TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ECATEPEC

DIVISIÓN DE INGENIERÍA MECÁNICA, MECATRÓNICA E INDUSTRIAL

“DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN CONTROLADOR DIFUSO SEGUIDOR SOLAR DE DOBLE EJE.”

TESIS QUE PRESENTA.

ING. JUAN CARLOS GARCÍA LIMÓN.

PARA OBTENER EL GRADO DE

MAESTRO EN EFICIENCIA ENERGÉTICA Y ENERGÍAS RENOVABLES.

ASESOR.

DR. JESÚS DE LA CRUZ ALEJO.

DICIEMBRE DE 2018

GOBIERNO DEL
ESTADO DE MÉXICO

DICTAMEN DE LIBERACIÓN DE PROYECTO DE TESIS

FO-TESE-DA-90

TIT05-ANEXO PM

DIRECCIÓN ACADÉMICA
DIVISIÓN DE INGENIERÍA MECATRÓNICA E INDUSTRIAL
MAESTRÍA EN CIENCIAS EN INGENIERÍA MECATRÓNICA

Fecha: 15 DE NOVIEMBRE DE 2018

Datos generales

Nombre del alumno:

García	Limón	Juan Carlos
Apellido paterno	Apellido materno	Nombres (s)

Matrícula: 201622041

PROYECTO: DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN CONTROLADOR DIFUSO SEGUIDOR SOLAR DE DOBLE EJE.

DIRECTOR: DR. JESÚS DE LA CRUZ ALEJO

Dictamen:

Con base en el registro No. POSG-MPEER/DIMMI/2018/05 correspondiente al periodo escolar 2017-1, asignado al proyecto de tesis DISEÑO E IMPLEMENTACIÓN EN FPGA DE UN CONTROLADOR DIFUSO SEGUIDOR SOLAR DE DOBLE EJE y presentado por el alumno referido, y después de haber sido desarrolladas las actividades requeridas, el Comité Tutorial, conformado por los integrantes que firman al calce, ha determinado que SI (X) NO ( ) han sido cubiertos en su totalidad los requisitos que permitan a esta División Académica liberar al alumno de los trámites académicos para tal efecto e iniciar el trámite de obtención de grado de maestría en la Unidad de Registro y Certificación de este Tecnológico de Estudios Superiores de Ecatepec.

Observaciones:

NINGUNA

Comité tutorial

Dr. ROGELIO FRANCISCO ANTONIO

Dr. CARLOS ROMAN MARIACA GASPAR

Dr. JESÚS DE LA CRUZ ALEJO

Elaboró:	Revisó y Visto Bueno:	Enterado:
Dr. AGUSTIN MORA ORTEGA Coordinador del Posgrado	M. en C. HÉCTOR RODRÍGUEZ CARMONA Jefe de División	M. en C. ARMANDO ALCALDE MARTÍNEZ Director Académico

Dedicatoria.

A mis hijos y nietos, por ese amor sincero e incondicional, que fortalecen mi deseo de continuar adelante a pesar de los múltiples obstáculos a lo largo de este camino.

A Leticia por su apoyo incondicional

A mis padres.

A mis hermanos, sobrinos y todas aquellas personas que siempre me han brindado su aprecio y cariño.

Agradecimientos.

Con admiración y aprecio a mi asesor el Dr. Jesús de la Cruz Alejo por su enorme apoyo y paciencia a lo largo de estos dos años de formación.

A los Doctores, maestros y compañeros de la maestría MEEyER y la maestría en Ciencias en Mecatrónica del TESE.

Al CONACYT por la beca otorgada y a todas aquellas personas que de una manera directa o indirecta me apoyaron para la culminación de dicho proyecto.

Resumen.

Este trabajo describe el desarrollo y simulación de un sistema de control seguidor solar de dos ejes para un sistema fotovoltaico, con el propósito de mejorar el aprovechamiento de la incidencia solar. El sistema de control se diseña con lógica difusa tipo Mamdani, ya que, a diferencia de otros sistemas de control, el control difuso de este tipo ofrece la ventaja de no requerir un modelo dinámico del sistema para su implementación, basado principalmente a partir de una base de conocimiento proporcionada por un operador humano. Con el objetivo de definir el rango de desplazamiento del sistema de control se calcularon los ángulos azimut y de altitud para la zona de Ecatepec Estado de México, ya que estos se encuentran en función de la hora y el día del año. Mediante la utilización de sensores ópticos LDR (Resistencias dependientes de la luz), se realiza la etapa de adquisición de datos, en el que por cada par de sensores se obtienen las señales etiquetadas como *e1* y *e2*, estas señales son enviadas a la tarjeta Nexys 4 DDR, la cual cuenta con un FPGA de la familia Artix 7, estas dos señales, complementadas con las derivadas del error, etiquetadas como $\Delta e1$ y $\Delta e2$ activan las entradas requeridas para el controlador difuso (FLC). Diseñado con el lenguaje de descripción de hardware (VHDL), el sistema se diseñó con dos entradas y una salida de cinco funciones triangulares de membresía y 10 bits de resolución cada una, mediante las cuatro etapas de procesamiento requeridas por el método Mamdani; fuzzificación, inferencia, agregación y defuzzificación con el método de centros de área COSAA. A la salida del sistema se obtiene una señal de tipo digital la cual se puede acondicionar a los requerimientos de la etapa de potencia, para poder manipular el actuador deseado, el cual pueda ejercer la acción de control. Los resultados obtenidos por el control difuso desarrollado fueron validados mediante la herramienta Test Bench de ISE Design Suite 14.7 y comparados con valores de respuesta teóricos, así como con los valores obtenidos con la herramienta Fuzzy Logic Designer de Matlab mediante el método de defuzzificación del centroide, obteniendo un sistema exacto.

Abstract.

This work describes the development and simulation of a two-axis solar tracking system for a photovoltaic system, with the purpose of improving the use of solar incidence. The control system is designed with fuzzy logic Mamdani type, since, unlike other control systems, fuzzy control of this type offers the advantage of not requiring a dynamic model of the system for its implementation, based mainly on a knowledge base provided by a human operator. In order to define the displacement range of the control system, the azimuth and altitude angles for the Ecatepec State of Mexico area were calculated, since they are based on the time and day of the year. Through the use of optical sensors LDR (Light dependent resistors), the data acquisition stage is performed, in which for each pair of sensors the signals labeled *e1* and *e2* are obtained, these signals are sent to the Nexys board 4 DDR, which has an FPGA of the Artix 7 family, these two signals, complemented with the derivatives of the error, labeled as $\Delta e1$ and $\Delta e2$ activate the inputs required for the fuzzy controller (FLC). Designed with the hardware description language (VHDL), the system was designed with two inputs and one output of five triangular membership functions and 10 bits of resolution each, through the four processing steps required by the Mamdani method; fuzzification, inference, aggregation and defuzzification with the method of area centers COSAA. At the output of the system a signal of digital type is obtained which can be conditioned to the requirements of the power stage, in order to be able to manipulate the desired actuator; which can exercise the control action. The results obtained by the developed diffuse control were validated by means of the Test Bench tool of ISE Design Suite 14.7 and compared with theoretical response values, as well as with the values obtained with the Matlab Fuzzy Logic Designer tool by means of the centroid defuzzification method, obtaining an exact system.

Índice General.

Lista de figuras.

Lista de tablas.

Introducción General.

El sol es la fuente principal de energía del planeta, la temperatura y presión en el interior del sol da origen a reacciones nucleares que liberan enormes cantidades de energía que llega al planeta de forma directa, difusa o reflejada en las partículas de la atmósfera, en las nubes y demás objetos en el ambiente. La disponibilidad de esta energía depende de la ubicación geográfica del lugar en donde se quiera aprovechar [1-2].

Como una fuente de energía limpia y renovable, la energía solar ha generado cada vez mayor atención, especialmente en el campo de la generación de electricidad, debido a la escasez y contaminación de combustibles fósiles. El proceso de conversión de energía solar en energía eléctrica es realizado principalmente a través de sistemas fotovoltaicos CSP (Colectores solares fotovoltaicos). La producción de energía que estos sistemas podrían producir depende de varios factores, como mantener la eficiencia de la celda solar empleando un sistema de seguimiento solar o maximizar la conversión de energía, utilizando las diferentes tecnologías para la fabricación de las celdas solares [2,4,5].

Actualmente, en el estado del arte relacionado con los seguidores solares, hay una gran cantidad de literatura relacionada con el estudio de los sistemas de posicionamiento solar, variando principalmente en el tipo de control utilizado para manipular a los seguidores solares, ya que son variadas las técnicas aplicadas, podemos encontrar controladores inteligentes utilizando Redes Neuronales Artificiales (ANN), lógica difusa (FLC), redes neuro-difusas (FNNs), o en el caso del control clásico, sistemas de control (PID), entre otros, como en [3]. Las alternativas son diferentes como en [11], donde el diseño de un sistema de seguimiento solar se realizó con un (Marcador de Posición1) controlador de lógica difusa (FLC), utilizando Matlab combinado con un control PI. Los resultados obtenidos con el control PI se comparan con los resultados de la lógica difusa. Se concluyó que las diferencias durante el período transitorio de posición, ambos controladores siguen la entrada de referencia variable, la respuesta de posición de FLC parece ser más lenta que la obtenida con PI cuando hay cambios de paso en el nivel de entrada de la posición de referencia [4 -5]. Mientras que en [10], se presentó un sistema de seguimiento solar de doble eje de posicionamiento automatizado con precisión, que se encuentra que el seguidor solar es capaz de posicionarse automáticamente en función del algoritmo de trayectoria solar con una precisión de

$\pm$ 0.5°. El sistema de posicionamiento proporcional integral derivativo (PID) mejora el seguimiento de ángulos de elevación y acimut. Revela que el seguidor solar propuesto puede generar 26.9% y 12.8% más de potencia que el sistema fotovoltaico de inclinación fija en condiciones claras y nubladas, respectivamente. Otro caso es [12], donde se diseñó un nuevo diseño del sistema de control del panel fotovoltaico utilizando MPSoC basado en FPGA. Esta propuesta presenta un diseño de subsistema basado en FPGA para controlar un seguidor solar de un eje y el seguimiento de punto de máxima potencia (MPPT) utilizando tres procesadores de núcleo blando. La principal ventaja de la arquitectura propuesta es su capacidad para manejar tareas simultáneas que requieren un gran tamaño de memoria y cálculos en punto flotante.

A diferencia de los sistemas mencionados anteriormente, la importancia de esta propuesta radica en mejorar la precisión y la velocidad de respuesta, así como reducir la complejidad de todo el sistema FLC, mediante la implementación de look up tables y Alpha levels(α) para aplicarse en las cuatro etapas del método Mamdani y su implementación en una plataforma FPGA que usa la descripción del lenguaje VHDL, en la tarjeta Nexys 4 DDR.

Por otro lado, debido a la complejidad involucrada en implementar y modelar sistemas de control aplicados a problemas del mundo real, una alternativa es el control difuso porque proporciona un proceso formal para representar, manipular e implementar el conocimiento empírico para controlar sistemas de una manera más sencilla [22-26].

Justificación.

Debido a los altos niveles de emisión de contaminantes de los combustibles fósiles y a los efectos negativos que produce la utilización de estos, como lo son el efecto invernadero, la lluvia acida, el adelgazamiento de la capa de ozono entre otros. Es necesario fomentar el uso de las energías renovables, que reduzca y complemente el uso de los energéticos fósiles, en el caso del de la energía solar, México posee una posición privilegiada en lo que respecta al potencial energético solar a nivel global, por lo cual la generación de electricidad por este medio representa una alternativa a los medios de generación tradicionales, con las progresivas reducciones del costo de la tecnología asociada, la energía solar fotovoltaica se puede convertir en un elemento crítico para reducir las emisiones de CO2 dentro del suministro de energía eléctrica.

La investigación en torno al aprovechamiento de las energías renovables es cada vez más requerida tanto por la industria privada como por el sector público, debido al aumento constante de los precios de los combustibles fósiles, y a que en la actualidad es posible para un usuario doméstico generar energía e interconectarse con la red suministradora, por tal se puede hacer uso de un sistema amigable con el medio ambiente sin perder el respaldo de la red convencional.

Es por ello de la importancia y trascendencia de desarrollar dispositivos que eficienticen la captación solar, buscando mayor exactitud, menor complejidad y menor costo a los ya existentes en el mercado. En este trabajo se propone una técnica de control inteligente con lógica difusa tipo Mamdani, implementado en una tarjeta Nexys 4 DDR de Digilent, la cual cuenta con un FPGA de la familia Artix -7, y desarrollado con lenguaje de descripción de hardware VHDL, el cual podrá manipular cualquier estructura mecánica de seguidor solar de doble eje en donde sea acoplado.

Objetivos.

Objetivo General.

- El objetivo general de este trabajo de investigación es proponer y desarrollar un sistema de control inteligente, mediante lenguaje VHDL en una plataforma FPGA, utilizando los principios de lógica difusa tipo Mamdani, implementados con operadores max-min, look up tables y niveles alfa, mediante el método COSAA para la etapa de defuzzificación.

Objetivos Particulares.

Con la finalidad de cumplir con el objetivo general se plantean y determinan los objetivos particulares descritos a continuación:

- Análisis de las coordenadas astronómicas necesarias para determinar el seguimiento solar, de esta forma es posible determinar el posicionamiento angular del seguidor solar.
- Diseñar el sistema de adquisición de datos, mediante la utilización de sensores de luz tipo LDR.
- Diseñar la etapa de conversión analógica - digital mediante la herramienta XADC, en Ise Design Suite.
- Diseñar cada una de las etapas del control difuso con lenguaje VHDL.
- Validar los valores obtenidos con la herramienta de simulación Test Bench.
- Instanciar en un solo bloque cada una de las etapas del control, así como el convertidor XADC.
- Instanciar el bloque del seguidor solar con los bloques obtenidos, uno para el movimiento de azimut y otro para el movimiento de elevación e implementarlo en la tarjeta Nexys 4.
- Validar resultados obtenidos, con valores teóricos y la herramienta de simulación Simulink de Matlab.

Organización de la tesis.

La tesis se integra por cuatro capítulos, a continuación, se hace una breve descripción del contenido de cada uno de ellos.

Capítulo 1.

Se analiza los aspectos que involucran el entorno de la energía solar fotovoltaica como es el nivel de radiación solar, descripción de las ecuaciones astronómicas que determinan el seguimiento solar para los movimientos azimut y de elevación, así como el principio de funcionamiento de una celda solar y los tipos de materiales utilizados en la fabricación de estas.

Capítulo 2.

Se analizan las bases del entorno teórico de la lógica difusa, así como la descripción de las cuatro etapas que involucra el modelo Mamdani, se muestran los tipos de dispositivos electrónicos utilizados para la implementación de la etapa de control, finalmente se realiza una descripción de los diferentes tipos de arreglos programables y de la tarjeta Nexys 4 DDR la cual se utilizó para verificar el funcionamiento del control difuso diseñado.

Capítulo 3.

Se muestra los bloques que integran la etapa de adquisición de datos, la descripción del algoritmo utilizado para la implementación de las etapas de fuzzificación, inferencia (máximos y mínimos) y la etapa de defuzzificación, se muestra la simulación en Fuzzy Logic Designer de Matlab.

Capítulo 4.

En este capítulo se muestra el procesamiento y la implementación de cada una de las etapas mediante el lenguaje de descripción de hardware VHDL, se muestran la validación y verificación de cada uno de los bloques del sistema, así como la instanciación y simulación de todo el sistema, finalmente se presenta la implementación en la tarjeta de desarrollo Nexys 4 DDR.

Capítulo 1
Radiación Solar.

En este capítulo se describen, las ecuaciones que permiten calcular los niveles de radiación extraterrestres, así como la radiación global que se recibe en el plano terrestre, se analizan aspectos relevantes del estado del arte referente a sistemas de control utilizados para la manipulación de controladores para seguidores solares.

Se muestran aspectos que involucran a la radiación solar y los factores que intervienen para poder determinar los niveles que se alcanzan en la zona de Ecatepec, Estado de México, México, así como los movimientos de la tierra en torno al sol, se analizan las ecuaciones necesarias para determinar los movimientos de elevación y azimut.

1.1 Radiación Solar.

El sol es una esfera de materia gaseosa intensamente caliente con un diámetro de $1.39x10^{9}\,m$ y en promedio se encuentra a $1.5x10^{11}\,m$ de distancia de la tierra, en la figura 1.1 se pueden apreciar las dimensiones del sol y de la tierra, así como el valor de la constante solar *Gs*, que es la energía del sol por unidad de tiempo recibida en un área unitaria de superficie perpendicular a la dirección de propagación de la radiación a distancia media tierra-sol fuera de la atmósfera [4,5,6]. En el interior de la esfera solar, son liberadas enormes cantidades de energía, de la cual una gran cantidad es recibida por la tierra, pero debido al efecto de la atmósfera, aproximadamente la mitad de la radiación solar se refleja, dispersa o absorbe llegando al suelo.

1.1.1 Tipos de radiación.

La radiación solar incidente sobre la superficie terrestre se puede manifestar de tres maneras:

- Directa: Es la que se recibe directamente desde el sol en línea recta, sin que se desvíe en su paso por la atmósfera.
- Difusa: Es la que se recibe del sol después de ser desviada por dispersión atmosférica. Es radiación difusa la que se recibe a través de las nubes, así como la que proviene del cielo azul.
- Reflejada: Es la radiación directa y difusa que se recibe por reflexión en el suelo u otras superficies próximas.

La luz solar disponible varía según la ubicación y el tiempo [4,5,6,8,9], la figura 1.2, muestra los niveles de radiación recibida según la zona geográfica a nivel mundial, se pude apreciar que para el caso de México los niveles de radiación son de los índices más alto dentro de la esfera terrestre, por lo que se requiere la implementación de tecnologías que propicien el máximo aprovechamiento de esta.

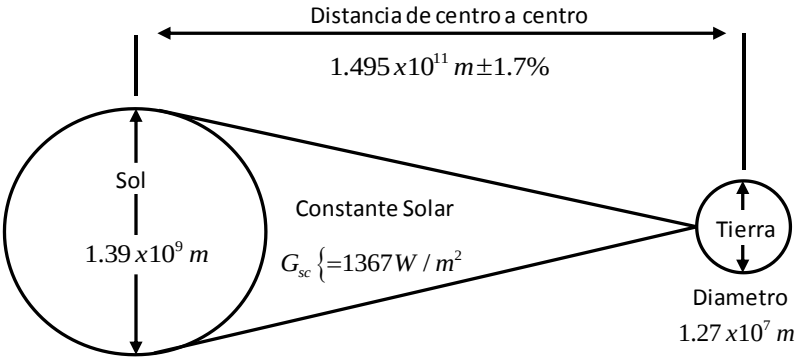


Figura 1.1 Magnitudes del sol y la tierra, fuente MIT open course.

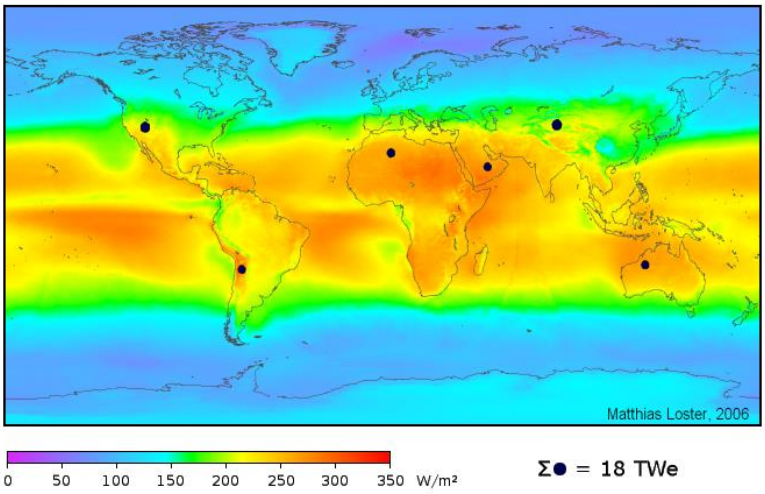


Figura 1.2 Energía solar a nivel mundial, fuente Matthias Loster 2006.

En este trabajo de investigación se calculó la radiación incidente extraterrestre en el plano normal, definida como G_{on} para los 365 días del año, mediante la siguiente ecuación:

$$G_{on} = G_{sc}\left(1 + 0.033\cos\frac{360n}{365}\right) \tag{1.1}$$

En donde

n= Número de día que corresponde, como la tierra tarda en dar la vuelta al sol 365 días, 5 horas, 48 minutos y 56 segundos, se establece que el año tiene 365 días, para compensar el desajuste cada 4 años se añade un día más.

En la figura 1.3 se muestra la tendencia de la curva generada por el comportamiento de la radiación extraterrestre, mostrando niveles máximos en los meses iniciales y finales de cada año, en la tabla 1.1, se muestran los valores obtenidos de radiación incidente extraterrestre a diferentes fechas a lo largo del año, es posible apreciar que los niveles más bajos de radiación se alcanzan, en el mes de junio, caso contrario para los primeros y últimos meses del año.

Tabla 1.1 Muestra de los resultados obtenidos con 1.1

Número de día	Fecha	*Gon*
1	01-ene	1412.10
2	02-ene	1412.08
3	03-ene	1412.05
4	04-ene	1412.00
5	05-ene	1411.94
152	01-jun	1327.96
153	02-jun	1327.58
154	03-jun	1327.21
155	04-jun	1326.85
156	05-jun	1326.50
335	01-dic	1406.23
336	02-dic	1406.60
337	03-dic	1406.97
338	04-dic	1407.32
339	05-dic	1407.67

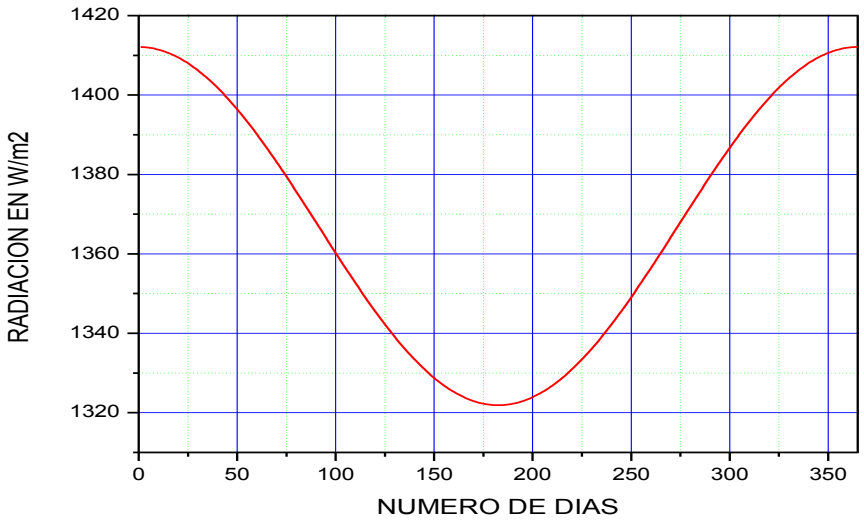


Figura 1.3 Comportamiento de la radiación solar extraterrestre a lo largo de un año.

