)
plot ([cell2mat (i (3:500))'],'b')
hold on
plot ([cell2mat (yp)'],'r')
hold off
figure (2)
plot (cell2mat (v (3:500)'), cell2mat (yp),'.')
hold on
plot (cell2mat (v (3:500)'), cell2mat (i (3:500)),'r')
hold off

%%%%% Extracción de pesos y polarizaciones
net.IW{1,1}
net.IW{1,2}
net.b{1}
net.b{2}
net.LW{2,1}
```

Apéndice C

Sección 1. *Descripción del código para la generación de la base datos.*

```
%Neural Network Nonlinear Autoregressive with Exogenous Inputs
%%%%%%%%%%%%%% Creación de la base de datos formato VHDL
%%%%%%%%%%%%%

load DCV_norm.dat                %Perfil de voltaje senoidal 500
datos
validacion_R = DCV_norm(:,2);

hex = dec2hex(bin2dec(dec18binv(validacion_R))); %Proceso
Conversión

aux=0;                                                  %Formato VHDL
fid=fopen('hex_sinc.txt','w');
for i=1:length(hex)
    if aux == 10
       fprintf(fid,'\n');
       aux=0;
    end
    fprintf(fid,'X"%s", ',hex(i,:));
    aux=aux+1;
end
fprintf(fid,'\n\n\n');
```

Sección 1.1. Generación de la cadena al tamaño de palabra.

```
function b=dec18binv(x)
l=length(x);

b=char(zeros(l,18));
for i=1:l
   aux=dec18bin(x(i));
   b(i,:)=aux;
end
```

Sección 1.2. Conversión de formato decimal a binario.

```
%Esta función Recibe un número real Menor a 512
%Retorna una cadena Binaria de 18 bits

function b=dec18bin(x)
if ischar(x)
    error('El argumento no puede ser una string');
end

if ~isreal(x)
    error('El argumento debe ser un real y menor a 16');
end

com=sign(x);                          %Verifica si es Numero
positivo
x=abs(x);
```

```
if x>=16
    error('Numero real menor a 16');


end
ent=floor(x);                   %Separa parte entera y decimal
dec=x-floor(x);

bent='';                        %Algoritmo para generar parte
entera
for k=0:4
i=4-k;
    if ent >= (2^i);
    aux='1';
    ent=ent-(2^i);
    else
    aux='0';
    ent=ent;
    end
bent=[bent,aux];
end

bdec='';                        %Algoritmo para generar parte
decimal
for i=1:13
    if dec >= 1/(2^i)
        aux='1'; dec=dec-(1/(2^i));
    else
        aux='0'; dec=dec;
    end
    bdec=[bdec,aux];
end

sal=[bent,bdec];                %Une trama de 18 bits

if com>=0                       %Si argumento de entrada es
positivo
    b=sal;
else                            %De lo contrario saca complemento
a 2
    b=sal;
    l=length(b);

for i=1:l
    if b(i)=='1'
    b(i)='0';
    else
    b(i)='1';
    end
end

ca='1';
for i=1:l
```

```
k=(l+1)-i;
if (b(k)=='1') & (ca=='1')
b(k)='0';
ca='1';
else
    if ca=='1'
    b(k)='1';
```

Apéndice D

Sección 1. *Implementación de red neuronal artificial NARX en VHDL.*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.Pack_SistemaNarx.all;

entity SistemaNarx is
port (clk, rst, we, push: in std_logic;
      tx: out std_logic);
end SistemaNarx;

architecture Behavioral of SistemaNarx is

signal clk_div, enable, etx: std_logic;
signal I: std_logic_vector (17 downto 0);

begin
----------------- COMPONENTES -------------------------
U1: Controlador port map (clk=>clk, rst=>rst, push=>push, we=>we,
clk_div=>clk_div, enable=>enable, etx=>etx);
U2: NARX port map (clk=>clk_div, ena=>enable, rst=>rst,
push=>push, sal_y=>I);
U3: Salida port map (tx=>tx, I=>I, enable=>enable, we=>we,
rst=>rst, push=>push, clk=>clk, etx=>etx);

end Behavioral;
```

Sección 1.1. Módulo de entrada (Implementación del sistema NARX).

