UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de clasificación de fallas en sistemas eléctricos mediante un algoritmo de aprendizaje supervisado

TESIS

Que para obtener el título de

Ingeniero Eléctrico - Electrónico

P R E S E N T A N

Paola Buendía Anaya
Cristian Torres González

DIRECTOR DE TESIS

Dr. Mario Roberto Arrieta Paternina

Ciudad Universitaria, Cd. Mx., 2020

Contenido

Índice de Figuras

Índice de Tablas

Glosario de Abreviaturas

SVM – Máquina de Soporte Vectorial (*Support Vector Machine*)

FPGA – Arreglo de Compuertas Lógicas Programables (*Field-Programmable Gate Array*)

RBM – Máquina de Boltzmann Restringida (*Restricted Boltzmann Machine*)

CNN – Red Neuronal Convolutiva (*Convolutional Neural Network*)

HVDC – Corriente Directa de Alto Voltaje (*High Voltage Direct Current*)

PSD – Densidad Espectral de Potencia (*Power Spectral Density*)

TC - Transformador de corriente

TP - Transformador de potencial

IIR – Respuesta infinita al Impulso (*Infinite Impulse Response*)

FIR – Respuesta Finita al Impulso (*Finite Impulse Response*)

DFT – Transformada Discreta de Fourier (*Discrete Fourier Transform*)

FFT – Transformada Rápida de Fourier (*Fast Fourier Transform*)

ANN – Red Neuronal Artificial (*Artificial Neural Networks*)

KKT - *Karush-Kuhn-Tucker*

ADC – Convertidor Analógico Digital (*Analog to Digital Converter*)

SPI – Interfaz Periférica de Serie (*Serial Peripheral Interface)*

UART – Transmisor-Receptor Asincrono Universal (*Universal Asynchronous Receiver-Transmitter)*

ARM - *Acorn RISC Machine*

VHDL – Lenguaje de Descripción de Hardware / Circuito Integrado Alto (*Very High Integrated Circuit / Hardware Description Language)*

ROM - Memoria de sólo lectura (*Read-Only Memory*)

ASM – Máquina de Estados Algorítmica (*Algorithmic State Machine*)

FIFO – Primero Entrada - Primero Salida (*First Input – First Output*)

Agradecimientos

Quiero agradecer a las personas más importantes en mi vida, mi familia, por apoyarme y confiar en mí. A mi padre por inculcarme la idea de emprender, el valor de la perseverancia, la iniciativa y, sobre todo, a seguir mejorando día a día; a mi madre que me enseñó a ser una persona independiente, empática y a realizar mis proyectos personales con vitalidad y amor; y a mi hermano que me ha demostrado la importancia de seguir intentando hasta obtener lo deseado.

También agradezco a mi institución, por brindarme el entorno idóneo para desarrollarme académica y personalmente, además de ofrecerme la oportunidad de conocer gente maravillosa, como profesores, compañeros y amigos que me han motivado y proporcionado la pasión por la educación y la ingeniería.

Una de las personas que me acompaño durante este proceso ha sido Cristian Torres, compañero de tesis y de carrera, quién me ha brindado su amistad, conocimiento, tiempo y paciencia, quiero agradecerle por enseñarme a analizar cada detalle y siempre impulsarme a ser mejor estudiante. De la misma forma agradezco a mi asesor de tesis, el Dr. Mario Arrieta Paternina por ofrecerme la confianza y libertad suficiente para emprender este trabajo, me ha mostrado la importancia de realizar tus actividades con dedicación y convicción, además destaco mi admiración por su interés en ver crecer profesionalmente a sus alumnos. Asimismo, agradezco la ayuda económica recibida del Dr. Alejandro Zamora.

Al Dr. Giovanni Ramírez por ofrecernos su tiempo y compartir su conocimiento, su ayuda fue crucial para el desarrollo de esta tesis; gracias por mostrarme lo qué es una persona apasionada y dedicada a su trabajo e inspirarme confianza.

Gracias, a la gente con la que conviví a diario en el laboratorio de ingeniería eléctrica, en especial al ingeniero Alberto Mondragón por compartir su espacio de trabajo, por su humildad y por proporcionarme sus valiosos consejos. Agradezco a mi compañero Romel Cárdenas, por ser una persona dispuesta a brindar su ayuda e impecable en todo lo que hace.

Finalmente, quiero agradecer a la UNAM, en especial agradezco al proyecto PAPIIT con numero TA100819 y al proyecto estratégico PE-A-04 del CEMIE-Redes.

Paola Buendía Anaya

Resumen

Esta tesis tiene como objetivo desarrollar un sistema de clasificación de fallas eléctricas basado en mediciones fasoriales y en un algoritmo de aprendizaje supervisado. El resultado final es un dispositivo realizado con ayuda de un FPGA (por sus siglas en inglés, *field-programmable gate array*).

El dispositivo desarrollado realiza el procesamiento de señales de corrientes en un sistema de potencia con ayuda de la transformada discreta de Fourier y una clasificación de fallas eléctricas en tiempo real mediante la técnica de máquina de soporte vectorial (SVM, por sus siglas del inglés para *Support Vector Machine*). Ambos algoritmos mencionados se implementan en un lenguaje interpretado, como primera prueba de su funcionamiento; utilizando dos funciones en Matlab™ para obtener el modelo y etiquetas de los datos de la SVM. Posteriormente, los algoritmos son sintetizados en hardware mediante un FPGA.

El propósito es la implementación en un sistema trifásico, con fasores de corriente como valores de entrada a la SVM y el despliegue de estos datos mediante una interfaz gráfica.

1 Introducción

En este capítulo se da una visión general del tema y se presenta el estado del arte, el cual es necesario para determinar en qué aspectos se puede innovar. También se presenta la hipótesis, el planteamiento del problema, los objetivos generales y específicos que se pretenden lograr y la justificación; esta última detalla las ventajas de la implementación del proyecto.

Las fallas en líneas de transmisión desempeñan un papel clave en la confiabilidad de los sistemas de potencia, ya que su detección oportuna evita interrupciones, la suspensión de procesos industriales y/o apagones [1], [2]. Por lo tanto, es imprescindible que la detección sea rápida y eficiente para aislar el equipo que contiene fallas. Además, cada vez existe una mayor cantidad de líneas de transmisión y distribución en funcionamiento a lo largo de todo el mundo, una mala protección en estos elementos afecta a la continuidad del servicio de energía eléctrica. Por esta razón existe un gran interés en mejorar su confiabilidad ante fallas eléctricas, y para ello se requiere de su detección, clasificación y localización.

Los avances en las técnicas tales como: procesamiento de señales, inteligencia artificial, posicionamiento global y comunicaciones han permitido superar las técnicas de protección tradicionales. Asimismo, existe un mayor uso de sensores para el monitoreo de redes eléctricas, que aportan más datos y por ende permiten el desarrollo de sistemas de diagnóstico inteligente de fallas eléctricas [1], dentro de este diagnóstico se encuentra la clasificación de fallas.

Existen una gran variedad de técnicas de clasificación de fallas y entre ellas se encuentra la máquina de soporte vectorial. Una de las mayores ventajas que convirtió al algoritmo en una poderosa herramienta para la clasificación en las líneas de transmisión y sistemas de distribución es el mínimo riesgo de la presencia de falsas fallas [1].

Es importante destacar que el algoritmo de máquina de soporte vectorial es eficaz por su capacidad de ajustarse o “aprender” en un sistema que cambia, además de ser un modelo ya probado en simulación y considerado tradicional para la clasificación de señales. Al ser un método de aprendizaje supervisado se requiere la obtención de variables, por lo tanto, la técnica se vuelve más efectiva con variables más específicas.

1.1 Estado del arte

Se revisó la literatura existente sobre temas relacionados con clasificadores de fallas eléctricas que utilicen algoritmos de aprendizaje supervisado y principalmente acerca de clasificadores usando algoritmos de máquina de soporte vectorial con el propósito de encontrar antecedentes que ayuden al desarrollo de este proyecto. Entre la revisión realizada se destacan los siguientes trabajos:

El documento titulado “Detección y localización de fallas en los sistemas de energía mediante la técnica máquinas de soporte vectorial” del año 2006 realizado por Rodríguez Suárez [2], presenta un modelo de localización de fallas para sistemas de distribución, adaptando la SVM con distintos tipos de Kernel y métodos de optimización para determinar la metodología más adecuada.

La publicación presentada en el año 2008 con título “Ubicación única de fallas en sistemas de distribución por medio de zonas con SVM”, realizada por Morales España et al. [3], describe una metodología para la determinación de fallas en sistemas de distribución efectuando simulaciones en Matlab™ para el manejo de la SVM, con la determinación de descriptores y dividiendo al sistema eléctrico en zonas.

En [1], se realiza la revisión de métodos usados para detección, clasificación y localización de fallas en líneas de transmisión y distribución, describiendo al algoritmo de máquina de soporte vectorial como un modelo clásico para la clasificación de fallas y presentando nuevos métodos como las máquinas de Boltzmann restringidas (RBN) y redes neuronales convolutivas (CNN). Otros trabajos dedicados a la clasificación de fallas en líneas de transmisión son en [4] que cumple este objetivo mediante el procesamiento de fasores de voltaje y corriente. En [5], se propone un algoritmo de clasificación que consigue la identificación de fallas por medio de las secuencias cero y negativa.

En el año 2018, se encontraron varios documentos, uno de ellos, es el titulado “Transient signal identification of HVDC transmission lines based on wavelet entropy and SVM”, realizado por: Guomin Luo et al. En el cual se propone un método que identifica tres tipos de señales transitorias usando la máquina de soporte vectorial y la transformada de wavelet. Los resultados de la simulación demuestran que la identificación de las señales es confiable, además de tener una fuerte adaptabilidad en comparación al método tradicional de clasificación de señales basado en umbrales establecidos [6].

El siguiente documento, presentado en el 2018 por Daniel Guillén et al., “Fault detection and classification in transmission lines based on a PSD index”, se realiza la clasificación de fallas en líneas de transmisión usando un método basado en la densidad espectral de potencia (PSD) y machine-learning, entre estos clasificadores se encuentra la SVM. Los resultados demostraron que la SVM en conjunto con el método detecta y clasifica las fallas correctamente en redes radiales [7].

La identificación de fallas también es un procedimiento de interés abordado en técnicas difusas en [8], [9], [10] cuya principal característica es no precisar de procesos de entrenamiento. Asimismo, para la detección y clasificación de fallas se ha aplicado extensamente las Redes Neuronales Artificiales (RNA), donde se requiere un proceso de entrenamiento y un conjunto de información de un ciclo para la clasificación de eventos [11], [12], [13].

Lo anterior demuestra que distintas técnicas han sido propuestas, sin embargo, sólo algunas de ellas se han implementado como en [14], [15], [16]. Por ejemplo, se presenta una implementación de hardware con entrenamiento on-line en [14], lo que demuestra que la técnica SVM se ejecuta de manera más eficiente. Mientras que en [15], se propone un entrenamiento en tiempo de ejecución utilizando un sistema con FPGA que es capaz de re-entrenar, pero que sólo funciona eficientemente con señales simuladas. En [16], se plantea la importancia en el entrenamiento de la SVM para una correcta implementación y su capacidad al crearse arquitecturas escalables, permitiendo el cálculo de distintas funciones Kernel para sistemas complejos. Esto se consigue utilizando múltiples instancias en FPGA para aprovechar el procesamiento paralelo.

Como se logra observar en la literatura revisada, distintas técnicas han sido propuestas, sin embargo, sólo algunas de ellas se han implementado. Esto se debe a los desafíos de hardware que implica este proceso y la cantidad de datos requeridos como entrenamiento en la SVM. Este trabajo tiene como propósito contribuir al desarrollo e implementación en tiempo real de algoritmos de estimación fasorial en conjunto con algoritmos de aprendizaje supervisado, además de aportar mayor rapidez en los tiempos de procesamiento y la mejora en el diseño de arquitecturas de hardware que cumplan con los requerimientos específicos de las señales reales en sistemas de potencia.

1.2 Planteamiento del problema

Los sistemas de potencia día tras día se continúan modernizando y por esta razón están sujetos a condiciones cambiantes que generan la necesidad de dispositivos de protección que se adapten a estos cambios en el sistema.

Por consiguiente, la adaptabilidad en un dispositivo de protección es sumamente importante. Con una mejor adaptabilidad hace posible un nivel más alto de protección, dando como resultado una mejora en sensibilidad y selectividad [17].

Hoy en día se busca esta característica de adaptación mediante distintos algoritmos de decisión, entre ellos se incluyen la inteligencia artificial como una solución. Sin embargo, se continúa buscando mejorar los métodos reportados y proponer nuevos para alcanzar este fin.

1.3 Hipótesis

- Si se desarrolla un dispositivo clasificador de fallas con algoritmos de DFT y SVM entonces se puede obtener la adaptabilidad requerida en un dispositivo de protección.

- Si la implementación del clasificador se realiza en FPGA entonces se puede aprovechar su característica de paralelismo para una clasificación más rápida y eficiente.

1.4 Objetivos de la tesis

1.4.1 Objetivo general

Desarrollar un sistema de clasificación de fallas eléctricas basado en mediciones fasoriales y en el algoritmo de aprendizaje supervisado de máquinas de soporte vectorial, con el fin de lograr su implementación en un sistema embebido basado en FPGA.

1.4.2 Objetivos específicos

- Diseñar la arquitectura del sistema de clasificación de fallas eléctricas con la característica de adaptabilidad mediante mediciones fasoriales y una máquina de soporte vectorial.

- Simular distintos tipos de fallas eléctricas con ayuda del software PSCAD™/EMTDC™ con el propósito de obtener los conjuntos de datos correspondientes a las magnitudes fasoriales de corriente.

- Analizar los datos de las simulaciones y determinar conjuntos de validación y entrenamiento para la creación del modelo de la SVM.

- Modelar el sistema de clasificación de magnitudes de fasores de corriente mediante una SVM y evaluar su desempeño utilizando el software Matlab™, lo anterior con el fin de utilizar el modelo resultante para su implementación en FPGA.

- Analizar las variables del modelo SVM que servirán para ser sintetizados en el FPGA.

- Implementar el método de estimación fasorial de la transformada discreta de Fourier y el modelo de la SVM mediante su codificación algorítmica en un sistema embebido basado en FPGA, esto con el propósito de desarrollar un sistema de clasificación de fallas que permita discriminar su tipo en tiempo real.

1.5 Justificación

La mayoría de los métodos de clasificación adoptan modelos basados en la teoría del aprendizaje supervisado. Sin embargo, SVM tiene una alta precisión y requiere un menor número de muestras para el entrenamiento y posterior clasificación en comparación con otras técnicas, además de garantizar la solución óptima global y una mayor flexibilidad en problemas no lineales al permitir el uso de distintos tipos de Kernel [18].

Por otro lado, la ventaja de desarrollar un prototipo en una FPGA, es su característica de ser reprogramables, lo cual da cierta versatilidad en tareas específicas. Otra particularidad es el procesamiento de comandos simultáneos y ejecución de numerosas líneas de códigos.

1.6 Contribución

Esta tesis tiene como principal contribución el desarrollo de un sistema de clasificación de fallas en tiempo real mediante estimación fasorial y un algoritmo de aprendizaje supervisado, capaz de efectuar el procesamiento de las señales a mayor velocidad que un microcontrolador y confiabilidad bajo diferentes tipos de fallas.

Artículo de conferencia: Los resultados adquiridos en esta tesis han sido presentados en el siguiente artículo,

Paola Buendía, Cristian Torres, M. R. A. Paternina, A. Zamora, "*A real-time FPGA-embedded faults classification system powered by phasor estimation and supervised learning algorithms*" sometido a: 52th North American Power Symposium (NAPS), Junio 2020.

En el siguiente capítulo se exhiben los conceptos generales necesarios para entender cómo realizar la implementación del dispositivo, estos temas incluyen información sobre un sistema de potencia, relevadores digitales y la teoría sobre la SVM.

2 Conceptos generales

En este capítulo se presenta la teoría sobre fallas eléctricas, pero antes de eso, es útil conocer cómo se conforma y divide un sistema eléctrico de potencia. Más adelante se exponen los conceptos relacionados con un relevador digital, esto es necesario ya que la clasificación de fallas es una función que posee un relevador. Se continúa mostrando definiciones matemáticas para entender la transformada discreta de Fourier, paso importante para el procesamiento de la señal que nos permite finalmente implementar la clasificación de las señales mediante una SVM.

2.1 Sistema eléctrico de potencia

Para realizar el sistema de clasificación de fallas, primero se deben conocer las variables a clasificar, en este caso, fallas eléctricas en un sistema de potencia.

El conjunto de elementos como generadores, transformadores, líneas y cargas interactuando entre sí permiten el suministro de energía eléctrica requerida que las cargas conectadas demandan a éste. El sistema eléctrico se puede dividir en 4 partes, ilustradas en la **Figura 2-1** Partes de un sistema eléctrico de potencia, y puede presentar diferentes niveles de tensión, tal como se ilustran para el caso mexicano.

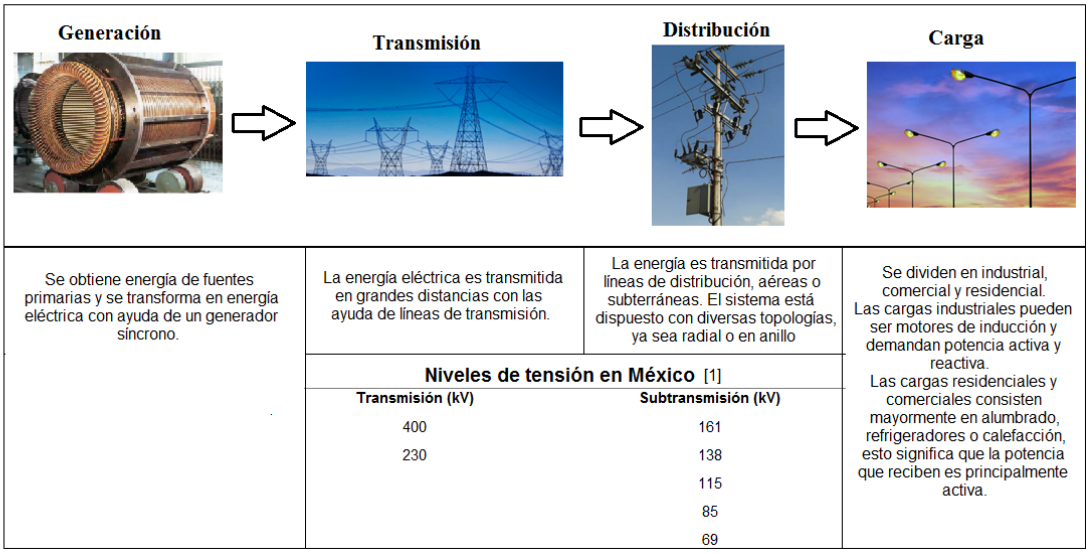


Figura 2-1.Partes de un sistema eléctrico de potencia

2.2 Fallas eléctricas en los sistemas de potencia

Las fallas eléctricas o desequilibrios en los sistemas de potencia es cualquier evento que interfiere con el flujo normal de corriente. Pueden ser en paralelo o en serie con la línea. La falla en derivación es producida debido a cortos circuitos. Mientras que la falla en serie es un evento donde la impedancia de las tres fases no es igual [19].

2.2.1 Fallas en serie

Este tipo de falla es provocada por conductores rotos en varias fases, pero frecuentemente sucede en una sola fase. En la **Figura 2-2** se muestra el ejemplo de este tipo de falla para la fase a. La corriente en la fase fallada tendrá un valor de cero y la corriente en las fases no falladas aumentarán con base a la potencia demandada de la carga.

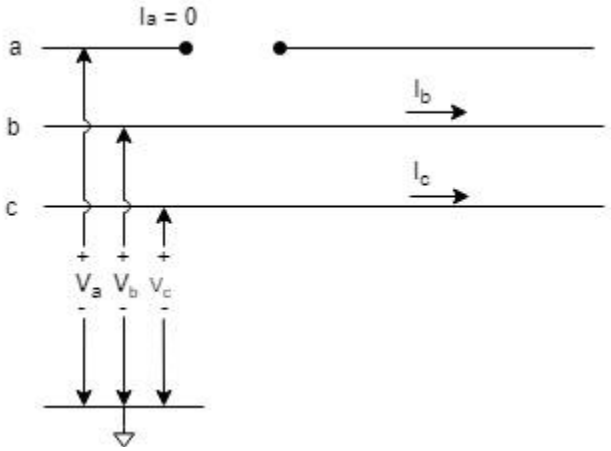


Figura 2-2. Falla en serie

2.2.2 Fallas en derivación

Estas fallas tienen un comportamiento variable a lo largo del tiempo donde corrientes y tensiones se modifican, teniendo en cuenta que los valores mayores de corriente se encuentran en los primeros ciclos del disturbio. Además, la magnitud de la corriente de falla depende de una impedancia interna en los generadores, la impedancia en el sistema y una impedancia de falla. La principal característica de fallas está relacionada con las impedancias involucradas durante la falla. Asimismo, se conoce que la impedancia del generador no es constante bajo una situación de corto circuito, esto provoca que existan los siguientes tres espacios de tiempo durante la falla: sub-transitorio, transitorio y finalmente el periodo estable.
Las fallas en derivación se pueden dividir en dos tipos: simétricas y asimétricas.

2.2.2.1 Fallas eléctricas simétricas

La falla eléctrica simétrica [20] se define como el corto circuito que ocurre a través de las tres fases de manera simultánea, conocida como falla trifásica, presentada en **Figura 2-3** . Las corrientes en las tres fases contienen la misma magnitud y están desplazadas entre sí 120°, por esta razón son llamadas fallas simétricas. Es inhabitual que ocurra. Sin embargo, es el tipo de falla más severo al involucrar las tres fases

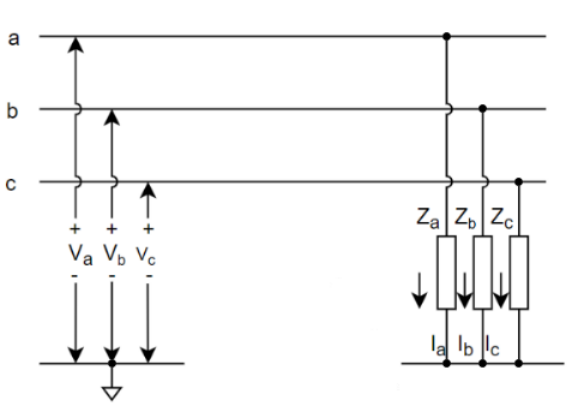


Figura 2-3.Falla trifásica.

$$Z = Z_a = Z_b = Z_c, \tag{2-1}$$

$$V = V_a = V_b = V_c, \tag{2-2}$$

$$I_a = I_b = I_c = \frac{V}{Z_{eq}} + Z, \tag{2-3}$$

donde I_a, I_b, I_c, son las corrientes de falla de cada fase, V_a, V_b, V_c los voltajes de falla de cada fase, Z_{eq} representa la impedancia equivalente de Thevenin del sistema y Z_a, Z_b, Z_c es la impedancia de falla en cada fase.

Los tipos de fallas en derivación asimétricas se muestran a continuación.

2.2.2.2 Fallas eléctricas asimétricas

Falla bifásica: El corto circuito ocurre entre dos de las tres fases del sistema, esto se observa en la **Figura 2-4**.

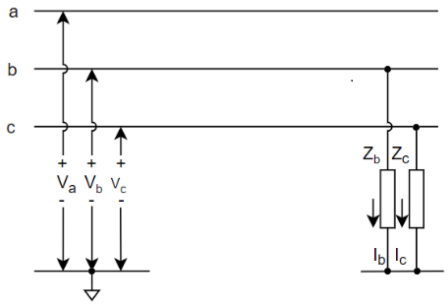


Figura 2-4. Falla bifásica.

Si la falla ocurre entre fase b y c, se tiene una corriente de falla igual a cero en la fase a, $I_a = 0$. En contraste, existirá una corriente de falla I_b e I_C definidas de la siguiente manera:

$$I_b = \frac{V_b - V_c}{Z_b + Z_c} \quad y \tag{2-4}$$

$$I_c = -I_b. \tag{2-5}$$

Falla bifásica a tierra: La unión ahora se presenta entre dos fases y la tierra. Esto se percibe en la **Figura 2-5**. En la configuración el corto circuito surge entre la fase b, c y tierra.