1.1.2 Masa de Aire.

La masa de aire es una relación que indica la cantidad de aire que la luz solar debe de atravesar antes de llegar al plano horizontal, cuando el sol está directamente en el zénit a nivel del mar la masa de aire es igual a 1, y para un ángulo zénit de 60º y a nivel del mar la masa de aire es igual a 1.5, para el cálculo de masa de aire (*AM*) se utilizó la siguiente ecuación:

$$AM = \frac{1}{\cos\theta_z + 0.50572\left(96.07995 - \theta_z\right)^{-1.634}} \tag{1.2}$$

En donde:

θ_z = Ángulo zénit

En la figura 1.4 se muestra la tendencia del comportamiento para la masa de aire, se aprecia que los niveles máximos se alcanzan entre los meses de diciembre y enero, a partir de este mes la tendencia es descendiente hasta llegar al mes de abril y hasta el mes de agosto la tendencia es casi constante en uno, a partir del octavo mes se muestra un constante crecimiento hasta llegar a su nivel máximo alcanzado en el mes de diciembre. En la tabla 1.2 se muestran los valores obtenidos de masas de aire, mes a mes, utilizando la ecuación 1.2 para su cálculo.

Tabla 1.2 Muestra de los valores obtenidos al calcular la masa de aire con 1.2

Mes del año		Masa de aire
1	Enero	1.2904
2	Febrero	1.1583
3	Marzo	1.0610
4	Abril	1.0092
5	Mayo	1.0009
6	Junio	1.0037
7	Julio	1.0010
8	Agosto	1.0088
9	Septiembre	1.0597
10	Octubre	1.1646
11	Noviembre	1.2960
12	Diciembre	1.3565

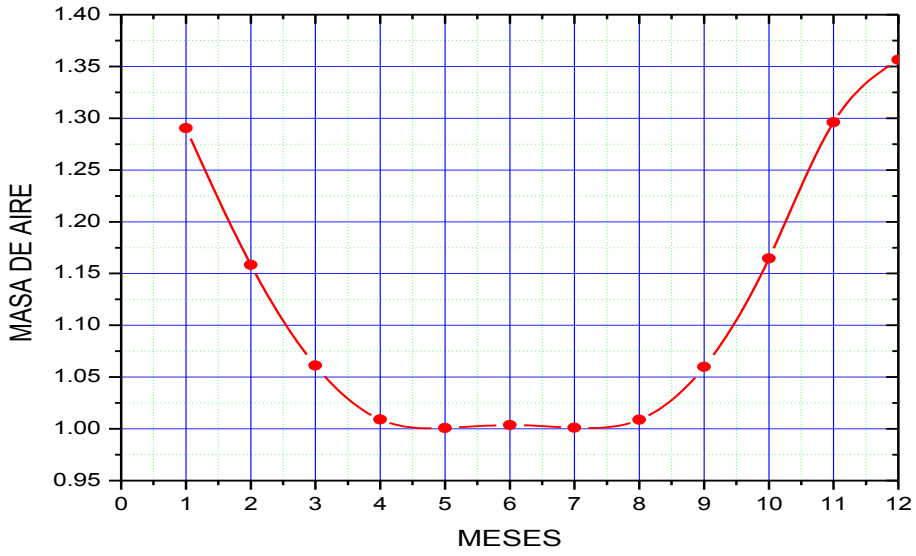


Figura 1.4 Comportamiento de la masa de aire en función de los meses.

1.1.3 Radiación directa.

Para poder calcular la radiación directa (I_d) de un lugar en particular es necesario conocer previamente la latitud del lugar así como la masa de aire, el cálculo (I_d) para la zona geográfica de Ecatepec, en el Estado de México (latitud 19°), los cálculos se realizaron al medio día, ya que en ese momento la posición solar se encuentra muy próximo al zenit en el que *(MA)* se encuentra muy cercano a 1, se utilizó la siguiente ecuación:

$$I_d = 1367 * 0.7^{AM^{0.678}} \tag{1.3}$$

En donde:

I_d = Radiación directa

MA = Masa de aire

1367 = Constante solar W / m^2

En la tabla 1.3 muestran los valores obtenidos de radiación directa en W/m^2, para los días 21 de cada mes, calculada mediante la ecuación 1.3, los resultados pertenecen al medio

día de cada mes, momento en el que hay una mayor aproximación al zenit, la tendencia se puede apreciar en la figura 1.5 un gráfico en forma de campana la cual refleja que los niveles más altos se alcanzan durante los meses de abril, mayo, junio y julio y los niveles más bajos, durante los meses de diciembre y enero.

Tabla 1.3 Radiación directa obtenida con 1.3

	Mes del año	Radiación directa W/m2
1	Enero	895
2	Febrero	922
3	Marzo	943
4	Abril	955
5	Mayo	957
6	Junio	956
7	Julio	957
8	Agosto	955
9	Septiembre	943
10	Octubre	920
11	Noviembre	893
12	Diciembre	882

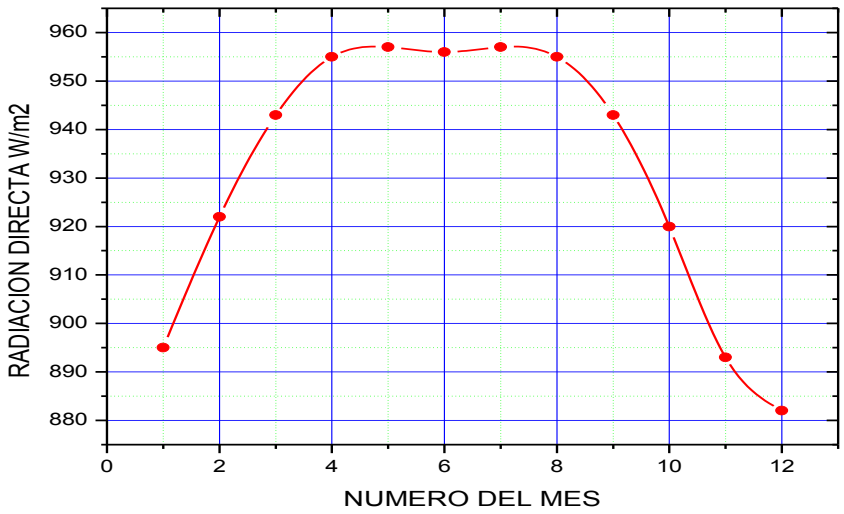


Figura 1.5 Radiación directa a lo largo del año en Ecatepec Estado de México.

1.2 Trayectoria solar.

La tierra tiene dos movimientos, uno de rotación sobre su propio eje que dura 24 horas 0 minutos y 57.33 segundos, ajustándose a 24 horas, compensándose con el año bisiesto cada cuatro años y que da origen al día y la noche originando una trayectoria circular con duración del día completo, y el de traslación movimiento alrededor del sol que origina el ciclo de las estaciones y tiene una duración de 365.24 días teniendo una trayectoria elíptica alrededor del astro [4,5]. El eje de la tierra tiene una inclinación de 23.44º respecto a la eclíptica, la cual es la proyección de la trayectoria alrededor del sol, para nuestro caso de estudio, esta declinación fue calculada para los 365 días del año mediante la siguiente ecuación:

$$\delta = 23.45^{\circ} \sin\left[\frac{360}{365}(284+n)\right] \tag{1.4}$$

En donde:

δ = Es la declinación solar en grados

n = Es el número de día correspondiente

La declinación fue calculada para cada uno de los días del año

En la figura 1.6, se puede observar la curva obtenida la cual varía estacionalmente debido a la inclinación de la Tierra, solo en los equinoccios de primavera y otoño el ángulo de declinación es igual a 0 °. Durante los solsticios muestra el extremo más alto y más bajo de 23.45º y de -23. 45º respectivamente, en la figura 1.7, muestra el movimiento terrestre alrededor del sol, y los periodos estacionales que este genera, así como el afelio (cuando la tierra se encuentra lo más alejada del sol) y el perihelio (momento de mayor cercanía al sol).

En la tabla 1.4 se pueden apreciar los valores obtenidos para el ángulo de declinación, para los primeros y últimos días del año, así como los valores para la primera mitad del año, los valores.

Tabla 1.4 Muestra de los resultados obtenidos con 1.4

Número de día	Fecha	Ángulo de declinación
1	01-ene	-23.01
2	02-ene	-22.93
3	03-ene	-22.84
4	04-ene	-22.75
5	05-ene	-22.65
152	01-jun	22.04
153	02-jun	22.17
154	03-jun	22.31
155	04-jun	22.42
156	05-jun	22.54
335	01-dic	-22.11
336	02-dic	-22.24
337	03-dic	-22.36
338	04-dic	-22.48
339	05-dic	-22.59

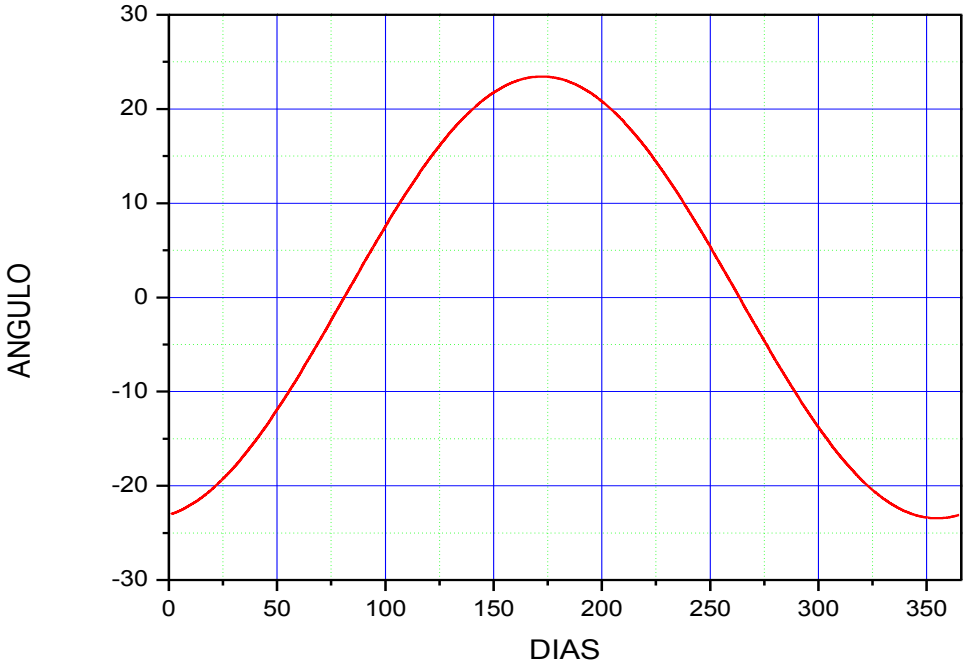


Figura 1.6 Tendencia del ángulo de declinación a lo largo del año.

.

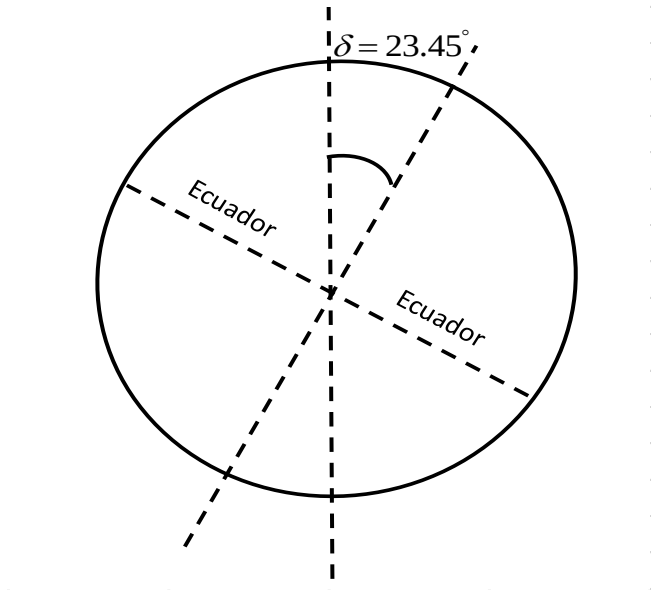


Figura 1.7 Ángulo de declinación de la tierra.

La figura 1.8 muestra el movimiento de rotación de la tierra en torno al sol, el cual genera las cuatro estaciones del año, los equinoccios para el primer y tercer cuarto del año, así como los solsticios para el segundo y cuarto periodo del año.

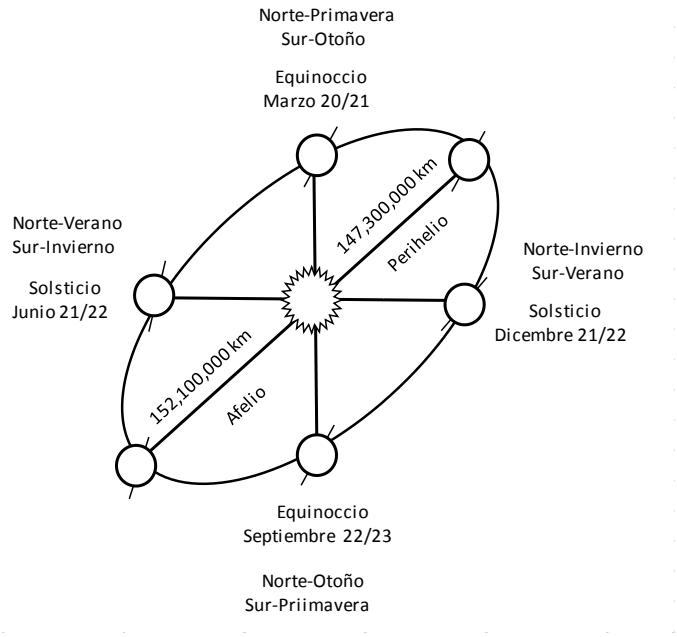


Figura 1.8 Movimiento terrestre alrededor del sol.

1.2.1 Latitud y Longitud.

Para localizar un punto en la Tierra se utilizan los ángulos de latitud (ϕ) y longitud propios del lugar, la latitud indica los grados del ángulo que se forma entre el Ecuador y el lugar en cuestión tomando como punto máximo +90º al Norte y -90º al Sur, mientras que el ángulo de longitud establece la posición de acuerdo al meridiano de Greenwich (plano de referencia internacional para definir el uso horario civil) [5,7,8,9] los lugares encontrados al Este u Oeste de Greenwich tendrán una longitud hasta 180º, como se aprecia en la figura 1.9.

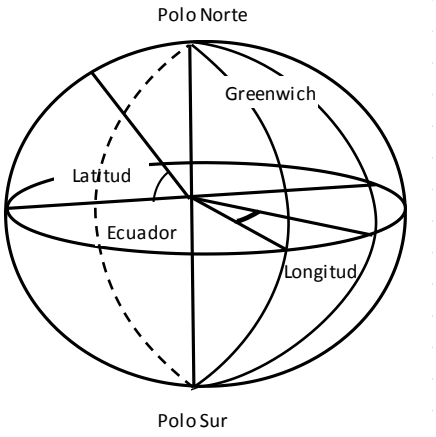


Figura 1.9 Ejes coordenados para determinar ángulos de latitud y longitud.

1.2.2 Movimiento de Altitud y Azimut.

1.2.2.1 Ángulo de altitud solar.

Para realizar el cálculo de los dos movimientos principales para determinar la posición del sol es importante determinar el ángulo de altitud solar (α), el cual es el ángulo que se forma entre la horizontal y la línea del sol, este ángulo es el complemento del ángulo zénit (θ_z) el cual es el ángulo entre la vertical y la línea del sol, como se puede observar en la figura 1.10.

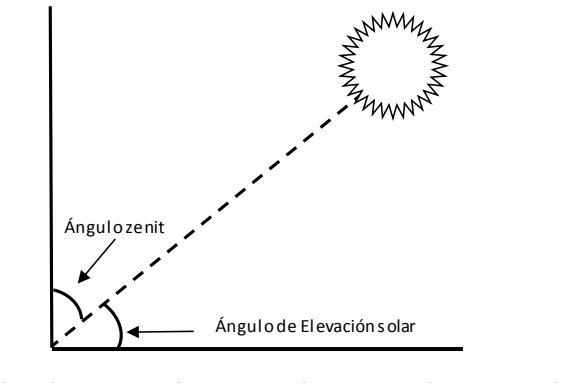


Figura 1.10 Esquema que muestra el ángulo zénit y de elevación.

El ángulo horario (w) se define como el ángulo entre la posición del sol con respecto al plano meridiano de cualquier ubicación directamente bajo el rayo del sol, antes de mediodía este ángulo es positivo, haciéndose 0 a las 12:00 pm y negativo posterior a esta hora, para calcularlo se utilizó la siguiente ecuación.

$$w=(ST-12)*15 \tag{1.4}$$

$ST=$ Es el tiempo solar local de 0 a 23.

Una vez calculados el ángulo de declinación y el en ángulo horario se cuenta con las variables necesarias para calcular el primero de los dos movimientos requeridos para el seguimiento solar, en este caso se realizó con la siguiente ecuación:

$$\alpha=\mathrm{sen}^{-1}\left[\mathrm{sen}\,\delta*\mathrm{sen}\,\phi+\cos\delta*\cos\phi*\cos\omega\right] \tag{1.5}$$

En donde:

$\delta=$ Es el ángulo de declinación.

$\phi=$ Latitud del lugar.

$w=$ Ángulo horario.

1.2.2.2 Ángulo azimut.

El ángulo de azimut (γ_s) se refiere a la magnitud del arco que forma la trayectoria solar desde un punto cardinal definido, para el caso de la zona geográfica norte, la trayectoria se mide de norte a sur tomando un valor de 90º en el este y de 270º en el oeste.

El ángulo acimut se calculó con la siguiente ecuación.

$$\gamma_s = sign(\omega)\left|\cos^{-1}\left(\frac{\cos\theta_z \sin\phi - \sin\delta}{\sin\theta_z \cos\phi}\right)\right| \tag{1.6}$$

La figura 1.11 muestra un comparativo entre el azimut solar y el ángulo de elevación, calculados para el municipio de Ecatepec con un rango que va de las 07:00 am a las 18:00 de un día en particular, en este caso para el 27 de noviembre de 2017. Como se puede observar, para la elevación a lo largo del día muestra valores mínimos de 0 y máximo de 50 grados, para el caso del azimut los valores van de 100 hasta los 250 grados.

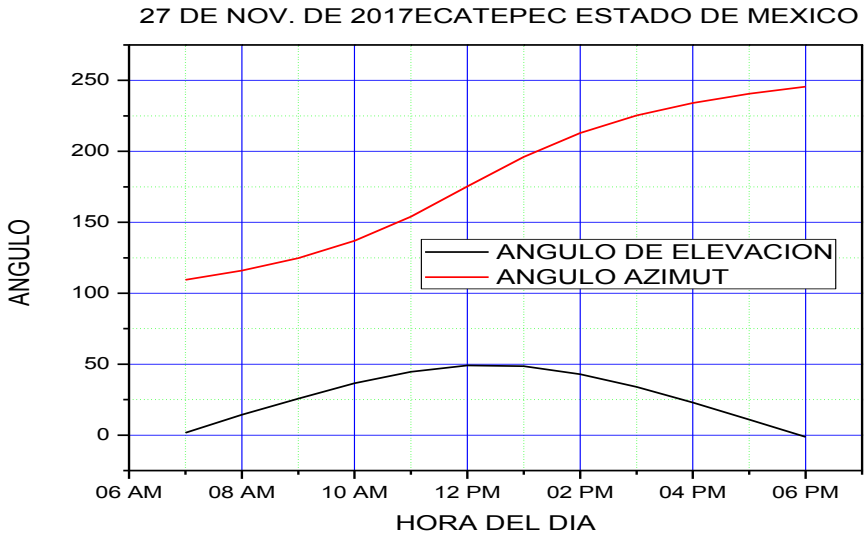


Figura 1.11 Ángulo azimut

En la figura 1.12, se muestra los dos movimientos necesarios para definir la trayectoria solar a lo largo del día, para el caso de Ecatepec en el Estado de México, México la salida del sol se da por el punto cardinal este, siguiendo una trayectoria hacia el sur para

culminar con la puesta por el punto cardinal oeste. Con base a la figura se aprecia que el ángulo azimut toma valores 0 a 270 grados, partiendo del norte haciendo un recorrido por el lado sur y finalizando en el oeste, el valor de la elevación toma un rango de 0 a 90 grados partiendo de la horizontal hacia el zenit y de 90 a 0 grados del zenit hacia la horizontal.

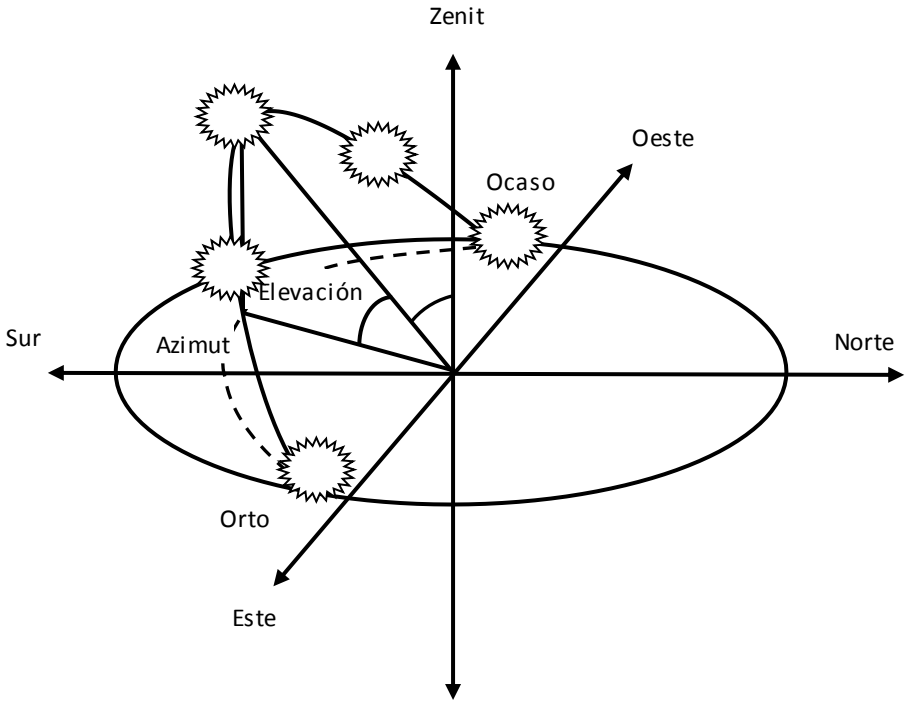


Figura 1.12 Seguimiento Solar acimut y de elevación.