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

entity NARX is
     port (clk,rst,push,ena : in  std_logic;
           sal_y : inout std_logic_vector (17 downto 0));
end NARX;

architecture arq_NARX of NARX is

component Entrada is
port(clk, rst, push: in std_logic;
     ent_u : out std_logic_vector (17 downto 0));
end component;

component TANSIG is
port (X : in std_logic_vector(17 downto 0);
          Y : out std_logic_vector(17 downto 0);
          Enable : in std_logic);
end component;

signal enable: std_logic;
signal ent_u,ent_u1,ent_u2,sal_y1,sal_y2 : std_logic_vector (17
downto 0);
signal SUM1_1, SUM2_1, SUM3_1, N1_2: std_logic_vector (35 downto
0);
signal N1_1, N2_1, N3_1: std_logic_vector (17 downto 0);

-- Pesos y Bias de la capa 1
constant W11_1: std_logic_vector (17 downto 0) := O"757523";
constant W12_1: std_logic_vector (17 downto 0) := O"770136";
constant W13_1: std_logic_vector (17 downto 0) := O"007267";
constant W14_1: std_logic_vector (17 downto 0) := O"007646";
constant W21_1: std_logic_vector (17 downto 0) := O"766451";
constant W22_1: std_logic_vector (17 downto 0) := O"012301";
constant W23_1: std_logic_vector (17 downto 0) := O"006571";
constant W24_1: std_logic_vector (17 downto 0) := O"001250";
constant W31_1: std_logic_vector (17 downto 0) := O"026340";
constant W32_1: std_logic_vector (17 downto 0) := O"751575";
constant W33_1: std_logic_vector (17 downto 0) := O"001773";
constant W34_1: std_logic_vector (17 downto 0) := O"003747";
constant B1_1 : std_logic_vector (17 downto 0) := O"030557";
constant B2_1 : std_logic_vector (17 downto 0) := O"764243";
constant B3_1 : std_logic_vector (17 downto 0) := O"011327";

--Pesos y Bias de la capa 2
constant W11_2: std_logic_vector (17 downto 0) := O"001415";
constant W12_2: std_logic_vector (17 downto 0) := O"023224";
constant W13_2: std_logic_vector (17 downto 0) := O"042343";
```

```
begin
--Retardo(Registro)
process (clk,rst,ent_u,ent_u1,sal_y,sal_y1)
begin
     if rst = '1' then
          ent_u1 <= "00" & X"0066";
          ent_u2 <= (others => '0');
          sal_y1 <= "00" & X"004A";
          sal_y2 <= "00" & X"0000";
     elsif (clk'event and clk ='1') then
          ent_u1 <= ent_u;
          ent_u2 <= ent_u1;
          sal_y1 <= sal_y;
          sal_y2 <= sal_y1;
     end if;
end process;

M1: Entrada port map (ena, rst, push, ent_u);

SUM1_1 <= ent_u1*W11_1 + ent_u2*W12_1 + sal_y1*W13_1 +
sal_y2*W14_1 + ("00000" & B1_1 & "0000000000000");
SUM2_1 <= ent_u1*W21_1 + ent_u2*W22_1 + sal_y1*W23_1 +
sal_y2*W24_1 + ("00000" & B2_1 & "0000000000000");
SUM3_1 <= ent_u1*W31_1 + ent_u2*W32_1 + sal_y1*W33_1 +
sal_y2*W34_1 + ("00000" & B3_1 & "0000000000000");

T1: TANSIG port map(SUM1_1(30 downto 13),N1_1,ena);
T2: TANSIG port map(SUM2_1(30 downto 13),N2_1,ena);
T3: TANSIG port map(SUM3_1(30 downto 13),N3_1,ena);


