Control Neuronal de un Sistema de Equilibrio (PÉNDULO INVERTIDO) en Dispositivos Lógicos Programables

Neuronal control of a balanced system (inverted pendulum) in a logical programmable device

Johnny Omar Medina Durán

Norbey Chinchilla Herrera

Ruby Daniela Vargas Quintero

Yesenia Restrepo Chaustre

Grupo de Investigación en automatización y Control (GIAC) Universidad Francisco de Paula Santander Cúcuta-Colombia. Correos elecrónicos: johnnyomarmd@ufps.edu.co, norbeych@ufps.edu.co, rubydanielavq@ufps.edu.co, yeseniarestrepo@ufps.edu.co

Información del artículo: recibido: enero de 2016, aceptado: abril de 2016
https://doi.org/10.17081/invinno.4.2.2488

Resumen

Este trabajo presenta la implementación de una Red Neuronal Feed-Foward para el control de equilibrio de un sistema sobre dos ruedas (péndulo invertido), en una tarjeta de desarrollo Nexys 2 de Digilent, que contiene una FPGA (Field Programmable Gate Array) XC3S500E. La herramienta utilizada para la creación, entrenamiento y simulación de la red neuronal fue la NNTool de Matlab. El algoritmo neuronal fue traducido a un modelo realizable en hardware, mediante diagramas de bloques, desarrollados con las herramientas Simulink y Xilinx System Generator (XSG). La validación de la red neuronal se realiza en un prototipo de equilibrio sobre dos ruedas. Este sistema tiene una unidad de medida inercial (IMU 6dof- MPU 6050), que incluyen un acelerómetro y un giroscopio de tres ejes cada uno, y 2 motorreductores con encoder magnético, utilizados como actuadores.

Palabras clave:

RNA, Péndulo invertido, Xilinx System Generator, NNTool, MPU - 6050.

Abstract

This paper presents the implementation of a neural network for controlling FeedFoward balance on two wheels system (inverted pendulum) in a Card Nexys 2 Digilent development containing a FPGA (Field Programmable Gate Array) XC3S500E. NNTool tool of Matlab was used for the creation, training and simulation of neural network. The neural algorithm is translated into a workable model in hardware, using block diagrams, using tools Simulink and Xilinx System Generator (XSG). The neural network validation is performed on a prototype of balance on two wheels, this system has an inertial measurement unit (IMU 6DOF-MPU 6050), which includes an accelerometer and three-axis gyroscope each and 2 geared motors with magnetic encoder , used as actuators.

Keywords:

CSR, ethics, responsibility, symbol, representation, signs

Introducción

En la actualidad existe dificultad para definir los sistemas de control inteligentes, debido precisamente a las diversas connotaciones que plantea la palabra inteligencia [1]: "Es de común consenso que para que un sistema actúe como un sistema inteligente debe emular las funciones de las criaturas vivas en cuanto a algunas de sus facultades mentales. Al menos, la inteligencia requiere la habilidad de percibir (agente) y adaptarse al entorno (aprendizaje), tomar decisiones y realizar acciones de control". En este sentido, cabe precisar que las redes neuronales artificiales (RNA) son sistemas compuestos por varios elementos de procesamiento (neuronas), cuya función principal es determinar la estructura de la red y conexiones entre nodos. Las RNA tienen la particularidad del aprendizaje mediante entrenamiento.

Las RNA están inspiradas en modelos biológicos, pero también pueden interpretarse como un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles (capas). En cada nivel existen interconexiones asociados a un peso (pesos sinápticos), que representa la información utilizada por las neuronas para resolver un problema específico [2].