1.2.3 Celdas Solares

La radiación solar se compone de un conjunto de ondas electromagnéticas a su vez formadas por una gran cantidad de agrupamientos energéticos denominados fotones que constituyen las unidades naturales de transporte de energía. De esta forma los rayos solares, forman una lluvia de fotones, cada uno de ellos transportando una cantidad muy pequeña de energía, que en su conjunto en un área determinada se convierten en un transporte considerable de energía. Los fotones se diferencian entre sí por el valor de su longitud de onda, o de su frecuencia, definida por el cociente entre su velocidad (velocidad de la luz) y la longitud de onda como en [5,8,9 ,10].

Las celdas solares fotovoltaicas que se utilizan en la actualidad, denominadas mono unión, están formadas por un semiconductor que tiene dos niveles de energía. Cuando sobre ellos actúa un fotón que no tiene la energía suficiente para excitar el electrón del nivel más bajo al superior no aprovechan esa energía. En el caso contrario, cuando el fotón tiene energía suficiente, lo que ocurre es que sí excita el electrón, pero la única energía que puede recuperar es la diferencia de energía de los dos niveles del semi-conductor, el resto se pierde.

Una celda solar puede ser descrita mediante el circuito equivalente como se muestra en la figura 1.13, en donde I_L representa la fuente de corriente eléctrica generada por la radiación luminosa, al alcanzar el efecto la superficie activa de la celda, esta corriente es constante para una determinada corriente interna unidireccional y la cual es función de la radiación incidente, la temperatura y el voltaje. La unión p-n funciona como un diodo que es atravesado por una corriente interna unidireccional I_D, R_s representa las perdidas eléctricas dentro de la celda y R_{sh} es una resistencia en paralelo que representa las fugas de corriente.

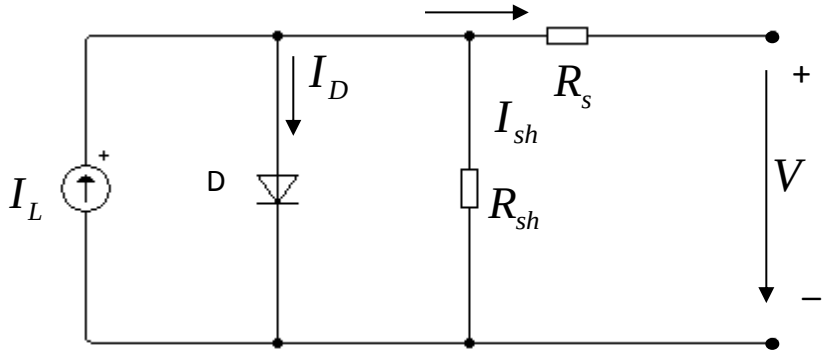


Figura 1.13 Modelo de una celda solar.

La ecuación que describe el circuito está dada de la siguiente forma:

$$I = I_L - I_D - I_{sh} = I_L - I_0\left[\exp\left(\frac{V + IR_s}{a}\right) - 1\right] - \frac{V + IR_s}{R_{sh}} \tag{1.7}$$

En donde:

$I = I_L - I_D - I_{sh}$ = Parámetro que representa la ley de corrientes de Kirchoff del circuito equivalente.

I_L = representa la fuente de corriente eléctrica fotogenerada por la radiación.

I_D = Corriente del diodo.

I_{sh} = Corriente de perdida por la resistencia en paralelo.

I_0 = Corriente inversa de saturación del diodo.

V = Voltaje del diodo

R_s = Resistencia en serie.

R_{sh} = resistencia shunt.

a = Factor de idealidad del diodo.

1.2.3.1 Voltaje de circuito abierto.

El voltaje de circuito abierto V_{OC} es el máximo valor del voltaje y se da cuando no tiene carga alguna conectada a la salida.

1.2.3.2 Corriente de corto circuito.

La corriente de corto circuito I_{SC} es el valor máximo de corriente en la celda y se da cuando la celda se encuentra en corto circuito.

1.2.3.3 Punto de potencia máxima.

El punto de potencia máxima de una celda solar fotovoltaica es un parámetro el cual expresa el punto en el que la potencia máxima es entregada y se obtiene al realizar el producto de la corriente máxima I_{SC} y voltaje máximo V_{OC}, obteniéndose el punto en el cual la celda solar opera a su mayor rendimiento. Como en la figura 1.14 y 1.15 se simuló el

funcionamiento de una celda solar en Simulink de Matlab, en la cual se obtiene la curva característica para el punto de potencia máxima de una celda solar. Los datos para la simulación fueron obtenidos de la hoja de datos de un panel solar modelo CORA 250 W de CORADIR S.A.

$I_{SC} = 8.74\ A$

$V_{OC} = .525\ V$

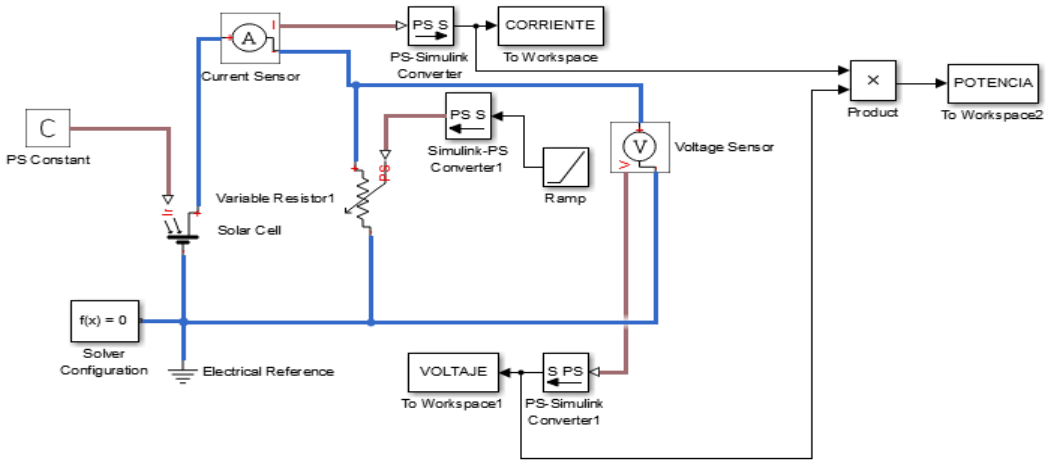


Figura 1.14 Simulación en simulink de una celda solar.

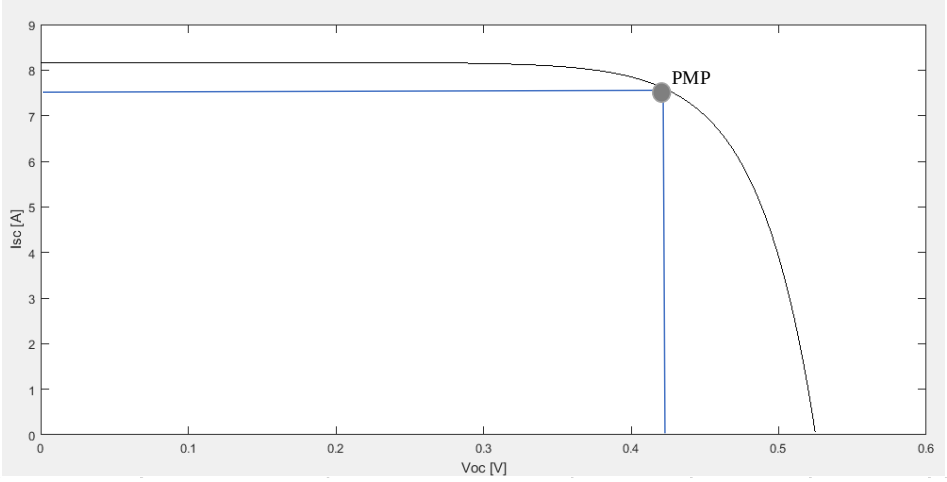


Figura 1.15 Punto de máxima potencia obtenido para una celda solar.

1.2.3.4 Celdas Solares Celdas solares multiunión.

Las celdas fotovoltaicas basan su funcionamiento en el efecto fotovoltaico y en la propiedad que tienen algunos materiales semiconductores. Estos materiales se caracterizan porque la anchura entre la banda de valencia y la de conducción, denominada banda prohibida, es normalmente inferior a 3 eV (Tabla 1.5), mientras que en el caso de los materiales aislantes es del orden de unos 7 eV [15-16].

Tabla 1.5 Saltos energéticos de diferentes semiconductores.
Fuente: Las células fotovoltaicas como alternativa energética.

Semiconductor	Eg (eV)
Te	0.33
Ge	0.67
$CuInSe_2$	1.05
Si Cristalino	1.12
Cu_2S	1.20
InP	1.34
GaAs	1.42
CdTe	1.45
CdSe	1.72
Si amorfo	1.75
Cu_2O	2.1
GaP	2.25
CdS	2.42
TiO_2	3

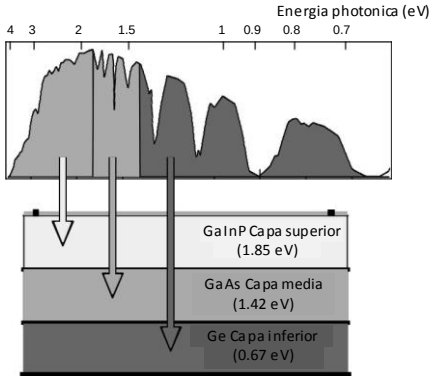


Figura 1.16 Funcionamiento de una celda solar triple unión.
Fuente (Journal homepage: www.elsevier.com. Progress and challenges for next-generation high-efficiency multijunction solar cells.)

Este tipo de celda consiste en la unión vertical de varias celdas solares, con distintos espacios de banda, conectadas en serie mediante uniones túnel. La posición de estas celdas debe ir en orden descendente en relación con su espacio de banda. Si se disponen en posición inversa la primera absorbería toda la radiación que tuviese más energía que su espacio de banda, pero solo se aprovecharía dicho espacio por lo que el resto de la energía se invertiría en calentar la celda. Actualmente las celdas multiunión de tres y cuatro uniones han alcanzado eficiencias de más del 40% [16,17,18,19].

La clave de la elevada eficiencia de estas celdas reside en su construcción, pues se componen de varias capas de materiales semiconductores que forman uniones P-N sintonizadas a diferentes longitudes de onda [17]. Como se puede observar en la figura 1.16 los fotones de menor longitud de onda (mayor energía) se absorben en la primera capa y los de mayor longitud de onda (menor energía) se absorben en las subsecuentes capas.

En la tabla 1.6, se muestran las eficiencias alcanzadas por los diferentes materiales utilizados para la fabricación de las celdas solares, así como el número de uniones utilizadas en el diseño, se puede apreciar que la máxima eficiencia se alcanza con una celda de triple unión.

Tabla 1.6 Eficiencias alcanzadas por tipo de material y materiales utilizados.

Material	Número de uniones	Eficiencia. (%)
Si	1	25.0 ± 0.5
GaAs	1	26.4 ± 0.8
CIGS	1	19.4 ± 0.6
CdTe	1	16.7 ± 0.5
GaInP/GaAs/GaInAs	3	35.8 ± 1.5
Si	1	27.6 ± 1.0
GaAs	1	29.1 ± 1.3
GaInP/GaInAs/Ge	3	41.6 ± 2.5

Capítulo 2
Lógica Difusa

En este capítulo se describe los aspectos teóricos sobre los cuales se basa el principio de funcionamiento y aplicaciones de la lógica difusa, se detallan las etapas de fuzzificación, inferencia, agregación y defuzzificación que involucra la lógica difusa tipo Mamdani así también se describe la tecnología FPGA y características del tipo de tarjeta NEXYS 4 DDR, la cual se utilizó para la realización de este trabajo.

2.1 Lógica Difusa.

En la lógica clásica las proposiciones pueden tomar solo dos valores de verdad, verdadero y falso, siglos atrás Aristóteles ya había formulado los principios fundamentales de la lógica clásica, el de no contradicción —nada puede ser y no ser al mismo tiempo — y el principio del tercer excluso —algo es o no es —[23-24]. En la década de los años veinte del siglo pasado, el filósofo polaco J. Lukasiewicz desarrolló los principios de la lógica multivaluada, cuyos enunciados pueden tener valores de verdad comprendidos entre el 0 (FALSO) y el 1 (CIERTO) de la lógica binaria clásica.

En 1965, L. Zadeh aplicó la lógica multivaluada a la teoría de conjuntos, estableciendo la posibilidad de que los elementos pudieran tener diferentes grados de pertenencia a un conjunto. Zadeh introdujo el término fuzzy (borroso, difuso) y desarrolló un álgebra completa para los conjuntos difusos, aunque estos conjuntos no tuvieron aplicación práctica hasta 1974 cuando Assilian y Mamdani en el Reino Unido desarrollaron el primer controlador difuso diseñado para la máquina de vapor [26,27,29].

El control difuso se implementa como una alternativa práctica para una variedad de aplicaciones de control ya que proporciona un método conveniente para construir controladores no lineales mediante el uso de información heurística. Dicha información puede provenir de un operador que ha actuado como un controlador "humano en el circuito" para un proceso. En la metodología de diseño de control difuso, se le pide al operador que escriba un conjunto de reglas sobre cómo controlar el proceso, luego se incorporan en un controlador difuso que emula el proceso de toma de decisiones del humano [27-28]. En otros casos, la información heurística puede provenir de cualquier persona que tenga conocimiento previo o dominio del proceso de control en particular.

La lógica difusa (fuzzy logic) permite tratar información imprecisa, como agua muy caliente, estatura baja o mucha fuerza, en términos de conjuntos difusos [20-22], estos conjuntos se relacionan con reglas difusa para determinar el tipo de acción a seguir, ejemplificando tenemos; que **SI** la habitación esta muy caliente **ENTONCES** enfriar mucho, es decir la lógica difusa hace uso de imprecisiones lingüísticas.

2.2 Sistemas Difusos.

Motivado por Zadeh y validado por Mamdani, el uso de los sistemas difusos ha sido aplicados en una gran variedad de áreas tales como el control automático, el procesamiento digital de señales, las comunicaciones, los sistemas expertos, la medicina, etc. Sin embargo, las aplicaciones más significativas de los sistemas difusos se han concentrado específicamente en el área del control automático tales como:

• Aeronaves y naves espaciales: Para el control de vuelo, control del motor, diagnóstico de fallas.

• Automóviles: Frenos, transmisión, suspensión y control del motor.

• Vehículos autónomos: Terreno y bajo el agua.

• Sistemas de fabricación: Programación y control de procesos (temperatura, presión y nivel, diagnóstico de fallas).

• Industria de la energía: Controles de motor, control para la distribución de energía.

• Robótica: control de posición y planificación de ruta [19-21].

Esencialmente un sistema difuso, es una estructura basada en conocimiento definido a través de un conjunto de reglas difusas del tipo si-entonces, las cuales, contienen una cuantificación lógica difusa de la descripción lingüística del experto de cómo realizar un control adecuado. Un sistema de lógica difusa requiere cuatro componentes clave: (1) la interfaz de Fuzzificación para transformar las entradas en términos lingüísticos para que puedan evaluarse utilizando la base de reglas; (2) la base de reglas establecidas sobre la base del conocimiento y la comprensión del sistema dinámico; (3) el mecanismo de inferencia establece aquellas reglas que deben aplicarse en un punto específico en el tiempo en función de las circunstancias prevalecientes y luego decide sobre los valores de salida que se aplicarán; y, (4) la interfaz de defuzzificación convierte los valores de decisión difusa establecidos mediante el mecanismo de inferencia en valores nítidos.

La figura 2.1 ilustra el diagrama a bloques y los componentes básicos de un sistema difuso tipo Mamdani.

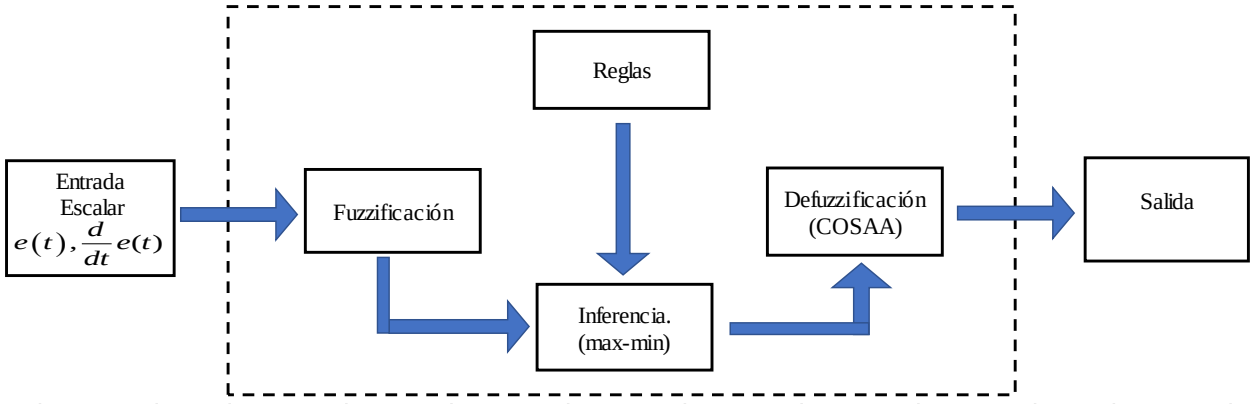


Figura 2.1 Diagrama Sistema Lógico Difuso.

2.3 Conjuntos clásicos y difusos.

Dentro de la teoría clásica de conjuntos se tiene definido un conjunto como una colección de elementos: Sea *U* un espacio de objetos y *x* un elemento de *U*, en un conjunto clásico *A*, se define la colección de elementos $x \in U$, de tal manera que cada *x* puede pertenecer ($x \in A$) o no pertenecer ($x \notin A$) al conjunto *A*, la función característica de un conjunto clásico solo puede tener dos valores 0 o 1.

$$A=\{x / x\varepsilon \, alumnos\, inscritos\, en\, la\, maestría\, de\, energías\, renovables\} \quad (2.1)$$

En este caso tenemos que solo los alumnos inscritos en la maestría de energía renovables pertenecen al conjunto *A* es decir los elementos del universo *U*, únicamente pueden o no pertenecer al conjunto *A*.

En el caso de los conjuntos difusos son aquéllos cuyos elementos que no tienen por qué pertenecer (grado de pertenencia 1) o no pertenecer (grado de pertenencia 0), sino que pertenecen según un cierto grado entre 0 y 1, donde el grado esta dado por la función de pertenencia del conjunto.

De esta forma la función de pertenencia de un conjunto *A* sobre un universo *U*, será de la forma:

$$A=\{(x,\mu_A(x))/\ x\varepsilon U\} \tag{2.2}$$

En donde:

$U=$ Universo de discurso

$A=$ El conjunto difuso

$\mu_A(x)=$ Es la membresía o pertenencia del elemento, al conjunto A.

2.3.1 Conjuntos continuos.

Cuando el universo *U* es continuo se denota de la siguiente forma:

$$A=\int_U \mu_A(x)/x \tag{2.3}$$

En donde:

$\int_U =$ Es la colección de todos los puntos $x\in U$ con la función $\mu_A(x)$.

Ejemplo de un conjunto difuso continuo:

Sea el universo denotado por las edades de las personas que tienen como habito la lectura en la ciudad de México, ver figura 2.2.

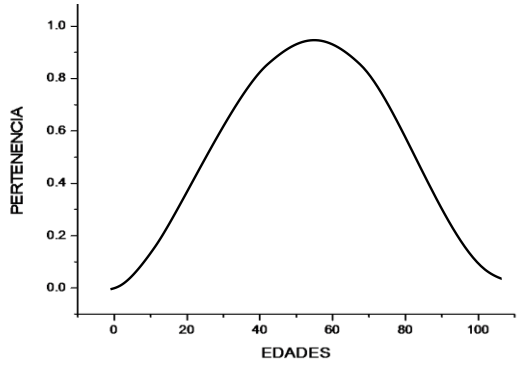


Figura 2.2 Conjunto difuso continuo.

2.3.2 Conjuntos discretos.

En el caso de que *U* sea discreto se denota de la siguiente forma:

$$A=\sum_{U}\mu_A(x)/x \tag{2.4}$$

En donde:

$\sum_{U}=$ Es la colección de todos los puntos $x\in U$ con la función $\mu_A(x)$.

2.4 Función de membresía $\mu(x)$.

La función de membresía o de pertenencia denotada por $\mu(x)$, indica el grado en que cada elemento de un universo dado pertenece a dicho conjunto, es el nivel de pertenencia a dicho conjunto dentro del rango [0,1], la figura 2.3 muestra gráficamente la función de pertenencia de un conjunto difuso.

$$0\le\mu(x)\le 1 \tag{2.5}$$

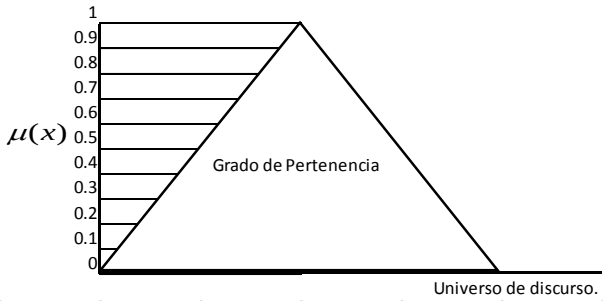


Figura 2.3 Función de membresía o de pertenencia.