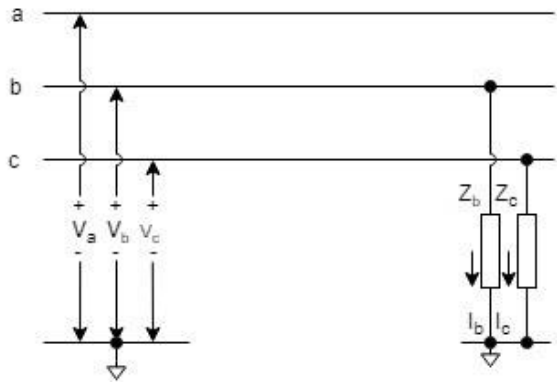


Figura 2-5. Falla bifásica a tierra.

Donde las corrientes tendrán los siguientes valores:

$$I_a = 0, \tag{2-6}$$

$$V_b = V_c = (I_c + I_b)(Z_a || Z_b). \tag{2-7}$$

Falla monofásica: Una de las tres fases sufre una conexión con la tierra como se ilustra en la **Figura 2-6**. Si la falla ocurre en la fase a, entonces existirá una corriente de falla en esta fase. Por el contrario, en la fase b y c la corriente de falla tendrá un valor de cero. Tenemos que:

$$I_a = \mathrm{V_a} * \quad y \tag{2-8}$$

$$I_c = I_b = 0. \tag{2-9}$$

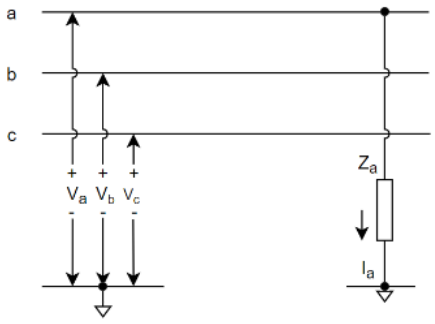


Figura 2-6. Falla monofásica.

2.2.3 Niveles de cortocircuito

Una vez, definidas las fallas eléctricas, es importante conocer los niveles de cortocircuito para advertir los valores reales que se tendrían que manejar. En el caso de México, la información sobre los niveles de cortocircuito de acuerdo con cada nivel de tensión se muestra a continuación [21]:

Tabla 2-1. Información sobre niveles de cortocircuito zona centro.

		Valores proyectados zona centro 2021			
Nivel de tensión [kV]	**No. Eventos**	**Máxima nivel de corto circuito [kA]**		**Mínimo nivel de corto circuito [kA]**	
		1F	**3F**	**1F**	**3F**
69-85	345	32.04	15.72	1.70	1.25
115	2,206	38.91	35.28	1.37	0.81
230	513	53.31	59.13	10.29	7.61
400	167	37.18	37.78	12.70	9.53

2.3 Relevadores digitales

Con la finalidad de lograr un buen diseño en el clasificador y comprender los pasos y métodos a utilizar, se revisa la arquitectura de los relevadores digitales, donde una de sus funciones es la clasificación de fallas eléctricas.

El relevador digital es un elemento esencial para la protección de sistemas eléctricos. Se define como un instrumento lógico que detecta cualquier cambio de la señal que recibe. Cuando la magnitud de la señal de entrada es distinta a un valor preestablecido, el relevador lo detectará y realizará una decisión de "disparo".

La estructura general del relevador digital se muestra en la **Figura 2-7** [22].

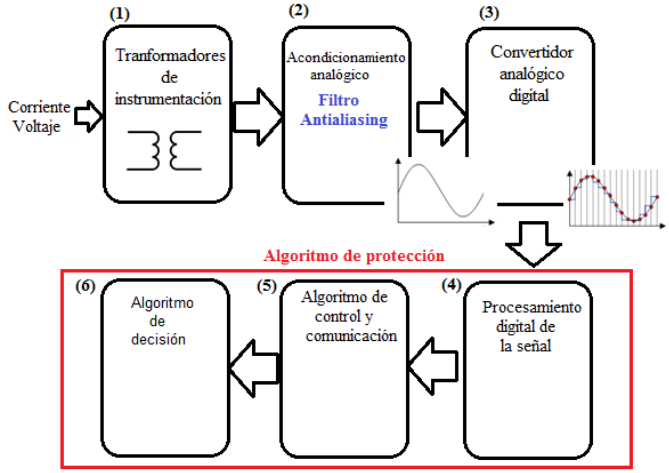


Figura 2-7. Estructura del relevador digital.

A continuación, se describe brevemente los elementos de la estructura del relevador digital de la **Figura 2-7.**

- Transformadores de instrumentación - Detecta la corriente y el potencial a través de un transformador de corriente (TC) y un transformador de potencial (TP), respectivamente.
- Filtro Antialasing – Filtro paso bajas que elimina frecuencias superiores a la una frecuencia determinada.
- Convertidor analógico digital - Estas muestras digitalizadas de corriente y tensión son proporcionadas a los microprocesadores del relevador.
- Procesamiento de la señal – En esta etapa se realiza un acondicionamiento digital de la señal con los filtros digitales IIR y FIR. También se obtienen componentes de la señal para el cálculo de valores, con ayuda de transformadas como: la transformada discreta de Fourier (DFT), o su implementación algorítmica como la FFT u otros métodos como la DTTFT, transformada discreta de Fourier recursiva, etc.
- Control y comunicación – Abarca entre otras funciones, el control de interruptores del circuito, control de alimentadores, el registro de valores medidos, comunicación con otros relevadores, etc.

2.4 Algoritmo de protección

Como se mencionó en la **Figura 2-7**, el algoritmo de protección se divide en un algoritmo de procesamiento, comunicación y decisión. Este trabajo se enfoca en los algoritmos de procesamiento y decisión.

El método de procesamiento más utilizado es la transformada discreta de Fourier, explicado en el siguiente capítulo. El funcionamiento del algoritmo de decisión se logra de distintas maneras, generalmente se crea con ayuda de esquemas lógicos que proporcionan los ajustes de funcionamiento del relevador. Dependiendo del tipo de protección se determinarán tipos de entrada y salidas, además de su lógica.

Por ejemplo: en el caso de un relevador de sobre corriente temporizado (51), según ANSI [23] se configura determinando umbrales de las corrientes de cada fase, además de retardos de tiempo si la aplicación lo requiere. La **Figura 2-8** ilustra un esquema lógico de este tipo de protección en un relevador SEL 551.

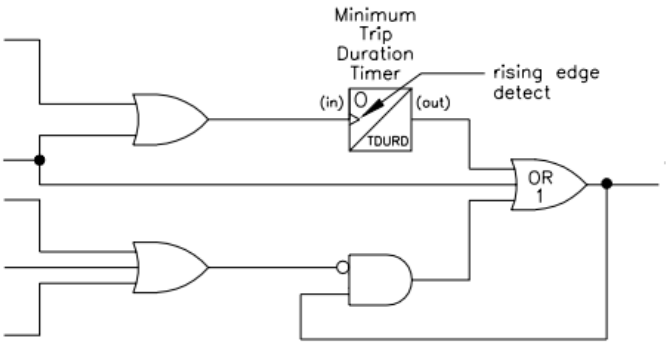


Figura 2-8. Esquema lógico de protección tipo (51) [24].

Además de los esquemas lógicos, otras técnicas utilizadas para la instrucción de clasificación de fallas contenida en el algoritmo de decisión se presentan a continuación:

- Lógica Fuzzy - La mayoría de las decisiones de los relevadores son de tipo discreto (0-1). No hay casos intermedios, es decir, nada a medio camino entre 0 y 1. En muchos casos, sin embargo, tomar la decisión apropiada es difícil debido a muchas razones y a veces la expresión "no sé" sería la mejor respuesta, si no se tiene seguridad o el valor medido está muy cerca de la frontera de la decisión.
- Redes Neuronales Artificiales – Este método sirve como predictor del rendimiento del sistema. Las redes neuronales poseen una característica de generalización para la identificación de patrones. Las ANN sirven para tareas de protección y control difíciles en sistemas eléctricos.
- Esquemas de decisión adaptiva - Es un proceso de ajuste automático a las condiciones cambiantes del sistema de potencia. La condición que permite esto, es la retroalimentación del relé con las señales anteriores en el procedimiento [22].

2.5 Transformada discreta de Fourier

La transformada discreta de Fourier se utiliza comúnmente para la obtención de la estimación del fasor.

Es importante recordar la importancia del fasor en los sistemas eléctricos de potencia, ya que el fasor es un ente matemático que permite obtener el cálculo de la magnitud y fase de una señal instantánea de voltaje o corriente que ayudan al análisis de fallas, y se define de la siguiente manera [25]:

Se considera una señal senoidal pura:

$$x(t) = x_m sen(\omega t + \varphi), \qquad (2\text{-}10)$$

donde $x(t)$ es la ecuación en el tiempo que representa una señal senoidal, x_m es la amplitud de esta señal, ω representa la frecuencia angular en [rad/s] y ϕ es ángulo de fase.

La ecuación (2-10) se puede expresar en el campo de los reales como se muestra en la siguiente ecuación

$$x(t) = \mathbb{R}e\{x_m cos(\omega t + \varphi) + jx_m sen(\omega t + \varphi)\}. \quad (2\text{-}11)$$

De acuerdo con la identidad de Euler, la ecuación (2-11) se reduce de la siguiente manera

$$x(t) = \mathbb{R}e\left\{x_m e^{j\varphi} e^{j\omega t}\right\}. \quad (2\text{-}12)$$

Finalmente, el fasor se representa como se muestra en (2-13) y (2-14). Donde X representa el fasor de la señal senoidal y X_{rms} es el fasor en valor rms

$$X = x_m e^{j(\varphi+\omega t)} \text{ ó} \quad (2\text{-}13)$$

$$X_{rms} = \frac{x_m}{\sqrt{2}} e^{j(\varphi+\omega t)}. \quad (2\text{-}14)$$

Dado que la definición de fasor sólo es para senoides puras y en la realidad, una señal de corriente en un sistema de potencia es afectada con señales de diferentes frecuencias llamadas armónicas, se utiliza la teoría de Fourier para poder "estimar" el comportamiento del fasor en el dominio de la frecuencia y ya que este fasor estimado es el resultado de una señal de entrada discreta, se utiliza como herramienta la transformada discreta de Fourier,

La DFT se representa mediante sus ecuaciones de síntesis (2-15) y de análisis (2-16) [26]:

$$\hat{x}_{[Nx1]} = \mathrm{W}_{[NxN]} \bullet \hat{\xi}(k)_{[Nx1]}, \quad (2\text{-}15)$$

$$\hat{\xi}(k)_{[Nx1]} = \mathrm{W}_{[NxN]}{}^{-1} \bullet x_{[Nx1]}, \quad (2\text{-}16)$$

donde N es el número de muestras por ciclo, $W_{[NxN]}$ representa la matriz de Fourier, $x_{[Nx1]}$ es la ventada de datos de la señal de entrada con valores de $n = 0,1,2,\dots,N-1$, $W_{[NxN]}{}^{-1}$ es la representación de la matriz inversa de Fourier y $\hat{\xi}(k)_{[Nx1]}$ son los coeficientes estimados de Fourier con valores de $k = 0,1,2,\dots,N-1$ [27].

La matriz de Fourier se expresa de la siguiente manera [28]:

$$\mathrm{W}_{[NxN]} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix}. \quad (2\text{-}17)$$

Los términos ω se nombran "factores de fase" y se determinan como:

$$\omega_N^k = e^{j\frac{2\pi k}{N}}. \quad (2\text{-}18)$$

Finalmente, al realizar la multiplicación entre la matriz inversa de Fourier y la ventana de datos se obtiene $\hat{\xi} \in \mathbb{C}$ y la información que contiene es:

$$\xi[k] = \begin{bmatrix} \hat{\xi}_0 \\ \hat{\xi}_1 \\ \hat{\xi}_2 \\ \hat{\xi}_3 \\ \hat{\xi}_k \end{bmatrix} \begin{matrix} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{matrix} \begin{matrix} Componente\ DC \\ Componente\ fundamental \\ Primer\ armónica \\ Segunda\ armónica \\ Armónica\ "k" \end{matrix} \quad (2\text{-}19)$$

En este trabajo se obtiene exclusivamente el dato estimado $\hat{\xi}_1$ que contiene la información de la componente fundamental, en capítulos posteriores se explica cómo reducir la implementación de la estimación fasorial algorítmicamente [26]. En las ecuaciones (2-20) y (2-21) se muestra finalmente como obtener los estimados de amplitud ($\hat{a}$) y fase ($\hat{\varphi}$), en este trabajo solo se utilizará el valor de la magnitud

$$\hat{a} = \left|\hat{\xi}_1\right|, \quad (2\text{-}20)$$

$$\hat{\varphi} = arctan\left(\hat{\xi}_1\right). \quad (2\text{-}21)$$

2.6 Generalidades de la SVM

Prosiguiendo con la teoría de la SVM para el desarrollo del algoritmo de decisión. La SVM tiene como fundamento la teoría del aprendizaje estadístico y fue introducida en los años 90 por Vapnik y sus colaboradores. Inicialmente fueron creadas para resolver problemas de clasificación binaria, pero hoy en día se usan para resolver problemas más complejos como: regresión, agrupación, multi-clasificación, entre otros [29].

La SVM funciona mediante aprendizaje supervisado, esto significa que requiere de dos conjuntos de datos: entrenamiento y validación. Ambos conjuntos son grupos de datos que entran al algoritmo de la SVM, la diferencia radica en que el conjunto de entrenamiento contiene los resultados de la clasificación, llamados etiquetas, esto con la finalidad de que la SVM sea entrenada. Mientras que el grupo de validación al no poseer las etiquetas servirá para verificar que el modelo resultante responda de la manera esperada, prediciendo la clasificación en base a su algoritmo.

Para lograr esta clasificación de datos se requiere de un separador. Las SVMs pertenecen a la categoría de clasificadores lineales, ya que utilizan separadores lineales, también conocidos como hiperplanos; ya sea en el espacio original de los datos, si éstos son separables o cuasi-separables, o en un espacio transformado, si los ejemplos no son separables linealmente en el espacio original. Como se muestra en la **Figura 2-9**. Particularmente, en el último caso mencionado, la búsqueda del hiperplano de separación en estos espacios transformados, normalmente de muy alta dimensión, se hará de forma implícita utilizando las denominadas funciones kernel [30] [31].

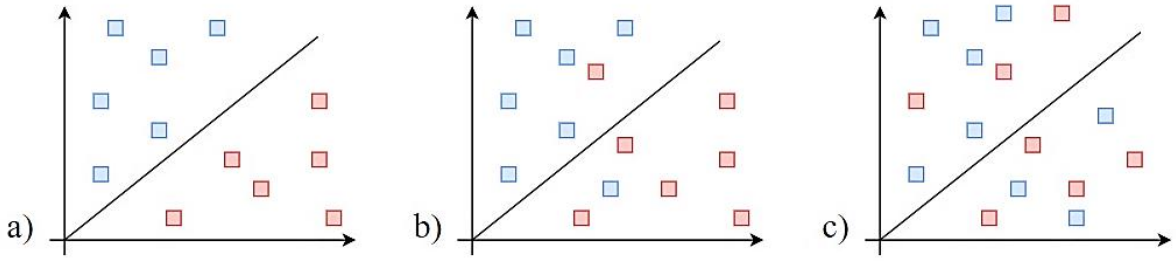


Figura 2-9. Casos de clasificación. a) Separable; b) cuasi-separable; y c) no separable.

La manera de encontrar un hiperplano de separación óptimo se alcanza seleccionando los datos más cercanos de cada muestra a éste, llamados vectores de soporte.

Desde un punto de vista algorítmico, hallar este hiperplano representa un problema de optimización, ya que existen un número infinito de ellos, pero el seleccionado será aquel que tenga la máxima distancia entre el hiperplano y los vectores de soporte, como se muestra en la **Figura 2-10**.

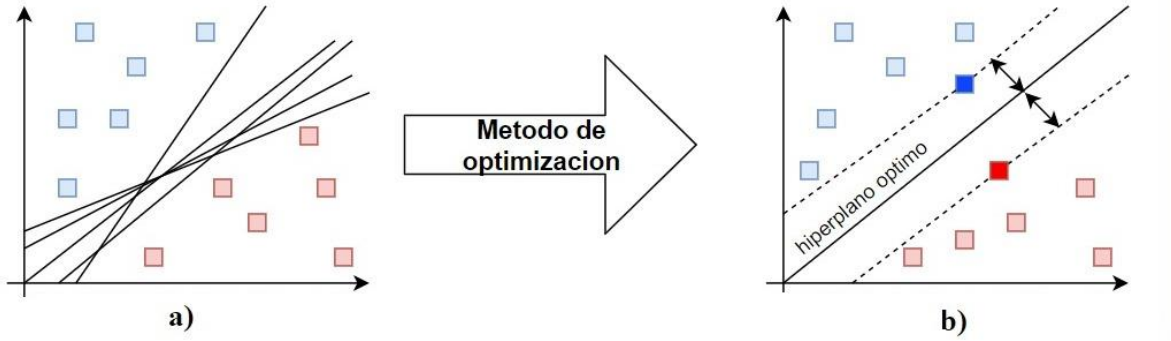


Figura 2-10. Hiperplano de separación en un conjunto de datos separables. a) Posibles hiperplanos; y b) hiperplano óptimo.

2.7 Fundamentación matemática de la SVM

Para resolver el problema de optimización se requieren dos tipos de funciones: función objetivo y funciones de restricción. La función objetivo en este caso representa el hiperplano a optimizar y las funciones de restricción son los puntos de referencia donde se comienza a optimizar la función objetivo. A continuación, se presentan los pasos para la definición de estos elementos.

Función del hiperplano: Con el objetivo de facilitar el análisis matemático, solo se presentará la clasificación binaria cuando los datos son separables.

Comenzando con un conjunto de datos $s = \{(x_1, y_1), \dots, (x_n, y_n)\}$, donde $x_i \in R^2$ y $y_i \in \{-1,1\}$, en otras palabras x_i son los datos y cado uno tiene asociado una

etiqueta y_i .Se procede definiendo la ecuación de un hiperplano en R^3, la cual se puede representar como [30]:

$$w_1x_1 + w_2x_2 + w_3x_3 + b = 0\,, \qquad (2\text{-}22)$$

donde: w_1, w_2, w_3 son los valores del vector normal al plano, x_1, x_2, x_3 son los valores del vector de posición de un punto en el plano y b es una contante de intersección.

La ecuación (2-22) se puede presentar de la siguiente manera:

$$(w_1w_2, w_3) \bullet (x_1, x_2, x_3) + b = 0. \qquad (2\text{-}23)$$

Simplificando tenemos que:

$$D(x) = \boldsymbol{w} \bullet \boldsymbol{x} + b = 0\,, \qquad (2\text{-}24)$$

donde: la función $D(x)$ provee como resultado la información necesaria para inferir la etiqueta del dato x. Si $D(x) \geq 1$ el dato pertenece a una clase, si $D(x) \leq -1$ el dato pertenece a la clase contraria. Los nombres de las variables se renombran de la siguiente manera [30]:

Donde: w representa un vector de pesos, x un vector de datos y b es una contante de ajuste.

Si $D(x) = 0$ nos encontramos sobre el hiperplano. Conforme nos alejamos de éste, los valores crecen, como se muestra a continuación. En el ejemplo de **Figura 2-11**. tenemos dos tipos de clases, clase a y clase b, representados mediante cuadros rojos y azules, respectivamente.

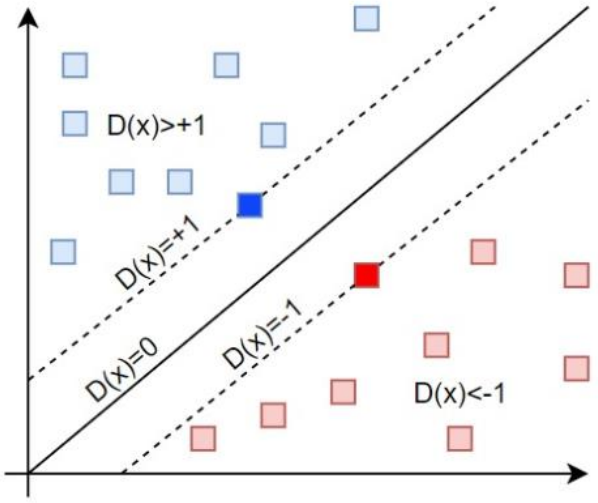


Figura 2-11. Función del hiperplano en un conjunto de datos separables.

Función objetivo: El siguiente paso es determinar la distancia máxima entre el hiperplano y los datos del vector soporte de cada clase, utilizando la ecuación de un plano a un punto obtenemos que [30]:

$$d = \frac{|\boldsymbol{w} \bullet \boldsymbol{x} + b|}{\| \boldsymbol{w}\|}, \tag{2-25}$$

donde: d es la distancia entre el hiperplano y un punto, $\| \mathbf{w}\|$ es el módulo del vector de pesos y $|\mathbf{w} \bullet \mathbf{x} + \mathrm{b}|$ es el valor absoluto de la función evaluada en un conjunto de puntos.

Usando de referencia los datos del vector soporte tenemos que el valor de (2-24) es igual a uno, por lo tanto:

$$d = \frac{1}{\| \boldsymbol{w}\|}, \tag{2-26}$$

El valor de $\| \boldsymbol{w}\|$ es inversamente proporcional a d, por lo tanto:

$$max\ d \xrightarrow{equivale\ a} min\|\boldsymbol{w}\|\ , \tag{2-27}$$

El módulo del vector $\boldsymbol{w}$ de la expresión anterior también se puede expresar como:

$$min\|\boldsymbol{w}\| \xrightarrow{equivale\ a} min\ \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}. \tag{2-28}$$

La expresión (2-28) se modifica con la finalidad de obtener una función coherente para el uso de la optimización cuadrática. Por lo tanto, la función a optimizar resultante es [30]:

$$f(\boldsymbol{w}) = \frac{1}{2}\ \boldsymbol{w} \cdot \boldsymbol{w}. \tag{2-29}$$

Funciones de restricción: Una vez definida la función objetivo, es necesario la definición de una función restricción. Utilizamos los vectores de soporte como las restricciones de la función del hiperplano $D(x)$, por lo tanto:

$$\boldsymbol{x} \cdot \boldsymbol{w} + b \geq 1\ , \tag{2-30}$$

$$\mathrm{y_i} = 1, \tag{2-31}$$

$$\boldsymbol{x} \cdot \boldsymbol{w} + b \leq -1\ , \tag{2-32}$$

$$\mathrm{y_i} = -1, \tag{2-33}$$

donde: $\boldsymbol{y}$ es el vector de etiquetas.

Al realizar la multiplicación de (2-30) con (2-31) y (2-32) con (2-33) obtenemos la expresión (2-34):

$$y_i\ (\boldsymbol{x} \cdot \boldsymbol{w} + b) \geq 1. \tag{2-34}$$

Por lo tanto, el problema de optimización se plantea con las siguientes expresiones [30]:

$$funcion\ objetivo \quad \rightarrow \quad f(\boldsymbol{w}) = \frac{1}{2}\ \boldsymbol{w} \cdot \boldsymbol{w}\ , \tag{2-35}$$

$$funcion\ restriccion \quad \rightarrow \quad y_i(\boldsymbol{x} \cdot \boldsymbol{w} + b) - 1 \geq 0. \tag{2-36}$$

Minimización de función objetivo: Obtenidas las funciones de objetivo y restricción el siguiente paso es la aplicación del método de los multiplicadores de Lagrange [30]:

$$L(\boldsymbol{w},b,\lambda) = \frac{1}{2}\ \boldsymbol{w}\cdot\boldsymbol{w} - \sum_{i=1}^{n}\lambda_i\,(y_i(x_i\cdot\boldsymbol{w}+b)-1)\,, \tag{2-37}$$

donde: λ representa los multiplicadores de Lagrange y $\lambda_i \geq 0$.

Realizando una generalización del método de los multiplicadores de Lagrange mediante las condiciones de Karush-Kuhn-Tucker (KKT), se obtiene [30]:

$$\frac{\partial L(\boldsymbol{w},b,\lambda)}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n}\lambda_i\, y_i x_i = 0\,, \tag{2-38}$$

$$\frac{\partial L(\boldsymbol{w},b,\lambda)}{\partial \boldsymbol{w}} = \sum_{i=1}^{n}\lambda_i\, y_i = 0\,. \tag{2-39}$$

Al resolver la ecuación (2-37) mediante programación cuadrática, se obtienen los valores de w y b [16]:

$$\boldsymbol{w} = \sum_{i=1}^{n}\lambda_i\, y_i x_i\,, \tag{2-40}$$

$$b = \frac{1}{s}\sum_{i=1}^{s} y_i - x_i\cdot\boldsymbol{w}, \tag{2-41}$$

donde: s es el número de datos del vector de soporte.

Una vez presentados los conceptos teóricos necesarios para el desarrollo de la DFT y SVM, en el siguiente capítulo se propone la arquitectura para la implementación de la estimación fasorial y la clasificación de fallas mediante un algoritmo SVM en tiempo real.