N1_2 <= N1_1*W11_2 + N2_1*W12_2 + N3_1*W13_2 + ("11111" & B1_2 &
"0000000000000");

process (ena,rst,N1_2)
begin
     if rst ='1' then
          sal_y  <= (others => '0');
     elsif (ena'event and ena ='0') then
        sal_y <= N1_2(30 downto 13);
     end if;
end process;

end arq_NARX
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;

entity Entrada is
     port(clk, rst, push: in std_logic;
        ent_u : out std_logic_vector (17 downto 0));
end Entrada;

architecture Arq_Ent of Entrada is

signal address : integer:=0;
type memoria is array (0 to 499) of std_logic_vector (19 downto
0);

signal V1: memoria:= (
                    X"000CD", X"00134", X"0019B", X"00202",
X"00269", X"002CF", X"00336", X"0039C", X"00402", X"00468",
X"004CE", X"00534", X"00599", X"005FF", X"00664", X"006C8",
X"0072D", X"00791", X"007F5", X"00858", X"008BC", X"0091E",
X"00981", X"009E3", X"00A45", X"00AA6", X"00B07", X"00B67",
X"00BC7", X"00C27", X"00C86", X"00CE4", X"00D42", X"00D9F",
X"00DFC", X"00E59", X"00EB4", X"00F10", X"00F6A", X"00FC4",
X"0101D", X"01076", X"010CE", X"01125", X"0117C", X"011D1",
X"01227", X"0127B", X"012CF", X"01322", X"01374", X"013C5",
X"01416", X"01465", X"014B4", X"01502", X"0154F", X"0159C",
X"015E7", X"01632", X"0167C", X"016C4", X"0170C", X"01753",
X"01799", X"017DE", X"01822", X"01865", X"018A8", X"018E9",
X"01929", X"01968", X"019A6", X"019E3", X"01A1F", X"01A5A",
X"01A94", X"01ACD", X"01B04", X"01B3B", X"01B70", X"01BA5",
X"01BD8", X"01C0A", X"01C3B", X"01C6B", X"01C9A", X"01CC7",
X"01CF4", X"01D1F", X"01D49", X"01D72", X"01D9A", X"01DC0",
X"01DE6", X"01E0A", X"01E2C", X"01E4E", X"01E6E", X"01E8E",
X"01EAC", X"01EC8", X"01EE4", X"01EFE", X"01F17", X"01F2F",
X"01F45", X"01F5A", X"01F6E", X"01F81", X"01F92", X"01FA3",
X"01FB1", X"01FBF", X"01FCB", X"01FD6", X"01FE0", X"01FE8",
X"01FEF", X"01FF5", X"01FFA", X"01FFD", X"01FFF", X"02000",
X"01FFF", X"01FFD", X"01FFA", X"01FF5", X"01FEF", X"01FE8",
X"01FE0", X"01FD6", X"01FCB", X"01FBF", X"01FB1", X"01FA3",
X"01F92", X"01F81", X"01F6E", X"01F5A", X"01F45", X"01F2F",
X"01F17", X"01EFE", X"01EE4", X"01EC8", X"01EAC", X"01E8E",
X"01E6F", X"01E4E", X"01E2C", X"01E0A", X"01DE6", X"01DC0",
X"01D9A", X"01D72", X"01D49", X"01D1F", X"01CF4", X"01CC7",
X"01C9A", X"01C6B", X"01C3B", X"01C0A", X"01BD8", X"01BA5",
X"01B70", X"01B3B", X"01B04", X"01ACD", X"01A94", X"01A5A",
X"01A1F", X"019E3", X"019A6", X"01968", X"01929", X"018E9",
X"018A8", X"01865", X"01822", X"017DE", X"01799", X"01753",
X"0170C", X"016C4", X"0167C", X"01632", X"015E7", X"0159C",
X"0154F", X"01502", X"014B4", X"01465", X"01416", X"013C5",
```