En el eje de las abscisas se representa el universo de discurso, mientras que en el eje de las ordenadas se sitúan los grados de pertenencia en el intervalo de [0,1]. La función de membresía representa una parte del conocimiento humano, por tal normalmente esta aproximación requerirá de un ajuste más fino.

2.4.1 Funciones de membresía más comunes dentro de la lógica difusa.

Triangular: Definida mediante el límite inferior a, el superior c y el valor modal b, tal que $a < b < c$, este tipo de función es la de mayor uso, en la figura 2.4 se observa una función de tipo triangular.

$Si\ a \le \Delta e1 \le b, si\ b \le \Delta e1 \le c, 0\ en otros.$

$$Triangular(x:a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{2.6}$$

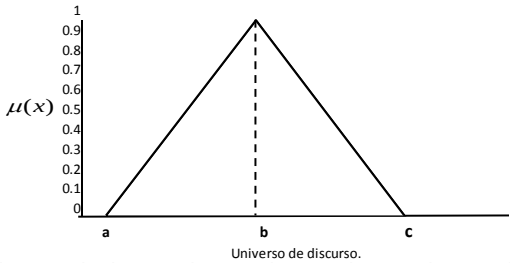


Figura 2.4 Función de membresía tipo triangular.

Trapezoidal: Figura 2.5 definida mediante el límite inferior a, el superior d y los valores de soporte b y c, tal que $a < b < c < d$.

$Si\ a \le x \le b, si\ b \le x \le c, si\ c \le x \le d, otros.$

$$Trapezoidal(x:a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{2.7}$$

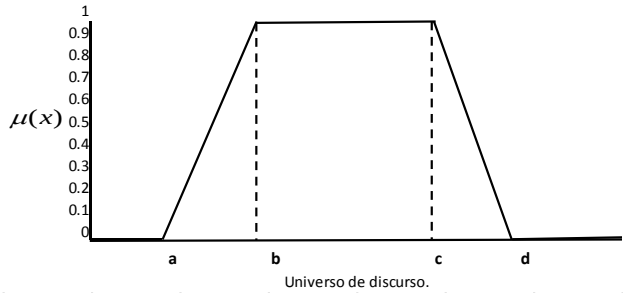


Figura 2.5 Función de membresía tipo trapezoidal.

Gaussiana y de Campana; Este tipo de función transforma un valor numérico real $x^* \in U$ en un conjunto difuso *A* en *U*, el que posee una función de membresía gaussiana que responde a la expresión:

$$\mu_A(x) = e^{-\left(\frac{x-x_C}{2\sigma 2}\right)} \tag{2.8}$$

Donde x_C es el valor de *x* para el cual la función es máxima y σ es la desviación estándar.

La figura 2.6, ilustra el tipo de función Gaussiana.

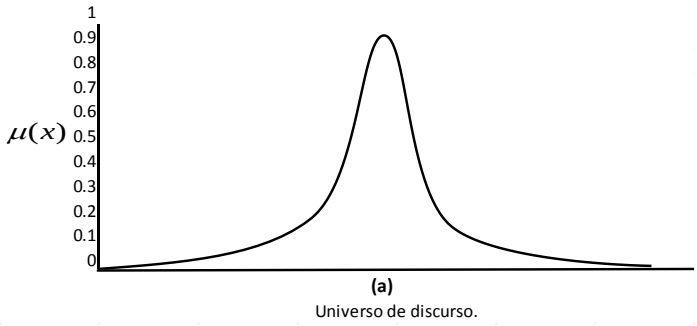


Figura 2.6 Función de membresía tipo Gaussiana.

2.5 Fuzzificación.

Fuzzificación (1): Consiste en la transformación de un conjunto clásico con datos escalares de entrada a su correspondiente conjunto difuso, es el proceso en el cual el diseñador define o determinan los grados de membresía o de pertenencia para cada una de las variables de entrada.

En el siguiente ejemplo se evalúa la temperatura del agua, asignándole un valor de pertenencia dentro del conjunto difuso, formado por tres funciones de membresía, en donde a cada una se etiqueta lingüísticamente, figura 2.7.

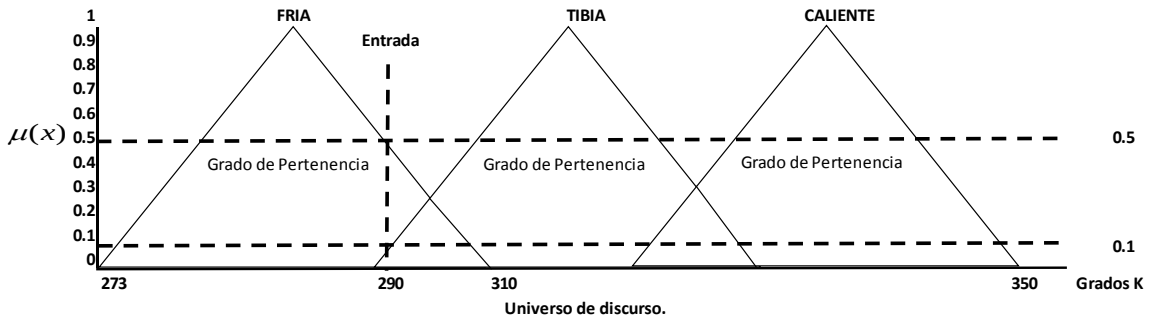


Figura 2.7 Conjunto difuso.

En este caso se tiene que:

Para la variable lingüística fría $\mu(x=fria)=0.5$ y para la variable lingüística tibia $\mu(x=tibia)=0.1$

2.6 Reglas difusas.

Las reglas difusas son las instrucciones que sigue la máquina de inferencia del controlador, estas relacionan las entradas con la salida. La elaboración de las reglas plasma la respuesta del algoritmo a las diferentes situaciones con las que se enfrente el sistema difuso [29-31]. Para la elaboración de las reglas difusas es necesaria la utilización de variables lingüísticas, esto debido a que las reglas difusas son una descripción lingüística que describe el sistema difuso de entradas y salidas, para elaborar las reglas se establece inicialmente las funciones de membresía definidas por el operador experto.

2.7 Inferencia.

En esta etapa, mediante el mapeo entre las reglas difusas se relacionan las entradas con las salidas mediante la premisa (SI- ENTONCES), se toman las entradas anteriores y se aplican a los antecedentes de las reglas difusas, este proceso realiza un mapeo entre cantidades difusas. Si una regla tiene múltiples antecedentes, se utiliza el operador AND u

OR para obtener un único número que represente el resultado de la evaluación. Este número (el valor de verdad) se aplica al consecuente.

La unión de *A* y *B* es un conjunto difuso en *U*, denotado por $A \cup B$ cuya función de pertenencia está definida como:

$$\mu A \cup B(x) = \max\left[\mu A(x), \mu B(x)\right] \tag{2.9}$$

La intersección de *A* y *B* es un conjunto difuso $A \cap B$ cuya función de pertenencia está definida como:

$$\mu A \cap B(x) = \min\left[\mu A(x), \mu B(x)\right] \tag{2.10}$$

Si x es A entonces, y es B en donde A y B son valores lingüísticos definidos por un conjunto difuso en un universo X e Y respectivamente. Si, un antecedente, es una condición en el dominio de aplicación; la cláusula entonces, una consecuencia, es una acción de control dado al proceso bajo control [26,31,33,34]. Con el conjunto de reglas difusas, el mecanismo de inferencia difusa es capaz de derivar una acción de control para un conjunto de valores de entrada.

2.8 Defuzzificación.

El proceso de defuzzificación realiza la conversión de los valores difusos de salida a valores reales el cual es el valor atribuido a la acción de control. Para la etapa de defuzzificación existen diversos métodos entre los que se encuentran el método del centroide, también conocido como Centro de Área o Centro de gravedad (COG) [28,30,32,34], en el que un punto representa el centro de gravedad del conjunto difuso, el cual está dado por la siguiente ecuación.

$$COG = \frac{\int_a^b \mu_A(x)\, x\, dx}{\int_a^b \mu_A(x)\, dx} \tag{2.11}$$

En donde:

$\mu_A =$ Es la función de pertenencia del conjunto de salida *A.*

$x =$ Es la variable de salida.

$a, b =$ Es el rango de integración.

Para la etapa de defuzzificación en este proyecto se utilizó el método de centro de porciones de área promedio (COSAA), este método permite trabajar con niveles alpha (α) [32], la ecuación 2.9 define este método:

$$COSAA = \frac{\sum_{i=0}^{\alpha_{\max}} \left(\frac{\left(x_f^{\alpha f} - x_0^{\alpha i} \right)}{2} + x_0^{\alpha i} \right)}{Número\ de\ conjuntos\ activados} \tag{2.12}$$

En donde:

$\alpha_f =$ El valor del nivel alfa final

$\alpha_i =$ El valor del nivel alfa inicial

2.9 Arreglos Programables.

A los circuitos digitales que son programables en hardware usando lenguajes como VHDL o Verilog se les conoce como dispositivos lógicos programables (PLDs, Programmable Logic Devices), estos componentes electrónicos son usado para construir circuitos digitales reconfigurables [36-37]. A diferencia de una compuerta lógica que tiene una función fija, los PLDs salen de fábrica sin una función específica, por lo tanto, necesitan ser programados o reconfigurados antes de poder ser usados, se trata de dispositivos que se pueden personalizar mediante diversas técnicas de programación. Existen diversos tipos de PLDs, entre los que se encuentran:

- SPLD (Simple Programmable Logic Devices):
    1. Consiste en una compuerta OR y alguna lógica de salida.

2. Puede configurarse para lógica combinacional, lógica de registro o una combinación de ambos.
3. Lógica de registro se refiere que hay un flip-flop para proporcionar funciones lógicas secuenciales.

- CPLD (Complex Programmable Logic Devices): es un dispositivo que contiene múltiples SPLDs y proporciona mayor capacidad para diseños más grandes.
- PAL (Programmable Array Logic): Se programa solo una vez.
- GAL (Generic Array Logic): Es un tipo de PAL que es reprogramable.
- FPGA (Field-Programmable Gate Array).

2.10 Arreglos de Compuertas Programables en Campo (FPGA).

Un FPGA es un dispositivo semiconductor que contiene componentes lógicos programables e interconexiones programables entre ellos. Los componentes lógicos programables pueden ser programados para duplicar la funcionalidad de puertas lógicas básicas tales como AND, OR, XOR y NOT o funciones combinacionales más complejas tales como decodificadores o simples funciones matemáticas. En muchos FPGA, estos componentes lógicos programables, también incluyen elementos de memoria, los cuales pueden ser simples flip-flops o bloques de memoria más complejos [38-39].

Una de las razones por la que los FPGAs son tan eficientes es que a diferencia de un ASIC (Aplication Specific Integrate Circuit, circuitos integrados de propósito o aplicación especifica), no se establece el diseño del circuito y se puede volver a configurar una FPGA tantas veces como se quiera. A pesar de que los ASIC dominan la tendencia en el desarrollo de aplicaciones en electrónica, entre los inconvenientes que se pueden encontrar es el alto costo y tiempo para crearlos, así como la imposibilidad para reprogramarlos [37-40].

La arquitectura de un CPLD consiste en bloques PAL/GAL con interconexiones programables. El FPGA difiere en que no usa arreglos tipo PAL/GAL.

Los tres elementos básicos en un FPGA son:

- Bloque Lógico configurable (CLB)

- Interconnexions.
- Bloques de entrada y salida (I/O Blocks)

En la figura 2.8 se puede apreciar los tres principales bloques de un FPGA (CLB, Interconexiones e I/O Blocks).

Los bloques lógicos contienen memoria para configuración del bloque, usan LUT'S que implementa lógica combinacional, flip-flops y multiplexores a la entrada y salida del bloque lógico. Las terminales de entrada y salida del FPGA usan celdas especiales de E/S que son diferentes de los bloques lógicos. Además, tienen un esquema de interconexión programable que permite la conexión entre las celdas de elementos lógicos entre sí, y con las celdas de E/S. La programación de las interconexiones y de los elementos lógicos puede o no ser permanente, eso depende de la tecnología de programación usada [36-40].

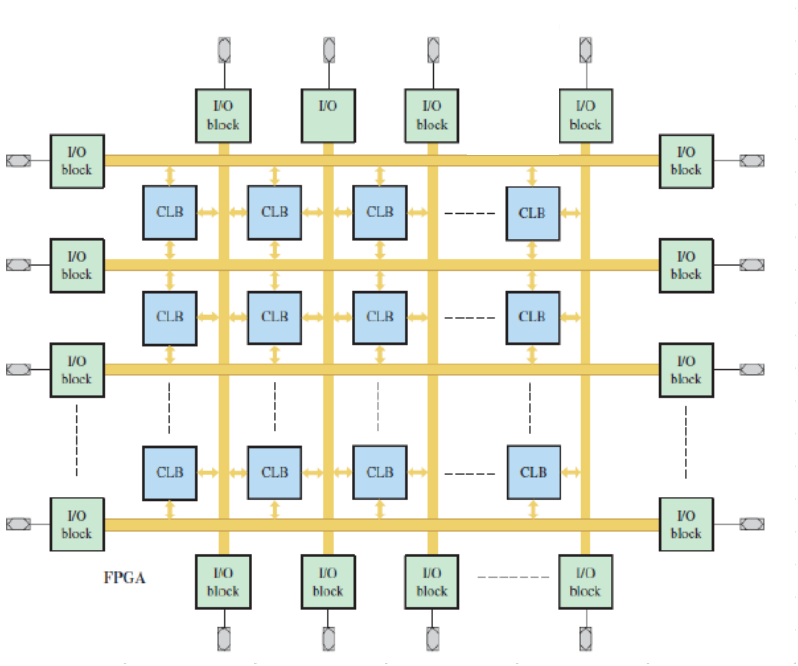


Figura 2.8 Estructura de un FPGA.

En la tabla 2.1 se muestra un comparativo entre los CPLD'S y los FPGA, en el que se puede apreciar la enorme diferencia entre las compuertas equivalentes, de aquí la diferencia para determinar el uso entre ambas tecnologías.

Tabla 2.1 Comparativo entre CPLD y FPGA.

	CPLD	FPGA
Arquitectura	Utilizan AND y OR	Utilizan LUT´S
Tecnología	Flash	RAM
Densidad	Baja a media (1000 compuertas equivalentes)	Media-alta (millones de compuertas equivalentes)

2.11 Tarjeta Nexys 4 DDR.

La tarjeta Nexys4 DDR cuenta con un FPGA de última generación Artix-7 ™ de Xilinx®, el cual está optimizado para la lógica de alto rendimiento, ofreciendo mayor capacidad, mayor rendimiento y mayor número de recursos que los diseños anteriores Artix-7 100T, en la figura 2.9 se muestra la imagen física del FPGA.

Entre las características que destacan este FPGA se encuentran:

- 15,850 segmentos lógicos, cada uno con cuatro LUTs 6-entrada y 8 flip-flops
- 4,860 Kbits de RAM.
- 240 Segmentos DSP
- Reloj interno con velocidades superiores a 450 MHz
- On-chip de conversión analógica-digital (XADC)

Figura 2.9 FPGA Artix-7

LA Nexys4 DDR incluye:

- 16 LEDs de usuario
- Dos indicadores de 7 segmentos de 4 dígitos
- Conector de tarjeta micro SD
- Salida VGA de 12 bits
- Salida de audio PWM
- Micrófono PDM
- Acelerómetro de 3 ejes
- Sensor de temperatura
- 128MB DDR2
- Serial flash
- Cuatro puertos pmod
- Pmod para señales XADC
- Puerto USB-JTAG Digilent para la programación de FPGAs y la comunicación
- Host USB para conectar mouse, teclados y tarjetas de memoria [35].

En la tabla 2.2 y figura 2.10 se describen las funcionalidades de los componentes externos de la tarjeta Nexys 4 DDR.

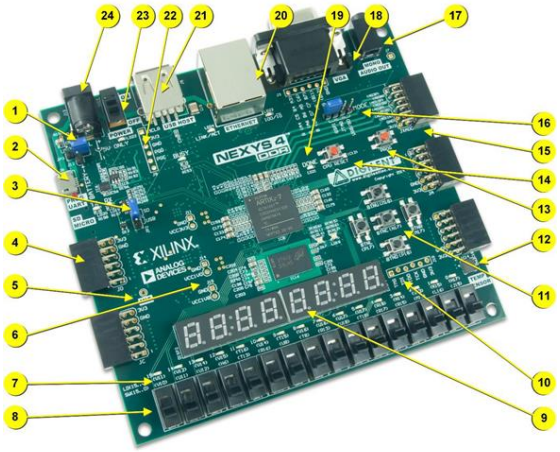


Figura 2.10 Tarjeta Nexys 4 DDR.

Tabla 2.2 Descripción de componentes Nexys 4 DDR

Número	Aplicación	Número	Aplicación
1	Selector de alimentación	13	Botón de reinicio FPGA
2	Puerto JTAG-USB	14	Botón de reinicio CPU
3	Selector (SD/USB)	15	Conector analógico-digital XADC
4	Conectores Pmod	16	Selector modo de programación
5	Micrófono	17	Conector de audio
6	Punto de prueba de fuente de alimentación.	18	Conector VGA
7	Leds (16)	19	Led indicador de programación
8	Interruptores deslizantes	20	Conector Ethernet
9	Display (8) de 7 segmentos	21	Conector puerto USB
10	Puerto JTAG	22	PIC24 Puerto de programación (fábrica)
11	Interruptores pulsadores	23	Interruptor de alimentación
12	Sensor de temperatura	24	Conector de alimentación.

Capítulo 3 Metodología para el Diseño de un FLC.

En este capítulo se detalla el diseño para la implementación de cada una de las etapas del sistema de control lógico difuso en el lenguaje de descripción de hardware VHDL, mediante el software de diseño ISE Design Suite. Se muestra la implementación de un FLC en Matlab con el Fuzzy Logic Designer, bajo las mismas condiciones que el FLC que se desarrolló, para comparar una muestra de los valores obtenidos entre ambos sistemas de control.

3.1 Sistema de Control.

Para esta tesis se diseñó un controlador de dos entradas y una salida. Las entradas físicas del sistema son los voltajes provenientes de una etapa de adquisición, las entradas de la máquina de inferencia son el error y la derivada del error, la salida de la máquina de inferencia es un valor difuso el cual será defuzzificado mediante el método COSAA, previamente descrito en esta tesis, a la salida del FLC tenemos un solo valor el cual nos permitirá posicionar el actuador en base a los cálculos obtenidos para el posicionamiento del seguidor solar, el diagrama a bloques de la figura 3.1 describe el sistema de control.

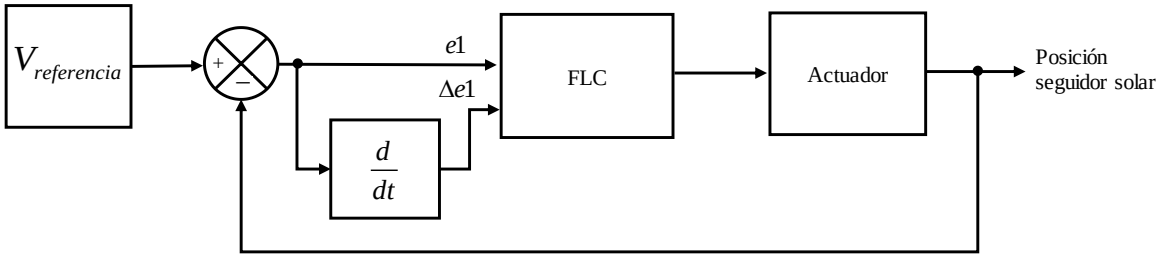


Figura 3.1 Diagrama a bloques sistema de control.

3.2 Adquisición de datos.

La primera fase para poder implementar el FLC, fue el generar las variables de entrada, para lo que se diseñó y desarrollo un sistema de adquisición de datos, el cual consta de 4 sensores de luz, de tipo resistivo (LDR), mediante estos, se generan los voltajes de entrada al XADC. En la operación de rastreo, los sensores LDR miden la intensidad de la luz solar y el desequilibrio en los voltajes generados por el sensor LDR generan un voltaje de error de retroalimentación. El voltaje de error es proporcional a la diferencia entre la ubicación de la luz solar y la ubicación del panel fotovoltaico, para cada sistema de control (elevación y azimut), se requieren un par de entradas, las cuales se etiquetaron como error ($e1$) y derivada del error ($\Delta e1$), para el primer caso así como $e2$ y $\Delta e2$ para el segundo caso, en donde la derivada del error se define como:

$$\frac{de}{dt} = Error\ actual\text{-}Error\ anterior \tag{3.0}$$

Esta etapa se desarrolló en una plataforma de hardware libre (tarjeta Arduino nano) el cual cuenta con un Microcontrolador: AT Mega 328, una memoria flash de 32 KB, una frecuencia de trabajo de 16 MHz y un voltaje de operación de 5 volts, en donde los sensores resistivos LDR se implementaron en un arreglo divisor de voltaje. El error y cambio en el error generados en la etapa de adquisición de datos como se muestra en la figura 3.2, son enviados, a la tarjeta Nexys 4 los cuales representan los valores escalares de entrada del sistema de control difuso.

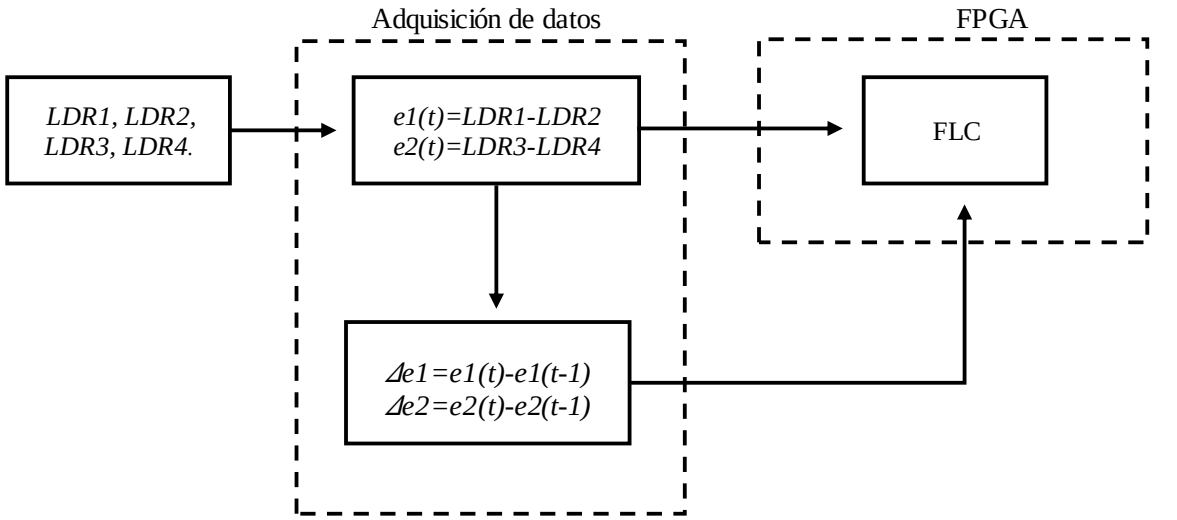


Figura 3.2 Diagrama a bloques para la adquisición de datos.