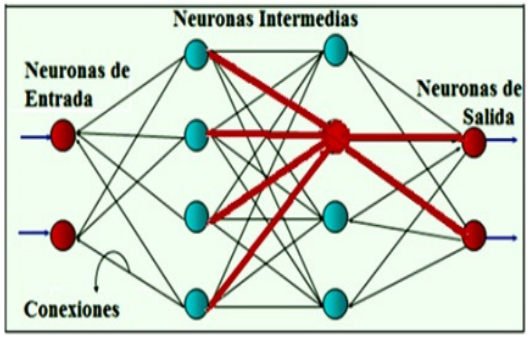

Figura 1. Elementos de una red neuronal artificial

Los elementos básicos de una red neuronal artificial son las neuronas de entrada (capa de entrada), las neuronas de salida (capa de salida), las neuronas intermedias (capas ocultas) y el conjunto de conexiones o pesos sinápticos entre las neuronas.
Las redes Feedfoward solo tienen conexiones hacia adelante y unidireccionales, suelen distinguirse porque los nodos son los elementos básicos de procesamiento y porque la arquitectura de la red esta descrita por las conexiones ponderadas entre los nodos, que pueden ser monocapa o multicapa según la necesidad del diseñador [3]

Las redes monocapa solo tienen una capa de neuronas, correspondiente a la red neuronal más sencilla, ya que proyectan las entradas a una capa de neuronas de salida, donde se realizan los diferentes cálculos. Por su parte, las redes multicapa comprenden una generalización de las redes monocapa, pero también existe un conjunto de capas intermedias entre la entrada y salida (capas ocultas). Este tipo de red puede estar total o parcialmente conectada [4]

Antecedentes

Red neuronal artificial

Alan Turing, en 1936, fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron WARREN McCulloch, un neurofisiólogo, y Walter Pitts, un matemático. En 1943, estos últimos elaboraron una teoría acerca de la forma en que trabajan de las neuronas. Luego, en 1949, Donald Hebb explicó por primera vez los procesos del aprendizaje (elemento básico de la inteligencia humana) desde un punto de vista psicológico. Aun hoy, este es el fundamento de la mayoría de las funciones de aprendizaje que pueden hallarse en una red neuronal.

En la década del 50, muchos investigadores participaron en la búsqueda de nuevas teorías acerca de las redes neuronales. En sus series de ensayos, Karl Lashley encontró que la información no se almacena en el cerebro en forma centralizada, sino que se distribuye encima de él. Mientras que, en 1956, en el Congreso de Dartmouth surgió la llamada Inteligencia Artificial (IA). Bernard Widroff y Marcian Hoff desarrollaron en 1960 el modelo Adaline (ADAptative LINear Elements). Esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas), que se ha utilizado comercialmente durante varias décadas.

En 1977, Stephen Grossbergy presentó su Teoría de Resonancia Adaptada (TRA). Esta teoría es una arquitectura de red que se diferencia de todas las demás previamente inventadas por simular otras habilidades del cerebro: memoria a largo y corto plazo. Y en 1985, John Hopfield propició el renacimiento de las redes neuronales con su libro Computación neuronal de decisiones en problemas de optimización. Por otra parte, en 1986, David Rumelhart y G. Hinton redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation). A partir de este año, el panorama fue alentador con respecto a las investigaciones y el desarrollo de las redes neuronales. En la actualidad, se realizan y publican numerosos trabajos sobre las aplicaciones nuevas que surgen (sobre todo en el área de control).

Dispositivos lógicos programables

Los Programmable Logic Device (PLD's) son circuitos integrados en los que se pueden programar ecuaciones lógicas Booleanas, tanto combinacionales como secuénciales. Están formados por una matriz de puertas AND conectada a otra matriz de puertas OR más biestables. Cualquier circuito lógico se puede implementar, por tanto, como suma de productos [5].

Las FPGA, introducidas por Xilinx en 1985, también se denominan Arreglos de Celdas Lógicas (LCA), y consisten en una matriz bidimensional de bloques configurables que se conectan mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una FPGA son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques. Además, tienen la ventaja de ser reprogramables, lo que aumenta una enorme flexibilidad al flujo de diseño; así, los circuitos se ejecutan más rápido que en otros dispositivos ya que su ejecución es en paralelo, por lo que no necesitan competir por los mismos recursos. Cada tarea de procesos se asigna a una sección específica del dispositivo y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica.