3 Desarrollo del modelo de clasificación de fallas basado en SVM y la estimación fasorial mediante la DFT

Con ayuda de la estructura del relevador digital mostrada en la **Figura 2-7**, y la información presentada, el modelo propuesto para la clasificación de fallas es el siguiente:

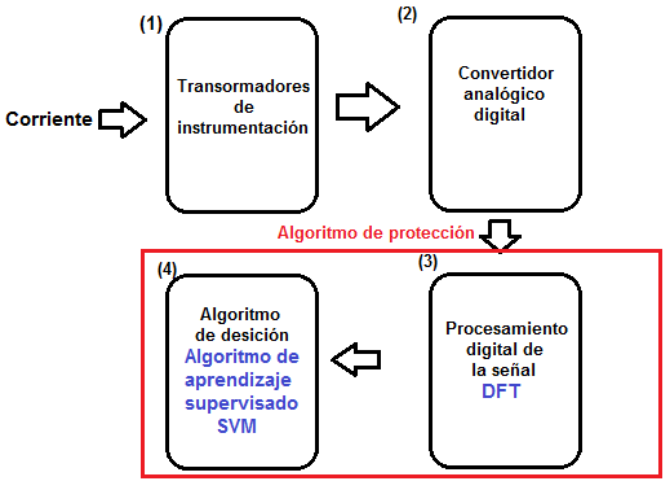


Figura 3-1. Primer modelo para clasificación.

En la **Figura 3-1** se observa la estructura que inicialmente sirvió de apoyo para comenzar el proyecto, constituida por los componentes que se consideraron útiles para el proyecto.
Una vez adoptada una estructura de partida, se comienza el desarrollo del clasificador. Para la implementación se propone dividir en dos fases el proceso, como se exhibe en **Figura 3-2**.

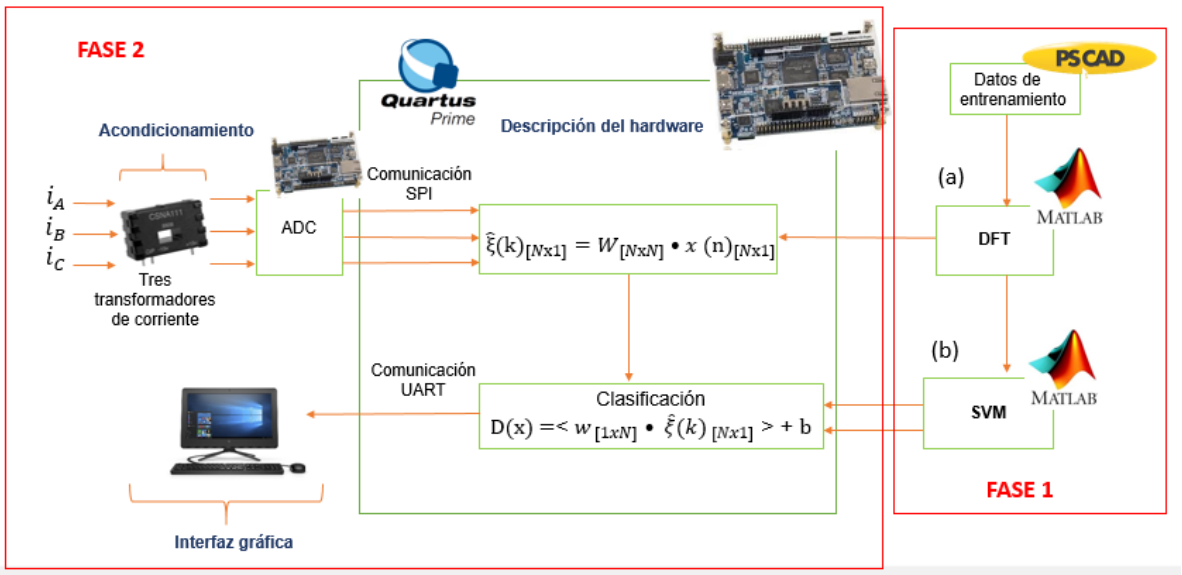


Figura 3-2. Dispositivo propuesto para el clasificador.

La fase 1 representa el procedimiento ejecutado fuera del FPGA, por otro lado, la fase 2 son todos los procesos efectuados en tiempo real para la clasificación de las fallas. La fase 2 trabaja con información recibida de la fase 1 pero es importante mencionar que la comunicación entre estas dos fases no se realiza en tiempo real.

Fase 1, sirve principalmente para el entrenamiento y validación de la SVM, los datos utilizados para el entrenamiento son corrientes en estado de falla, obtenidas de simulaciones de diferentes tipos de fallas permanentes en un sistema de prueba implementado en PSCAD/EMTDC™. En este contexto, fue necesario realizar dos algoritmos en Matlab™ para realizar la estimación fasorial y obtener el modelo de la SVM que integran el sistema de clasificación. De modo que el primero permite calcular los fasores estimados de los datos obtenidos de las simulaciones mediante la implementación de la DFT, ya que las señales resultantes en PSCAD/EMTDC™ son señales instantáneas y senoidales. Este algoritmo también es útil para la obtención de la matriz de Fourier (W) utilizada en la implementación. Y el segundo permite realizar el entrenamiento y obtener la función de un hiperplano optimizado, variable bias (b) y el vector de pesos (w). Además, se valida el resultado del modelo obtenido y posteriormente se utilizan en la implementación.

La fase 2 se compone de 3 etapas:

- Acondicionamiento – Se requiere un sensor de corriente modelo CSNA111 por cada corriente de fase, además se implementa un arreglo de resistencias para mantener un voltaje de entrada de 0-4 V en el ADC del FPGA.

- Descripción del hardware – La comunicación del ADC y el FPGA se realiza a través de SPI. Después de adquirir la ventana de datos, $X_{[Nx1]}$, se efectúa el producto punto con la inversa de la matriz de Fourier (${W_{[1xN]}}^{-1}$) obtenida en el proceso de estimación fasorial, con el fin de adquirir los fasores estimados. Posteriormente, se hace la clasificación, es necesario utilizar la constante bias y el vector de pesos resultantes del modelo de la SVM obtenida para este propósito. Previamente se deben etiquetar los datos, con el propósito de discriminar cada dato en su correspondiente clase; y finalmente el valor resultante se envía mediante comunicación UART a una interfaz gráfica.
- Interfaz – Con ayuda de la herramienta GUI de Matlab™, se realiza la interfaz gráfica, en ella se presentan gráficas de los fasores estimados y la clasificación resultante.

Los diagramas de flujo correspondientes a la estimación fasorial mediante la aplicación de la DFT (a) y el modelo de la SVM (b) ilustrados en **Figura 3-2** se muestran en la siguiente sección. Lo anterior, representa la fase 1 de este desarrollo.

3.1 Algoritmo de estimación fasorial

Debido a que las señales de corriente al ser procesadas se actualizan constantemente, se necesita implementar un muestreo simultaneo y consecutivo de las señales de corriente, por lo que se define una "ventana de datos". La cantidad de muestras de la ventana de datos se ajustan de acuerdo con el número de muestras por ciclo de la señal y respetando el teorema de Nyquist. Si las muestras de un ciclo de la señal coinciden con el número de muestras de la ventana de datos se genera una mayor precisión en la estimación.

Para obtener toda la información de las señales de corriente es necesario actualizar la ventana de una forma deslizante tal como lo describe la **Figura 3-3**. Donde se observa que la ventana 1 comienza en la muestra 0 y termina en la muestra N-1 de la señal, al actualizarse la ventana toma del dato 1 al dato N y así sucesivamente.

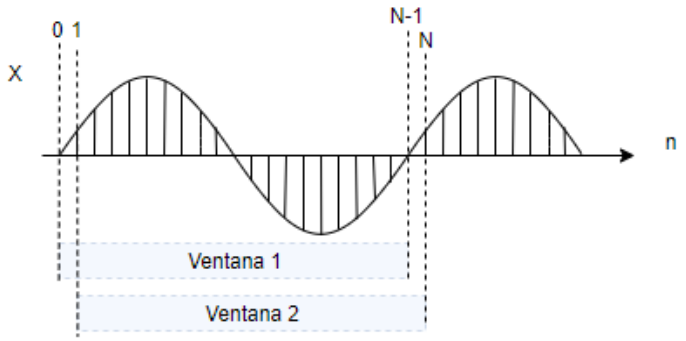


Figura 3-3. Ventana de datos deslizante.

Una vez la ventana de datos deslizante se encuentra disponible, se procede a obtener los fasores de las señales instantáneas de corriente obtenidas en PSCAD/EMTDC™ mediante el algoritmo de la DFT implementado en Matlab™. El cual se divide en dos diagramas de flujo: el diagrama de la **Figura 3-4**, fundamentado en la teoría de DFT (ver sección 2.5) y el diagrama de la **Figura 3-5** incluyen la obtención de la magnitud del fasor correspondiente a la ventana deslizante.

Para realizar la codificación del diagrama de flujo de la **Figura 3-4**, se realizaron los siguientes pasos:

- Se genera un vector cuyas componentes son 0,1, hasta N-1, recordando que N es el número de muestras por ciclo, en este caso N=8.
- Se multiplica el vector antes obtenido (U) por el mismo vector, pero transpuesto, de modo que $A = UU^T$.
- A partir de la matriz A obtenida en el numeral anterior, se genera la matriz de Fourier W .
- Se obtiene la matriz inversa de Fourier ($P = W^{-1}$)
- Se toman los valores correspondientes a la segunda fila y se asigna la parte real e imaginaria en $a = Re\{P(2,:)\}$ y $b = Im\{P(2,:)\}$, respectivamente.

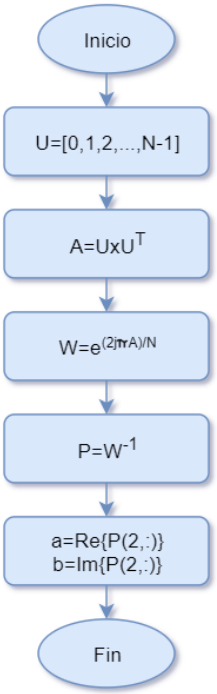


Figura 3-4. Diagrama de flujo para algoritmo de DFT.

Se utiliza un número de muestras N=8, por consiguiente, se obtiene una matriz de Fourier cuya dimensión es de 8x8, entonces la segunda fila que se utiliza tiene una dimensión que será de 1x8 donde cada componente es compleja y se muestran en la **Tabla 3-1**

Tabla 3-1. Segunda fila de la matriz inversa de Fourier.

A	0.1250	0.0884	0	-0.0884	-0.1250	-0.0884	0	0.0884
B	0	-0.0884	-0.1250	-0.0884	0	0.0884	0.1250	0.0884

Una vez obtenidos a y b, se procede a realizar la estimación del fasor de la señal de corriente, mediante la implementación algorítmica de la secuencia deslizante de datos, tal como se ilustra en **Figura 3-5**.

Los pasos para obtener la magnitud del fasor de una señal (s) mediante el algoritmo de la **Figura 3-5** son:

- Se establecen los límites de la ventana de la señal (ini, fin), empezando en 1 y terminando en N para definir su tamaño.
- En este paso se propone el ciclo for, en donde: i empieza en 1 y continúa incrementando hasta alcanzar el valor total de muestras en la señal (T=N).
- Se toman los valores de la ventana de la señal (s) y se guarda en x.
- Se obtiene una parte real del fasor (R $\in \mathbb{R}$) al realizar el producto punto entre el vector b y la ventana de la señal (x). La parte compleja del fasor (C $\in \mathbb{C}$)

se obtiene al realizar el producto punto entre el vector a y la ventana de la señal (x).

- Se obtiene el módulo del fasor (f) a partir de la parte compleja(C ∈ ℂ) y real (R ∈ ℝ)

- Se actualiza la ventana al aumentar en uno los límites de la señal.

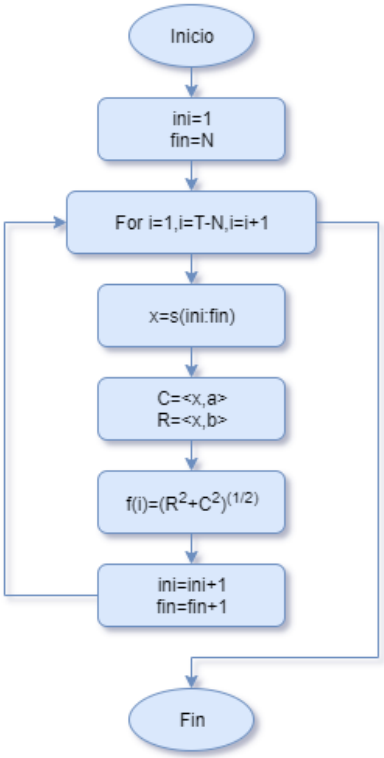


Figura 3-5. Diagrama de flujo para la estimación fasorial de una secuencia consecutiva de una señal instantánea.

Finalmente, es posible observar en **Figura 3-6** el resultado de la estimación fasorial mediante la implementación en Matlab™ de los dos diagramas de flujo descritos y presentados en **Figura 3-4** y **Figura 3-5**.

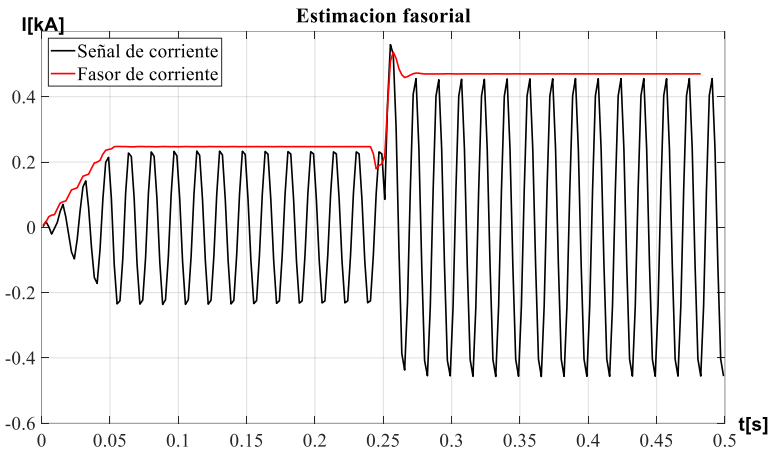


Figura 3-6. Fasor de señal de corriente obtenido mediante la implementación de la DFT.

3.2 Modelado de la SVM

En la **Figura 3-7** se presenta el diagrama de flujo utilizado para modelar el sistema de clasificación de fallas basado en SVM y considerando la estimación fasorial y la preparación del experimento consistente en el ordenamiento de los datos, la separación de los datos de entrenamiento/validación, etiquetado de datos y la validación del modelo obtenido.

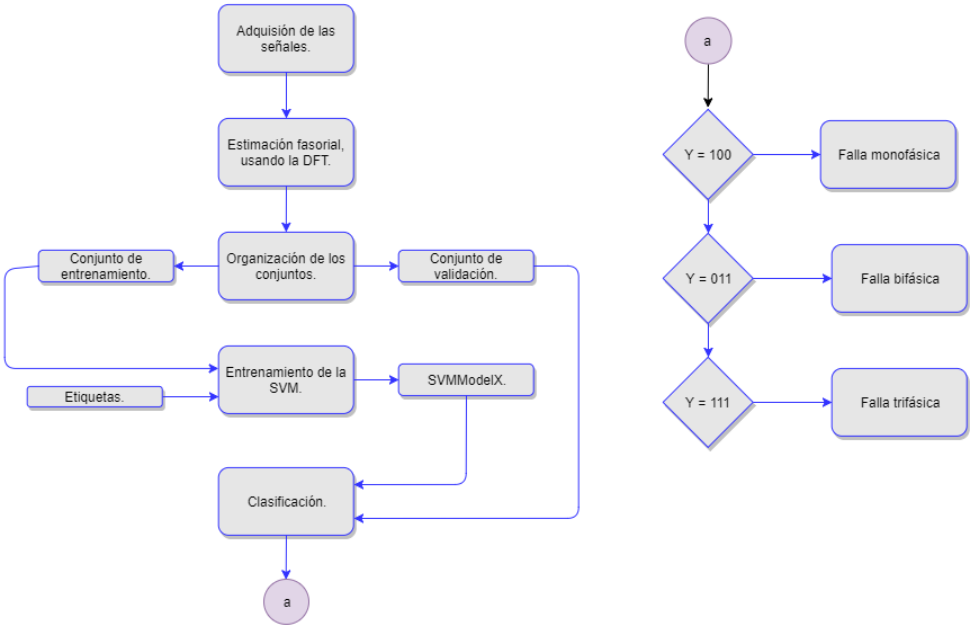


Figura 3-7. Diagrama de flujos para la implementación de SVM.

Los pasos del diagrama de flujo son:

- Se realizan las simulaciones de fallas monofásicas, bifásicas y trifásicas en PSCAD/EMTDC™ para obtener las señales instantáneas de corriente.

- Se aplica el algoritmo de estimación fasorial de la DFT para la obtención de fasores de datos simulados. Estos fasores constituyen los datos de entrenamiento y validación para la SVM.

- Los datos de experimentación se separan en dos conjuntos: entrenamiento (80%) y validación (20%). Se asume 80/20 en este proyecto como un rango adecuado para este experimento.

- Entrenamiento de la SVM con ayuda de las etiquetas asignadas a los datos de entrenamiento.

- Obtención del modelo de la SVM para la realización de la clasificación.

- De acuerdo con los resultados de la clasificación binaria, definir si se trata de una falla monofásica, bifásica, o trifásica.

En este capítulo se mostró, entre otras cosas, dos algoritmos necesarios para la clasificación de la SVM. Para poder ejecutar el algoritmo (a) son indispensables los datos de entrada, estos valores se consiguen mediante las simulaciones presentadas en el siguiente capítulo. Realizado este proceso, es posible ejecutar el algoritmo (b), donde se efectúa el entrenamiento y validación. Debido a esto, después de adquirir los datos de las simulaciones, es importante una correcta organización de ellos, en grupos de validación y entrenamiento, este procedimiento de organización también se describe de mejor manera en el capítulo 4.

4 Sistema eléctrico de potencia de prueba

Como se mencionó en capítulo 3, es necesario un sistema de prueba que permita generar los datos de experimentación y posteriormente los conjuntos de entrenamiento y validación. Por consiguiente, en este capítulo se explica la topología del sistema de potencia implementada en PSCAD/EMTDC™ para obtener los datos de entrenamiento, el ordenamiento de los fasores resultantes del algoritmo (a) en grupos de entrenamiento y validación, con sus respectivas etiquetas. Por último, se presenta el vector de pesos y la variable bias resultantes del algoritmo (b).

4.1 Topología en PSCAD/EMTDC™

El sistema propuesto contiene dos generadores y dos líneas de transmisión, como se muestra en la **Figura 4-1** . Sobre el sistema de prueba se realizan 3 tipos de fallas: monofásica en fase A, bifásica en fase B y C, por último, falla trifásica; las cuales se efectúan en la mitad de las dos líneas, con una duración de 0.25 [s], se aplican en 0.25 [s] y el tiempo de simulación es 0.5 [s]. La frecuencia de muestreo se define de 960 [Hz], ya que es la frecuencia mínima que permite obtener la simulación, posteriormente en Matlab se disminuye está frecuencia en un factor de 2 para obtener 8 muestras por ciclo.

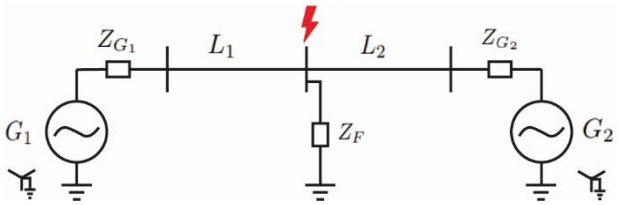


Figura 4-1. Sistema eléctrico de potencia simulado en PSCAD/EMTDC™.

Las características de los elementos del sistema se observan en la **Tabla 4-1**.

Tabla 4-1. Especificaciones del sistema simulado.

Características del sistema				
Generadores				
	Voltaje [kV]	Desfase [°]	R [Ω]	X [Ω]
Gen 1	230	0	9.186	52.02477
Gen 2	230	20	9.186	52.0277
Líneas				
	R [Ω]	X [Ω]	Longitud [Km]	
Line 1	3.2170	45.6986	90	
Line 2	0.3574	5.0776	10	
Potencia base: 100 [MVA]				

Casos de estudio: Los casos de estudio de la topología propuesta se observan en la ***Tabla 4-2***. Se presenta la falla monofásica, bifásica y trifásica con una variedad de cinco resistencias de falla, en total obtendremos 45 señales senoidales para las tres fases.

Tabla 4-2. Resistencias de falla de sistema simulado.

Tipo de falla	Resistencia de falla [Ω]
Monofásica a tierra	0.01
	10
	20
	50
	70
Bifásica	0.01
	10
	20
	50
	70
Trifásica	0.01
	10
	20
	50
	70

En la siguiente ***Figura 4-2*** se muestran las señales de corrientes instantánea obtenidas de la simulación, donde la resistencia de falla es igual a 70 [Ω] para los tres tipos de falla. La amplitud máxima de la corriente en estado estable antes de la falla es 0.25 [kA] y sus valores máximos durante falla depende del tipo de falla.

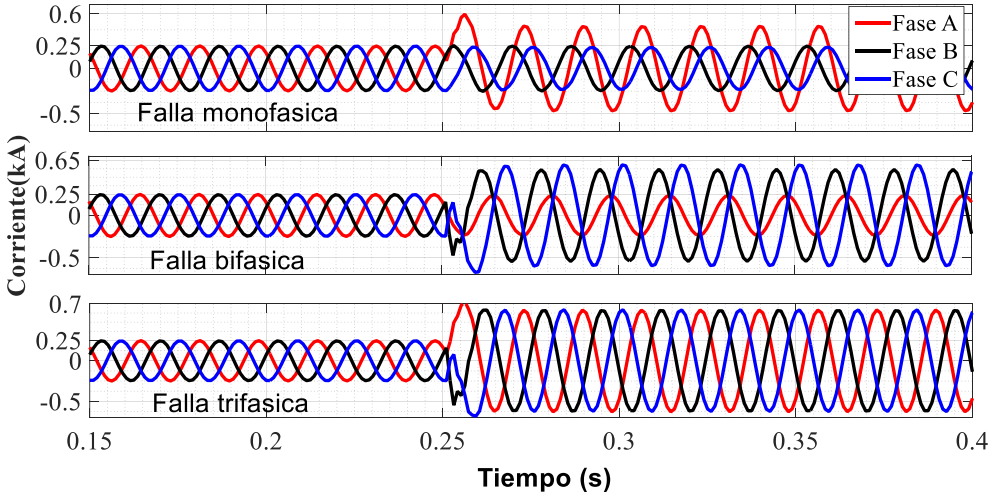


Figura 4-2. Corrientes instantáneas obtenidas de la simulación.

4.2 Desarrollo del modelo del sistema de clasificación basado en SVM

Una vez las señales instantáneas de corrientes son obtenidas mediante PSCAD™/EMTDC™ y sus fasores son estimados a partir de la estimación fasorial llevada a cabo por la DFT, se procede a extraer sólo la amplitud de los fasores de corriente para preparar los datos de experimentación acorde se describe en **Figura 4-3**. Se ilustran los 45 datos obtenidos de las señales de corriente de la simulación. Estos datos se organizarán para obtener varias ventanas y utilizarlas para el entrenamiento y validación de la SVM.