Una vez que se obtienen los valores analógicos, resultado del proceso de adquisición de datos, estos son enviados al pmod XADC de la tarjeta Nexys 4 en donde estos valores son convertidos en valores digitales. Dichos valores corresponden a los valores de entrada del FLC.

Para la implementación del sistema de control difuso tipo Mamdani, se definieron 5 funciones de membresía de tipo triangular para cada uno de los conjuntos difusos, dos entradas ($e1$, $e2$, $\Delta e1$ y $\Delta e2$), y una salida, etiquetándolas de la siguiente manera: Muy bajo (MB), Bajo (B), Medio (M), Alto (A) y muy Alto (MA), como se aprecia en la figura 3.3.

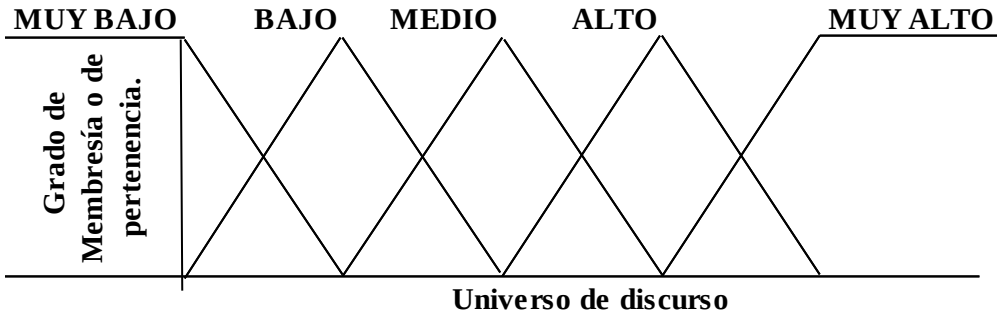


Figura 3.3 Funciones de membresía.

3.3 Fuzzificación.

Previamente, para la etapa de fuzzificación, se diseñaron tablas de búsqueda en una hoja de cálculo del programa informático Excel de Microsoft office, en esta etapa se asignó su correspondiente valor de resolución para cada uno de los valores posibles de entrada. En la figura 3.4 se muestra el tipo de funciones obtenidas con los valores asignados para cada entrada.

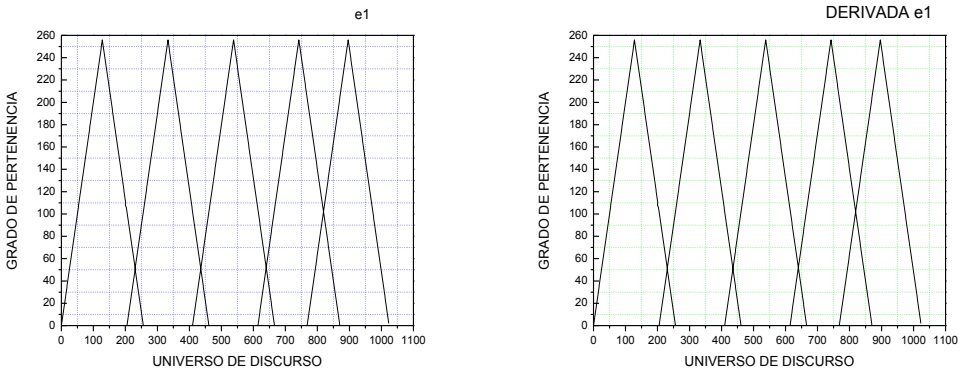


Figura 3.4 Conjuntos difusos variable de entrada.

Es en esta etapa en donde, se transforman los valores escalares de entrada en valores difusos, para el caso de este trabajo de investigación se implementó con una resolución de 10 bits, para el universo de discurso o eje de las abscisas ; $2^n - 1$ en donde $n = 10$ el cual nos permite tener un rango difuso de 0 a 1023 para los valores reales de 0 a 1.25 volts el caso de los movimientos de elevación y azimut, es decir 0 grados corresponde a 0 bits en la conversión a su valor de resolución y a 1.25 volts le corresponde 1023 bits en su valor de

resolución. Para el caso del grado de membresía o de pertenencia , eje de las ordenadas se tiene una resolución de 8 bits; $2^n - 1$ en donde $n = 8$, obteniendo 255 bits de resolución, la resolución en valores difusos y valores reales se pueden apreciar en la figura 3.5 para las entradas $e1 \, y \, e2$.

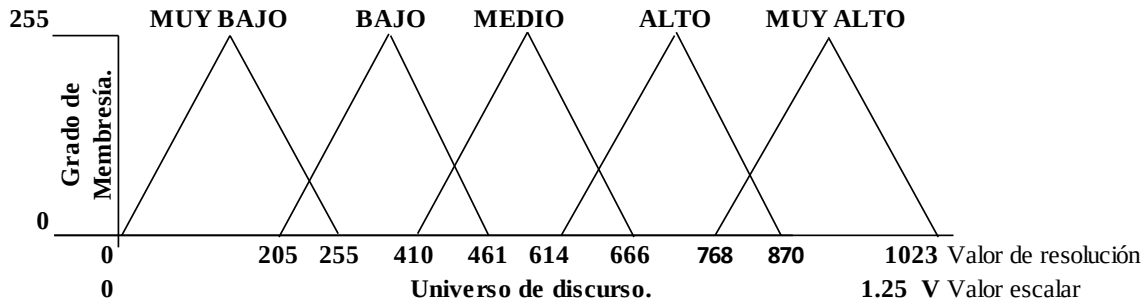


Figura 3.5 Conjuntos difusos variable de entrada e1 y e2.

3.4 Inferencia.

3.4.1 Mínimos.

En la etapa de inferencia los valores obtenidos en la etapa de fuzzificación se organizan de tal forma que se forma una matriz de 5x5 en la cual se realiza un mapeo entre las funciones de membresía, en donde la primera función de membresía de la entrada $e1$ se compara con cada una de las funciones de membresía de la entrada $\Delta e1$, registrando los valores mínimos obtenidos en cada una de las comparaciones, en la primera fila de la matriz obtenida. En la figuras 3.6 y 3.7 se aprecia la matriz resultante del barrido que se realiza entre las funciones de membresía, en el caso de (1,1) se registra el valor mínimo entre la función de membresía MB de $e1$ y MB de $\Delta e1$, sucesivamente hasta llegar a (5,5) en donde el valor mínimo se obtiene de *MA* de $e1$ y *MA* de $\Delta e1$.

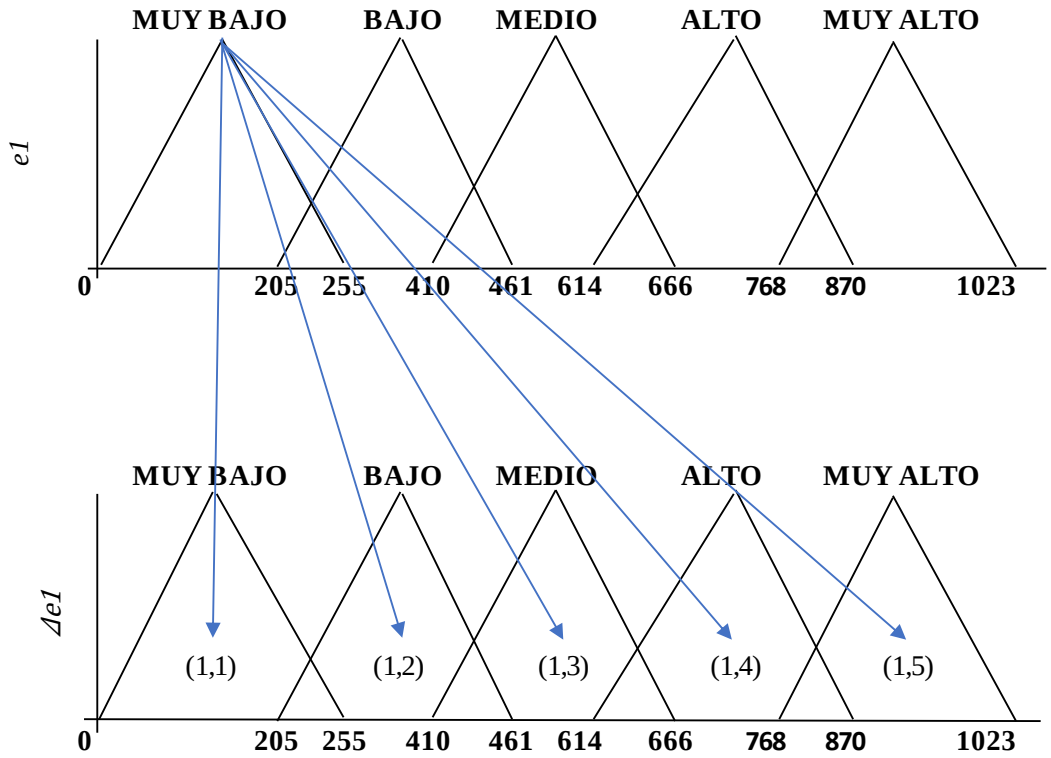


Figura 3.6 Obtención de los valores mínimos.

Δe1 \ e1					
min →	(1 , 1)	(1 , 2)	(1 , 3)	(1 , 4)	(1 , 5)
min →	(2 , 1)	(2 , 2)	(2 , 3)	(2 , 4)	(2 , 5)
min →	(3 , 1)	(3 , 2)	(3 , 3)	(3 , 4)	(3 , 5)
min →	(4 , 1)	(4 , 2)	(4 , 3)	(4 , 4)	(4 , 5)
min →	(5 , 1)	(5 , 2)	(5 , 3)	(5 , 4)	(5 , 5)

Figura 3.7 Matriz valores mínimos.

3.5 Agregación.

3.5.1 Máximos.

Una vez que se ha conformado la matriz de valores mínimos, la siguiente etapa es la agregación en donde el vector de función agregada almacena los valores máximos de cada regla difusa activada. Estos valores se interpolan mediante el operador max con los conjuntos difusos de salida para obtener el arreglo que contendrá los valores de la función agregada el siguiente paso es realizar un mapeo para encontrar los valores máximos, en este caso se realiza un barrido vertical en el que se busca el valor máximo de cada una de las columnas, para efecto del lenguaje de VHDL mediante la sentencia IF se realizó la comparación de los dos primeros datos de la columna 1, de igual forma para los datos 3 y 4 los resultados de estas dos comparaciones se mapean con entre ellos para obtener el valor máximo de estos dos datos, se etiqueto como maxc1, este valor se compara con el último de los valores de la primer columna, de esta forma se obtiene el valor máximo de la primer columna etiquetada como maxcolumna 1 de igual forma para las siguientes cuatro columnas, de tal forma que de los 25 valores obtenidos en el bloque para la obtención de los valores mínimos, este se redujo a 5 valores, los cuales son los valores de entrada de la etapa de defuzzificación. La secuencia para la obtención de los valores máximos se puede apreciar en la figura 3.8.

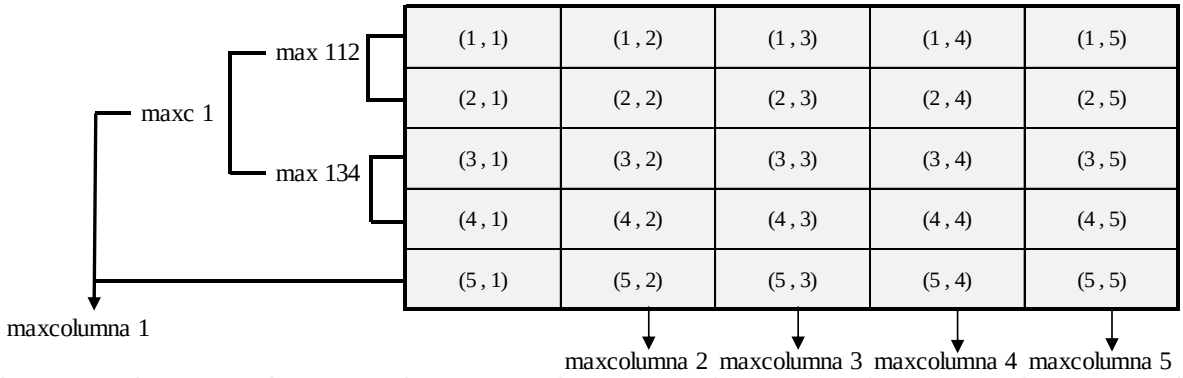


Figura 3.8 Matriz para encontrar los valores máximos.

3.6 Defuzzificación.

Finalmente, en la etapa de defuzzificación los valores difusos obtenidos en la etapa de agregación son convertidos a valores reales o escalares. Mediante la utilización de los niveles de cuantización, que en este caso tiene un rango de 0 a 1023 niveles, en el que se utilizó el método Centro de Áreas Promedio o Center of Slice Area Average por sus siglas en inglés (COSAA), para lo cual se utilizó la ecuación (2.9), en la figura 3.9 se muestran la etapa de defuzzificación mediante los niveles alpha:

$$COSAA = \frac{\sum_{i=0}^{\alpha_{\max}} \left(\frac{\left(x_f^{\alpha f} - x_0^{\alpha i} \right)}{2} + x_0^{\alpha i} \right)}{Número\ de\ conjuntos\ activados} \tag{2.9}$$

En donde:

α_f = El valor del nivel alfa final

α_i = El valor del nivel alfa

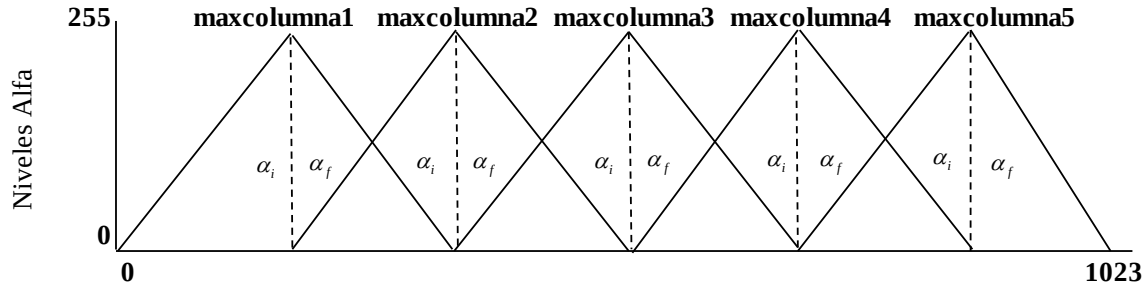


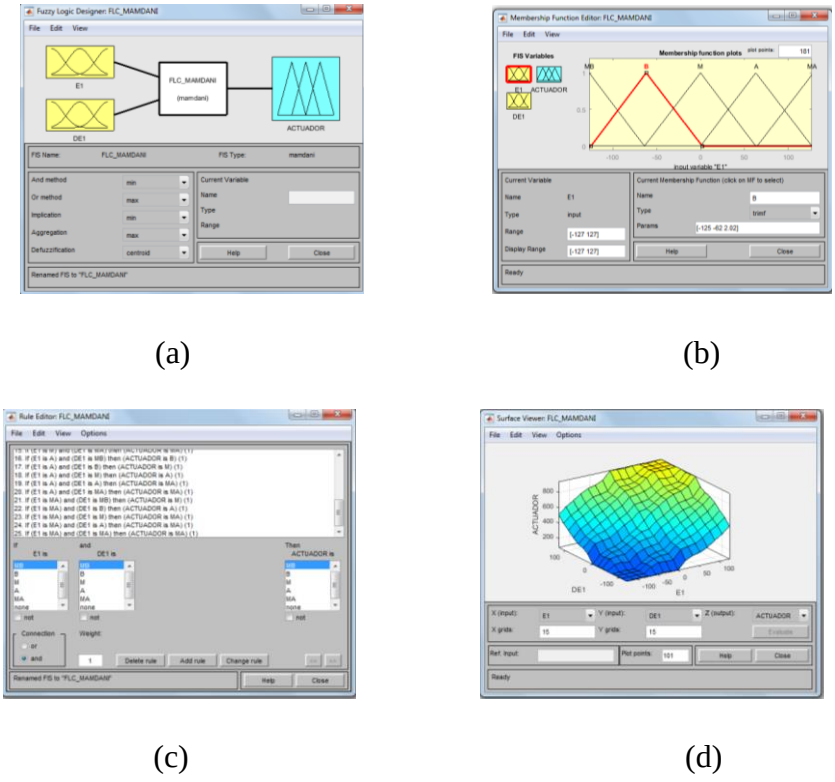
Figura 3.9 Defuzzificación con niveles alfa.

3.7 Simulación en Fuzzy Logic Designer de Matlab.

Un FLC en el Fuzzy Toolbox de Matlab se denota como FIS (Fuzzy Inference System), es una interfaz gráfica de usuario que permite la elaboración de sistemas lógico difuso en el entorno de Matlab.

Un FIS elaborado con esta herramienta permite definir el número de entradas y salidas para cada conjunto difuso, en este caso se implementaron dos entradas y una salida, y se etiquetaron igual que FLC diseñado, los rangos del universo de discurso, así como la asignación de las reglas también se asignaron bajo los mismos parámetros. Esta herramienta también permite seleccionar entre un FLC tipo Mamdani y un tipo Takagi Sugeno, así como el método de defuzzificación, que para el caso de la simulación fue hecho con el método del centroide o centro de gravedad.

Las figuras 3.10 muestran la simulación del FLC con el Fuzzy Toolbox de Matlab, en la figura (a) se definió el tipo de FLC, el número de entradas y la salida, así como el tipo de defuzzificación, en la figura (b) se definieron la cantidad de funciones de membresía, así como los parámetros del universo de discurso, en la figura (c) se editaron las reglas, finalmente en la figura (d) se puede apreciar las superficies que se obtuvieron con la simulación del sistema de control, estas superficies son suaves y no reflejan sobre saltos.



(a) (b)

(c) (d)

Figura 3.10 (a), (b), (c) y (d) Simulación con el Fuzzy Toolbox de Matlab.

Capítulo 4 Implementación con VHDL.

En este capítulo se describe la implementación del sistema de control difuso tipo Mamdani, para lo cual se utilizó el lenguaje de descripción de hardware VHDL mediante el software de desarrollo ISE Design Suite 14.7. Para la implementación se diseñó individualmente cada una de las etapas del sistema de control difuso, validando el funcionamiento de cada una de ellas para finalmente unir cada uno de los bloques, mediante las herramientas componente e instanciación. El sistema de control completo se implementó en una tarjeta de desarrollo NEXYS 4 DDR, que cuenta con un FPGA de la familia Artix-7 de Xilinxs.

Los resultados de la respuesta del seguidor se presentan al final del capítulo.