Péndulo Invertido

El péndulo invertido es un sistema mecánico clásico para probar nuevas ideas en la disciplina del control. Es uno de los ejemplos más conocidos de sistemas a estabilizar, por ende, ha sido motivo de muchos estudios. Si bien existen variantes en la aplicación, en esta investigación, basada en la estabilización del péndulo sobre vehículo de dos ruedas independientes, es también llamado vehículo auto balanceado, pues consiste en un móvil que integra una red inteligente de sensores, ensamblajes mecánicos y un sistema de control que le permite desplazarse manteniendo su estabilidad, teniendo dos ruedas como punto de apoyo.

Modelo matemático

Es una de las herramientas más interesantes de que actualmente se dispone para analizar y representar un proceso. Este modelo es de gran ayuda para la toma de decisiones, ya que sus resultados son inteligibles y útiles [6].

Para iniciar el modelamiento matemático, se debe partir del modelado físico del sistema, así como conocer los sensores, los actuadores y sus características. Los parámetros físicos del sistema se muestran en la Tabla 1.

Tabla 1. Fuerzas y parámetros del sistema

SÍMBOLO	DESCRIPCIÓN
N	Fuerza normal de reacción del carro sobre el péndulo, en la unión entre ambos
-N F	uerza normal de reacción del péndulo sobre el carro (ley de acción y reacción)
N_c	Fuerza normal de la superficie sobre el carro
F	Fuerza aplicada para mover el carro
F_f	Fuerza d e fricción e ntre e l carro y la s uperficie d e desplazamiento
G_c	Fuerza de gravedad sobre el carro
G_p	Fuerza de la gravedad sobre el péndulo
θ	Angulo de desviación del péndulo respecto a la posición de equilibrio
w	Velocidad angular del péndulo
ϵ	Aceleración angular del péndulo
M_c	Masa del carro
M_p	Masa del péndulo
a_c	Aceleración lineal del carro
a_p	Aceleración lineal del péndulo
L	Distancia desde la unión entre el carro y el péndulo
g	Aceleración gravitatoria
μ_c	Coeficiente de fricción entre la superficie y el carro
μ_p	Coeficiente de fricción entre la articulación que une al carro y péndulo
I	Momento de inercia del péndulo relativo a la articulación
M	Sumatoria de los pares no inerciales que actúan en el péndulo, respecto a la articulación de unión.

A continuación, se describe el proceso de obtención de las ecuaciones de movimiento del sistema. :

Aplicando la segunda ley de newton al movimiento lineal del carro, se tiene que:

$$\sum F = m * a$$
$$F + F_f + G_c - N + N_c = M_c * a_c$$

(1)

Las fuerzas y la aceleración de la ecuación 1, se descomponen en sus términos correspondientes en eje ´´x´´ y en eje ´´y´´. Teniendo en cuenta que ux y uy son los vectores unitarios.

$$F = F * u_x,\ F_f = -F_f * u_x \pi r^2,\ G_c = M_c * g * u_y$$
$$N = N_x * u_x - N_y * u_y,\ N_c = -N_c * u_y,\ a_c = \ddot{x} * u_x$$

(2)

Para la fuerza de fricción se debe tener en cuenta que la superficie limita el movimiento del carro, hacia arriba y hacia abajo. En estas circunstancias, la fuerza de fricción será:

$$F_f = \mu_c * |N_c| * sgn(\dot{x} * N_c)$$
$$F_f = \mu_c * N_c * sgn(\dot{x} * N_c)$$

(3)

En segundo lugar, aplicando la ley de Newton al movimiento lineal del péndulo, se obtiene la siguiente ecuación

$$N + G_p = M_p * a_p$$

(6)

La fuerza ejercida por la gravedad sobre el péndulo, se define así:

$$G_p = M_p * g * u_y$$

(7)

Para descomponer la aceleración del péndulo, ap, debido a que se sitúa entre el centro de masas del péndulo, debe tomarse en cuenta:

La aceleración del carro al que está unido el péndulo.

$$a_c = \ddot{x} * u_x$$

(8)

Y la rotación del péndulo a una velocidad angular, que se puede describir como:

$$w = \dot{\theta} * u_z$$

(9)

Por lo tanto, la aceleración angular del péndulo es:

$$\varepsilon = \ddot{\theta} * u_z$$

(11)

Donde:

rp= es el vector que presenta la posición del centro de masa del péndulo respecto a la articulación sobre la cual rota el mismo.