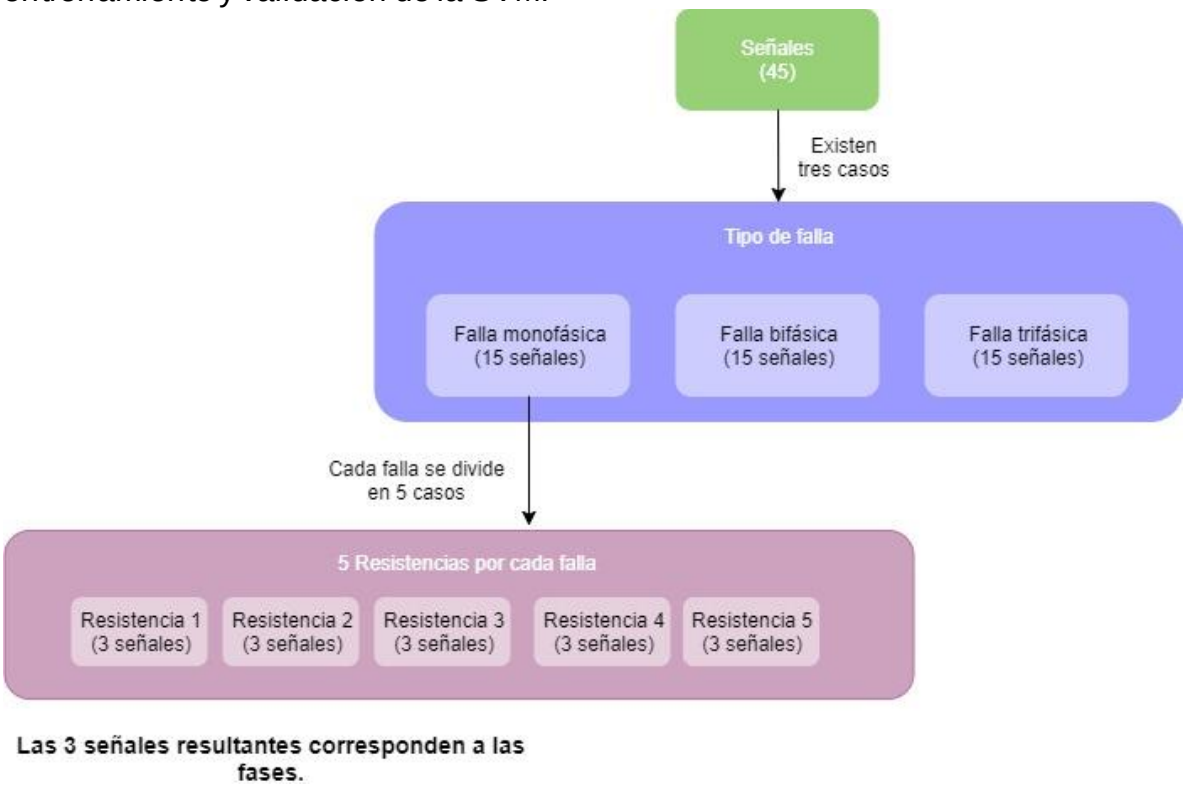


Figura 4-3. Distribución de datos de experimentación para entrenamiento y validación de la SVM.

La matriz final en Matlab™ que ilustra la manera de ordenar los datos, se observa en la **Figura 4-4**, considerando el tipo de falla y su fase para las filas y el número de muestras de cada ventana segmentada en las columnas. El número total de muestras por fase obtenido fue de 144 datos, divididos en 18 ventanas de 8 muestras; por lo tanto, el conjunto total de experimentación está formado por 810 ventanas. Los conjuntos de datos de entrenamiento y validación se dividen respectivamente, asumiendo un 80% de ventanas consecutivas, 648 en total, para el entrenamiento y el porcentaje restante para el conjunto de validación. En las siguientes figuras se ilustran sólo 5 ventanas, con el fin de ejemplificar.

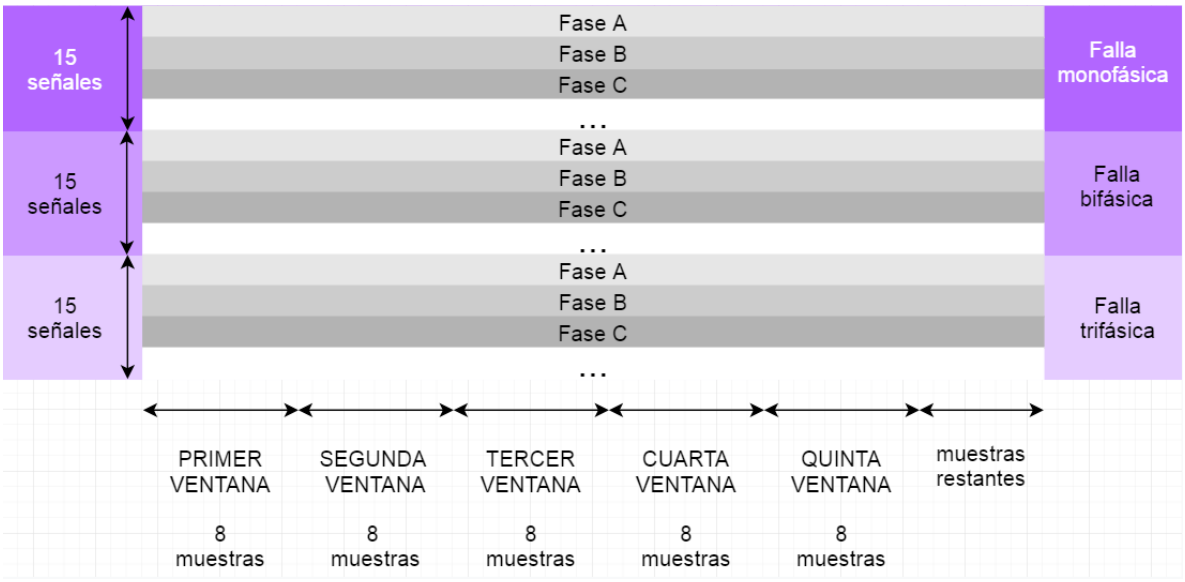


Figura 4-4. Organización de datos en una matriz.

Ahora, en la **Figura 4-5** se ejemplifica de manera gráfica las ventanas no deslizantes de 8 muestras en cada fase, para una falla trifásica con una resistencia de falla igual a 0.01 [Ω].

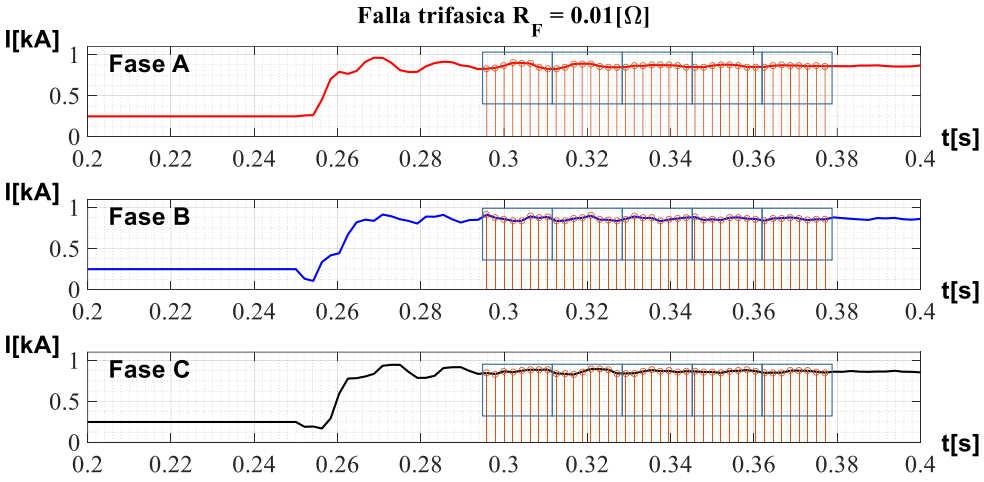


Figura 4-5. Organización gráfica de los datos.

La asignación de las etiquetas para el entrenamiento se presenta en la **Tabla 4-3**. Cada bit pertenece a una fase y si existe falla en esa fase el bit es igual a uno, en caso contrario es cero.

Tabla 4-3. Etiquetas para el conjunto de entrenamiento.

Tipo de Falla	Etiqueta		
	Fase A	Fase B	Fase C
Monofásica	1	0	0
	0	1	0
	0	0	1
Bifásica	0	1	1
	1	1	0
	1	0	1
Trifásica	1	1	1

Datos obtenidos de función optimizada de la SVM: Los datos de validación, entrenamiento y etiquetas mediante matrices al algoritmo (b), se ingresan a la función que se utilizó para el entrenamiento es la llamada "fitcsvm", se eligió esta función ya que sirve para la clasificación de una clase binaria en la máquina de soporte vectorial; los datos de entrada es el vector de etiquetas y la matriz de entrenamiento (80% del total de datos). El resultado obtenido es el vector de pesos (w) y la constante bias (b), presentados en la **Tabla 4-4**.

La segunda función utilizada se llama "predict" y sirve para obtener un vector de etiquetas (ver sección 3.2), los datos de entrada son el w y b, además de la matriz de validación (20%).

Tabla 4-4. Valores resultantes del algoritmo SVM en Matlab™

w =
0.2849
0.2845
0.2840
0.2849
0.2849
0.2852
0.2848
0.2843
b =
2.3337

Se continúa presentando los resultados de las etiquetas del conjunto de validación del algoritmo en la SVM, en ***Tabla 4-5***. Se observa que el resultado es consecuente al ordenamiento explicado anteriormente, ya que las etiquetas de cada fase se pueden utilizar para la clasificación de falla monofásica en A, bifásica en B y C, y trifásica para cada valor en la resistencia de falla. Por esta razón, es viable usar los valores de la **Tabla 4-4** para su implementación en el FPGA.

Tabla 4-5. Matriz de etiquetas resultante de algoritmo de SVM.

Y=

1	0	0	RF= 0.01 [Ω]
0	1	1	
1	1	1	
1	0	0	RF=10 [Ω]
0	1	1	
1	1	1	
1	0	0	RF=20 [Ω]
0	1	1	
1	1	1	
1	0	0	RF=50 [Ω]
0	1	1	
1	1	1	
1	0	0	RF=70 [Ω]
0	1	1	
1	1	1	
Fase A	Fase B	Fase C	

Existen dos consideraciones importantes para ingresar los datos de entrenamiento al algoritmo de la SVM en Matlab™. En primer lugar, el entrenamiento se realiza después de 2 ciclos ocurrida la falla; y la segunda consideración es la entrada de los datos normalizados respecto al valor máximo de la corriente en estado estable antes de la falla.

Concluido este capítulo, también da por terminada la fase 1, continuando con la implementación física, correspondiente a la fase 2. Para la implementación se ingresan los resultados de la segunda fila de la matriz inversa de Fourier en **Tabla 3-1** y el vector de pesos (w) y constante bias (b) de en la arquitectura del FPGA como se ilustró en la **Figura 3-2**.

5 Implementación física

En esta sección se presentan las tres etapas mencionadas previamente en el capítulo 3. Posteriormente, se describe el hardware utilizado y la implementación detallada de cada una de las etapas de la arquitectura del sistema de clasificación de fallas basado en estimación fasorial y SVM. Debido a que tiene que realizarse una etapa de acondicionamiento de las señales de corrientes, tal proceso se introduce, a continuación.

5.1 Acondicionamiento analógico de las señales de corriente

Para transducir las señales de corriente se utiliza el sensor de corriente altema CSNA111, el cual es capaz de sensar una corriente AC de ± 70 A_{pico} a ± 70 mA_{pico}. Esta corriente es transducida a una señal de voltaje al pasar a través de una resistencia R_t [32]. Cuando R_t=100 [Ω], existe un voltaje de salida V_{out} = 1 V por cada corriente de entrada I_{IN} = 10 A. Esto significa que se tiene un voltaje AC máximo de salida de $V=\pm 7V_{pico}$.

Una vez se caracteriza el sensor, se procede a la adecuación de su voltaje de salida acorde a las características del ADC embebido en el FPGA a utilizar. De modo que en la entrada del ADC se requiere una entrada de voltaje de 0-4 V AC, por lo que se implementa el arreglo de resistencias de la **Figura 5-1**.

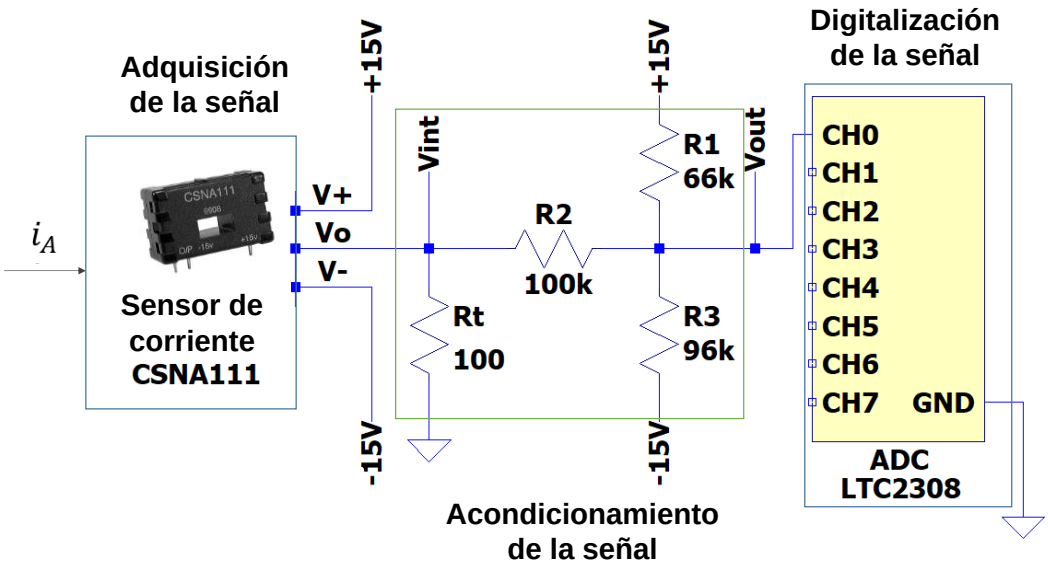


Figura 5-1. Circuito de acondicionamiento analógico.

5.2 Descripción de hardware

El modelo de la tarjeta de diseño utilizada para la creación del sistema clasificador es DE10-Nano, este dispositivo permite el diseño de hardware mediante un FPGA Intel System-on-Chip (SoC), que combina un procesador ARM Cortex-A9 con lógica programable Cyclone ® V.

El chip Cyclone ® V embebido en la tarjeta DE10-Nano trabaja con dos lenguajes de descripción: Verilog y VHDL, en este proyecto se utiliza VHDL [33] [34].

5.2.1 Diseño de hardware mediante VHDL

Definida la tarjeta y el lenguaje para utilizar en el FPGA, se menciona de manera muy general la estructura de diseño en VHDL, ya que es necesaria su definición para la correcta descripción de la arquitectura en el FPGA [35].

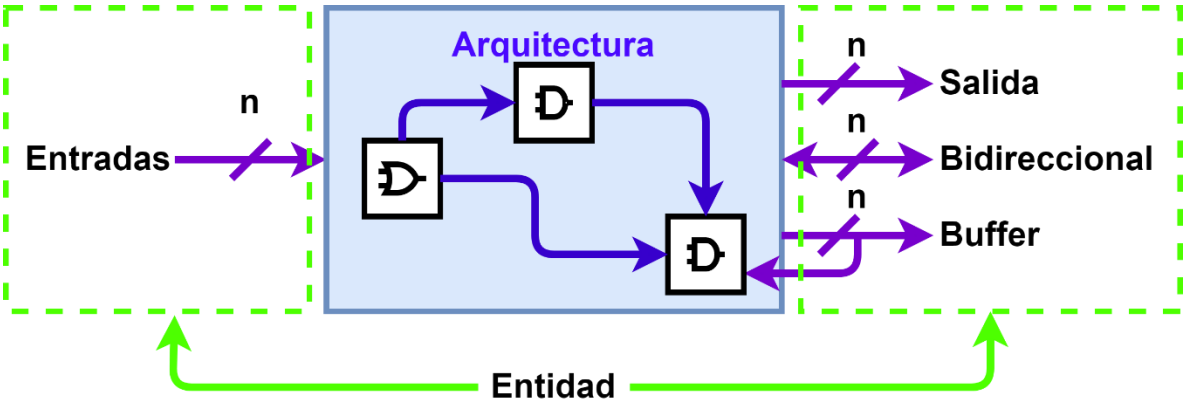


Figura 5-2. Estructura de diseño en VHDL.

En la **Figura 5-2** se muestra la estructura de diseño en VHDL de un circuito electrónico, esta se divide en dos partes:

- **Entidad**. En esta sección se describen los puertos con los que interactúa el circuito y su exterior. Para definir un puerto en la entidad se debe tener claro los siguientes puntos:
  - **Direccionalidad**. Definición del puerto como entrada o salida, ambos (bidireccional) o buffer (donde el puerto es un registro interno y un puerto de salida al mismo tiempo).
  - **Tamaño.** Número de bits usados por el puerto.
  - **Tipos de dato (Formato).** Aunque para el compilador (encargado de convertir el código en hardware) los puertos solo reciben o entregan bits, para el diseñador es útil definir un formato numérico a cada puerto, de esta forma se puede entender si el dato asignado al puerto proviene o próximamente será usado para una operación lógica o aritmética.

- **Arquitectura**. Descripción del comportamiento interno del circuito electrónico. Explicando operaciones, manejo y procesamiento de los datos.

Posteriormente, se anexa una tabla por cada circuito implementado donde se plasma su entidad, considerando los puntos expuestos anteriormente. También se agregarán sus respectivas cartas ASM, dado que de esta forma fueron diseñadas las arquitecturas de todos los circuitos en este proyecto.

5.2.2 Conexiones del FPGA

La tarjeta de desarrollo contiene 5 señales de reloj conectadas a distintos dispositivos embebidos en la tarjeta. Para la construcción de este proyecto se utilizaron los relojes CLK1_50 y CLK2_50, que están conectados al FPGA y ambos poseen una frecuencia de 50 [MHz], sus terminales asignadas se muestran en la **Tabla 5-1**.

Tabla 5-1. Asignación de terminales de relojes en FPGA [34].

Signal Name	FPGA Pin No.	Description	I/O Standard
FPGA_CLK1_50	PIN_V11	50 MHz clock input	3.3V
FPGA_CLK2_50	PIN_Y13	50 MHz clock input	3.3V
FPGA_CLK3_50	PIN_E11	50 MHz clock input (share with FPGA_CLK1_50)	3.3V
HPS_CLK1_25	PIN_E20	25 MHz clock input	3.3V
HPS_CLK2_25	PIN_D20	25 MHz clock input	3.3V

En la **Tabla 5-2** se observa las terminales dedicadas a los interruptores, existen 5 switches. En este caso, se empleó SW[0] como interruptor dedicado al reset del sistema.

Tabla 5-2. Asignación de terminales para interruptores [34].

Signal Name	FPGA Pin No.	Description	I/O Standard
SW[0]	PIN_Y24	Slide Switch[0]	3.3V
SW[1]	PIN_W24	Slide Switch[1]	3.3V
SW[2]	PIN_W21	Slide Switch[2]	3.3V
SW[3]	PIN_W20	Slide Switch[3]	3.3V

El ADC puede ser configurado para aceptar 8 señales de entrada, estas señales se conectan de ADC_IN0 a ADC_IN7, se utilizaron 3 terminales de entrada: ADC_IN0, ADC_IN1 y ADC_IN2. En la **Figura 5-3** se expone la localización de las terminales del ADC en la tarjeta, además de dos terminales AC24 y AD26, empleados como salidas del protocolo UART y el circuito de la SVM, respectivamente (diseños presentados en la sección 5.3).

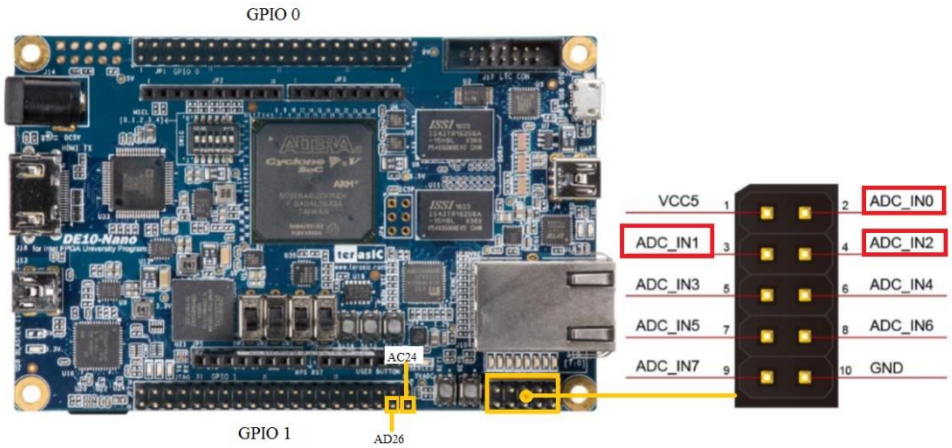


Figura 5-3. Terminales utilizadas para el diseño [34].

Las terminales correspondientes a las señales en el convertidor analógico digital se presentan en la **Tabla 5-3**.

Tabla 5-3. Asignación de terminales de convertidor analógico digital [34].

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CONVST	PIN_U9	Conversion Start	3.3V
ADC_SCK	PIN_V10	Serial Data Clock	3.3V
ADC_SDI	PIN_AC4	Serial Data Input (FPGA to ADC)	3.3V
ADC_SDO	PIN_AD4	Serial Data Out (ADC to FPGA)	3.3V

5.3 Arquitecturas embebidas en el FPGA

La descripción del hardware que se ilustró en la **Figura 3-2** requiere la implementación de arquitecturas encargadas de la comunicación SPI, la obtención de fasores estimados, la clasificación mediante el hiperplano de separación y la comunicación UART [35], [36], [37].

Las arquitecturas que realizan estos trabajos se muestran en la **Figura 5-4** Estas se ilustran por brevedad para un sistema monofásico, i.e. sólo monitoreando una señal de corriente, por lo que se explicará el funcionamiento para este caso. Posteriormente, para lograr el manejo de las 3 señales correspondientes al sistema trifásico, simplemente se replica este diseño dos veces más para las otras fases.

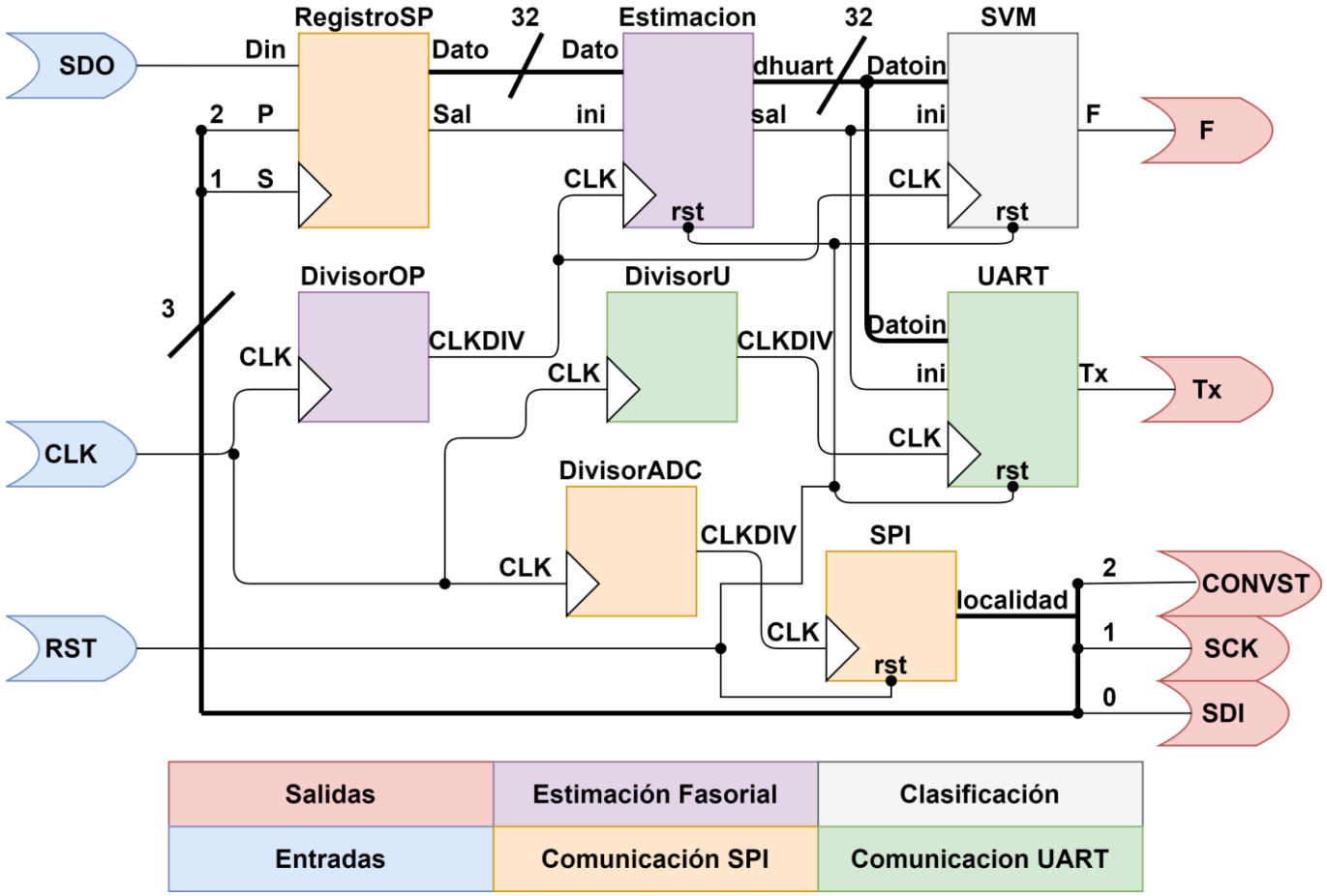


Figura 5-4. Arquitecturas para clasificador monofásico

A continuación, se explican brevemente las etapas en las que se divide el circuito mostrado en la **Figura 5-4**.