4.1 Conversión analógico-digital.

La conversión analógica-digital es una de las etapas fundamentales para el desarrollo del sistema de control, el cual consiste en la transcripción de señales analógicas en señales digitales, en el que la señal de la etapa de adquisición de datos (voltaje) es convertida a un número binario (valores discretos de voltaje) con el propósito de facilitar su procesamiento (codificación, compresión, etc.) y hacer la señal resultante (digital) más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas. Para la conversión analógico-digital del sistema de control, los datos provenientes de la etapa de adquisición de datos son ingresados a la tarjeta Nexys 4 mediante la interfaz de comunicación pmod y transformada mediante la herramienta XADC, para su implementación en la tarjeta se diseñó el siguiente código definiendo a dos puertos de entrada como VOLTAJE_IN y GND_IN y a la salida un puerto como OUTPUT_digital con una resolución de 10 bits, en la parte inferior se presenta una muestra del código requerido para su implementación.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
library unisim;
use unisim.vcomponents.all;
entity convertidor_a_d is
   port ( clk : in  std_logic;
   voltaje_in : in  std_logic;
   gnd_in : in  std_logic;
    output_digital : out  std_logic_vector (9 downto 0));
end convertidor_a_d;
architecture behavioral of convertidor_a_d is
component sensor_c
   port (
        daddr_in          : in  std_logic_vector (6 downto 0);
        dclk_in           : in  std_logic
        den_in            : in  std_logic
        di_in             : in  std_logic_vector (15 downto 0);
        dwe_in            : in  std_logic;
```

```
        reset_in        : in  std_logic;
        vauxp3          : in  std_logic;
        vauxn3          : in  std_logic;
        busy_out        : out  std_logic;
        channel_out     : out  std_logic_vector (4 downto 0);
        do_out          : out  std_logic_vector (15 downto 0);
        drdy_out        : out  std_logic;
        eoc_out         : out  std_logic;
        eos_out         : out  std_logic;
        alarm_out       : out std_logic;
        vp_in           : in  std_logic;
        vn_in           : in  std_logic);
end component;
```

La figura 4.1 muestra el esquemático de la conversión analógico-digital, en el que se implementan las entradas GND y Vcc para cada uno de los controles lógico-difusos, uno para el movimiento de elevación y otro para el movimiento azimut, estos voltajes se presentan de forma analógico para su conversión a digital con una resolución de 10 bits, cada salida es asignada como entrada para la etapa de fuzzificación del FLC.

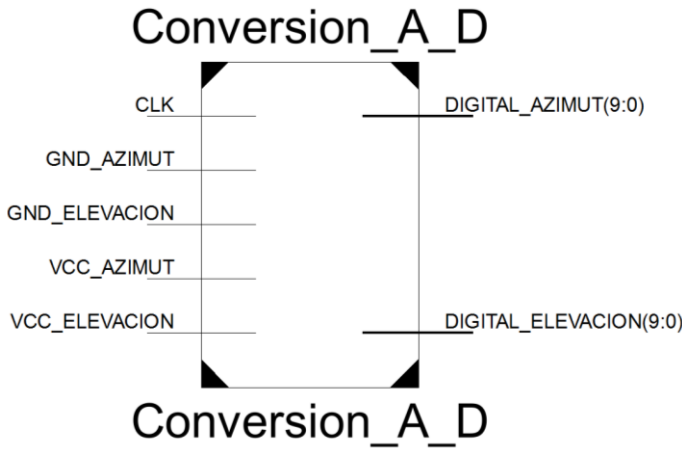


Figura 4.1 Esquemático de la conversión analógico - digital.

La figura 4.2 muestra la implementación en la tarjeta Nexys 4 del convertidor analógico-digital XADC, en este caso se implementó un arreglo divisor de voltaje con dos resistencias variables 1 Kohm, las cuales son reguladas para obtener el límite de voltaje máximo de 1.25 volts al cual trabaja el convertidor XADC. Este arreglo permite ajustar los

parámetros de trabajo del convertidor analógico-digital, los voltajes que son recibidos de la etapa de adquisición de datos son ajustados a un rango de 0 a 1.25 volts, estos datos son enviados a la primera etapa del control lógico difuso.

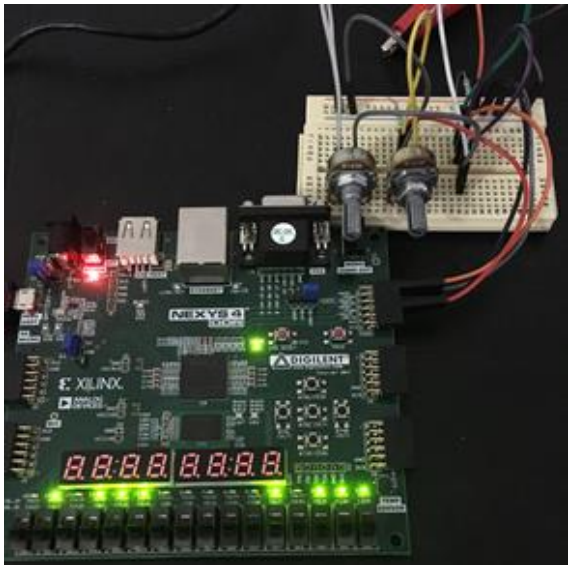


Figura 4.2 Implementación de la conversión analógico - digital.

4.2 Fuzzificación.

Para la etapa de fuzzificación se implementaron tablas de búsqueda o lookup tables por su traducción al inglés, mediante la utilización del condicional when, en el software de diseño electrónico ISE Design Suite 14.7, para lo cual se definieron las 2 entradas del sistema provenientes del sistema de adquisición de datos con una longitud de 10 bits de resolución, para las salidas se definieron las etiquetas utilizadas para las 5 funciones de membresía de cada conjunto difuso, con una longitud de resolución de 8 bits.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity fuzzificacion1 is
   Port (
        E1 : in  std_logic_vector (9 downto 0);
```

```
        DE1 : in  std_logic_vector (9 downto 0);
        MB1 : out  std_logic_vector (7 downto 0);
        B1 : out  std_logic_vector (7 downto 0);
        M1 : out  std_logic_vector (7 downto 0);
        A1 : out  std_logic_vector (7 downto 0);
        MA1 : out  std_logic_vector (7 downto 0);
        MB2 : out  std_logic_vector (7 downto 0);
        B2 : out  std_logic_vector (7 downto 0);
        M2 : out  std_logic_vector (7 downto 0);
        A2 : out  std_logic_vector (7 downto 0);
        MA2 : out  std_logic_vector (7 downto 0));
end fuzzificacion1;
architecture Behavioral of fuzzificacion1 is
```

En la figura 4.3 se muestra el esquemático obtenido en la etapa de fuzzificación, se aprecian las entradas E1 y DE1 con una resolución de 10 bits, estos valores de entrada son fuzzificados y los valores de salida son arrojados para cada una de las 10 posibilidades de salida con su respectiva etiqueta definida previamente con la función de membresía triangular.

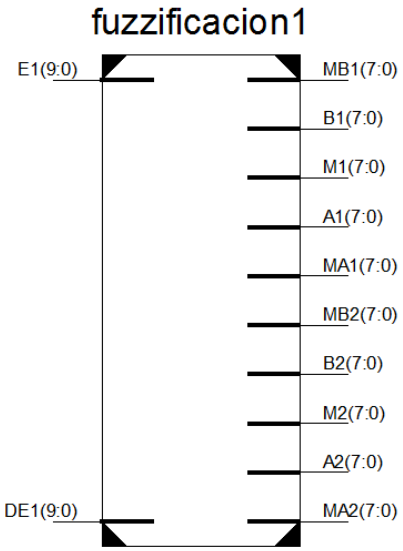


Figura 4.3 Esquemático para la etapa de fuzzificación.

En la figura 4.4 se muestra la simulación realizada para esta etapa, en el que se tomaron valores aleatorios de entrada. Los cuales fueron calculados teóricamente para poder verificar

el correcto funcionamiento de la etapa de fuzzificación implementada. Los valores de entrada se pueden apreciar como e1 y de1 así, como los diferentes valores de salida obtenidos. Se pudo validar el correcto funcionamiento del bloque de fuzzificación.

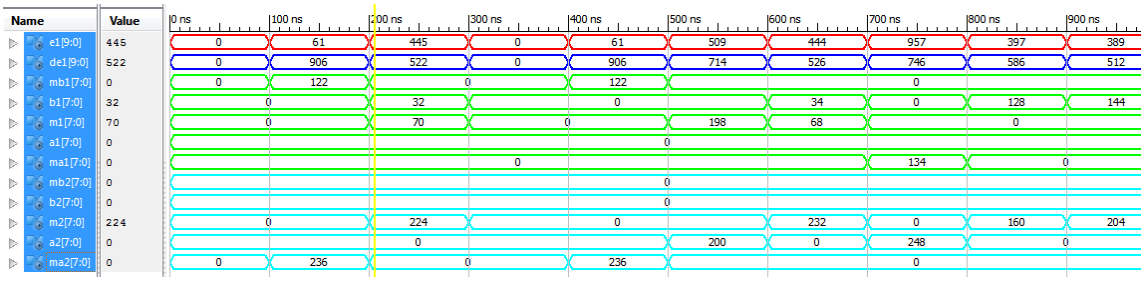


Figura 4.4 Simulación para la etapa de fuzzificación mediante la herramienta Test Bench.

4.3 Inferencia.

Para el caso de la inferencia del sistema difuso, para el diseño en VHDL se implementaron como entradas las salidas de la etapa anterior, es decir las entradas de la segunda etapa son las etiquetas de las salidas de membresía, para la salida se implementaron 25 salidas etiquetadas como valores mínimos, tanto las entradas como las salidas se implementan con 8 bits de resolución. Mediante el bucle loop se implementó la matriz de 5x5 en la cual se realiza un barrido entre cada una de las funciones de membresía para obtener los valores mínimos, es decir se tienen 2 conjuntos difusos de 5 funciones de membresía, obteniendo así 25 valores posibles, en la parte inferior se presenta una muestra de la implementación del código en VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity minimos1 is
port ( reset : in std_logic ;
        MB1 : out  std_logic_vector (7 downto 0);
        B1 : out  std_logic_vector (7 downto 0);
```

```
M1 : out  std_logic_vector (7 downto 0);
A1 : out  std_logic_vector (7 downto 0);
MA1 : out  std_logic_vector (7 downto 0);
MB2 : out  std_logic_vector (7 downto 0);
B2 : out  std_logic_vector (7 downto 0);
M2 : out  std_logic_vector (7 downto 0);
A2 : out  std_logic_vector (7 downto 0);
MA2 : out  std_logic_vector (7 downto 0));
min_11 : out  std_logic_vector (7 downto 0);
min_12 : out  std_logic_vector (7 downto 0);
min_13 : out  std_logic_vector (7 downto 0);
min_14 : out  std_logic_vector (7 downto 0);
min_15 : out  std_logic_vector (7 downto 0);
```

En la figura 4.5 se muestra el esquemático obtenido en la etapa de inferencia, se aprecian las entradas etiquetadas con el correspondiente nombre de la función de transferencia para E1 y DE1 con una resolución de 8 bits. Los valores de salida son los valores mínimos obtenidos de la matriz implementada de 5x5, de tal manera que a la salida se muestra 25 valores etiquetados como min_11 hasta min_55, los valores de salida tienen una resolución de 8 bits.

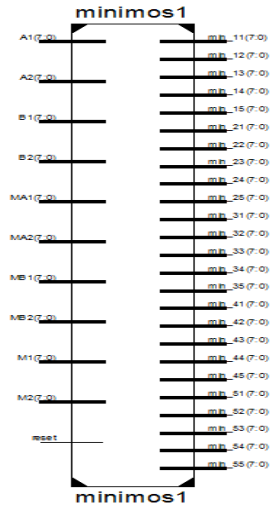


Figura 4.5 Esquemático para la etapa de inferencia.

En la figura 4.6 se muestra la simulación realizada para esta etapa, en el que se tomaron valores aleatorios de entrada. Los cuales fueron calculados teóricamente para poder verificar el correcto funcionamiento de la etapa de inferencia implementada. Los valores de entrada se pueden apreciar como mb1,b1,a1,ma1 y a2, los valores de salida se obtienen a través de min_11 hasta min_55. Se pudo validar el correcto funcionamiento del bloque de fuzzificación.

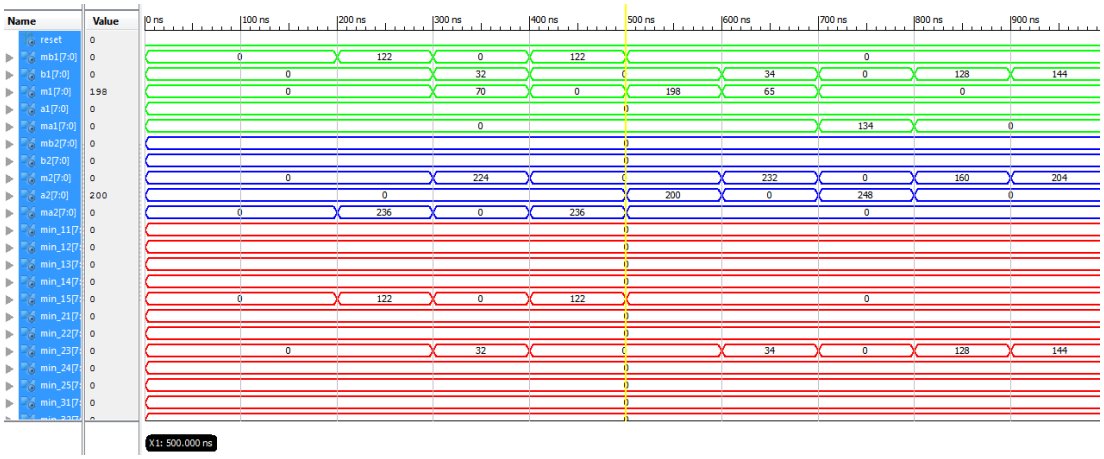


Figura 4.6 Simulación para la etapa de fuzzificación mediante la herramienta Test Bench.

4.4 Agregación.

Para desarrollar la etapa de agregación, se habilitaron 25 entradas, las cuales son los valores mínimos obtenidos de la etapa de inferencia, estos valores fueron reducidos a 5 valores a la salida, en esta etapa los valores mínimos de entrada fueron comparados entre cada una de las columnas formadas con la matriz de 5x5, de esta forma el valor resultado de cada columna es el valor máximo de cada una de ellas, en la parte inferior se presenta una muestra de la implementación del código en VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity maximos1 is
```

```
   port ( min_11 : in  std_logic_vector (7 downto 0);
          min_12 : in  std_logic_vector (7 downto 0);
          min_13 : in  std_logic_vector (7 downto 0);
          min_14 : in  std_logic_vector (7 downto 0);
          .
          .
          .
          min_55 : in  std_logic_vector (7 downto 0);
          max_01 : out  std_logic_vector (7 downto 0);
          max_02 : out  std_logic_vector (7 downto 0);
          max_03 : out  std_logic_vector (7 downto 0);
          max_04 : out  std_logic_vector (7 downto 0);
          max_05 : out  std_logic_vector (7 downto 0));
end maximos1;
```

En la figura 4.7 se muestra el esquemático obtenido en la etapa de agregación, se aprecian las entradas etiquetadas como min_11 hasta min_55, provenientes de la etapa de inferencia, una vez que se realiza el proceso de obtención de los valores máximos para cada columna, a la salida se muestran los cinco valores resultantes, etiquetados como max_01, max_02, max_03, max_04 y max_05 con una resolución de 8 bits.

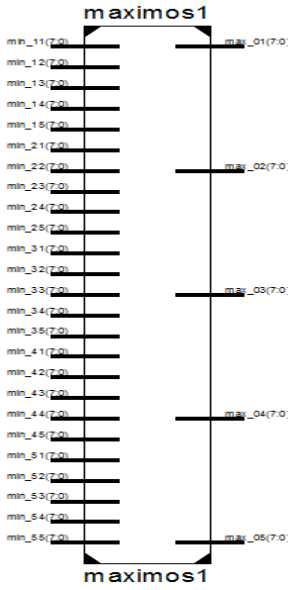


Figura 4.7 Esquemático para la etapa de agregación.

En la figura 4.8 se muestra la simulación realizada para la etapa de agregación, en el que se tomaron valores aleatorios de entrada. Los cuales fueron calculados teóricamente para poder verificar el correcto funcionamiento de la etapa implementada. Los valores de entrada se pueden apreciar como min_11 hasta min_, los valores de salida se obtienen a través de max_01 hasta max_55. Con los valores obtenidos se pudo validar el correcto funcionamiento del bloque de fuzzificación.

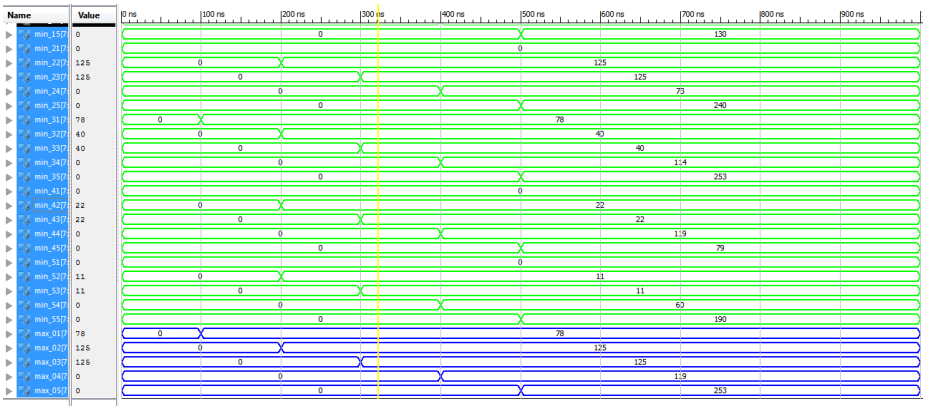


Figura 4.8 Simulación para la etapa de agregación mediante la herramienta Test Bench.