$$rp = L(\sin(\theta) * u_x - \cos(\theta) * u_y$$

(12)

Al sustituye las equivalencias anteriores en (11), se obtiene:

$$a_p = \ddot{x} * u_x + \ddot{\theta} * u_z * L * (\sin(\theta) * u_x - \cos(\theta) * u_y) + \dot{\theta}^2 * u_z * L * (u_z(\sin(\theta) * u_x - \cos(\theta) * u_y))$$

(13)

También se obtiene el siguiente resultado:

$$a_p = \ddot{x} * u_x + \ddot{\theta} * L * (\sin(\theta) * u_y + \cos(\theta) * u_x) - \dot{\theta}^2 * L * (\sin(\theta) * u_x - \cos(\theta) * u_y)$$

(14)

Al aplicar la segunda ley de Newton al movimiento rotatorio del péndulo, alrededor de la articulación, se obtiene la siguiente ecuación:

$$M = I * \varepsilon + rp * a_c$$

(15)

Donde:

$$M = rp * G_p - \mu_p * \dot{\theta} * \mu_z$$

(16)

M es el sumatorio de los pares no inerciales, actuando en el péndulo respecto de la articulación que los une.

$$I = \frac{4}{3} * M_p * L^2$$

(17)

I es el momento de inercia del péndulo respecto de la articulación que lo une con el carro.

$$-rp * a_t$$

(18)

El producto anterior se puede interpretar como el par generado por la fuerza inercial causada por la aceleración del carro.

Sustituyendo, entonces, (18), (17), (16) en (15), se obtiene.

$$M_p * g * L\sin(\theta) - \mu_p * \dot{\theta} = \frac{4}{3} * M_p * L^2 * \ddot{\theta} + M_p * \ddot{x} * L\cos(\theta)$$

(19)

Una vez realizado el análisis, se reordenan y agrupan las ecuaciones obtenidas. De este modo, en primer lugar, se sustituirá Nx de (16) en (3).

$$\ddot{x} = \frac{F - F_f + M_p * L(\ddot{\theta} * \cos(\theta) - \dot{\theta}^2 * \sin(\theta))}{M_c + M_p}$$

(20)

Ahora, sustituyendo (20) en (19), se obtiene:

$$\ddot{\theta} = \frac{g * \sin(\theta) + \cos(\theta) * \left[\frac{-F - M_p * L * \dot{\theta}^2 * \sin(\theta) + F_f}{M_c + M_p}\right] - \frac{\mu_p * \dot{\theta}}{M_p * L}}{L * [\frac{4}{3} - \frac{M_p * \cos(\theta)^2}{M_c + M_p}]}$$

(21)

Suponiendo, por último, términos de fricción nulos se tiene:

$$\ddot{\theta} = \frac{g * \sin\theta + \cos\theta * \left[\frac{-F - M_p * L\dot{\theta}^2 * \sin(\theta)}{M_c + M_p}\right]}{L * \left[\frac{4}{3} - \frac{M_p * \cos(\theta)^2}{M_c + M_p}\right]}$$

(22)

$$\ddot{x} = \frac{F + M_p * L * (\ddot{\theta} * \cos(\theta) - \dot{\theta}^2 * \sin(\theta))}{M_c + M_p}$$

(23)

RNA Para el Control de Equilibrio del Péndulo Invertido

Como ya se advirtió, las redes neuronales consisten en una simulación de las propiedades observadas en los sistemas neuronales biológicos, a través de modelos matemáticos recreados mediante mecanismos artificiales [7]

Para verificar el comportamiento de las RNAs implementadas en FPGAs, se tomó la aplicación del péndulo invertido. Aunque este sistema ha sido objeto de estudio en muchas investigaciones, sigue siendo hoy en día el principio de muchos procesos industriales, que comprenden desde la simulación del vuelo de una aeronave hasta los robots bípedos [8].

Matlab cuenta con una caja de herramientas (Neural Network Toolboox) que permite diseñar RNA´s. Cuando se crea una RNA en Matlab, las capas generadas incluyen la combinación de pesos, amén de la operación de multiplicación y suma, así como de operaciones especiales como

mapeo de entrada y salida de la RNA.