```
X"01374", X"01322", X"012D0", X"0127B", X"01227", X"011D1",
X"0117C", X"01125", X"010CE", X"01076", X"0101D", X"00FC4",
X"00F6A", X"00F10", X"00EB4", X"00E59", X"00DFC", X"00D9F",
X"00D42", X"00CE4", X"00C86", X"00C27", X"00BC7", X"00B67",
X"00B07", X"00AA6", X"00A45", X"009E3", X"00981", X"0091E",
X"008BC", X"00858", X"007F5", X"00791", X"0072D", X"006C8",
X"00664", X"005FF", X"00599", X"00534", X"004CE", X"00468",
X"00402", X"0039C", X"00336", X"002CF", X"00269",

X"00202", X"0019B", X"00134", X"000CD", X"00066", X"00000",
X"3FF9A", X"3FF33", X"3FECC", X"3FE65", X"3FDFE", X"3FD97",
X"3FD31", X"3FCCA", X"3FC64", X"3FBFE", X"3FB98", X"3FB32",
X"3FACC", X"3FA67", X"3FA01", X"3F99C", X"3F938", X"3F8D3",
X"3F86F", X"3F80B", X"3F7A8", X"3F744", X"3F6E2", X"3F67F",
X"3F61D", X"3F5BB", X"3F55A", X"3F4F9", X"3F499", X"3F439",
X"3F3D9", X"3F37A", X"3F31C", X"3F2BE", X"3F261", X"3F204",
X"3F1A7", X"3F14C", X"3F0F0", X"3F096", X"3F03C", X"3EFE3",
X"3EF8A", X"3EF32", X"3EEDB", X"3EE84", X"3EE2F", X"3EDD9",
X"3ED85", X"3ED30", X"3ECDE", X"3EC8C", X"3EC3B", X"3EBEA",
X"3EB9B", X"3EB4C", X"3EAFE", X"3EAB1", X"3EA64", X"3EA19",
X"3E9CE", X"3E984", X"3E93C", X"3E8F4", X"3E8AD", X"3E867",
X"3E822", X"3E7DE", X"3E79B", X"3E758", X"3E717", X"3E6D7",
X"3E698", X"3E65A", X"3E61D", X"3E5E1", X"3E5A6", X"3E56C",
X"3E533", X"3E4FC", X"3E4C5", X"3E490", X"3E45B", X"3E428",
X"3E3F6", X"3E3C5", X"3E395", X"3E366", X"3E339", X"3E30C",
X"3E2E1", X"3E2B7", X"3E28E", X"3E266", X"3E240", X"3E21A",
X"3E1F6", X"3E1D4", X"3E1B2", X"3E191", X"3E172", X"3E154",
X"3E138", X"3E11C", X"3E102", X"3E0E9", X"3E0D1", X"3E0BB",
X"3E0A6", X"3E092", X"3E07F", X"3E06E", X"3E05D", X"3E04F",
X"3E041", X"3E035", X"3E02A", X"3E020", X"3E018", X"3E011",
X"3E00B", X"3E006", X"3E003", X"3E001", X"3E000", X"3E001",
X"3E003", X"3E006", X"3E00B", X"3E011", X"3E018", X"3E020",
X"3E02A", X"3E035", X"3E041", X"3E04F", X"3E05D", X"3E06E",
X"3E07F", X"3E092", X"3E0A6", X"3E0BB", X"3E0D1", X"3E0E9",
X"3E102", X"3E11C", X"3E138", X"3E154", X"3E172", X"3E192",
X"3E1B2", X"3E1D4", X"3E1F6", X"3E21A", X"3E240", X"3E266",
X"3E28E", X"3E2B7", X"3E2E1", X"3E30C", X"3E339", X"3E366",
X"3E395", X"3E3C5", X"3E3F6", X"3E428", X"3E45B", X"3E490",
X"3E4C5", X"3E4FC", X"3E533", X"3E56C", X"3E5A6", X"3E5E1",
X"3E61D", X"3E65A", X"3E698", X"3E6D7", X"3E717", X"3E758",
X"3E79B", X"3E7DE", X"3E822", X"3E867", X"3E8AD", X"3E8F4",
X"3E93C", X"3E984", X"3E9CE", X"3EA19", X"3EA64", X"3EAB1",
X"3EAFE", X"3EB4C", X"3EB9B", X"3EBEA", X"3EC3B", X"3EC8C",
X"3ECDE", X"3ED31", X"3ED85", X"3EDD9", X"3EE2F", X"3EE84",
X"3EEDB", X"3EF32", X"3EF8A", X"3EFE3", X"3F03C", X"3F096",
X"3F0F0", X"3F14C", X"3F1A7", X"3F204", X"3F261", X"3F2BE",
X"3F31C", X"3F37A", X"3F3D9", X"3F439", X"3F499", X"3F4F9",
X"3F55A", X"3F5BB", X"3F61D", X"3F67F", X"3F6E2", X"3F744",
X"3F7A8", X"3F80B", X"3F86F", X"3F8D3", X"3F938", X"3F99C",
X"3FA01", X"3FA67", X"3FACC", X"3FB32", X"3FB98", X"3FBFE",
X"3FC64", X"3FCCA", X"3FD31", X"3FD97", X"3FDFE", X"3FE65",
X"3FECC", X"3FF33", X"3FF9A",X"00000", X"00066");
```