4.5 Defuzzificación.

La etapa de defuzzificación afecta directamente la precisión del resultado y el rendimiento del sistema difuso. Por esta razón, se encuentran diversas formas diferentes de defuzzificar, algunos de ellos relacionados con la precisión, y algunos otros preocupados por la eficiencia, en este caso con la finalidad de aumentar la velocidad de respuesta del sistema de control, el método utilizado para implementar la etapa de defuzzificación es el método de COSAA, ya que a diferencia del método de centro de gravedad COG este método muestra una velocidad de respuesta mayor, aunque el método COG, tiene una mejor exactitud, en comparación con el método MOM este tiene mayor velocidad de respuesta, por esta razón se decide implementar el método COSAA, ya que este tiene velocidad y exactitud. Para implementarlo en VHDL, se utilizó la ecuación 2.9, en este caso se implementaron 5 valores de entrada, que son los valores que provienen de la etapa de agregación a la salida tenemos un solo valor, el cual es el valor esperado del FLC, el cual se va a acondicionar para manipular al actuador.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity defuzzificacion1 is
   port ( max_01 : in  std_logic_vector (7 downto 0);
          max_02 : in  std_logic_vector (7 downto 0);
          max_03 : in  std_logic_vector (7 downto 0);
          max_04 : in  std_logic_vector (7 downto 0);
          max_05 : in  std_logic_vector (7 downto 0);
          out_flc : out  std_logic_vector (9 downto 0));
end defuzzificacion1;
architecture behavioral of defuzzificacion1 is
signal alfa_inicio1 : std_logic_vector (9 downto 0);
signal alfa_final1 : std_logic_vector (9 downto 0);
signal alfa_inicio2 : std_logic_vector (9 downto 0);
signal alfa_final2 : std_logic_vector (9 downto 0);
```

En la figura 4.9 se muestra el esquemático obtenido en la etapa de defuzzificación, en el se pueden apreciar las entradas etiquetadas como max_01 hasta max_55 provenientes de la etapa de agregación, en este bloque es en donde se implementó el método para defuzzificar COSAA, a la salida se obtienen los valores defuzzificados con una resolución de 10 bits, se etiqueto como out_flc.

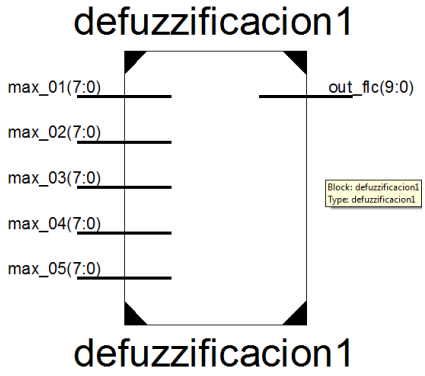


Figura 4.9 Esquemático para la etapa de defuzzificación.

En la figura 4.10 se muestra la simulación realizada con Test Bench de Ise Design para la etapa de defuzzificación, en el que se tomaron valores aleatorios de entrada. Los cuales fueron calculados teóricamente para poder verificar el correcto funcionamiento de la etapa implementada. Los valores de entrada se pueden apreciar como max_01 hasta max_, los valores de salida se obtienen a través de out_flc. Con los valores obtenidos se pudo validar el correcto funcionamiento del bloque de fuzzificación.

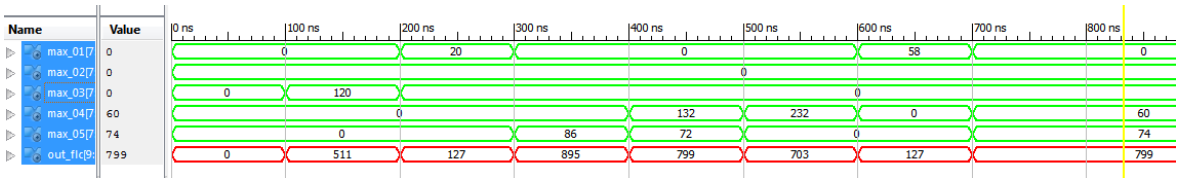


Figura 4.10 Simulación para la etapa de defuzzificación mediante la herramienta Test Bench.

4.6 Instanciación FLC.

En la etapa de instanciación se crea un solo bloque a partir de otros bloques de menor jerarquía, en este caso en las figuras 4.11 se muestra el esquemático, que se obtiene al integrar cada uno de los bloques diseñados, para generar el sistema de control que permita generar los movimientos necesarios para el seguimiento solar. Como se puede apreciar en la figura se cuenta con dos entradas, por las cuales ingresaran los voltajes provenientes de la etapa de adquisición de datos, a la salida se tienen los valores defuzzificados con una resolución de 10 bits, los cuales se podrán acondicionar a una etapa de potencia que permita generar el movimiento de seguimiento solar.

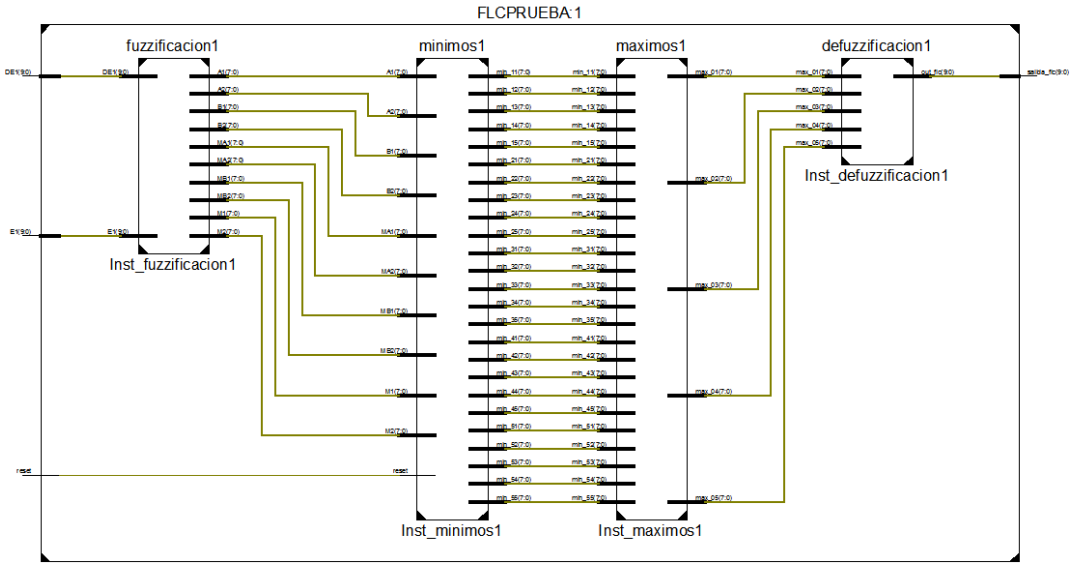


Figura 4.11 Esquemático FLC.

En la figura 4.12 se muestra la simulación realizada para verificar el funcionamiento del FLC completo, en el que se tomaron valores aleatorios de entrada. Los cuales fueron calculados teóricamente para poder verificar el correcto funcionamiento del sistema de control difuso. Los valores de entrada se pueden apreciar como e1 y de1, los valores de salida se obtienen a través de salida_flc. Con los valores obtenidos se pudo validar el correcto funcionamiento del sistema.

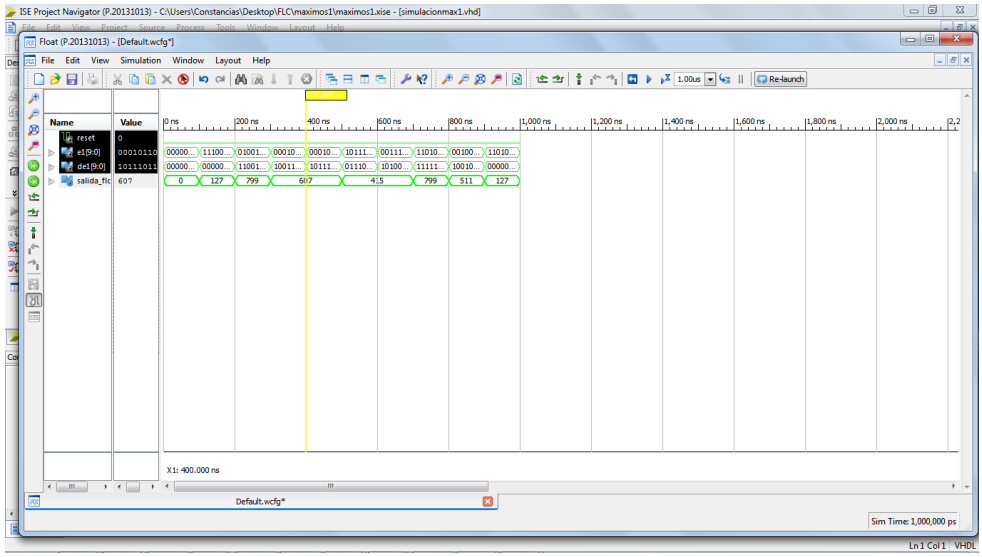


Figura 4.12 Simulación FLC mediante la herramienta Test Bench.

4.7 Resultados.

Con la finalidad de verificar el funcionamiento del sistema de control difuso diseñado, se dispuso a realizar las pruebas de validación. Dicha prueba consistió en evaluar el desempeño del controlador mediante su implementación en la tarjeta Nexys 4 DDR, los valores obtenidos fueron validados, con la herramienta de simulación Test Bench de Ise Design Suite, la herramienta Fuzzy Logic Tools de Matlab y los valores teóricos calculados Esta prueba consiste en evaluar el desempeño del FLC en la tarjeta de desarrollo FPGA.

La tabla 4.1, muestra algunos resultados experimentales obtenidos durante la realización de esta prueba, en el que se tomó un valor constante de 780 para la entrada x y valores aleatorios para la entrada y, se puede apreciar que los valores obtenidos entre Matlab y el FPGA son semejantes, en la figura 4.13 se muestra gráficamente el comportamiento de ambas herramientas, se puede observar que el comportamiento del FLC diseñado es el mismo que los valores que arroja Matlab.

Tabla 4.1 FPGA-MATLAB.

Muestra	*x*	*y*	FPGA	MATLAB
1	780	50	128	127
2	780	100	128	127
3	780	150	128	127
4	780	200	128	127
5	780	250	223	223
6	780	300	319	319
7	780	350	319	319
8	780	400	319	319
9	780	450	415	415
10	780	500	511	511
11	780	550	511	511
12	780	600	511	511
13	780	650	607	607
14	780	700	703	703
15	780	750	703	703
16	780	800	799	799
17	780	850	799	799
18	780	900	896	895
19	780	950	896	895
20	780	1000	896	895
21	780	1020	896	895

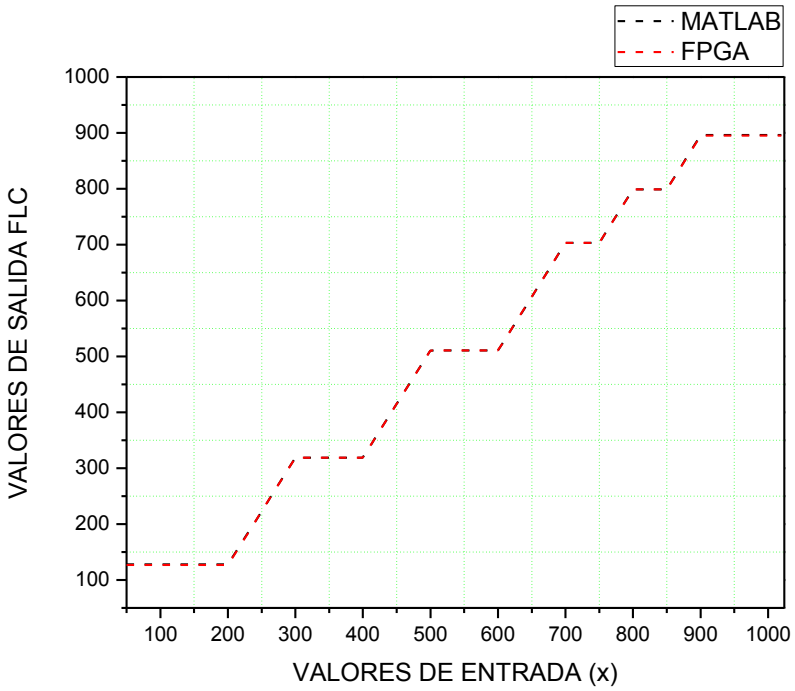


Figura 4.13 Valores obtenidos FPGA-Matlab

Una segunda prueba, consistió en validar el funcionamiento mediante los datos de entrada de la etapa de adquisición de datos, en la figura 4.14 se muestra la implementación física para la validación del funcionamiento en el que los sensores y periféricos son interconectados al pmod XADC de la tarjeta Nexys 4 DDR, para esta prueba se tomaron mediciones a lo largo del día 07 de octubre de 2018, de 7 a 18 horas con un lapso de tiempo de 1 hora, para poder validar el funcionamiento se tomaron los valores de salida de la etapa defuzzificación, a los valores obtenidos (digitales con una resolución de 10 bits), se le asigno su correspondiente valor en grados, una salida para el ángulo de elevación y otra para el ángulo azimut, en la tabla 4.2 se aprecian los valores obtenidos.

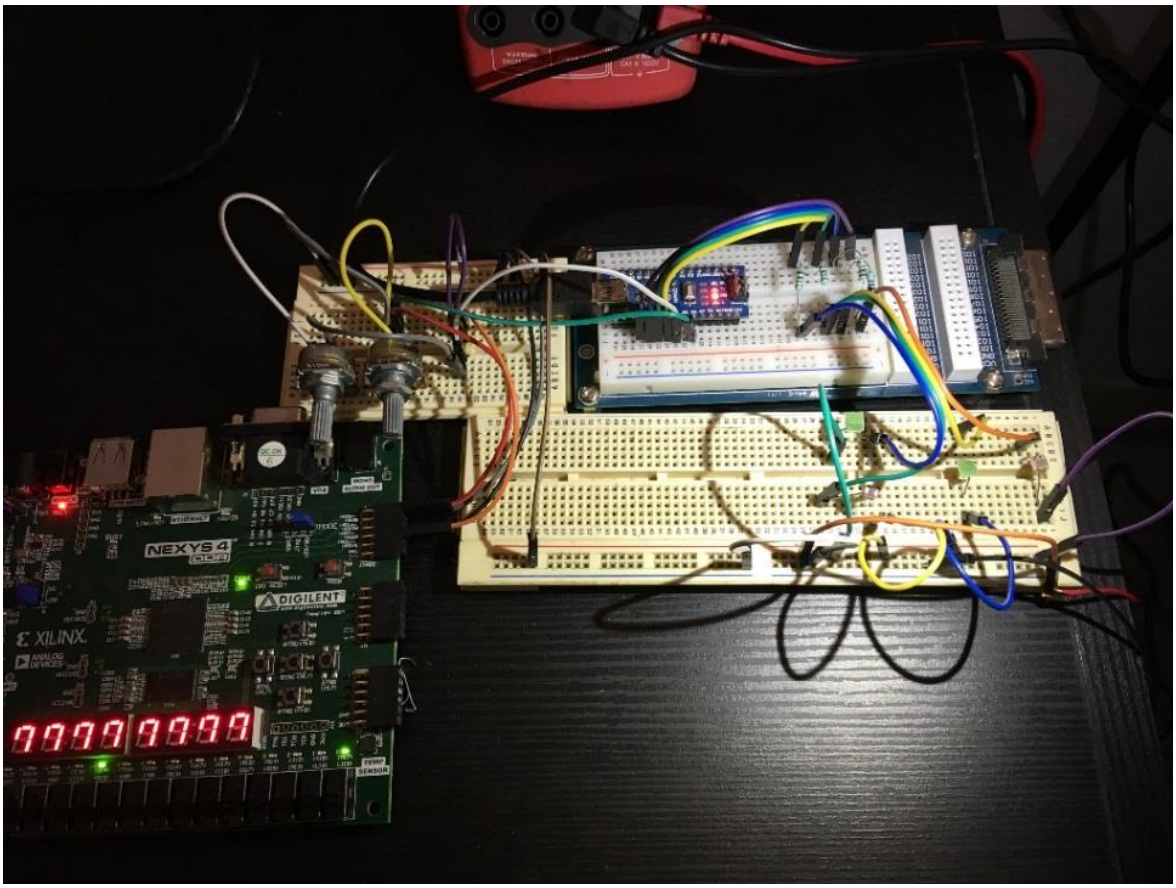


Figura 4.14 Implementación sistema completo.

Tabla 4.2 Valores obtenidos mediante la simulación para los movimientos de elevación y azimut.

Hora del día 24 hrs.	Elevación teórico	Valor defuzzificación	Correspondiente valor en grados elevación.	Azimut teórico.	Valor defuzzificación FLC	Correspondiente valor en grados azimut
7	1.6	61	3	109.38012	458	112
8	14.4	348	17	115.90007	479	117
9	25.7	573	28	124.72029	516	126
10	36.5	675	33	137.02509	569	139
11	44.6	982	48	153.9905	642	157
12	49.1	1023	50	175.3648	720	176
13	48.6	962	47	196.0095	814	199
14	42.9	818	40	212.97491	884	216
15	34	634	31	225.27971	925	226
16	23	368	18	234.09993	974	238
17	11	205	10	240.61988	986	241
18	-1.3	0	0	245.64255	1015	248

En la figura 4.15 se muestra gráficamente el comparativo entre los valores obtenidos para el ángulo de elevación calculados teóricamente y los valores que arroja el FLC diseñado, se puede apreciar que los valores obtenidos con las pruebas físicas el comportamiento es semejante a los valores calculados y mostrados en el capítulo 1.

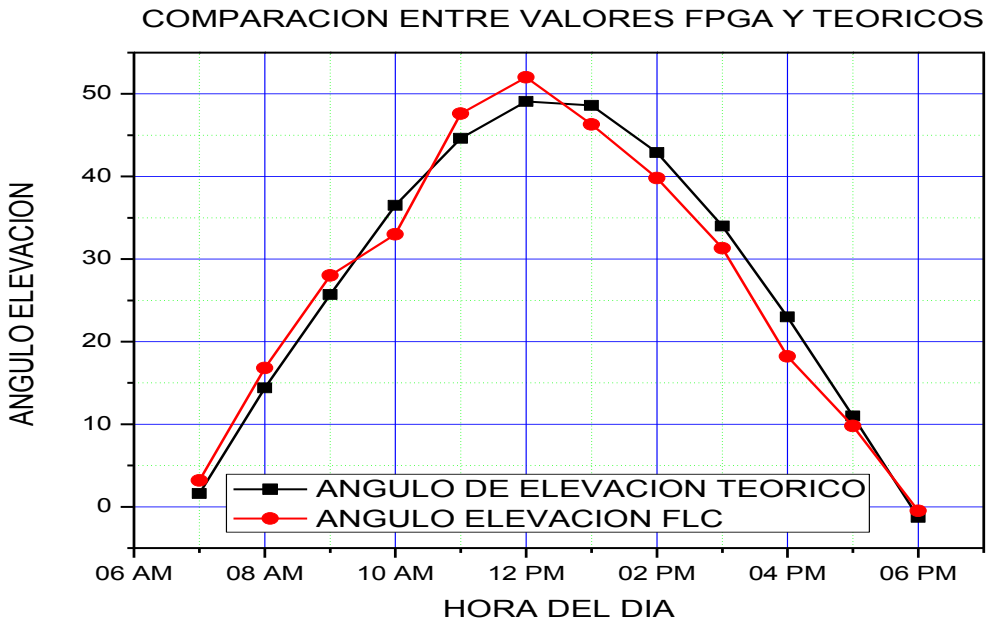


Figura 4.15 Valores obtenidos con el FLC y valores teóricos para el movimiento de elevación.

En la figura 4.16 se muestra gráficamente el comparativo entre los valores obtenidos para el ángulo azimut calculados teóricamente y los valores que arroja el FLC diseñado, se puede apreciar que los valores obtenidos con las pruebas físicas el comportamiento es semejante a los valores calculados y mostrados en el capítulo 1, para este tipo de movimiento.

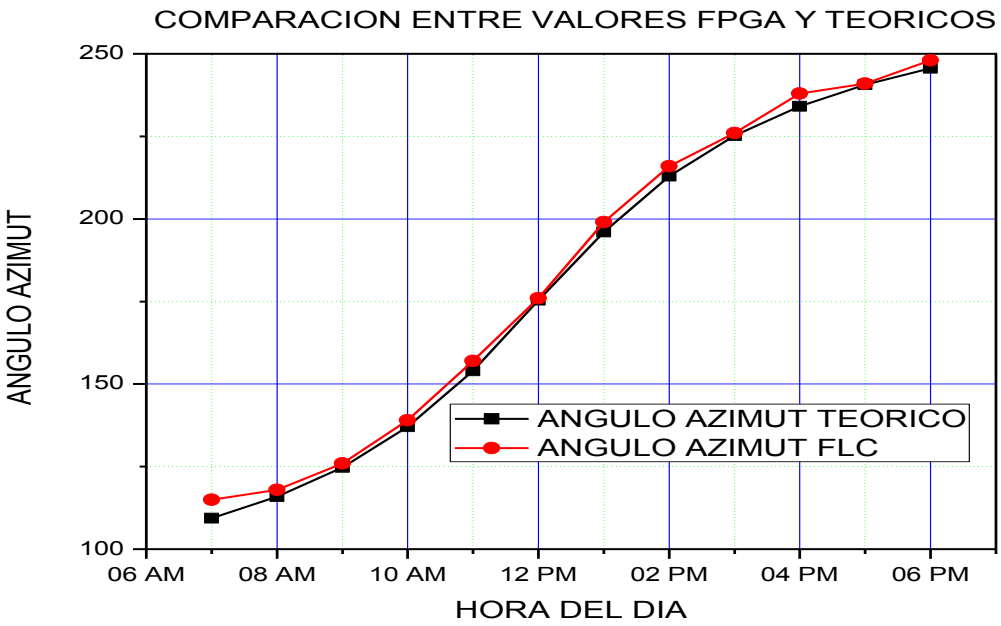


Figura 4.16 Valores obtenidos con el FLC y valores teóricos para el movimiento azimut.

Por tanto, en ambos casos se verificó el correcto funcionamiento del control difuso diseñado y el correcto funcionamiento de la etapa implementada con el método de defuzzificación (COSAA).

En la tabla 4.3 se muestra un comparativo entre los dispositivos lógicos utilizados por el FLC en la tarjeta de trabajo Nexys 4 DDR, se puede observar que los elementos de hardware utilizados en esta investigación son notablemente menores a los disponibles en la tarjeta.

Tabla 4.3 Elementos utilizados para la implementación del sistema de control difuso en la tarjeta de trabajo Nexys 4 DDR.

Resumen de los elementos utilizados en el diseño del FLC		
Elemento	Utilizados	Disponibles
Número de registros	40	126,800
Núumero de LUTS	30	63,400
Número completo de pares LUT-FF	20	41
Número IOBs	24	210

4.8 Conclusiones.

En el presente trabajo de investigación se diseñó y simuló el algoritmo para un sistema de seguimiento solar mediante lógica difusa tipo Mamdani, en un dispositivo FPGA, con la finalidad que dicho algoritmo puedo ser acondicionado a una etapa de potencia y adaptado a cualquier estructura de seguimiento solar que cuente con dos grados libertad.

Con base en los resultados obtenidos, se concluye que:

La selección de la tarjeta Nexys 4 DDR que cuenta con un FPGA de la familia Artix™-7, para la implementación del sistema de control difuso fue el idóneo, por su versatilidad de respuesta y por la gran cantidad de recursos con los que cuenta.

De las cuatro etapas diseñadas para el sistema de control difuso, la que requiere mayor tiempo de procesamiento, así como mayor demanda de recursos de hardware fue la etapa de fuzzificación, lo cual fue contrarrestado con la implementación de las look-up tables.

Los resultados obtenidos entre la respuesta de salida en la FPGA, los valores teóricos calculados, la herramienta de simulación de ISE Design suite 14.7 y los valores obtenidos mediante la herramienta de simulación Fuzzy Toolbox de Matlab, concuerdan y validan la exactitud del sistema.

La determinación de aplicar el método de defuzzificación COSAA fue el óptimo, que, aunque ofrece una menor exactitud que el método COG, facilita la implementación del mismo por la reducción en el número de operaciones implicadas para su inserción en el código VHDL.

Las pruebas del sistema y la simulación realizada permiten obtener un panorama general del funcionamiento del sistema bajo diferentes condiciones de operación. La ventaja de este sistema de control radica en que su uso y aplicación puede generalizarse a cualquier dispositivo capaz de ser controlado, con el único requerimiento de ajustar los parámetros de las funciones de membresía, tomando en cuenta los factores que afectan su rendimiento, así como el conocimiento que tenga el experto humano del sistema.

4.9 Trabajo a Futuro

Una vez concluido con el desarrollo, implementación y validación en FPGA del algoritmo de control difuso, este trabajo requiere en un futuro, primeramente, acoplar una etapa de potencia, así como el diseño de la estructura mecánica, en la cual se pueda aplicar el sistema diseñado.

Este trabajo permitirá implementar el algoritmo, no solamente en sistemas de seguimiento solar; el cual fue el propósito de dicha investigación, si no que podría extenderse y ser aplicado a cualquier sistema capaz de ser manipulado mediante sistemas de control inteligente.

Así también el algoritmo se puede complementar con otro tipo de control inteligente como una red neuronal que permita generar un sistema neuro-difuso.

Referencias.

[1] Is.Brahami “Optimization of a photovoltaic generator with a new solar tracker”, Journal of Electrical Engineering, 2015.

[2] Aldo V.Darosa, "Fundamental of Renewable Energy Processes", Elsevier INC, 2005.

[3] M. A. Usta, "Design and Performance of Solar Tracking System with Fuzzy Logic Controller" 6th International Advanced Technologies Symposium (IATS’11), 16-18 May 2011, Elazığ, Turkey

[4] John A. Duffie, Solar Engineering of Thermal Processes, four edition, John Wiley, USA 2013.

[5] C. Julian Chen, “Physics of Solar Energy”, John Wiley & Sons, Inc. New Jersey, USA 2011.

[6] José Beltrán Adán, Tesis de Maestría, “Prototipo fotovoltaico con seguimiento del Sol para procesos electroquímicos”, Centro Nacional de Investigación y Desarrollo Tecnológico 2007.

[7] Cotal, H., Fetzer, C., Boisvert, J., Kinsey, G., King, R., Hebert, P., ... & Karam, N. (2009). III–V multijunction solar cells for concentrating photovoltaics. *Energy & Environmental Science*, *2*(2), 174-192.

[8] Retratos de la Conexión fotovoltaica a la red, seguidores y huertas solares, Eduardo Lorenzo, 2003, Instituto de energía solar – Universidad Politécnica de Madrid.

[9] La energía solar, aplicaciones prácticas, colectivo, promotora general de estudios, España, quinta edición 2009.

[10] M.H.M. Sidek, N. Azis, W.Z.W. Hasan, M.Z.A. Ab KadirS, Shafie and M.A.M. Radzi "Automated positioning dual-axis solar tracking system with precision elevation and azimuth angle control". Elsevier, Journal home page, February 2017.

[11] Sahu, B. K., Pati, S., Mohanty, P. K., & Panda, S. (2015). Teaching–learning based optimization algorithm based fuzzy-PID controller for automatic generation control of multi-area power system. *Applied Soft Computing*, *27*, 240-249.

[12] Dhimish, M., Holmes, V., Mehrdadi, B., Dales, M., & Mather, P. (2017). Photovoltaic fault detection algorithm based on theoretical curves modelling and fuzzy classification system. *Energy*, *140*, 276-290.

[13] Friedman, D. J. (2010). Progress and challenges for next-generation high-efficiency multijunction solar cells. *Current Opinion in Solid State and Materials Science*, *14*(6), 131-138.

[14] Gad HH, Haikal AY, Ali HA. New design of the PV panel control system using FPGA-based MPSoC. Sol Energy 2017; 146:243–56.

[15] Wang, A., & Xuan, Y. (2017). A detailed study on loss processes in solar cells. *Energy*.

[16] Essig, S., Ward, S., Steiner, M. A., Friedman, D. J., Geisz, J. F., Stradins, P., & Young, D. L. (2015). Progress towards a 30% efficient GaInP/Si tandem solar cell. *Energy Procedia*, *77*, 464-469.

[17] Tendencias tecnológicas mundiales en el desarrollo y aplicación de paneles solares fotovoltaicos, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT) e IALE Tecnología, S.L. Mayo 2014

[18] Miller, D. C., & Kurtz, S. R. (2011). Durability of Fresnel lenses: a review specific to the concentrating photovoltaic application. *Solar Energy Materials and Solar Cells*, *95*(8), 2037-2068.

[19] Nishioka, K., Ikematsu, K., Ota, Y., & Araki, K. (2012). Sandblasting durability of acrylic and glass Fresnel lenses for concentrator photovoltaic modules. *Solar Energy*, *86*(10), 3021-3025.

[20] Xie, W. T., Dai, Y. J., Wang, R. Z., & Sumathy, K. (2011). Concentrated solar energy applications using Fresnel lenses: A review. *Renewable and Sustainable Energy Reviews*, *15*(6), 2588-2606.

[21] Wang, G., Chen, Z., Hu, P., & Cheng, X. (2016). Design and optical analysis of the band-focus Fresnel lens solar concentrator. *Applied Thermal Engineering*, *102*, 695-700.

[22] Wilamowski B. M. "Neuro-Fuzzy Systems and its applications" tutorial at 24th IEEE International Industrial Electronics Conference (IECON'98) August 31 - September 4, 1998, Aachen, Germany, vol. 1, pp. t35-t49.

[23] M.H.M. Sidek, N. Azis, W.Z.W. Hasan, M.Z.A. Ab KadirS, Shafie and M.A.M. Radzi "Automated positioning dual-axis solar tracking system with precision elevation and azimuth angle control". Elsevier, Journal home page, February 2017.

[24] Yager, R. R., & Zadeh, L. A. (Eds.). (2012). *An introduction to fuzzy logic applications in intelligent systems* (Vol. 165). Springer Science & Business Media.

[25] Hwang, G. C., & Lin, S. C. (1992). A stability approach to fuzzy control design for nonlinear systems. *Fuzzy sets and Systems*, *48*(3), 279-287.

[26] K Passino, S. Yurkovich, "Fuzzy Control", 1st ed, Addison Wesley, California, USA 1998.

[27] Li, Y., Tong, S., & Li, T. (2015). Observer-based adaptive fuzzy tracking control of MIMO stochastic nonlinear systems with unknown control directions and unknown dead zones. *IEEE Transactions on Fuzzy Systems*, *23*(4), 1228-1241.

[28] Collotta, M., Bello, L. L., & Pau, G. (2015). A novel approach for dynamic traffic lights management based on Wireless Sensor Networks and multiple fuzzy logic controllers. *Expert Systems with Applications*, *42*(13), 5403-5415.

[29] Araceli Grande Meza, tesis de maestría, "Observadores Difusos y Control Adaptable Difuso Basado en Observadores, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional 2003.

[30] Ross, T. J. (2009). *Fuzzy logic with engineering applications*. John Wiley & Sons.

[31] Sundarabalan, C. K., Selvi, K., & Kubra, K. S. (2015). Performance investigation of fuzzy logic controlled MPPT for energy efficient solar PV systems. In *Power Electronics and Renewable Energy Systems* (pp. 761-770). Springer, New Delhi.

[32] de la Cruz-Alejo, J., Antonio-Méndez, R., & Salazar-Pereyra, M. Fuzzy logic control on FPGA for two axes solar tracking. *Neural Computing and Applications*, 1-15.

[33] Hernández Zavala, M. A. (2009). *Arquitectura de alto rendimiento para procesadores difusos* (Doctoral dissertation).

[34] Zavala, A. H., Ruelas, J. A. H., & Nieto, O. C. (2013). Centre of Slice Area Average Defuzzifier For Digital Fuzzy Systems and Its Hardware Implementation. *Journal of Multiple-Valued Logic & Soft Computing*, *21*.

[35] https://reference.digilentinc.com/_media/nexys4-ddr:nexys4ddr_rm.pdf

[36] Sharma, A. K. (1998). *Programmable Logic Handbook: PLDs, CPLDs, and FPGAs* (pp. 99-171). New York: McGraw-Hill.

[37] Dally, W. J., & Chang, A. (2000, June). The role of custom design in ASIC chips. In *Proceedings of the 37th Annual Design Automation Conference* (pp. 643-647). ACM.

[38] Iida, M. (2018). What Is an FPGA?. In *Principles and Structures of FPGAs* (pp. 23-45). Springer, Singapore.

[39] Pellerin, D., & Thibault, S. (2005). *Practical fpga programming in c*. Prentice Hall Press.

[40] Cong, J., & Ding, Y. (1994). FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *13*(1), 1-12.

ANEXO A: Código para la etapa de adquisición de datos en Arduino.

Código implementado en Arduino para la etapa de adquisición de datos.