1) La comunicación del ADC se constituye del divisor de frecuencia, una memoria ROM contenida en el circuito nombrado SPI y un registro serie-paralelo, llamado RegistroSP.

2) Con el nombre Estimación, este circuito tiene como objetivo realizar las operaciones necesarias para obtener los fasores estimados de las señales de corriente.

3) El circuito nombrado SVM determina si la señal adquirida se encuentra en estado de falla o no, sustituyendo en la función del hiperplano la constante bias, el vector de pesos y el fasor obtenido.

4) Por último, UART es un circuito que comunica el FPGA y una computadora.

En la **Tabla 5-4** se presentan las señales de entrada y salida de la arquitectura final para el sistema de clasificación para una sola señal de corriente.

Tabla 5-4. Entidad del circuito clasificador de una señal.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
CLK	Entrada	1	Lógico	Reloj de FPGA a 50 [MHz]
RST	Entrada	1	Lógico	Al recibir un “1” lógico se reinicia el circuito
SDO	Entrada	1	Lógico	Señal de salida del ADC
SCK	Salida	1	Lógico	Reloj del ADC
SDI	Salida	1	Lógico	Señal de entrada del ADC
CONVST	Salida	1	Lógico	Inicia la recepción de datos en el ADC
Tx	Salida	1	Lógico	Señal que permite la transmisión de datos del FPGA a la computadora
F	Salida	1	Lógico	Resultado final de la clasificación

El funcionamiento del clasificador comienza con la comunicación SPI. Con ayuda de ésta se adquiere la señal de entrada del ADC en el puerto SDI (Serial Data Input), que es un voltaje AC de 0-4 V. El dato entra bit por bit de manera serial en el circuito RegistroSP, la salida de éste es un dato en paralelo de 12 bits el cual se utiliza en el bloque llamado Estimación para obtener el fasor estimado mediante la teoría de la DFT, el fasor estimado es asignado a un registro nombrado “duart” que contiene 32 bits. A continuación, “duart” se envía al circuito llamado SVM con el propósito de evaluar la ventana de datos en el hiperplano de la máquina de soporte vectorial (2-24) y clasificar los valores en falla o no falla. Al mismo tiempo, este dato se envía

al circuito UART con la finalidad de enviar los fasores estimados a la interfaz gráfica. Por último, el valor “F” del circuito llamado SVM contiene 1 ó 0 y también es enviado a la interfaz gráfica mediante UART y 3 LEDS del FPGA para su visualización.

Explicada de manera general la arquitectura de clasificación para una señal monofásica, se expone de manera particular cada circuito, pero antes de profundizar en las etapas de implementación es necesario mencionar el uso de divisores de frecuencia, ya que la frecuencia del reloj principal en el FPGA es de 50 [MHz] y este valor es grande si requerimos obtener 8 muestras por ciclo de la señal de corriente. Asimismo, realizar las operaciones necesarias para obtener los fasores estimados requieren una frecuencia igual a la de muestreo, por lo tanto, utilizar un valor de 50 MHz resulta inadecuado. Por último, para el caso de la comunicación UART es imprescindible ajustar el divisor a un valor cercano a la velocidad de baudaje que se está utilizando.

5.3.1 Divisores de frecuencia

La principal utilidad de los divisores de frecuencia es operar como reloj en los circuitos de la comunicación SPI y UART y para realizar las operaciones de la estimación fasorial, nombrados como DivisorADC, DivisorU, DivisorOP, respectivamente.

Con ayuda de una variable, nombrada “aux” y el periodo del reloj principal del FPGA se busca generar una señal con una nueva frecuencia. El valor de “aux” comienza en cero y cada ciclo del reloj de 50 [MHz] incrementa en uno. Otro parámetro nombrado “n” define hasta qué valor contará “aux”, para después reiniciar la cuenta nuevamente desde cero. La salida del divisor de frecuencia conmuta en 1 y 0 usando de referencia estos dos parámetros y resultando en una señal con una nueva frecuencia basada en el valor “n”. En la **Figura 5-5** se ilustra este proceso mediante una carta ASM

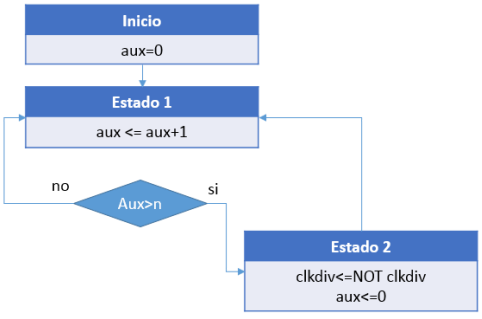


Figura 5-5. Carta ASM para divisor de frecuencia.

En **Tabla 5-5** se muestra la entidad del circuito que se constituye de 2 entradas y una salida.

Tabla 5-5. Entidad de los circuitos divisores.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
CLK	Entrada	1	Lógico	Reloj de FPGA a 50 [MHz]
RST	Entrada	1	Lógico	Al recibir un "1" lógico se reinicia el circuito.
clkdiv	Salida	1	Lógico	Señal modificada en la frecuencia por el divisor.

Cálculo para los divisores de frecuencia

Para calcular la variable "n" en cualquier divisor, utilizamos la ecuación (5-1).

$$n = \frac{f_p}{2f_n}, \qquad (5\text{-}1)$$

Donde: f_n es la frecuencia de la nueva señal y f_p es la frecuencia de la señal de reloj del FPGA.

Para implementar los 3 divisores, se consiguieron los siguientes resultados de "n" en **Tabla 5-6**:

Tabla 5-6. Constantes de divisores.

Nombre	$f_n[kHz]$	*N*
DivisorOP	13.44	1860
DivisorU	115	217
DivisorADC	13.44	1860

Una vez diseñados circuitos para adecuar el reloj principal del FPGA con el fin de poder entregar la señal adecuada a cada etapa del clasificador. También es necesario construir un circuito que facilite el manejo de los datos seriales entregados por el ADC, como se menciona a continuación.

5.3.2 Registro serie-paralelo

Debido a que el ADC envía información a el FPGA de forma serial y esta es requerida en forma paralela, entonces se usa un registro serie-paralelo. En este circuito los bits en serie se actualizan cada flanco de subida de la señal SCK, al completar los 12 pulsos de SCK se espera el pulso en CONVST para actualizar el dato de 12 bits en paralelo. Simultáneamente, un pulso alto en "sal" se presenta con la misma duración de CONVST para indicar la salida del dato, al finalizar este tiempo regresa a estado bajo, hasta la próxima actualización de un nuevo dato en paralelo.

La **Tabla 5-7** muestra la entidad del circuito, donde "s" toma el valor de SCK, "p" representa el valor de CONVST, "din" son los bits que se van adquiriendo de SDO y "dato" es el resultado final que contiene 12 bits:

Tabla 5-7. Entidad registro serie-paralelo.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
S	Entrada	1	Lógico	Cuando recibe un flanco de subida entra un bit del dato serie.
P	Entrada	1	Lógico	Cuando es "1" envía dato de 12 bits en paralelo.
Din	Entrada	1	Lógico	Bit serial obtenido del ADC.
Dato	Salida	12	Binario	Registro que obtiene el dato en paralelo.
Sal	Salida	1	Lógico	Siempre está en 0, solo cambia en 1 cuando se envía "dato".

A continuación, en la **Figura 5-6** se muestra la respectiva carta ASM para su implementación.

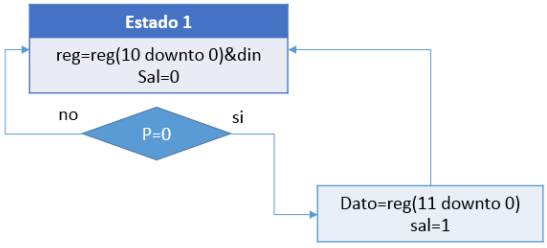


Figura 5-6. Carta ASM de registro serie-paralelo.

El diagrama de bloques implementado para el registro se muestra en la **Figura 5-7**.

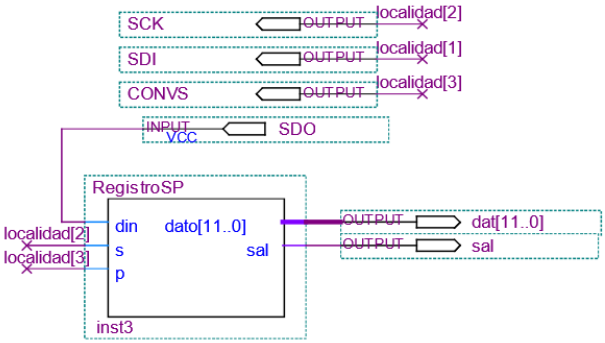


Figura 5-7. Diagrama de bloques registro serie-paralelo.

Terminado el circuito, nombrado RegistroSP se conecta la terminal SDO del ADC al puerto de entrada "din" como se muestra en la **Figura 5-7**, para poder convertir los datos obtenidos en paralelo y enviarlos al próximo circuito que calcula el valor del fasor estimado.

La descripción detallada de la obtención del dato en el puerto SDO del ADC se presenta a continuación, esta descripción abarca la comunicación SPI requerida y configuración de los 3 canales utilizados en el ADC.

5.3.3 Comunicación SPI

El ADC requiere el uso de una comunicación serial y bidireccional que le permita la conexión al FPGA, en este caso es compatible con el protocolo SPI. La representación de esta comunicación se observa en la **Figura 5-8** [38].

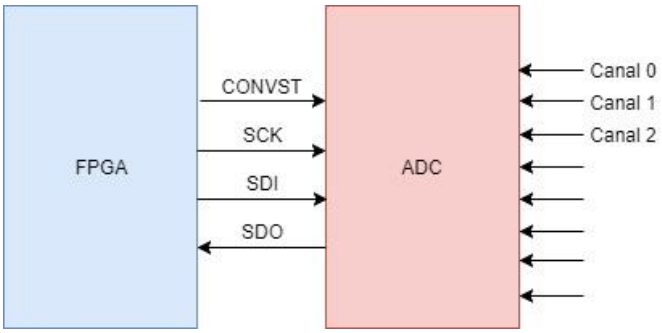


Figura 5-8. Comunicación ADC-FPGA

Para su uso se requieren cuatro puertos sincronizados, mencionados anteriormente: señal de inicio de protocolo (CONVST), señal de reloj (SCK), envío (SDO) y recepción (SDI). El FPGA envía señales al puerto CONVST, SCK y SDI con ayuda de tramas de comunicación y recibe un dato a través del puerto SDO. En la **Tabla 5-8** se presentan estas señales.

Tabla 5-8. Descripción señales para comunicación SPI.

Señal	**Descripción**
CONVST	Su tarea consiste en iniciar el protocolo de comunicación.
SCK (*Serial Data Clock*)	Es la señal de reloj. Hace posible la sincronización de envío y recepción de datos.
SDI (*Serial Data Input*)	Configuración de canales. Contiene seis bits, la manera en cómo opera el ADC dependerá de ellos.
SDO (*Serial Data Output*)	Señal de salida del ADC. Es responsable de entregar 12 bits de salida al FPGA.

El diseño de las tramas de comunicación para el protocolo SPI se realizó con ayuda de una memoria ROM y su respectivo contador además del divisor de frecuencia, como se muestra en la ***Figura 5-9***.

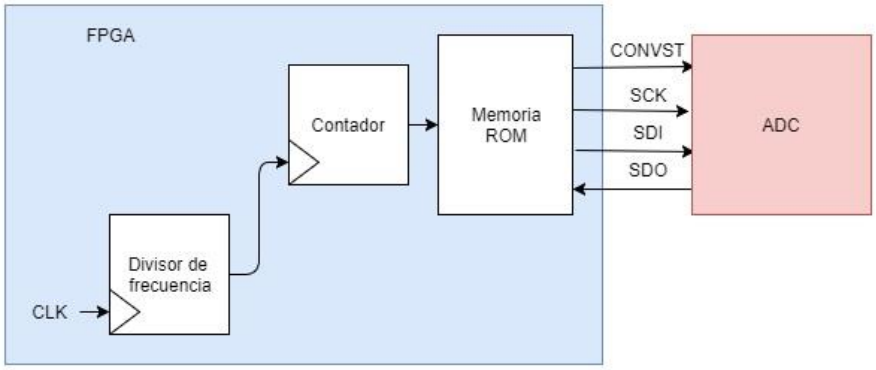


Figura 5-9. Arquitectura propuesta para uso del ADC.

Los cálculos y consideraciones necesarias para la implementación de la memoria ROM y el contador se presentan a continuación

Tramas de comunicación

La trama de comunicación cambia con base en el canal que se desea activar ya que el puerto SDI se inicializa de manera distinta. Los canales utilizados en este trabajo son el 0,1 y 2. En la **Figura 5-10** se presenta el diagrama de tiempos a diseñar para la comunicación del ADC en el canal 1.

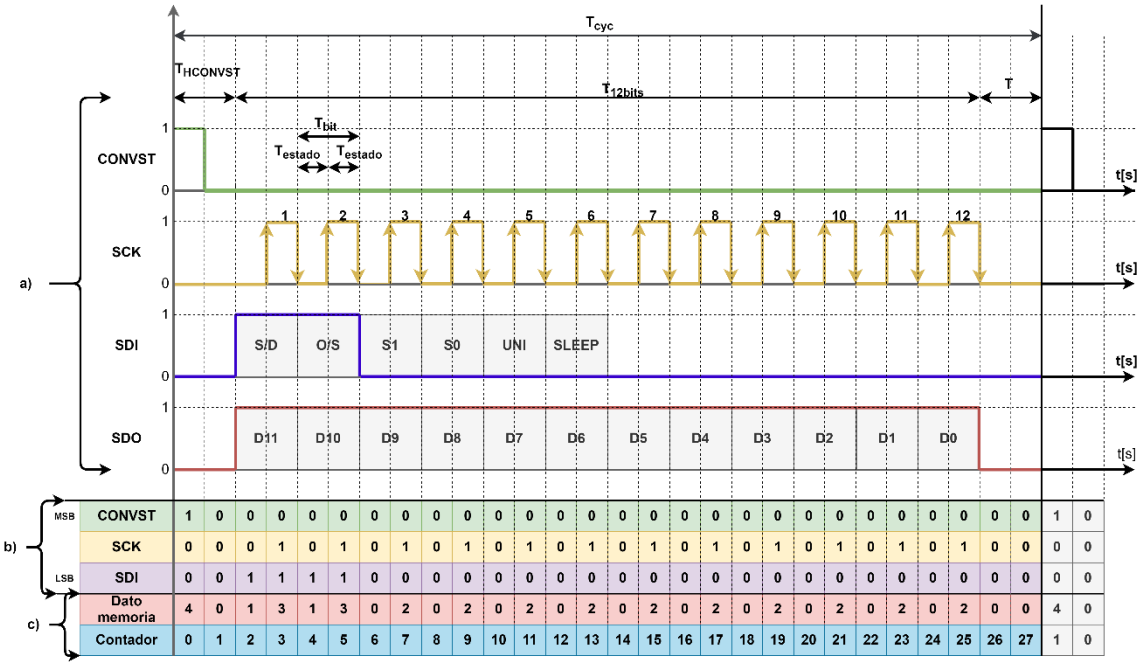


Figura 5-10. Diagrama de tiempos para comunicación ADC.

Los nombres de los tiempos que requieren ser calculados para la trama de comunicación se observan en la **Figura 5-10** (a). La variable nombrada T_{estado} es el tiempo mínimo en el protocolo de comunicación y del mismo modo representa cada cambio de estado en SCK. La siguiente variable por considerar es T, ya que

este es un tiempo de espera en el ADC antes de recibir el siguiente pulso de CONVST. La variable $T_{HCONVST}$ representa el tiempo que tarda en enviar el primer bit del dato después del flanco de subida en CONVST. Otra variable que se observa es T_{bit}, que es el tiempo de duración de cada bit del dato. Por último, se observa T_{CYC}, este es el tiempo que tarda el protocolo en obtener los 12 bits del dato considerando $T_{HCONVST}$ y T.

Cálculos

Conociendo la trama de comunicación en el protocolo, el siguiente paso es realizar los cálculos correspondientes para determinar el valor del contador a utilizar en la memoria ROM, esto permite además definir el número de localidades de la memoria.

Se define la conversión con 8 muestras por ciclo de la señal de corriente. Por lo tanto, el tiempo de muestreo, denominado en este caso como T_{CYC} es:

$$T_{CYC} = \frac{T_s}{N}, \tag{5-2}$$

Donde: T_s es tiempo de un ciclo fundamental de la señal de 60 [Hz] y N representa el número de muestras

$$\therefore T_{CYC} \approx 2.0833\ [ms].$$

En el diagrama se observa que, si cada bit del dato en el canal SDO tiene la misma duración entonces:

$$T_{CYC} = 12 * T_{bit} + T_{HCONVST} + T. \tag{5-3}$$

De acuerdo con el funcionamiento del ADC, $T_{HCONVST} > 20$ [ns] y $T > 20$ [ns]. Por lo tanto, es viable considerar que:

$$T_{HCONVST} = T_{bit}, \tag{5-4}$$

$$T = T_{bit}, \tag{5-5}$$

Entonces:

$$T_{CYC} = 14 * T_{bit}. \tag{5-6}$$

El resultado que se obtiene, si $T_{CYC} \approx 2.0833\ [ms]$, es el siguiente:

$$T_{bit} \approx 148.80[\mu s]\ ,$$

Finalmente, para obtener el tiempo mínimo del protocolo, con ayuda de la **Figura 5-10** deducimos que:

$$T_{estado} = \frac{T_{bit}}{2}, \tag{5-7}$$

$$\therefore T_{estado} = 74.40[\mu s].$$

Con ayuda de T_{estado} se puede dividir el tiempo de obtención del dato, $\mathrm{T_{CYC}}$, en "fragmentos" iguales, cada "fragmento" representa una localidad de la memoria ROM. Para obtener el número total de localidades (l) se realiza la siguiente lo siguiente:

$$l = \frac{T_{CYC}}{T_{estado}}, \tag{5-8}$$

$$l \approx 28.$$

Una vez el número de localidades se define en 28, se ingresa el valor correspondiente para formar la trama de comunicación SPI. En la **Figura 5-10**(b), se ilustra el valor que contiene cada localidad de acuerdo con el bit asignado en CONVST, SCK y SDI y en **Figura 5-10**(c) se muestra la relación de estos datos de localidad guardados en la memoria y el su contador asignado. En consecuencia, se concluye que el contador debe ser capaz de direccionar la totalidad de estas localidades, teniendo una cuenta desde 0 hasta 27.

Por lo tanto, la frecuencia requerida para el divisor de frecuencia llamado "DivisorADC" es:

$$f_{conv} = \frac{1}{T_{estado}}, \tag{5-9}$$

$$f_{conv} = 13.44[kHz].$$

Esta frecuencia expresa el recorrido en las 28 localidades de memoria, esto significa que se obtiene un sólo dato de 12 bits. Por lo tanto, para obtener 8 muestras por ciclo de la señal, f_{conv} se multiplica por 8, asimismo para adquirir un dato de los 3 canales, es necesario triplicar esta frecuencia ya que el ADC no adquiere las señales de manera paralela.

El diagrama de bloques que cumple la función de comunicar el FPGA y ADC se muestra a continuación, en la **Figura 5-11** se observa la conexión de registro serie paralelo y la comunicación SPI.

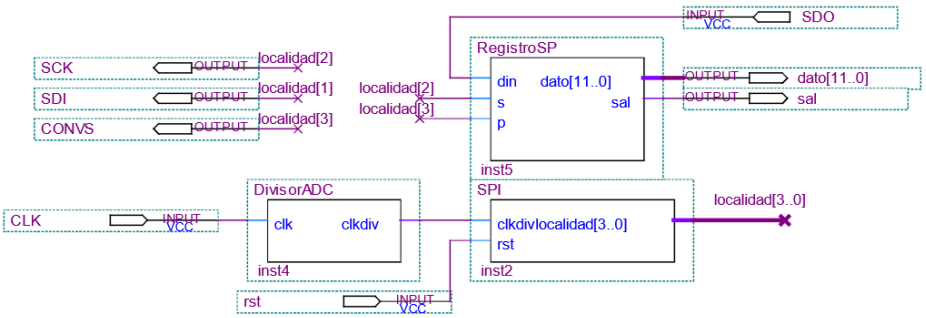


Figura 5-11. Diagrama de bloques comunicación SDI.

Se propone la siguiente entidad en la **Tabla 5-9**:

Tabla 5-9. Entidad Memoria ROM con contador.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
Localidad	Salida	3	Binario	Contiene los bits requeridos para CONVST, SCK y SDI, guardados en la memoria.
Clkdiv	Entrada	1	Lógico	Reloj de entrada
Rst	Entrada	1	Lógico	Al recibir un "1" lógico se reinicia el circuito

La carta ASM de la memoria ROM y el contador se puede observar en la **Figura 5-12**, donde "dir" es la variable del contador que apunta a las 28 localidades de memoria.

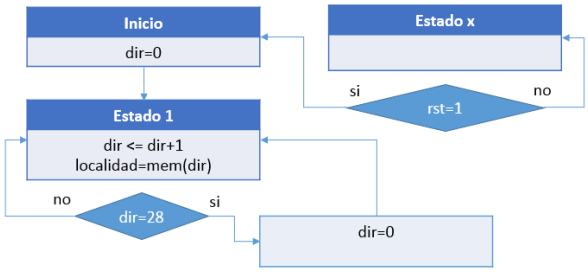


Figura 5-12. Carta ASM de memoria ROM y contador.

Una vez finalizada la etapa de comunicación del FPGA con el ADC para la obtención de datos de los sensores de corriente, se puede continuar con la siguiente etapa que consiste en estimar el fasor a partir de un conjunto de muestras en la señal de la corriente.

5.3.4 Estimación fasorial

El diseño de este circuito se basa en el algoritmo de la **Figura 3-5** con algunas diferencias, dado que la estimación se realizará en tiempo real.

Primero los datos mostrados en la **Tabla 3-1**, donde a y b representan un valor real e imaginario, se guardan cada uno en dos memorias de 8 localidades. Los datos de la segunda fila denominados "a" se guarda en un arreglo llamado *re* y b se guarda en un arreglo llamado *im*.

Los arreglos *im* y *re* se multiplica con otra memoria de 8 localidades, que representa la ventana deslizante de los datos de entrada, para replicar la **ec. 2.11** de la teoría de la DFT. Se obtienen dos nuevos arreglos que representan la parte real e imaginaria de los fasores estimados $\hat{\xi}_1$, para resultado de $\hat{a}$ será necesario obtener el módulo de este valor. Estas operaciones se repiten cada que el sistema recibe una nueva muestra.

Para la implementación de la etapa de medición fasorial se presenta el diagrama de bloques con su respectivo divisor en la **Figura 5-13**.

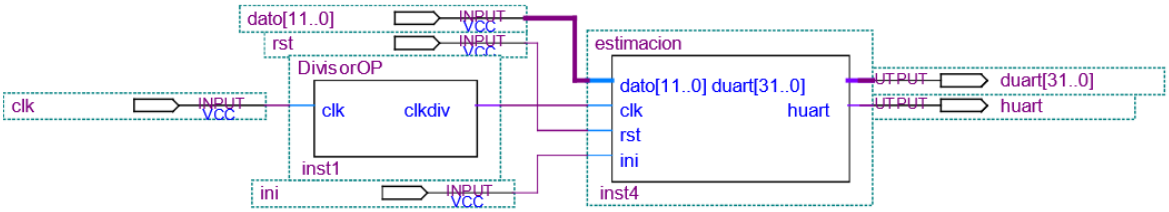


Figura 5-13. Diagrama de bloques estimación fasorial.

Su entidad se presenta en la **Tabla 5-10**:

Tabla 5-10. Entidad estimación fasorial.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
Clk	Entrada	1	Lógico	Reloj de entrada
Dato	Entrada	12	Binario	Contiene el dato recibió del registro serie paralelo
Ini	Entrada	1	Lógico	Habilita la captura de datos
Rst	Entrada	1	Lógico	Al recibir un "1" lógico se reinicia el circuito
Huart	Salida	1	Lógico	Indica que los datos de salida están listos
Duart	Salida	31	Binario	Registro que recibe los datos que se enviaran por UART

Esta etapa fue implementada con una carta ASM, como se muestra en la **Figura 5-14**.