```
begin
      lectura: process (clk,rst)
      begin
      if (rst = '1') then
                  ent_u <= (others => '0');
            elsif (falling_edge (clk)) then
                  ent_u <= V1(address)(17 downto 0);
            end if;
      end process lectura;

      direccion: process (clk,rst,push) is
      begin
            if (rst = '1' or push = '1') then
                  address<= 0;
            elsif (rising_edge (clk)) then
                  if address < V1'high then
                        address <= address + 1;
                  else
                        address <= address;
                  end if;
            end if;
      end process direccion;
end Arq_Ent;
```

Sección 1.1.2. Código de la implementación de la función de transferencia (3 Neuronas).

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

entity TANSIG is
Port (X : in std_logic_vector(17 downto 0);
          Y : out std_logic_vector(17 downto 0);
          Enable : in std_logic);
end TANSIG;

architecture Arq of TANSIG is

constant a0:integer:= 8128; constant a1:integer:= 7568;
constant a2:integer:= 6624; constant a3:integer:= 5472;
constant a4:integer:= 4320; constant a5:integer:= 3280;
constant a6:integer:= 2432; constant a7:integer:= 1760;
constant a8:integer:= 1248; constant a9:integer:= 880;
constant aa:integer:= 624;  constant ab:integer:= 432;
constant ac:integer:= 304;  constant ad:integer:= 208;
constant ae:integer:= 208;  constant af:integer:= 96;

constant b0:integer:= 0;    constant b1:integer:= 112;
constant b2:integer:= 464;  constant b3:integer:=1120;
constant b4:integer:= 1984; constant b5:integer:= 2960;
constant b6:integer:= 3904; constant b7:integer:= 4784;
constant b8:integer:= 5552; constant b9:integer:= 6176;
constant ba:integer:= 6656; constant bb:integer:= 7056;
constant bc:integer:= 7344; constant bd:integer:= 7568;
constant be:integer:= 7552; constant bf:integer:= 7856;

constant x0:integer:= 0;    constant x1:integer:= 1536;
constant x2:integer:= 3072; constant x3:integer:= 4608;
constant x4:integer:= 6144; constant x5:integer:= 7680;
constant x6:integer:= 9216; constant x7:integer:= 10752;
constant x8:integer:= 12228;constant x9:integer:= 13824;
constant xa:integer:= 15360;constant xb:integer:= 16896;
constant xc:integer:= 18432;constant xd:integer:= 19968;
constant xe:integer:= 21504;constant xf:integer:= 23040;
constant x10:integer:= 28672;

signal abs_x:   std_logic_vector(17 downto 0);
signal latch:   std_logic_vector(17 downto 0);
signal Y_S:     std_logic_vector(17 downto 0);
signal abs_y:   std_logic_vector(35 downto 0);
signal a,b:     std_logic_vector(17 downto 0);
signal mult:    std_logic_vector (17 downto 0);

begin
abs_x<=abs(X);
process(abs_x)
```

```
begin

      if (abs_x>= X0)and(abs_x< X1) then
            a<=conv_std_logic_vector(a0,18);
            b<=conv_std_logic_vector(b0,18);
      elsif (abs_x< X2) then
            a<=conv_std_logic_vector(a1,18);