```
int valorPot0=0;
int valorPot1=0;
int valorPot2=0;
int valorPot3=0;
void setup() {
  // put your setup code here, to run once:
pinMode(11,OUTPUT);// ENTRADA A0===SALIDA 1
pinMode(10,OUTPUT);// ENTRADA A1===SALIDA 2
pinMode(9,OUTPUT);//  ENTRADA A2===SALIDA 3
pinMode(3,OUTPUT);//  ENTRADA A3===SALIDA 4
Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
valorPot0=analogRead(A0);
valorPot0=map(valorPot0,0,1023,0,80);
analogWrite(11,valorPot0);
delay(1000);
valorPot0=analogRead(A0);
Serial.println(valorPot0);
delay(1000);
//////////////////////////////// SALIDA 2
valorPot1=analogRead(A1);
valorPot1=map(valorPot1,0,1023,0,80);
analogWrite(10,valorPot1);
delay(1000);
valorPot1=analogRead(A1);
Serial.println(valorPot1);
delay(1000);
```

```
////////////////////////////////SALIDA 3
valorPot2=analogRead(A2);
valorPot2=map(valorPot2,0,1023,0,80);
analogWrite(9,valorPot2);
delay(1000);
valorPot2=analogRead(A2);
Serial.println(valorPot2);
delay(1000);
////////////////////////////////SALIDA 4
valorPot3=analogRead(A3);
valorPot3=map(valorPot3,0,1023,0,80);
analogWrite(3,valorPot3);
delay(1000);
valorPot3=analogRead(A3);
Serial.println(valorPot3);
delay(1000);
}
```

ANEXO B: Código para la conversión analógica-digital.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
library UNISIM;
use UNISIM.VComponents.all;

entity Conversion_A_D is
   Port ( CLK : in  STD_LOGIC;
          GND_AZIMUT : in  STD_LOGIC;
          GND_ELEVACION : in  STD_LOGIC;
          VCC_AZIMUT : in  STD_LOGIC;
          VCC_ELEVACION : in  STD_LOGIC;
          DIGITAL_AZIMUT : out  STD_LOGIC_VECTOR (9 downto 0);
          DIGITAL_ELEVACION : out  STD_LOGIC_VECTOR (9 downto 0));
end Conversion_A_D;
architecture Behavioral of Conversion_A_D is
component converter
   port (
        DADDR_IN          : in  STD_LOGIC_VECTOR (6 downto 0);     -- Address bus for
the dynamic reconfiguration port
        DCLK_IN           : in  STD_LOGIC;
        DEN_IN            : in  STD_LOGIC;
        DI_IN             : in  STD_LOGIC_VECTOR (15 downto 0);
        DWE_IN            : in  STD_LOGIC;
        RESET_IN          : in  STD_LOGIC;
        VAUXP3            : in  STD_LOGIC;
        VAUXN3            : in  STD_LOGIC;
        VAUXP11           : in  STD_LOGIC
        VAUXN11           : in  STD_LOGIC;
        BUSY_OUT          : out  STD_LOGIC;
        CHANNEL_OUT       : out  STD_LOGIC_VECTOR (4 downto 0);
        DO_OUT            : out  STD_LOGIC_VECTOR (15 downto 0);
        DRDY_OUT          : out  STD_LOGIC;
        EOC_OUT           : out  STD_LOGIC;
        EOS_OUT           : out  STD_LOGIC;
        ALARM_OUT         : out STD_LOGIC;
        VP_IN             : in  STD_LOGIC;
        VN_IN             : in  STD_LOGIC);
end component;
```

```
SIGNAL CLK2 : STD_LOGIC;
SIGNAL CQ : STD_LOGIC_VECTOR(28 DOWNTO 0);
SIGNAL ENABLE : STD_LOGIC;
SIGNAL ES     : STD_LOGIC;
SIGNAL READY  : STD_LOGIC;
SIGNAL DATA   : STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL ADDRESS_IN    : STD_LOGIC_VECTOR(6 DOWNTO 0);
SIGNAL DI     : STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL DW     : STD_LOGIC;
SIGNAL RE     : STD_LOGIC;
SIGNAL BU     : STD_LOGIC;
SIGNAL CHA    : STD_LOGIC_VECTOR(4 DOWNTO 0);
SIGNAL ALA    : STD_LOGIC;
SIGNAL VP     : STD_LOGIC;
SIGNAL VN     : STD_LOGIC;
Begin
PROCESS (CLK)
BEGIN
IF (CLK'EVENT AND CLK = '1') THEN
CQ<=CQ+1;
CLK2<=CQ(28);
END IF;
END PROCESS;
PROCESS (CLK2)
BEGIN
IF (CLK2'EVENT AND CLK2 = '1') THEN
IF (READY = '0') THEN
DIGITAL_AZIMUT                                                    <=
(DATA(15),DATA(14),DATA(13),DATA(12),DATA(11),DATA(10),DATA(9),DATA(8)
,DATA(7),DATA(6));
--DIGITAL_ELEVACION                                               <=
(DATA(15),DATA(14),DATA(13),DATA(12),DATA(11),DATA(10),DATA(9),DATA(8)
,DATA(7),DATA(6));
END IF;
END IF;
END PROCESS;
```

ANEXO C: Código para la implementación de la etapa de fuzzificación.

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

entity fuzzificacion1 is
   Port (
        e1 : in  STD_LOGIC_VECTOR (9 downto 0);
        de1 : in  STD_LOGIC_VECTOR (9 downto 0);
        MB1 : out  STD_LOGIC_VECTOR (7 downto 0);
        B1 : out  STD_LOGIC_VECTOR (7 downto 0);
        M1 : out  STD_LOGIC_VECTOR (7 downto 0);
        A1 : out  STD_LOGIC_VECTOR (7 downto 0);
        MA1 : out  STD_LOGIC_VECTOR (7 downto 0);
        MB2 : out  STD_LOGIC_VECTOR (7 downto 0);
        B2 : out  STD_LOGIC_VECTOR (7 downto 0);
        M2 : out  STD_LOGIC_VECTOR (7 downto 0);
        A2 : out  STD_LOGIC_VECTOR (7 downto 0);
        MA2 : out  STD_LOGIC_VECTOR (7 downto 0));
end fuzzificacion1;


architecture Behavioral of fuzzificacion1 is

begin

process (E1)

begin

case e1 is
when   "0000000000"=>      MB1   <=      "00000000"    ;
when   "0000000001"=>      MB1   <=      "00000010"    ;
when   "0000000010"=>      MB1   <=      "00000100"    ;
when   "0000000011"=>      MB1   <=      "00000110"    ;

process (DE1)
begin
case de1 is
when   "0000000000"=>      MB2   <=      "00000000"    ;
when   "0000000001"=>      MB2   <=      "00000010"    ;
```

```
when   "0000000010"=>     MB2   <=     "00000100"   ;
when   "0000000011"=>     MB2   <=     "00000110"   ;
when   "0000000100"=>     MB2   <=     "00001000"   ;
when   "0000000101"=>     MB2   <=     "00001010"   ;
when   "0000000110"=>     MB2   <=     "00001100"   ;
when   "0000000111"=>     MB2   <=     "00001110"   ;
when   "0000001000"=>     MB2   <=     "00010000"   ;
when   "0000001001"=>     MB2   <=     "00010010"   ;
when   "0000001010"=>     MB2   <=     "00010100"   ;


when   "1111110110"=>     MA2   <=     "00010100"   ;
when   "1111110111"=>     MA2   <=     "00010010"   ;
when   "1111111000"=>     MA2   <=     "00010000"   ;
when   "1111111001"=>     MA2   <=     "00001110"   ;
when   "1111111010"=>     MA2   <=     "00001100"   ;
when   "1111111011"=>     MA2   <=     "00001010"   ;
when   "1111111100"=>     MA2   <=     "00001000"   ;
when   "1111111101"=>     MA2   <=     "00000110"   ;
when   "1111111110"=>     MA2   <=     "00000100"   ;
when   "1111111111"=>     MA2   <=     "00000010"   ;
when others => MA2 <=     "00000000";

end case;
end process;

end Behavioral;
```

ANEXO D: Código para la implementación de la etapa de inferencia.

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

entity minimos1 is
   Port ( reset : in STD_LOGIC ;
                        MB1 : in  STD_LOGIC_VECTOR (7 downto 0);
          B1 : in  STD_LOGIC_VECTOR (7 downto 0);
          M1 : in  STD_LOGIC_VECTOR (7 downto 0);
          A1 : in  STD_LOGIC_VECTOR (7 downto 0);
          MA1 : in  STD_LOGIC_VECTOR (7 downto 0);
          MB2 : in  STD_LOGIC_VECTOR (7 downto 0);
          B2 : in  STD_LOGIC_VECTOR (7 downto 0);
          M2 : in  STD_LOGIC_VECTOR (7 downto 0);
          A2 : in  STD_LOGIC_VECTOR (7 downto 0);
          MA2 : in  STD_LOGIC_VECTOR (7 downto 0);
          min_11 : out  STD_LOGIC_VECTOR (7 downto 0);
          min_12 : out  STD_LOGIC_VECTOR (7 downto 0);
          min_13 : out  STD_LOGIC_VECTOR (7 downto 0);
          min_14 : out  STD_LOGIC_VECTOR (7 downto 0);
          min_15 : out  STD_LOGIC_VECTOR (7 downto 0);
begin

--PROCESO PARA ENCONTRAR LOS MINIMOS-
process (reset, comp1, min , MB1, B1, M1, A1, MA1, MB2, B2, M2, A2, MA2)

variable i,j: integer range 0 to 7;

begin
min (0,0) <=0;
min (0,1) <=0;
min (0,2) <=0;
min (0,3) <=0;
min (0,4) <=0;
min (1,0) <=0;
min (1,1) <=0;
min (1,2) <=0;
min (1,3) <=0;
min (1,4) <=0;
```

```
comp2 (0) <=conv_integer (MB2);
comp2 (1) <=conv_integer (B2);
comp2 (2) <=conv_integer (M2);
comp2 (3) <=conv_integer (A2);
comp2 (4) <=conv_integer (MA2);

If (reset = '1') then
comp1 (0) <= 0;
comp1 (1) <= 0;
comp1 (2) <= 0;
comp1 (3) <= 0;
comp1 (4) <= 0;

comp2 (0) <= 0;
comp2 (1) <= 0;
comp2 (2) <= 0;
comp2 (3) <= 0;
comp2 (4) <= 0;
else
for i in 0 to 4 loop
for j in 0 to 4 loop

min_51 <= conv_std_logic_vector (min (4,0),8);
min_52 <= conv_std_logic_vector (min (4,1),8);
min_53 <= conv_std_logic_vector (min (4,2),8);
min_54 <= conv_std_logic_vector (min (4,3),8);
min_55 <= conv_std_logic_vector (min (4,4),8);

end Behavioral;
```

ANEXO E: Código para la implementación de la etapa de agregación.

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

entity maximos1 is
   Port ( min_11 : in  STD_LOGIC_VECTOR (7 downto 0);
          min_12 : in  STD_LOGIC_VECTOR (7 downto 0);
          min_13 : in  STD_LOGIC_VECTOR (7 downto 0);
          min_14 : in  STD_LOGIC_VECTOR (7 downto 0);
          min_15 : in  STD_LOGIC_VECTOR (7 downto 0);

begin
max1 <=conv_integer (min_11);
max2 <=conv_integer (min_21);
max3 <=conv_integer (min_31);
max4 <=conv_integer (min_41);
max5 <=conv_integer (min_51);

----------máximo columna 1----------
if max1 > max2 then
max112 <= max1 ;
else
max112 <= max2;
end if;

if max3 > max4 then
max134 <= max3;
else
max134 <= max4;
end if;

if max112 > max134 then
maxc1 <= max112;
else
maxc1 <= max134;
end if;

if maxc1 > max5 then
maxcolumna1 <= maxc1;
```

```
else
maxcolumna1 <= max5;
end if;
max_01 <= conv_std_logic_vector ((maxcolumna1),8);
max_02 <= conv_std_logic_vector ((maxcolumna2),8);
max_03 <= conv_std_logic_vector ((maxcolumna3),8);
max_04 <= conv_std_logic_vector ((maxcolumna4),8);
max_05 <= conv_std_logic_vector ((maxcolumna5),8);
end Behavioral;
```

ANEXO F: Código para la implementación de la etapa de defuzzificación.

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

entity defuzzificacion1 is
   Port ( max_01 : in  STD_LOGIC_VECTOR (7 downto 0);
          max_02 : in  STD_LOGIC_VECTOR (7 downto 0);
          max_03 : in  STD_LOGIC_VECTOR (7 downto 0);
          max_04 : in  STD_LOGIC_VECTOR (7 downto 0);
          max_05 : in  STD_LOGIC_VECTOR (7 downto 0);
end defuzzificacion1;

architecture Behavioral of defuzzificacion1 is

signal alfa_inicio1 : std_logic_vector (9 downto 0);

signal alfa_final1 : std_logic_vector (9 downto 0);

signal alfa_inicio2 : std_logic_vector (9 downto 0);

signal alfa_final2 : std_logic_vector (9 downto 0);

signal alfa_inicio3 : std_logic_vector (9 downto 0);

signal alfa_final3 : std_logic_vector (9 downto 0);

signal alfa_inicio4 : std_logic_vector (9 downto 0);

signal alfa_final4 : std_logic_vector (9 downto 0);

signal alfa_inicio5 : std_logic_vector (9 downto 0);

signal alfa_final5 : std_logic_vector (9 downto 0);

signal alfa_A, alfa_B, alfa_C, alfa_D, alfa_E : integer;

signal alfa_inicio1A : integer;

signal alfa_final1A : integer;

signal alfa_inicio2A : integer;

if alfa_C > 0 and alfa_D > 0 then
  out_3 <= (alfa_C+alfa_D)/2;
 elsif  alfa_D < 1 then
  out_3 <= alfa_C;
  end if;
```

```
if alfa_D > 0 and alfa_E > 0 then
  out_4 <= (alfa_D+alfa_E)/2;
 elsif  alfa_E < 1 then
  out_4 <= alfa_D;
  end if;
 if alfa_D < 1 then
  out_5 <= alfa_E;
  end if;
 if out_1 /= 0 then
 out_6 <= out_1;
 elsif out_2 /= 0 then
 out_6 <= out_2;
 elsif out_3 /= 0 then
 out_6 <= out_3;
 elsif out_4 /= 0 then
 out_6 <= out_4;
 elsif out_5 /= 0 then
 out_6 <= out_5;
 elsif out_1 < 1 and out_2 < 1 and out_3 < 1 and out_4 < 1 and out_5 < 1 then
 out_6 <= 0;
 end if;
end process;
end Behavioral;
```

ANEXO G: Código para la instanciación en un solo bloque.

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

entity FLC_SEGUIDOR_AZIMUT is
   Port ( DE1 : in  STD_LOGIC_VECTOR (9 downto 0);
          E1 : in  STD_LOGIC_VECTOR (9 downto 0);
                        RESET : in  STD_LOGIC;
          SALIDA_AZIMUT : out  STD_LOGIC_VECTOR (9 downto 0));

end FLC_SEGUIDOR_AZIMUT;

architecture Behavioral of FLC_SEGUIDOR_AZIMUT is
signal MB1: std_logic_vector (7 downto 0);
signal B1: std_logic_vector (7 downto 0);
signal M1: std_logic_vector (7 downto 0);
signal MA1: std_logic_vector (7 downto 0);
signal out_flc : std_logic_vector(7 downto 0);

        COMPONENT fuzzificacion1
        PORT(
                E1 : IN std_logic_vector(9 downto 0);
                DE1 : IN std_logic_vector(9 downto 0);
                MB1 : OUT std_logic_vector(7 downto 0);
                B1 : OUT std_logic_vector(7 downto 0);
                M1 : OUT std_logic_vector(7 downto 0);
                A1 : OUT std_logic_vector(7 downto 0);
                MA1 : OUT std_logic_vector(7 downto 0);
                MB2 : OUT std_logic_vector(7 downto 0);
                B2 : OUT std_logic_vector(7 downto 0);
                M2 : OUT std_logic_vector(7 downto 0);
                A2 : OUT std_logic_vector(7 downto 0);
                MA2 : OUT std_logic_vector(7 downto 0)
                );
        END COMPONENT;

                COMPONENT minimos1
        PORT(
                reset : IN std_logic;
                MB1 : IN std_logic_vector(7 downto 0);
```

```
            B1 : IN std_logic_vector(7 downto 0);
            M1 : IN std_logic_vector(7 downto 0);
            A1 : IN std_logic_vector(7 downto 0);
            MA1 : IN std_logic_vector(7 downto 0);
            MB2 : IN std_logic_vector(7 downto 0);
            B2 : IN std_logic_vector(7 downto 0);
            M2 : IN std_logic_vector(7 downto 0);
            A2 : IN std_logic_vector(7 downto 0);
            MA2 : IN std_logic_vector(7 downto 0);

END COMPONENT;

            COMPONENT maximos1
        PORT(
            min_11 : IN std_logic_vector(7 downto 0);
            min_12 : IN std_logic_vector(7 downto 0);
            min_13 : IN std_logic_vector(7 downto 0);
            min_14 : IN std_logic_vector(7 downto 0);
            min_15 : IN std_logic_vector(7 downto 0);
            min_21 : IN std_logic_vector(7 downto 0);
            min_22 : IN std_logic_vector(7 downto 0);
            min_23 : IN std_logic_vector(7 downto 0);
            min_24 : IN std_logic_vector(7 downto 0);
            min_25 : IN std_logic_vector(7 downto 0);

END COMPONENT;
        COMPONENT defuzzificacion1
        PORT(
            max_01 : IN std_logic_vector(7 downto 0);
            max_02 : IN std_logic_vector(7 downto 0);
            max_03 : IN std_logic_vector(7 downto 0);
            max_04 : IN std_logic_vector(7 downto 0);
            max_05 : IN std_logic_vector(7 downto 0);
            out_flc : OUT std_logic_vector(9 downto 0)
            );
        END COMPONENT;
begin
        Inst_fuzzificacion1: fuzzificacion1 PORT MAP(
            E1 => E1,
            DE1 => DE1,
            MB1 => MB1,
            B1 => B1,
            M1 => M1,
```

```
            A1 => A1,
            MA1 => MA1,);
    Inst_minimos1: minimos1 PORT MAP(
            reset => RESET,
            MB1 => MB1,
            B1 => B1,
            M1 => M1,
            A1 => A1,
            MA1 => MA1,
            MB2 => MB2,
            B2 => B2,
            M2 => M2,
            A2 => A2,
            MA2 => MA2, );
    Inst_maximos1: maximos1 PORT MAP(
            min_11 => min_11,
            min_12 => min_12,
            min_13 => min_13,
            min_14 => min_14,
            min_15 => min_15,
            min_21 => min_21,
            min_22 => min_22,
            min_23 => min_23,
            min_24 => min_24,
            min_25 => min_25, );
    Inst_defuzzificacion1: defuzzificacion1 PORT MAP(
            max_01 => max_01,
            max_02 => max_02,
            max_03 => max_03,
            max_04 => max_04,
            max_05 => max_05,
            out_flc => SALIDA_AZIMUT
    );
end Behavioral;
```