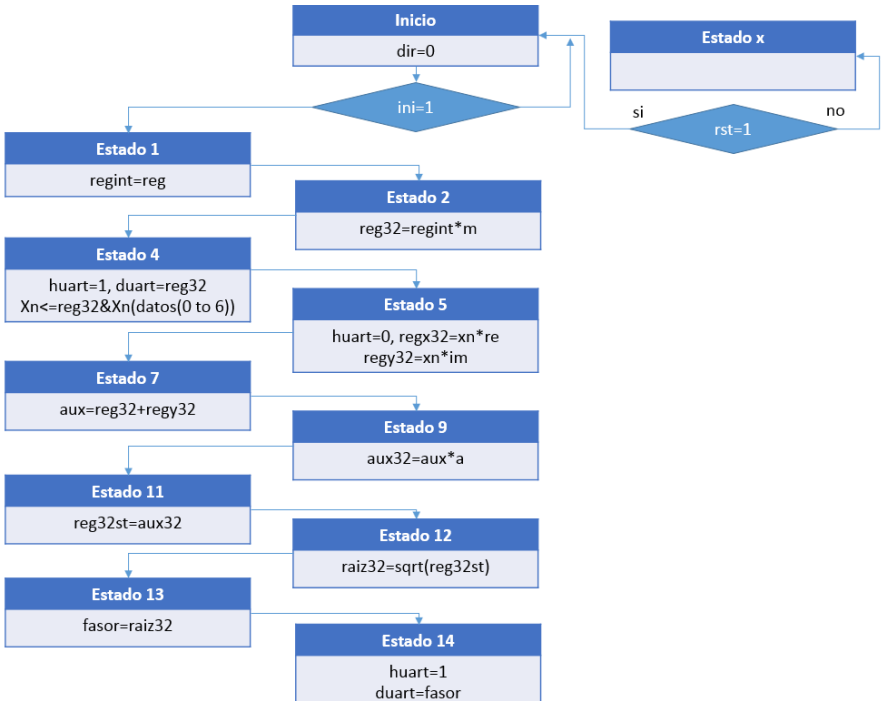


Figura 5-14. Carta ASM de transformada discreta de Fourier.

Este circuito contiene un puerto de entrada en paralelo de 12 bits y un puerto de salida de 32 bits, ambos utilizan un formato de punto fijo; sólo en el caso del cálculo del módulo del fasor se utilizó formato en punto flotante.

En el estado *inicio*, todos los registros internos y puertos de salida se mantienen en cero. El sistema se mantiene en el estado de *inicio* mientras que puerto *ini*=0, cuando el puerto *ini* recibe 1, el registro llamado *reg* recibe el valor de los datos del ADC y el sistema avanza al *Estado 1*.

Para el *Estado 1*, en el dato de entrada se realiza un desplazamiento hacia la izquierda de 16 bits y un desplazamiento hacia la derecha de 4 bits con el fin de poder interpretarlo en formato punto fijo signado con 16 bits de parte de entera y 16 de parte fraccionaria.

Continuamos con el *Estado 2*, donde se realiza una conversión de los 32 bits con la finalidad de obtener una cantidad real de corriente, considerando el valor de cuantificación en el ADC y el factor de transformación del sensor CSNA111. El resultado convertido se guarda en el registro nombrado *reg32*.

Luego, se crea una ventana de 8 muestras en el *Estado 4*. En el circuito, se utiliza la lógica de un buffer tipo *FIFO* implementado en una memoria conformada por un número de localidades igual al número de muestras por ciclo de la señal, *reg32* se posiciona en la primera localidad de memoria, y cada nuevo ciclo esta memoria se actualiza recorriendo los datos.

Después, en el *Estado 5* se realiza la multiplicación de la matriz que representa la ventana deslizante de datos *Xn* con los arreglos *re* e *im*. Se obtendrán un valor que representa el valor real, nombrado *regx32* y otro representando el valor imaginario, llamado *regy32*.

Con estos dos valores, *regx32* y *regy32*, se obtiene el módulo del fasor estimado en los siguientes estados, para calcular la raíz cuadrada se replica una arquitectura que utiliza formato en punto flotante, por esta razón es necesario su respectivo cambio de formato, una vez realizada la operación de la raíz cuadrada, módulo del fasor se vuelve a interpretar en formato punto fijo, *Estado 13*, para un manejo más rápido en la clasificación SVM y posteriormente en la interfaz gráfica.

Diseñado el circuito para obtener la magnitud del fasor estimado en 60 Hz a partir de 8 muestras de la señal de corriente, es necesario, ahora, usar estos valores en el circuito clasificador de fallas basado en SVM para determinar si existe o no falla en la señal.

5.3.5 Implementación de la SVM

En esta etapa se describe la implementación en el FPGA del modelo del sistema de clasificación baso en SVM entrenado y validado en Matlab. Como resultado del modelo antes mencionados se obtuvieron, el vector de pesos w y la constante de bias b mostradas en la **Tabla 4-4**. Estas componentes son guardadas en una memoria llamada w y registro llamado b.

La entrada de este circuito es un coeficiente estimado en formato de punto fijo de 32 bits y la salida corresponde a la etiqueta 1 o 0. Nuevamente, la lógica FIFO se usa en una memoria de 8 ubicaciones para almacenar los fasores estimados y actualizarlos continuamente. La **ec. 2.19** se ejecuta en tiempo real con la operación MAC de los datos de la memoria w con los fasores obtenidos de la implementación DFT y la adición del registro b. Cuando se ha calculado este valor, es posible definir falla o no falla en la señal y asignar este bit a un led FPGA.

En la **Figura 5-15**, se observa el divisor correspondiente y el bloque utilizado para implementar el clasificador.

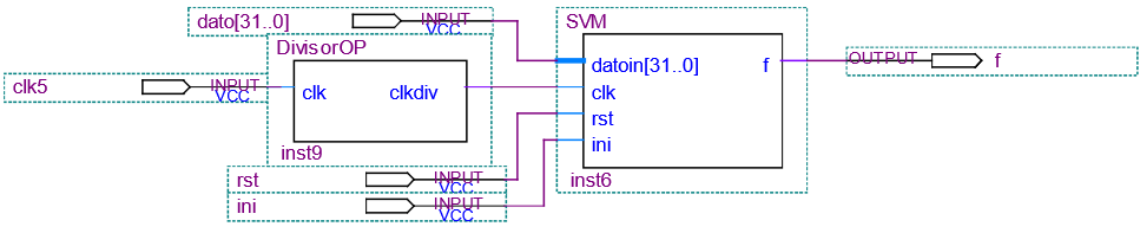


Figura 5-15. Diagrama de bloques de clasificador SVM.

Para la implementación del clasificador se propone el siguiente circuito, donde la entidad está definida con la siguiente **Tabla 5-11**:

Tabla 5-11. Entidad del circuito clasificador.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
F	Salida	1	Lógico	Envía un “1” al detectar una falla y un “0” mientras no exista.
Ini	Entrada	1	Lógico	Habilita la entrada de datos ante un “1” lógico
Rst	Entrada	1	Lógico	Al recibir un “1” lógico se reinicia el circuito
Datoin	Entrada	32	Punto fijo Signado Q16	Datos obtenidos de la estimación
Clk	Entrada	1	Lógico	Señal de reloj obtenida del divisorOP

La siguiente carta ASM mostrada en la **Figura 5-16** expresa el funcionamiento del circuito.

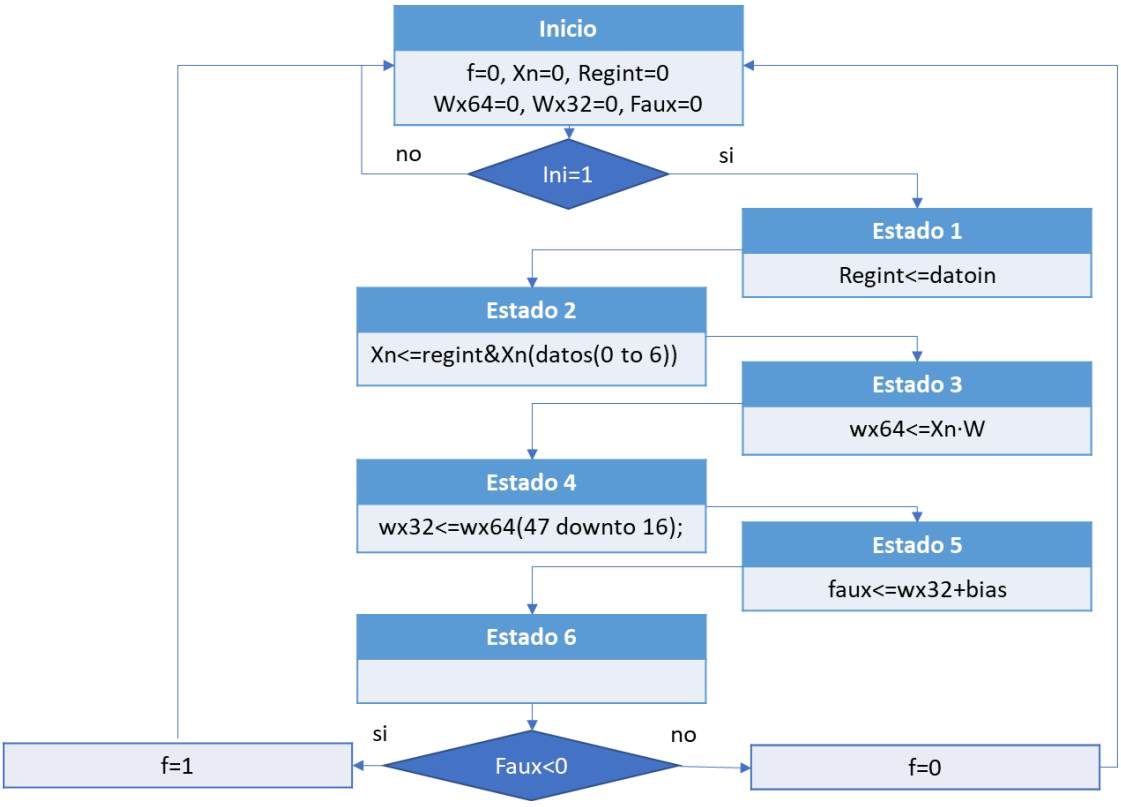


Figura 5-16. Carta ASM de clasificador SVM.

Hasta esta sección se explicó el diseño de los circuitos necesarios para el sistema clasificador monofásico, ahora es necesario explicar el circuito que realiza la comunicación entre el sistema y una computadora, con el fin de poder mostrar los fasores y los disparos de falla por medio de una interfaz gráfica. La comunicación

antes mencionada se realiza mediante el protocolo UART y su implementación se explica en la siguiente sección.

5.3.6 Protocolo UART

Para la comunicación entre FPGA y la interfaz gráfica, se diseñó un circuito que funciona como protocolo UART, con un puerto paralelo que recibe 32 bits y una salida en serie como Tx. El diagrama de bloques en **Figura 5-17** ilustran la implementación de esta comunicación y su respectivo divisor de frecuencia.

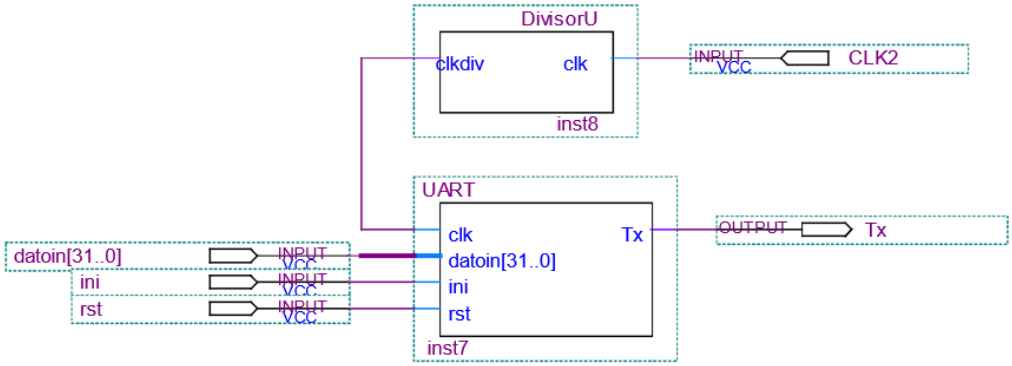


Figura 5-17. Diagrama de bloques protocolo UART.

Cabe anotar que el bloque UART en **Figura 5-17** sigue la parametrización de la **Tabla 5-12**:

Tabla 5-12. Entidad circuito UART.

Nombre	Direccionalidad	Tamaño [bits]	Formato	Descripción
Clk	Entrada	1	Lógico	Reloj de entrada
datoin[31..0]	Entrada	32	Binario	Contiene el dato recibió la estimación
Ini	Entrada	1	Lógico	Habilita la captura de datos
Rst	Entrada	1	Lógico	Al recibir un “1” lógico se reinicia el circuito
Tx	Salida	1	Lógico	Puerto que transmite los bits

Los datos de entrada de 32 bits contienen el valor del coeficiente estimado; mientras Tx forma una trama de comunicación, guardando un bit de inicio y 8 bits de información, por lo tanto, se requieren 4 tramas de datos para enviar un coeficiente estimado a la interfaz gráfica. El bit de inicio Tx = 1 representa una comunicación nula, por lo que la comunicación comenzará cuando en Tx se le asigne el valor cero.

La carta ASM de la **Figura 5-18** expresa el proceso del envío de las 4 tramas de datos, el circuito divide el dato recibido en 4 registros llamados RegA, RegB, RegC y RegD.

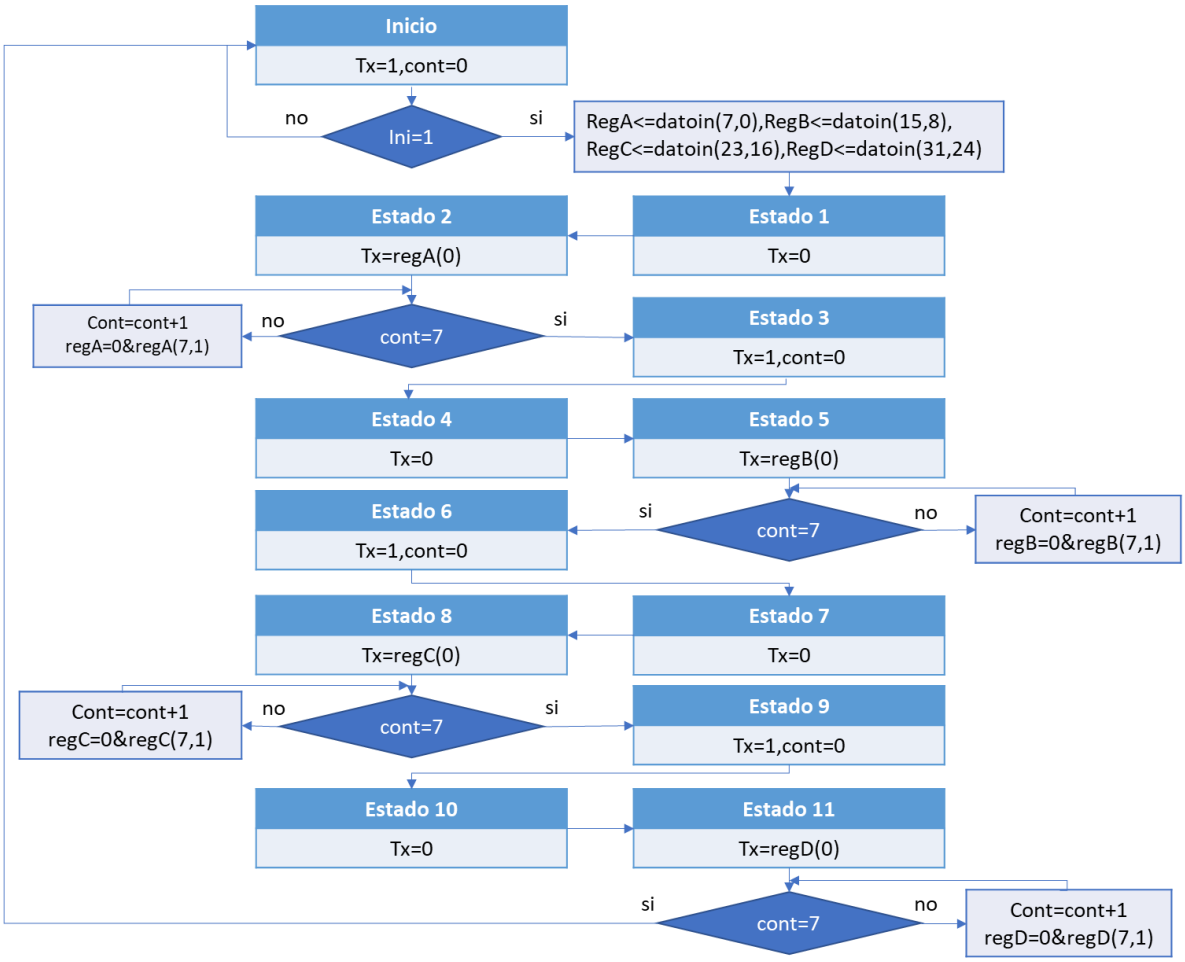


Figura 5-18. Carta ASM de protocolo UART.

Los circuitos explicados anteriormente unidos constituyen un sistema clasificador de fallas monofásicas, capaz de enviar datos mediante UART a una interfaz gráfica. Este sistema se replica dos veces más para el diseño del clasificador trifásico, con las consideraciones que se explicaran a continuación.

5.3.7 Circuito clasificador de sistemas trifásicos

En la **Figura 5-19** se muestra el sistema clasificador de fallas eléctricas para un sistema trifásico, i.e. para sensar tres señales de corriente; anteriormente se explicó los bloques necesarios para el funcionamiento de un sistema monofásico (una sola señal de corriente).

Para lograr el objetivo de discriminar las fallas eléctricas en las tres fases es necesario repetir algunos bloques del circuito mencionado. Por lo que se reutilizaron los circuitos ya descritos anteriormente del sistema monofásico para la calificación y se incorporaron en un sistema trifásico. Para ello, las arquitecturas simplemente se repitieron tres veces, una por cada fase, con algunas diferencias, tales como:

- Se unieron los circuitos de “estimación” y SVM en la misma arquitectura, con el fin de evitar el uso excesivo de divisores de frecuencia.
- Se implementó un nuevo circuito llamado “secuenciador” y su respectivo reloj. Su función es multiplexar los circuitos SPI con el objetivo de habilitar una fase y deshabilitar las dos restantes.
- También se implementó un conjunto de circuitos “tri-state” para controlar en qué fase será recibido el dato del ADC.
- Por último, se multiplico por tres la frecuencia del divisor ADC con la finalidad de seguir obteniendo 8 muestras por ciclo de cada fase.

Esto es válido ya que el proceso para clasificar cada señal es el mismo y por ende requiere de las mismas arquitecturas.

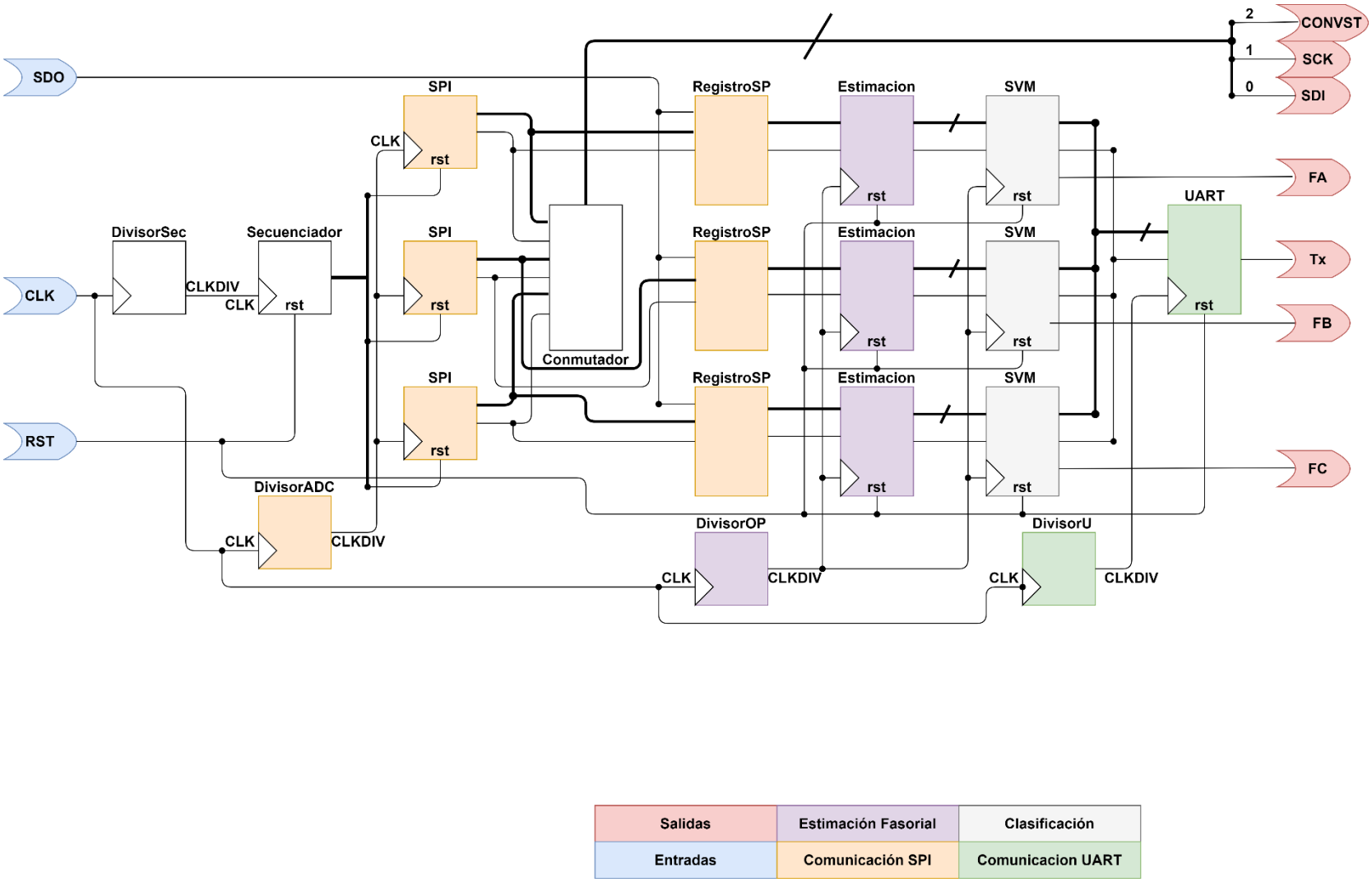


Figura 5-19. Circuito clasificador de señales trifásicas.

Una vez diseñado el sistema clasificador trifásico, es necesario una interfaz gráfica que reciba los datos enviados por el protocolo UART y sea capaz de mostrar los datos obtenidos de los coeficientes estimados y la clasificación de falla.

5.4 Interfaz gráfica

La interfaz gráfica se creó en GUIDE-Matlab®. En la **Figura 5-20** se presenta el diagrama de flujo que permite su funcionamiento.

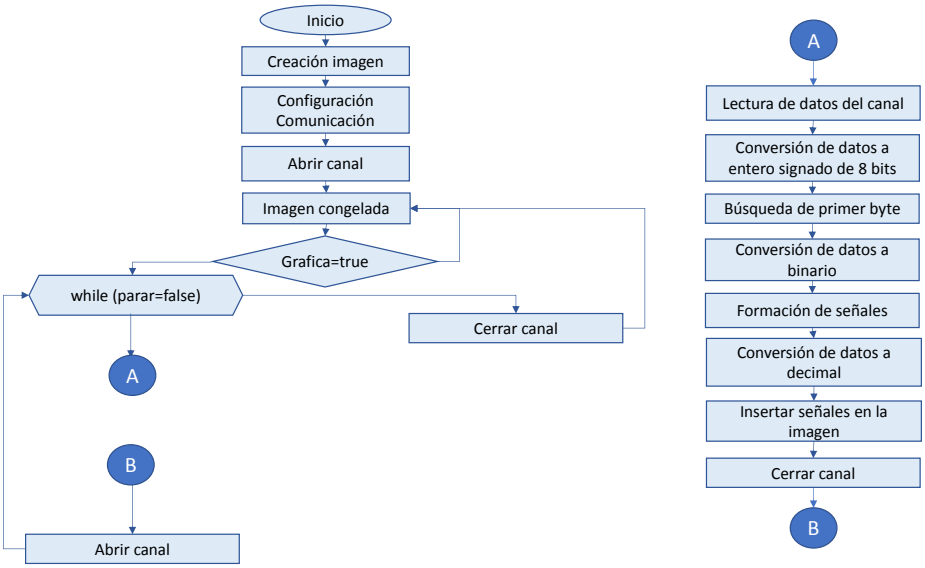


Figura 5-20. Diagrama de flujo interfaz gráfica.