            b<=conv_std_logic_vector(b1,18);
      elsif (abs_x< X3) then
            a<=conv_std_logic_vector(a2,18);
            b<=conv_std_logic_vector(b2,18);
      elsif (abs_x< X4) then
            a<=conv_std_logic_vector(a3,18);
            b<=conv_std_logic_vector(b3,18);
      elsif (abs_x< X5) then
            a<=conv_std_logic_vector(a4,18);
            b<=conv_std_logic_vector(b4,18);
      elsif (abs_x< X6) then
            a<=conv_std_logic_vector(a5,18);
            b<=conv_std_logic_vector(b5,18);
      elsif (abs_x< X7) then
            a<=conv_std_logic_vector(a6,18);
            b<=conv_std_logic_vector(b6,18);
      elsif (abs_x< X8) then
            a<=conv_std_logic_vector(a7,18);
            b<=conv_std_logic_vector(b7,18);
      elsif (abs_x< X9) then
            a<=conv_std_logic_vector(a8,18);
            b<=conv_std_logic_vector(b8,18);
      elsif (abs_x< XA) then
            a<=conv_std_logic_vector(a9,18);
            b<=conv_std_logic_vector(b9,18);
      elsif (abs_x< XB) then
            a<=conv_std_logic_vector(aa,18);
            b<=conv_std_logic_vector(ba,18);
      elsif (abs_x< XC) then
            a<=conv_std_logic_vector(ab,18);
            b<=conv_std_logic_vector(bb,18);
      elsif (abs_x< XD) then
            a<=conv_std_logic_vector(ac,18);
            b<=conv_std_logic_vector(bc,18);
      elsif (abs_x< XE) then
            a<=conv_std_logic_vector(ad,18);
            b<=conv_std_logic_vector(bd,18);
      elsif (abs_x< XF) then
            a<=conv_std_logic_vector(ae,18);
            b<=conv_std_logic_vector(be,18);
      elsif (abs_x< X10)then
            a<=conv_std_logic_vector(af,18);
            b<=conv_std_logic_vector(bf,18);
      else

a<=(others=>'0');
            b<=O"020000";
```

```
     end if;
end process;

abs_y<=(a*abs_x)+("00000" & b & "0000000000000");

process(X,abs_y)
begin
     if X(17)='1' then
          Y_S <= O"000000"-abs_y(30 downto 13);
     else
          Y_S <= abs_y(30 downto 13);
     end if;
end process;

latch <= Y_S when(Enable = '1') else latch;
Y <= latch;
end Arq;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.Pack_SistemaNarx.all;

entity Salida is
  port(tx: out std_logic;
       I: in std_logic_vector (17 downto 0);
       enable, we, clk, rst, push, etx: in std_logic);
end Salida;

architecture Behavioral of Salida is

signal sal1: std_logic_vector (17 downto 0);

begin
----------------- COMPONENTES --------------------------
S1: MemoriaWR port map (I=>I, clk1=>enable, clk2=>etx, rst=>rst,
we=>we, push=>push, Iwr=>sal1);
S2: serial port map (I=>sal1, etx=>etx, clk=>clk, push=>push,
tx=>tx);

end Behavioral;
```

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

entity MemoriaWR is
  port(I: in std_logic_vector (17 downto 0);
       clk1, clk2, rst, we, push: in std_logic;
       Iwr: out std_logic_vector (17 downto 0));
end MemoriaWR;

architecture Behavioral of MemoriaWR is

signal address: integer:=0;
signal clk: std_logic;

--type memoria is array(0 to 399) of STD_LOGIC_VECTOR(0 to 17);
type memoria is array (0 to 501) of STD_LOGIC_VECTOR(0 to 17);
signal Ipv: memoria;

begin
with we select
     clk <= clk1 when '1',   -- Escritura  enable
               clk2 when '0',   -- Lectura    etx
                 '0'  when others;


WR: process (clk, we)
    begin
      if (rising_edge(clk)) then   -- rising
         if(we='1') then
         Ipv(address) <= I;     -- Escritura de los datos
           else
              Iwr <= Ipv(address); -- Lectura de los datos
         end if;
         end if;
end process WR;

Direccion: process(clk, rst, address, Ipv, push) is
    begin
           if (rst = '1' or push = '1') then
              address <= 0;
           elsif (falling_edge (clk)) then   -- falling
                if address < Ipv'high then
                     address <= address + 1;
                else
                     address <= address;
                end if;
           end if;
end process Direccion;

end Behavioral;
```

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

entity serial is
  port (clk, push: in  STD_LOGIC;
        I: in STD_LOGIC_VECTOR(17 downto 0);
        etx: in STD_LOGIC;
             tx: out  STD_LOGIC);
end serial;

architecture Arq of serial is

signal regtx:  STD_LOGIC_VECTOR (7 downto 0);
signal cnttx: STD_LOGIC_VECTOR (15 downto 0):= "0000000000000000";
signal ttx: STD_LOGIC_VECTOR (3 downto 0):= "0000";
signal cargador: STD_LOGIC_VECTOR (1 downto 0);
signal I1,I2,I3: STD_LOGIC_VECTOR (7 downto 0);


-------------- VELOCIDAD DE TRANSMISIÓN --------------------------
--
---Para 9600 baudios a 100MHz---
---(1/9600)/10ns  = 10416.6666
constant baudtx: STD_LOGIC_VECTOR(15 downto 0):=
"0010100010110000";                                              --
10416  10100010110000
begin

-- Proceso para cargar los datos
 process(push, etx)
  begin
      if (push = '1') then
           I1<=(others=>'0');
           I2<=(others=>'0');
           I3<=(others=>'0');

     elsif(etx'event and etx='0')then
           I1<=I(7 downto 0);
           I2<=I(15 downto 8);
           I3<="000000"&I(17 downto 16);         -- RELLENAR CON
I(17)

     end if;
 end process;

-- Reloj de transmisión
 process (clk,push,etx,cnttx,ttx)
  begin
      if (push='1' or etx='1')then
            cnttx <= (others=>'0');
```

```
ttx   <= "0000";
         cargador <= "00";
     elsif (clk'event and clk='1' and etx='0')then
           cnttx <= cnttx+1;
          if (cnttx=baudtx)then
               ttx<=ttx+1;
               cnttx<=(others=>'0');
          if (ttx="1010")then
               cargador <= cargador + 1;
               ttx <= "0000";
          end if;
          end if;
     end if;
end process;

-- Asignacion de Caracter
with cargador select
     regtx <=   I1    when "00",
                      I2    when "01",
                      I3    when "10",
                      X"00" when others;

-- Protocolo de transmisión
with ttx select
          tx    <=    '1' when "0000",
                     '0' when "0001",   --bit de start
           regtx(0) when "0010",     --inicia transmision de dato
           regtx(1) when "0011",
           regtx(2) when "0100",
           regtx(3) when "0101",
           regtx(4) when "0110",
           regtx(5) when "0111",
           regtx(6) when "1000",
           regtx(7) when "1001",     --fin de transmision de dato
                     '1' when "1010",--bit de stop
                     '1'        when others;
end Arq;
```