A continuación, se describen los pasos a realizar en el diagrama de flujo:

- Creación de gráficas y figuras, determinación de posiciones y tamaño.
- Configuración del puerto serial, tamaño de matriz de datos de entrada y velocidad en baudios a 115200.
- Se abre el canal para permitir la entrada de datos.
- Existen dos funciones controladas por dos botones. De acuerdo con el botón presionado, se efectúa la función" graficar" o "detener".
- La función "detener", evita que la comunicación continúe, cerrando el canal. La función "graficar" efectúa un ciclo que permite graficar dato por dato.
- Se reciben datos enteros de 8 bits. Cuando comienza la recepción de datos, un bit identificador asignado en MSB ayuda a encontrar la señal de la fase a, como se muestra en la **Figura 5-21**. Una vez encontrados, se agrupan 4 bytes para obtener los primeros datos de la fase a, continuando con la fase

b y luego la fase c; este proceso se repite continuamente hasta que se reconstruyen las señales con apoyo de Matlab ®.

- Se cierra el canal.

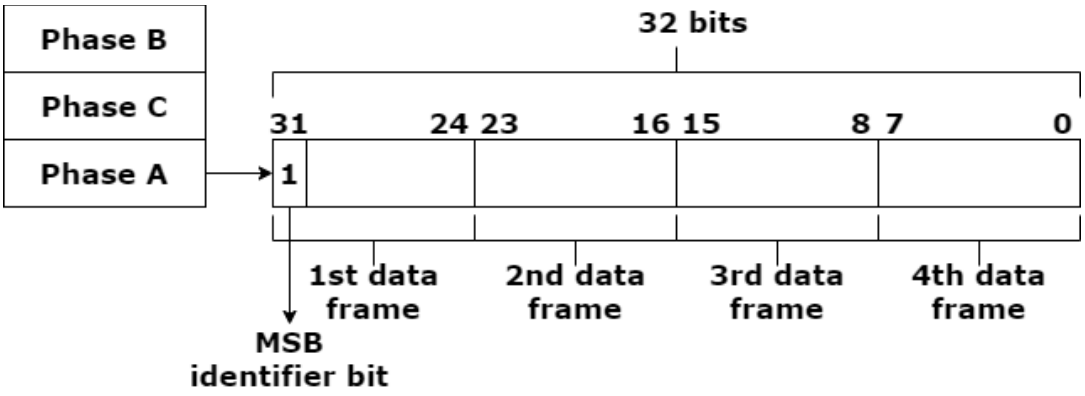


Figura 5-21. Trama de datos en GUI.

Finalmente, en **Figura 5-22** se muestra la interfaz, donde se puede observar la gráfica del fasor de las tres fases en tiempo real, considerando el tiempo de procesamiento de la señal.

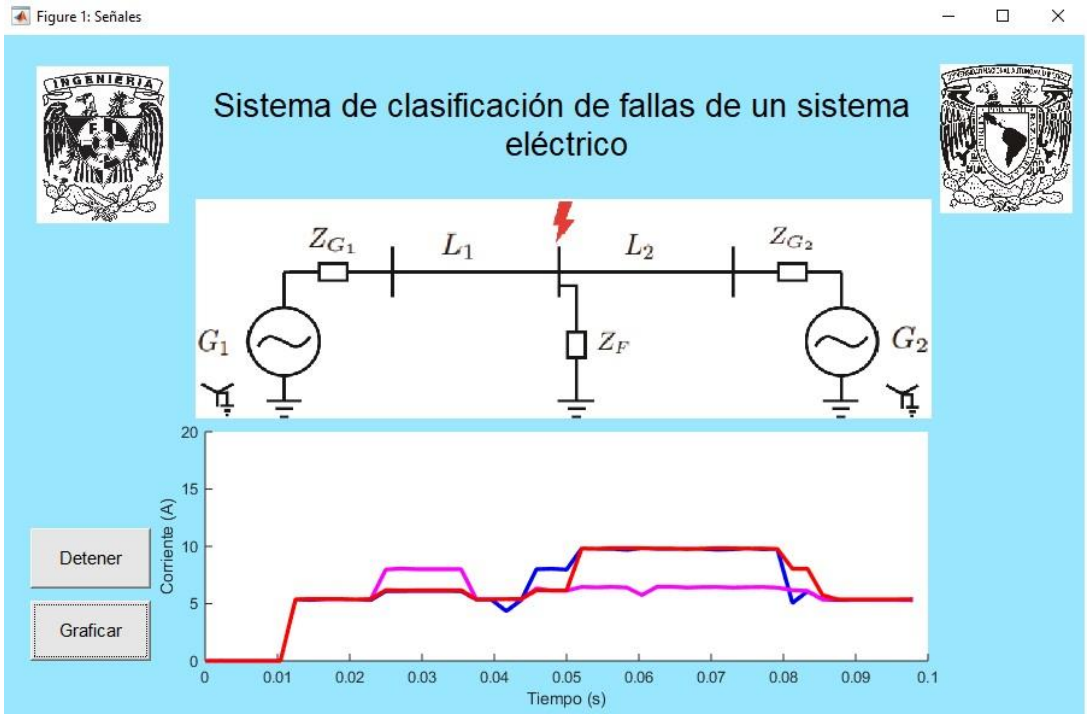


Figura 5-22. Interfaz gráfica.

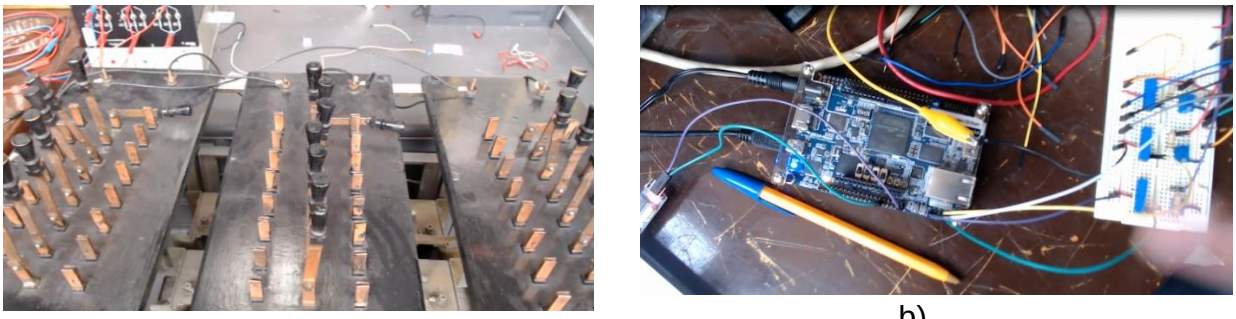
Con el término de este capítulo, se menciona el sistema de prueba utilizado en la implementación del clasificador de fallas trifásico con un algoritmo de aprendizaje supervisado y sus resultados.

6 Análisis de resultados

En este capítulo se presentan los resultados que se obtuvieron durante la realización del trabajo y su respectivo análisis, con los resultados de la implementación mediante la captura de los datos de los fasores y disparos después de diferentes fallas.

6.1 Resultados Implementación

El sistema de prueba conformado por una fuente en AC con un voltaje de 127 [V] por fase y una corriente máxima de 15 [A], se conectó a una carga trifásica en estrella, como se muestra en **Figura 6-1**(a) con un valor de impedancia mínima de 31 [Ω]. Para desbalancear el sistema, se aumentó la carga en la fase donde se deseaba obtener una sobre corriente. Cada sensor de corriente y su circuito de acondicionamiento, **Figura 6-1**(b), se conectó a cada fase para después realizar su procesamiento, clasificación y presentación en la interfaz gráfica.



a) b)

Figura 6-1. Implementación física. a) Carga trifásica, b) FPGA y acondicionamiento

Ahora, se presentan los resultados de los fasores e indicadores de falla obtenidos de la implementación en el FPGA. Se realizó falla monofásica en fase A, B y C, falla bifásica en A y B, por último, una falla trifásica

En **Figura 6-2**(a) puede distinguirse que la falla monofásica ocurre en la fase A al incrementarse la corriente respecto a las otras fases. En **Figura 6-2**(b), se observa que el valor del indicador es igual a “1” en la fase A, y su valor en las fases B y C es igual a “0”. Lo cual representa que la clasificación se realizó de manera correcta. Se muestra un incremento de la corriente en todas las fases, pero el incremento mayor se efectúa en la fase A, por esta razón el indicador sólo clasifica la falla en fase A.

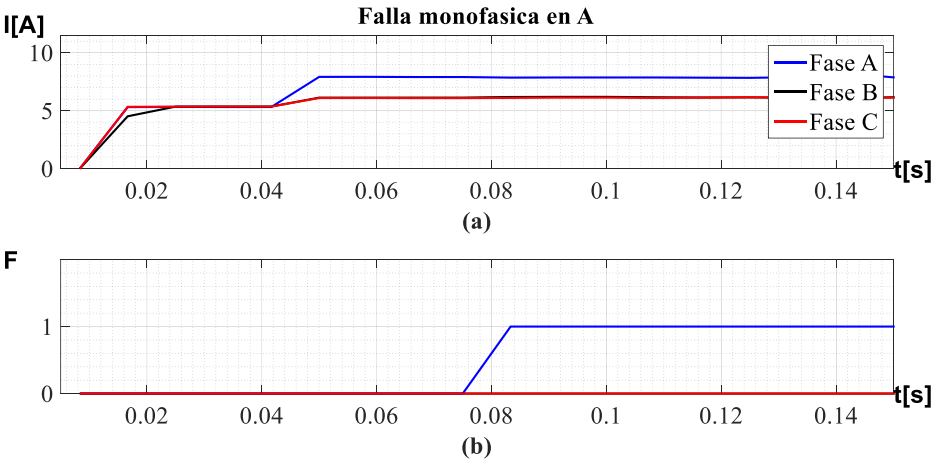


Figura 6-2. Resultados implementación en falla monofásica en A. (a) Fasores de corriente; y (b) señal de clasificación.

En la **Figura 6-3**(a) se muestra la falla monofásica en B y se observa que el comportamiento en los fasores de corriente e indicador de fallas es similar al anterior, en **Figura 6-3**(b) el indicador tiene un valor de uno en la fase B, indicando una falla monofásica en esta fase.

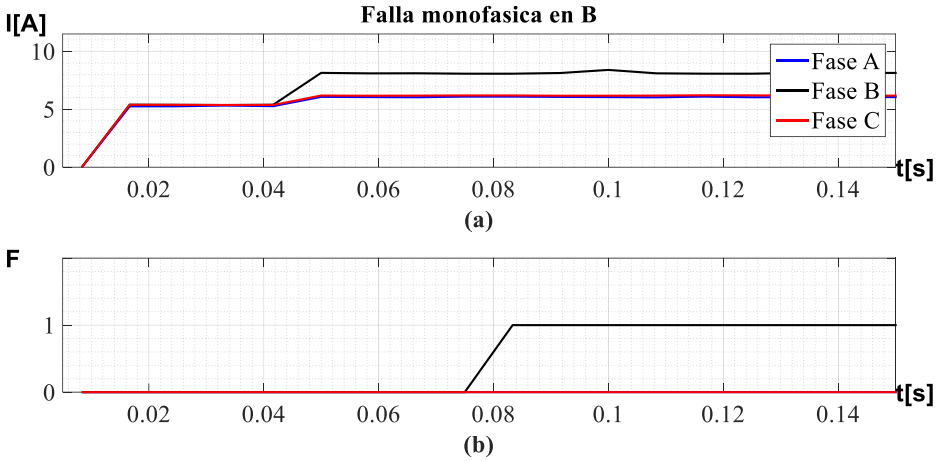


Figura 6-3. Resultados implementación en falla monofásica en B. (a) Fasores de corriente; y (b) señal de clasificación.

Figura 6-4(a) se muestra una falla monofásica en C, la estimación del fasor contiene una mayor distorsión que en los casos anteriores, pero sigue siendo favorable para la correcta clasificación. En la **Figura 6-4**(b). el indicador muestra la fase fallada de manera correcta.

En conclusión, las magnitudes de los fasores de las corrientes son semejantes sin importar en qué fase se efectúa la falla.

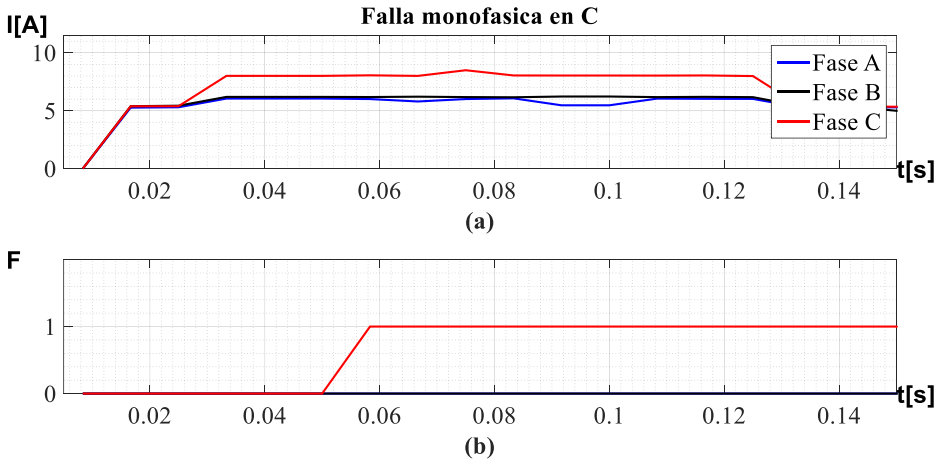


Figura 6-4. Resultados implementación en falla monofásica en C. (a) Fasores de corriente; y (b) señal de clasificación.

Ahora se presenta la falla bifásica en B y C. En la **Figura 6-5**(a) se observa que los fasores de corriente en estas fases son similares ya que la carga de desbalance utilizada fue la misma en ambas fases. El tiempo de respuesta del indicador, en la **Figura 6-5**(b), para las fases falladas es el mismo en tiempo de respuesta.

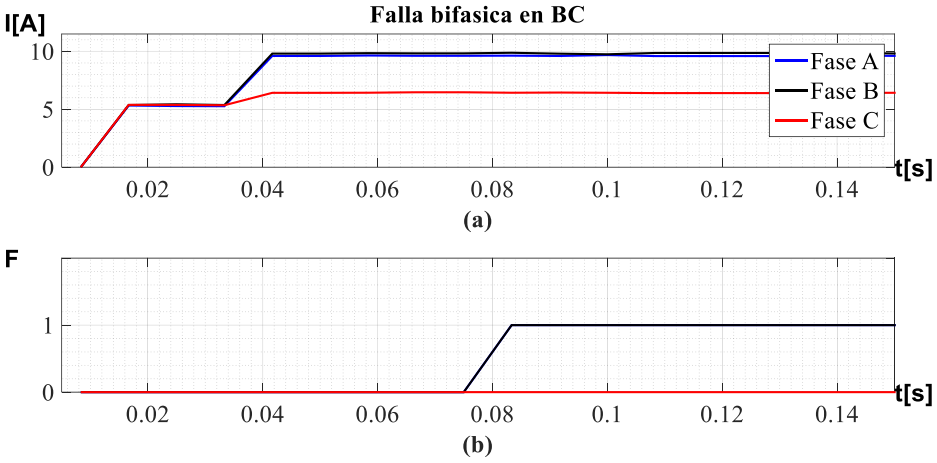


Figura 6-5. Resultados implementación en falla bifásica en B y C. (a) Fasores de corriente; y (b) señal de clasificación.

Para finalizar, en la **Figura 6-6**(a) se muestra la falla trifásica, todos los fasores de corrientes incrementan a un valor similar y al mismo tiempo. Sin embargo, en **Figura 6-6**(b) se observa que el tiempo de respuesta del indicador es diferente entre fases, ya que la fase C es más rápida comparándola con la fase B y C.

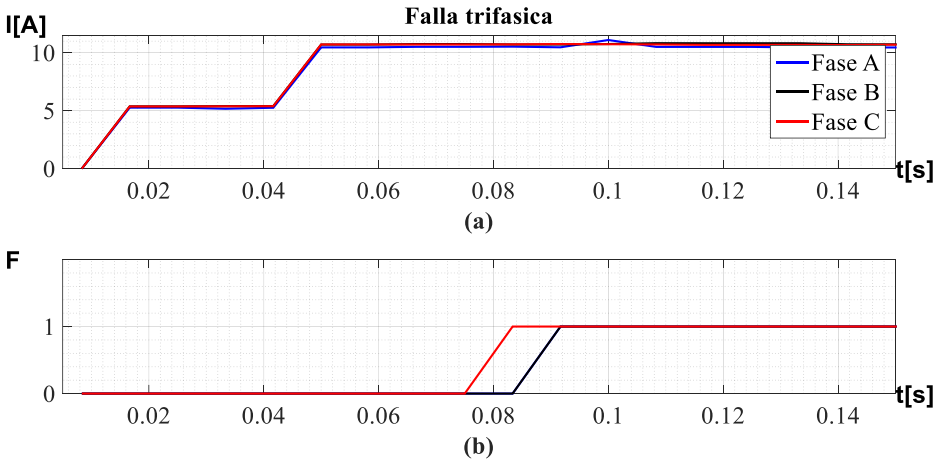


Figura 6-6. Resultados implementación en falla trifásica. (a) Fasores de corriente; y (b) señal de clasificación.

6.2 Comparación de resultados con relevador comercial

Para validar los resultados experimentales, se utiliza un sistema de protección SEL-351A. De este modo, tanto el dispositivo propuesto como el relé comercial se comparan bajo los mismos escenarios que consisten en cortocircuitos monofásicos en las fases B y C, como se ilustra en la **Figuras 6-7 y Figura 6-8**. Se observa en **Figura 6-7 (a)** y **Figura 6-8 (a)**, que las estimaciones de amplitud de corriente muestran magnitudes similares. Sin embargo, el relé comercial exhibe una ligera oscilación que no se presenta en el dispositivo propuesto. Las señales F también son similares en ambos casos, **Figura 6-7 (b)** y **Figura 6-8 (b)**, existiendo menos de 5 ms de diferencia de tiempo entre los dispositivos para la clasificación de fallas.

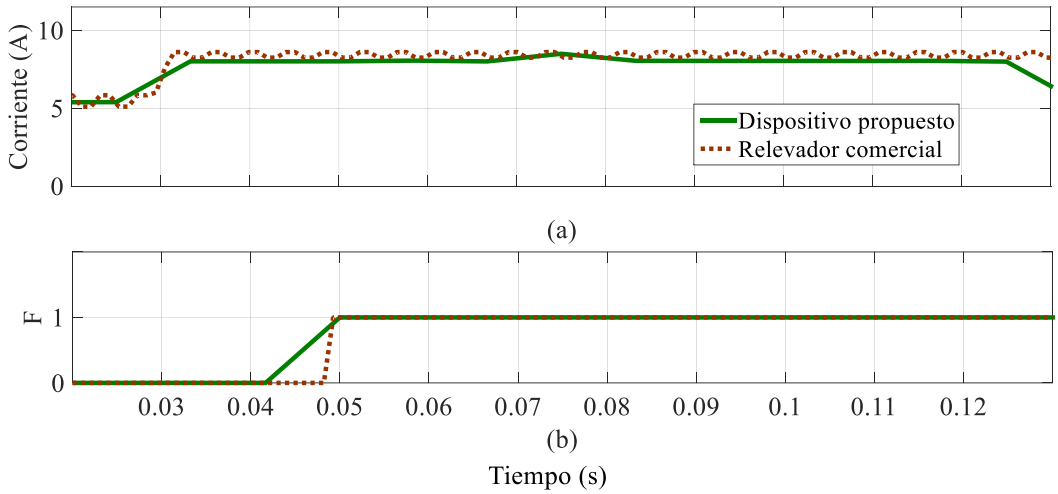


Figura 6-7. Falla monofásica en fase B. (a) Amplitud de fasor de corriente. (b) Clasificación de fallas con los valores F iguales a 1 para la fase B.

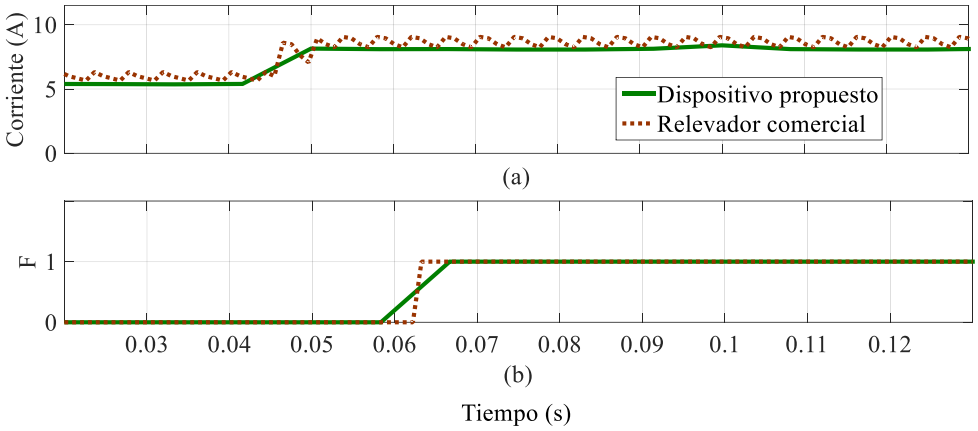


Figura 6-8. Falla monofásica en fase C. (a) Amplitud de fasor de corriente. (b) Clasificación de fallas con los valores F iguales a 1 para la fase C.

7 Conclusiones y trabajos futuros

7.1 Conclusiones

La implementación eficiente de un dispositivo clasificador de señales requiere, ante todo, del correcto desarrollo de algoritmos que satisfacen necesidades específicas; la DFT y la SVM otorgan las características adecuadas para este fin.

En el caso de la DFT, el proceso para la realización de las operaciones demandó un menor peso computacional, ya que, únicamente se utilizó la componente fundamental de las senoidales de corriente para la obtención de un fasor, sin considerar sus armónicos, permitiendo un procesamiento de la señal más rápido y enfocado a los requerimientos del dispositivo, y lograndose un error de estimacón inferior al 5%. Además, es conveniente destacar que el proceso de clasificación se realiza de manera más oportuna mediante el fasor obtenido de la DFT en lugar de los datos de las señales senoidales.

Por otro lado, el algoritmo de la SVM requiere de un adecuado entrenamiento para la correcta clasificación de nuevos valores, en este trabajo se realiza el entrenamiento con ayuda de datos obtenidos del simulador PSCAD/EMTDC™ mediante un sistema de potencia de dos buses y valores distintos en la resistencia de falla, estos casos de estudios fueron suficientes para la adquisición de resultados satisfactorios en la validación de la SVM mediante Matlab™ y para su posterior implementación en un sistema distinto al simulado, lograndose una precisión del 99%. Otro punto por destacar es que es el dispositivo funciona en cualquier sistema cuando se ingresa un valor relativo de la corriente en estado estable, esto significa que solo este valor es necesario conocer en el sistema para la clasificación de la falla.

Es importante mencionar que el vector de pesos y la constante bias obtenidas en Matlab™ se utilizan en la implementación, permitiendo de este modo, comprobar la adaptabilidad del clasificador al realizarse pruebas físicas en un sistema distinto al simulado con una resistencia ubicada entre el rango utilizado para el entrenamiento, específicamente de 0 a 70 Ω, esto significa que existe la limitante en fallas de alta impedancia.

Además de la adaptabilidad que posee el dispositivo gracias a la SVM entrenada, se logró comprobar que el clasificador funciona correctamente a pesar de que exista un incremento en todas las corrientes del sistema, identificando correctamente las fases falladas.

Finalmente, los procesos ya mencionados, DFT y SVM, al ejecutarse en una arquitectura de FPGA de forma paralela, esto implica una mayor velocidad en comparación con un microcontrolador, en este caso, el procesamiento y clasificación involucran 3 señales, pero esto significa una mayor ventaja entre más

señales se requiera clasificar y gracias a la velocidad del proceso se puede nombrar el dispositivo como un sistema con funcionamiento cercano al tiempo real.

7.2 Trabajos futuros

La implementación final se realizó en un sistema SOC, que une un FPGA y micro-controlador ARM. Si bien en el FPGA se puede implementar cualquier sistema digital, el ARM puede auxiliar al diseño con interfaces especializadas y mejorar aspectos de la implementación como: la comunicación y el almacenamiento de datos. Usando protocolos de comunicación óptimos y más complejos que UART y el uso de un servidor con base de datos. El trabajo futuro sería implementar este clasificador en conjunto con el ARM y no sólo con FPGA.

Para obtener todos los beneficios que el algoritmo de SVM puede ofrecer como clasificador de fallas, se pueden proponer diferentes formas de usar el algoritmo como distintas variables de entrada (tales como: el voltaje, la fase de la corriente, etc.), hasta usar una SVM multi-clases (en lugar de repetir tres clasificadores binarios). Comparando su desempeño y así elegir la forma de implementar más eficiente.