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

entity Controlador is
port (clk, rst, push, we: in std_logic;
      clk_div, enable, etx: inout std_logic);
end Controlador;

architecture Behavioral of Controlador is
signal cnt: std_logic_vector (4 downto 0);
signal aux: std_logic_vector (4 downto 0);
signal stop: std_logic_vector (8 downto 0);
signal contador: std_logic_vector (19 downto 0);

begin
clkdiv:process (clk, rst, clk_div, cnt)
     begin
       if (rst = '1' or cnt = "10100") then   -- 20
           clk_div <= '0';
           cnt <= (others => '0');
     elsif (clk'event and clk = '1') then
           cnt <= cnt + 1;
           if (cnt = "01001") then       --  9
            clk_div <= '1';
           else
       clk_div <= clk_div;
      end if;
     end if;
end process clkdiv;

enab:process (clk, rst, aux, clk_div)
     begin
       if (rst = '1' or aux = "10100") then   -- 20
           enable <= '1';
           aux <= (others => '0');
     elsif (clk'event and clk = '1') then
           aux <= aux + 1;
           if (aux = "00100") then  -- 4
            enable <= '0';
           else
       enable <= enable;
      end if;
     end if;
end process enab;

 process(contador, we, stop)
  begin
    if (we = '0') then
  if (contador < X"00014" or stop = "111110101") then -- 501
             etx<='1';
```

```
     else
             etx<='0';
       end if;
      end if;
end process;

-- Proceso para detener etx
process(etx,push,stop)
  begin
      if (push='1') then
         stop <= (others=>'0');
   elsif (etx'event and etx = '1') then
         stop <= stop + 1;
     end if;
end process;

process(clk, push, we, contador)
  begin
      if (push='1') then
           contador <= (others=>'0');
   elsif (clk'event and clk = '1') then
         if (contador < X"53ECB") then
             contador <= contador + 1;
           else
          contador <= (others=>'0');
         end if;
     end if;
end process;

end Behavioral;
```

Apéndice E

Sección 1. *Descripción del código para la recepción de los datos.*

```
% Programa para recibir información del FPGA
clear all;
clc;
%% Parámetros del protocolo serial y abre el puerto
s = serial('COM10');
set(s,'DataBits',8);
set(s,'StopBits',1);
set(s,'BaudRate',9600);
set(s,'Parity','none');
set(s,'Terminator','CR');
set(s,'InputBufferSize', 1506);
set(s, 'Timeout', 5);
set(s,'OutputBufferSize', 1506);

fopen(s);                           % Abre el puerto
  [a, count] = fread(s,1506);
fclose(s);                          % Cierra el puerto
```

Sección 2. *Procesamiento de los datos.*

```
% Programa para reconvertir la información
clear all;
clc;

load datosTeorico.dat;
load datosMS.dat;
load datosFPGA.dat;

a = datosFPGA;

bin = dec2bin(a);

max = length(a);
aux = 1;
for i = 3 : 3 : max
    if aux <= max
        var(aux,:) = [bin((i),:) bin((i-1),:) bin((i-2),:)];
        aux = aux + 1;
    end
end

max1 = length(var);
for i = 1 : max1
    b(i,:) = var(i,(7:24));
end

for i = 1:max1
    dec=[];
    if b(i,1)=='1'
        b(i,:)=com2bin(b(i,:));
        s(i,:) = 1;
    else
        s(i,:) = 0;
    end
```

```
    for j = 1 : -1 : -15
        if b(i,(3 - j)) == '1'
            dec = [dec 2^(j+2)];
        else
            dec = [dec 0];
        end
    end

    if s(i)==1
        dec2(i ,:) = sum(dec)*-1;
    else
        dec2(i ,:) = sum(dec);
    end
end

T = datosMS(:,1);
FPGA = dec2(:,1);

t = sum((T - FPGA).^2);

R = corr2(T,FPGA)
mse = t/500

figure(1)
plot(T)
hold on
plot(FPGA,'r')
hold off

Target = datosTeorico(:,2);
FPGA = dec2(:,1);

t = sum((Target - FPGA).^2);

R1 = corr2(Target,FPGA)
mse1 = t/500

figure(2)
plot(Target)
hold on
plot(FPGA,'r')
hold off

v=datosTeorico(:,1);
figure(3)
plot(v,Target)
hold on
plot(v,FPGA,'r')
hold off
```