En los sistemas eléctricos de protección, además de detectar y clasificar, es importante localizar una falla eléctrica. Como trabajo futuro se propone el uso de la SVM para localizar fallas en un sistema fijo.

Es importante destacar que el uso del FPGA acarrea consigo ciertos inconvenientes como la complejidad de la configuración, una desventaja de los FPGA es que no están optimizados para el consumo de energía dada sus velocidades de procesamiento, es por eso que muchos sistemas FPGA utilizan un microcontrolador en la placa para aumentar la eficiencia energética y aprovechar ventajas del microprocesador y FPGA.

Referencias

[1] C. H. J. H. Kunjin Chen, «Fault detection, classification and location for transmission lines and distribution systems: a review on the methods,» *High Volt,* vol. 1, pp. 25-30, 2016.

[2] Rodriguez Suárez J., Detección y localización de fallas en los sistemas de energía mediante la técnica máquinas de soporte vectorial, Colombia: Universidad Industrial de Santander, 2006.

[3] B. C. R. V. T. R. Morales España G., « Ubicación única de fallas en sistemas de distribución por medio de zonas con SVM,» *Revista Facultad de Ingeniería Universidad de Antioquia,* nº 47, pp. 187-196, 2009.

[4] B. M. a. J. G. Zhu, «Fault classification and faulted phaseselection based on the symmetrical components of reactive power forsingle-circuit transmission lines,» *IEEE transactions on power delivery,* vol. 28, nº 4, p. 2326–2332.

[5] A. R. a. R. Adhami, «A fault detection and classification tech-nique based on sequential components,» *IEEE Transactions on IndustryApplications,* vol. 50, nº 6, p. 4202–4209, 2014.

[6] C. Y. Y. T. Y. L. Guomin Luo, «Transient signal identification of HVDC transmission lines based on wavelet entropy and SVM,» *The 14th IET International Conference on AC and DC Power Transmission,* vol. 2019, pp. 2414-2419., 2019.

[7] Guillén D., Arrieta Paternina M., Ortiz-Bejar J., «Fault detection and classification in transmission lines,» *IET Generation, Transmission & Distribution,* vol. 12, pp. 4070-4078, 2018.

[8] S. G. S. M. R. a. M. J. S. Hasheminejad, «Traveling-wave-based protection of parallel transmission lines usingteager energy operator and fuzzy systems,» *IET Generation, Transmis-sion & Distribution,* vol. 10, nº 4, p. 1067–1074, 2016.

[9] A. Y. a. A. Swetapadma, «Enhancing the performance of trans-mission line directional relaying, fault classification and fault locationschemes using fuzzy inference system,» *IET Generation, Transmission& Distribution,* vol. 9, nº 6, p. 580–591, 2015.

[10] C. G. G. S. a. G. R. J. Upendar, «Pso and ann-based fault classification for protective relaying,» *IET generation, trans-mission & distribution,* vol. 4, nº 10, p. 1197–1212, 2010.

[11] S. Seyedtabaii, «Improvement in the performance of neural network-based power transmission line fault classifiers,» *IET generation, trans-mission & distribution,* vol. 6, nº 8, p. 731–737, 2012.

[12] H. Fathabadi, «Novel filter based ann approach for short-circuit faultsdetection, classification and location in power transmission lines,» *In-ternational Journal of Electrical Power & Energy Systems,* vol. 74, p. 374–383, 2016.

[13] A. J. a. J. M. A. Yusuff, «Determinant-based feature extrac-tion for fault detection and classification for power transmission lines,» *IET generation, transmission & distribution,* vol. 5, nº 12, p. 1259–1267, 2011.

[14] B. D. A. a. S. Ridella, «A digital architecture for supportvector machines:theory, algorithm, and fpga implementation,» *IEEEtransactions on neural networks,* vol. 14, nº 5, p. 993–1009, 2003.

[15] C.-L. C. Y.-C. W. Joe-Air Jiang, «A hybrid framework forfault detection, classification, and location—part ii: Implementation and test results,» *IEEE transactions on power delivery,* vol. 26, nº 3, p. 1999–2008, 2011.

[16] M. P. a. C.-S. Bouganis, «A scalable fpga architecturefor non-linear svm training,» *IEEE transactions on power delivery,* p. 337–340, 2008.

[17] J. S. A. W. Waldemar R., Digital Signal Processing in Power System Protection and Control, Londres: Springer, 2011.

[18] A. S. H. M. Y. A. M. P. R. H. A. H. F. A. H. &. S. R. Ahmad, «A review on applications of ANN and SVM for building,» *Renewable and sustainable energy reviews,* vol. 33, pp. 112-119, 2014.

[19] P. Anderson, Analysis of Faulted Power Systems, United State of America, 1999.

[20] G. Y. STEVENSON, Análisis de Sistemas de Potencia, México: McGraw Hill, 1998.

[21] CENACE, «Niveles de cortocircuito de la Red Nacional de Transmisión,» México, 2019.

[22] S. J. W. A. Rebizant W., Digital Signal Processing in Power System Protection and Control, Londres: Springer, 2011.

[23] CFE, «Ajustes de protecciones eléctricas de las unidades generadoras, transformadores de unidad e interruptor de potencia,» México, 2019.

[24] SEL-551, «Manual de Instrucciones-Relé de sobrecorriente,» 2017.

[25] A. Phadke y J. Thorp, Synchronized Phasor Measurements and Their Applications, New York: Springer, 2008.

[26] C.-J. R., «A Matlab and Power Factory – based WAMS Simulator,» 2019.

[27] A. V. Oppenheim y R. W. Schafer, Tratamiento de señales en tiempo discreto, Madrid: Pearson Educacion, 2011.

[28] J. G. Proakis y D. G. Manolakis, Tratamiento digital de señales, Madrid: Pearson Educations, 2007.

[29] L. Wang, Support Vector Machine: Theory and Applications, Singapore: Springer, 2005.

[30] E. Carmona, Tutorial de máquinas de soporte vectorial (SVM, Madrid: UNED, 2014.

[31] B. Scholkopf y A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, London, England: The MIT Press, 2002.

[32] M. S. S. a. Control, «Solid State Sensors CSN Series,» Honeywell, USA, 2005.

[33] DE10-Nano, «Getting Started Guide,» 2017.

[34] DE10-Nano, «User Manual,» 2018.

[35] O. Gazi, A Tutorial Introduction to VHDL Programming, Ankara. Turkey: Springer, 2019.

[36] R. Woods, J. McAllister, G. Lighbody y Y. Yi, FPGA-based Implementation of Signal Processing Systems, WILEY, 2017.

[37] A. A. Bazil Raj, FPGA-Based Embedded System Developer's Guide, New York, USA: CRC Press, 2018.

[38] M. Thair y K. Javed, ARM Microprocess Systems: Cortex -M Architecture, Programming, and Interfacing, New York, USA: CRC Press, 2017.

[39] P. Anderson, Power System Protection, United State of America, 1999.

[40] CFE, «Curso: Fundamentos de Sistemas Eléctricos de Potencia,» Primera Parte, México, 2010.

[41] I. J. R. E. Saha M.M, Fault Location on Power Networks, Londres: Springer, 2010.

[42] H. E. Gers J.M, Protection of Electricity Distribution Networks, United Kingdom., 2011.

[43] H. Saadat, Power System Analysis, United State of America, 1999.

[44] J. Glover, Power System - Analysis and Design, United State of America., 2012.

[45] C. D. Paulo.F.Ribeiro, Power Systems Signal Processing for smart grids, Wiley, 2014.

[46] R. Herbrich, Learning Kernel Classifiers Theory and Algorithms, London, Englad: The MIT Press, 2002.

A Anexos

A.1. Simulaciones en ModelSim

Para comprobar el funcionamiento correcto del dispositivo, se hicieron tres simulaciones en ModelSim:

1. Dedicada a los circuitos utilizados para la comunicación del ADC con el FPGA y la obtención de los datos.
2. Procesamiento y clasificación de los datos.
3. Comunicación UART.

Para facilitar las simulaciones, en cada una, el reloj principal es de 100 [ns] con un ciclo de trabajo del 50%.

-Comunicación ADC. En esta simulación, se observa cómo se comunica el FGPA con el ADC, para obtener los datos de los tres canales. Como ya se detalló, la FPGA envía un 1 lógico al puerto CONVST, para iniciar la comunicación, después el ADC recibe la configuración del canal por medio del puerto SDI, el tercer puerto SDO envía los bits del ADC a la FGPA, y finalmente la FPGA le envía al ADC pulsos para sincronizar la recepción de datos y él envió de la configuración del ADC por medio del puerto SCK.

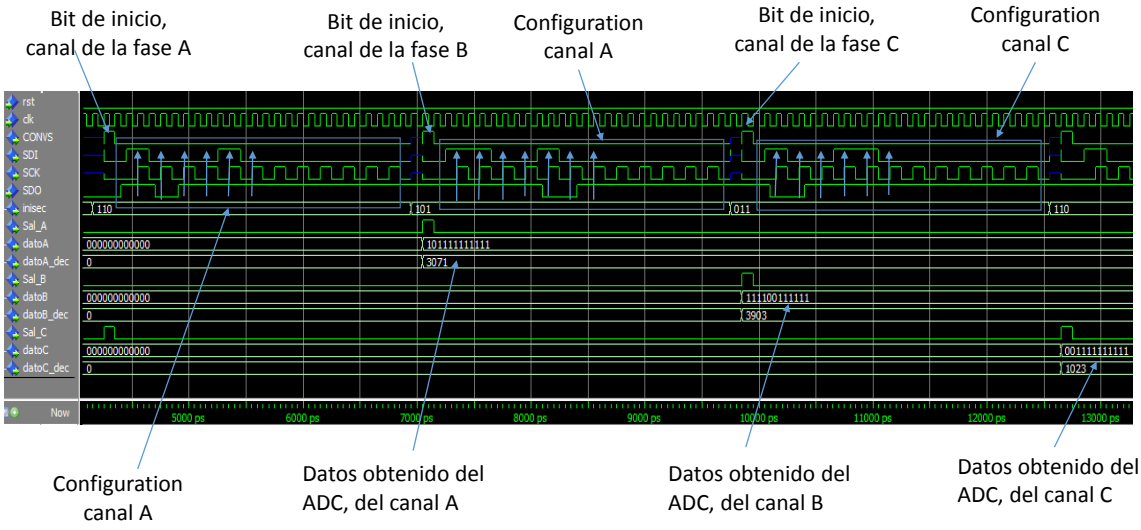


Figura A-1. Simulación interfaz SPI.

En la **Figura A-1** se señala en el diagrama de tiempos obtenido con la simulación, los bits de inicio en CONVST, la configuración y la obtención de datos de cada canal. Para entender mejor la figura, cada flecha azul horizontal significa un flanco de subida en el puerto SCK. En la configuración del canal A, se indica con la primera flecha azul el puerto SDI, que recibe un 1 lógico, al igual que SDO envía un 1 lógico, mientras que con la segunda flecha azul se indica que SDI recibe un 0 lógico y SDO también un 0 lógico, siguiendo la misma dinámica para los demás flancos de subida

en el SCK del canal A, se observa que SDI recibe el dato serial “100010000000” y SDO envía el dato serial “101111111111”, este último es convertido en paralelo y es enviado al circuito de estimación para ser procesado. Siguiendo la misma lógica se observa que SDI recibe un “110010000000” serial para configurar el canal B y envía el dato serial “111100111111” por SDO y para el canal C recibe por SDI el dato serial “100110000000” y envía el dato serial por SDO “001111111111”.

-Estimación y clasificación. Se procede a mostrar el funcionamiento del procesamiento y clasificación de los datos, una vez obtenido el dato del ADC de cada fase, se obtiene la estimación fasorial de la señal de corriente y posteriormente su clasificación, ambos circuitos fueron explicados en sección 5.3.4 y sección 5.3.5 del capítulo 5 .En esta simulación, se prueba con una señal de 17.38 [A], que se muestra en el puerto llamado “signalA” de la ***Figura A-2***, el fasor obtenido de la simulación es mostrado como señal y con su valor decimal, de esta forma se aprecia como el fasor se modifica y con el valor decimal sabemos que al fallar la señal llega a un máximo de 34.9 [A]. También se muestra el disparo de la clasificación de la falla en la variable llamada “FA”, que, si bien no responde al momento exacto de la falla, si responde casi medio ciclo de la señal sinodal simulada.

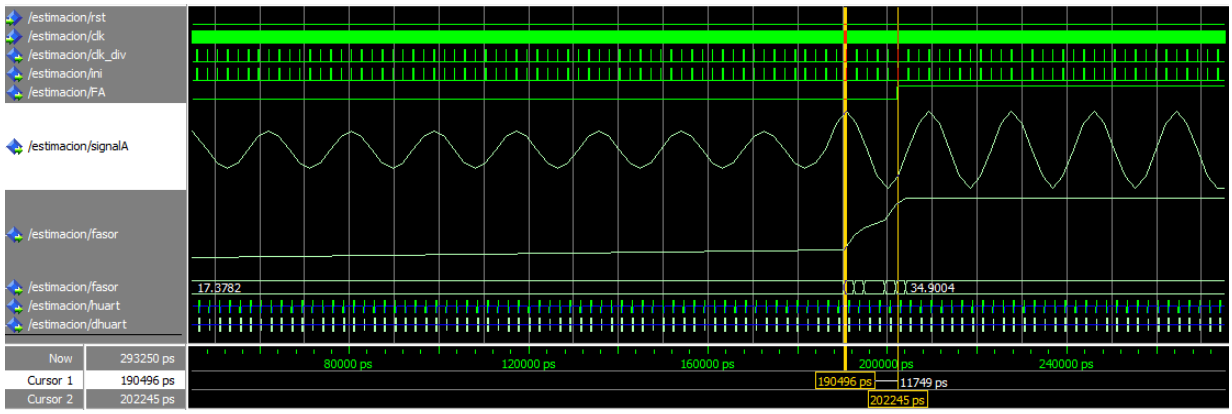


Figura A-2. Procesamiento y clasificación de señales.

También se puede observar, que en los puertos llamados “huart” y “dhuart”, alteman entre verde y azul, cuando está en verde es porque “huart” envía un 1 para habilitar la comunicación de la respectiva fase y el dato enviado al circuito de comunicación se encuentra en “dhuart”. Cuando ambos puertos se encuentran en color azul, significan que se encuentran en alta impedancia y otra fase está activando la comunicación de los datos. El funcionamiento de este circuito de comunicación se muestra a continuación.

-Comunicación UART. Es la última simulación necesaria para mostrar la funcionalidad del dispositivo y el funcionamiento es explicado en la sección 5.3.6, del capítulo 5. El circuito fue probado con el dato “10000001011111110000000101111011”. Mientras que el circuito reciba pulsos de reloj, y el puerto “rst” como “ini” se encuentren en 0 logico, el puerto Tx enviara un

"1" lógico, hasta que reciba un dato en el puerto "dhuart" y un 1 en el puerto "1" de la etapa de procesamiento.

El recibir el pulso de habilitación, Tx se pone en 0 para iniciar la comunicación, y se procede a enviar el dato de forma serial. Como se explicó anteriormente, el dato de entrada es de 32 bits, por lo tanto, este dato divido en 4 registros de 8 bits y se va enviando el byte almacenado en el registro detrás del otro. Como se muestra en la ***Figura A-3***

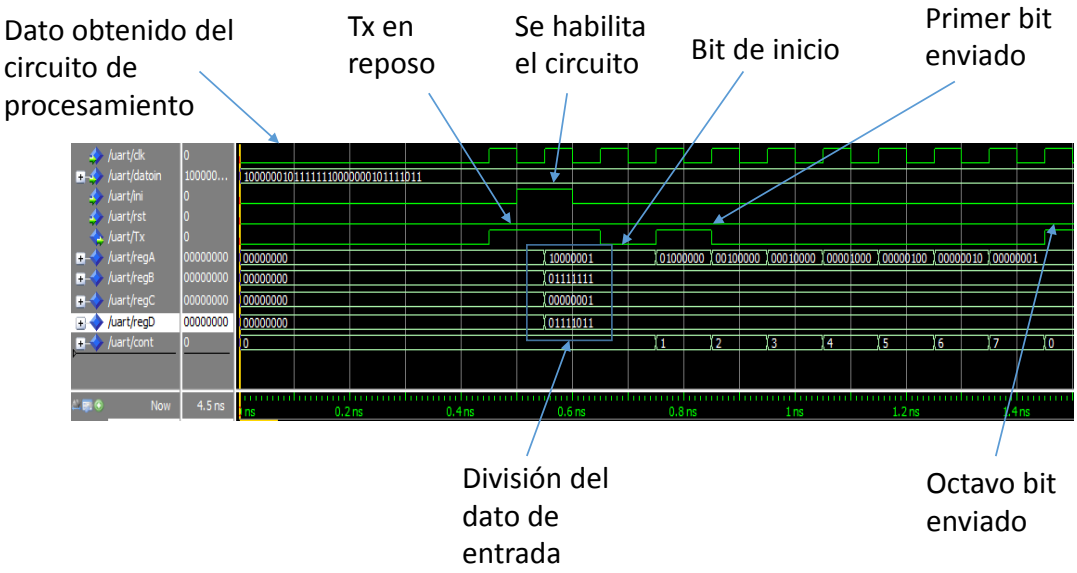


Figura A-3. Simulación comunicación UART.

A.2. Especificaciones del dispositivo propuesto

Tabla A-1. Especificaciones finales del dispositivo

Entrada de voltaje en CA	**Entrada de corriente en CA**
VA, VB, VC: 5 V_{RMS}	IA, IB, IC: 50 A_{RMS}
Fuente de alimentación: 12 Vdc nominal	
Salida de voltaje:	
Fasores (A,B,C) en 60 Hz: 3.3 [V]	Disparos (A, B, C): 3.3 [V]

B Anexos

B.1. Hoja de especificaciones sensor de corriente CSNA111

Solid State Sensors CSN Series

Closed Loop Current Sensors

FEATURES

- Current sensing up to 1200 amps
- Measures AC, DC and impulse currents
- Lowest cost/performance ratio
- Rapid response, no overshoot
- High overload capacity
- High level of electrical isolation between primary and secondary circuits
- Small size and weight

CLOSED LOOP SENSORS

Closed loop current sensors measure AC, DC and impulse currents over 0-25, 0-50, 0-100, 0-600 and 0-1200 Amp ranges. The CSN Series is based on the principles of the Hall effect and the null balance or zero magnetic flux method (feedback system). The magnetic flux in the sensor core is constantly controlled at zero. The amount of current required to balance zero flux is the measure of the primary current flowing through the conductor, multiplied by the ratio of the primary to secondary windings. This closed loop current is the output from the device and presents an image of the primary current reduced by the number of secondary turns at any time. This current can be expressed as a voltage by passing it through a resistor.

CATALOG NUMBER SYSTEM

PLEASE NOTE: This matrix is intended **only** to aid you in identifying sensor catalog listings. It is not all-inclusive, and **must not be used** to form new listings.

Example: CSNA111

CSN Closed Loop Current Sensor

Current Range (Peak/RMS nom.)

A ±70 A/50 A rms nom.
B ±100 A/50 A rms nom.
C ±90 A/50 A rms nom.
D ±22 A/15 A rms nom.
E ±36 A/25 A rms nom.
F ±150 A/100 A rms nom.
J ±600 A/300 A rms nom.
K ±1200 A/500 A rms nom.
L ±600 A/300 A rms nom.
M ±1200 A/500 A rms nom.
P ±90 A/50 A rms nom.
R ±200 A/125 A rms nom.
T ±150 A/50 A rms nom.

Supply Voltage

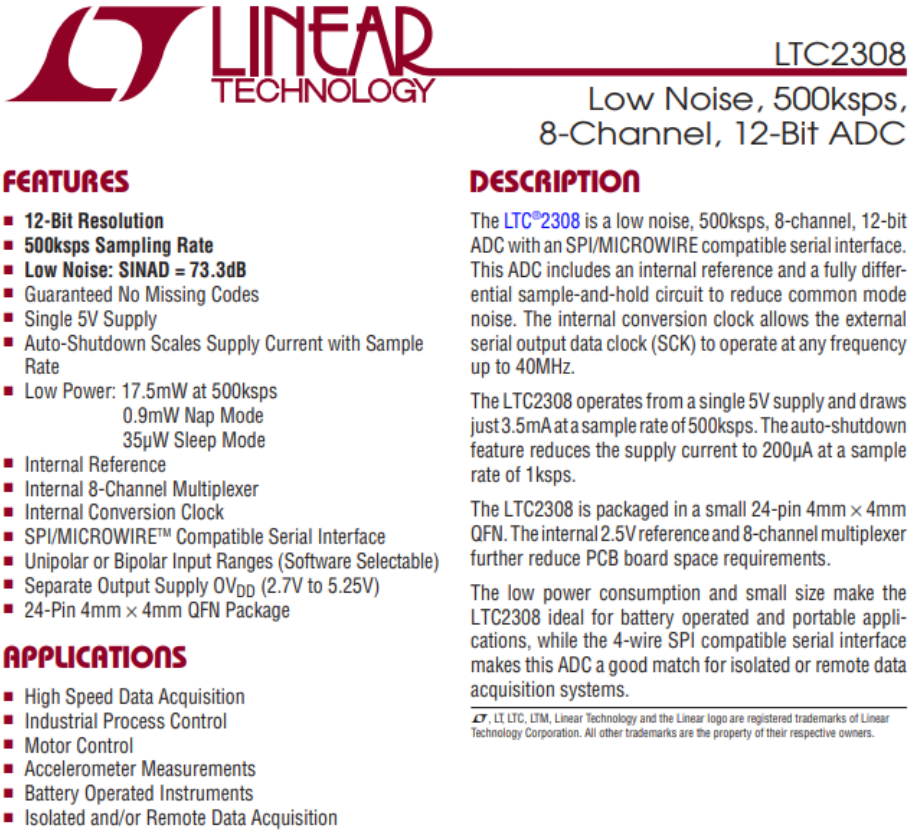
1 ±15 V
2 ±13 V
3 ±5 V
4 ±12 V to 18 V
5 ±15 V to 24 V
6 ±12 V to 15 V

Coil Characteristics

1 1:1000 turns/90 Ω @ 70°C
2 1:2000 turns/160 Ω @ 70°C
3 1:2000 turns/130 Ω @ 70°C
4 1:1000 turns/50 Ω @ 70°C
5 1:1000 turns/110 Ω @ 70°C
6 1:1000 turns/30 Ω @ 70°C
7 1:2000 turns/80 Ω @ 70°C
8 1:2000 turns/25 Ω @ 70°C
9 1:5000 turns/50 Ω @ 85°C

Figura B-1. Hoja de datos del sensor CSNA111

B.2. Hoja de especificaciones ADC LTC2308



LINEAR TECHNOLOGY

LTC2308

Low Noise, 500ksps, 8-Channel, 12-Bit ADC

FEATURES

- **12-Bit Resolution**
- **500ksps Sampling Rate**
- **Low Noise: SINAD = 73.3dB**
- Guaranteed No Missing Codes
- Single 5V Supply
- Auto-Shutdown Scales Supply Current with Sample Rate
- Low Power: 17.5mW at 500ksps
  0.9mW Nap Mode
  35µW Sleep Mode
- Internal Reference
- Internal 8-Channel Multiplexer
- Internal Conversion Clock
- SPI/MICROWIRE™ Compatible Serial Interface
- Unipolar or Bipolar Input Ranges (Software Selectable)
- Separate Output Supply OV_{DD} (2.7V to 5.25V)
- 24-Pin 4mm × 4mm QFN Package

APPLICATIONS

- High Speed Data Acquisition
- Industrial Process Control
- Motor Control
- Accelerometer Measurements
- Battery Operated Instruments
- Isolated and/or Remote Data Acquisition

DESCRIPTION

The LTC®2308 is a low noise, 500ksps, 8-channel, 12-bit ADC with an SPI/MICROWIRE compatible serial interface. This ADC includes an internal reference and a fully differential sample-and-hold circuit to reduce common mode noise. The internal conversion clock allows the external serial output data clock (SCK) to operate at any frequency up to 40MHz.

The LTC2308 operates from a single 5V supply and draws just 3.5mA at a sample rate of 500ksps. The auto-shutdown feature reduces the supply current to 200µA at a sample rate of 1ksps.

The LTC2308 is packaged in a small 24-pin 4mm × 4mm QFN. The internal 2.5V reference and 8-channel multiplexer further reduce PCB board space requirements.

The low power consumption and small size make the LTC2308 ideal for battery operated and portable applications, while the 4-wire SPI compatible serial interface makes this ADC a good match for isolated or remote data acquisition systems.

Figura B-2. Hoja de datos del ADC LTC2308