Antes de crear la red neuronal, se deben tener en cuenta los siguientes parámetros

Número de capas ocultas

Las capas ocultas dan a la red la habilidad de generalizar. Las redes neuronales con una ó dos capas ocultas son las más utilizadas en la práctica, teniendo un buen desempeño. El incremento en el número de capas también incrementa el tiempo de procesamiento y el peligro de sobreajuste, lo que conduce a un pobre desempeño en la predicción fuera de muestra. El sobreajuste ocurre cuando un modelo de predicción tiene muy pocos grados de libertad. En otras palabras, se tienen relativamente pocas observaciones en relación con sus parámetros y, por lo tanto, puede memorizar datos individuales en lugar de aprender patrones generales. Por esta razón, se recomienda que todas las redes neuronales comiencen de preferencia con una o a lo mucho dos capas [9].

Número de neuronas ocultas

No existen reglas generales o teorías para determinar el número de neuronas en la capa oculta, aunque sí sugieren algunas recomendaciones.

La red debe tener una topología piramidal, esto es, debe tener el mayor número de neuronas en la capa de entrada y menos en las posteriores [10]. Se recomienda que el número de neuronas en cada se halle más o menos entre la mitad de la capa siguiente y el doble de la capa anterior.

Lo recomendable es probar con un número pequeño de neuronas ocultas y solo incrementarlo gradualmente si la red neuronal parece no aprender. De esta forma, se puede reducir el problema del sobreajuste al existir más pesos (parámetros) que muestras de datos, r. Este es el método a emplear para determinar el número de neuronas en la capa oculta de la red, aunque la utilización de redes más grandes ayuda a reducir tanto el error de entrenamiento como el de generalización.

Número de neuronas de salida

Decidir el número de neuronas de salida es algo más sencillo porque hay muchas razones para emplear una sola neurona. Por ejemplo, las redes neuronales con múltiples salidas, especialmente si tales salidas están ampliamente espaciadas, producen resultados inferiores en comparación con una red de una única salida. Lo recomendable al respecto es tener una red especializada para cada una de las salidas deseadas en cada predicción.

Funciones de transferencia

Siempre es recomendable estudiar los histogramas de las variables escaladas de entrada y salida, de manera que se pueda identificar la necesidad de realizar un escalamiento que produzca una distribución uniforme y emplear, así, de manera eficiente, el número de neuronas disponibles.

Acorde con lo anterior, la red neuronal seleccionada consta de dos capas, con cinco neuronas en la capa inicial y una neurona en la capa de salida. La función de activación seleccionada fue la de activación tansig, que genera una tangente hiperbólica al obtener, a la salida de la neurona, valores comprendidos entre 1 y -1, cuando se evalúan desde menos infinito hasta infinito.

La red seleccionada es una Feedfoward,

red muy utilizada en sistemas digitales por su eficiencia y simplicidad. La topología de red neuronal creada en NNTool es la propuesta en la Figura 2.

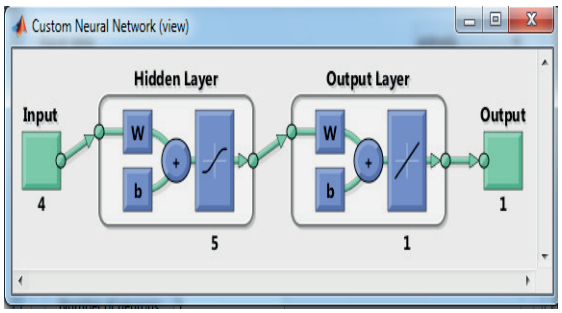

Figura2. Topología de la RNA seleccionada

entrenamiento de la red

Entrenar una red neuronal para aprender patrones involucra presentar ejemplos de las respuestas correctas de manera iterativa. El objetivo del entrenamiento es encontrar un conjunto de pesos entre las neuronas que determinan el mínimo global de la función de error [11]. A menos que el modelo esté sobreajustado, el conjunto de pesos debería proporcionar una buena generalización. Un término de momento y de cinco a diez conjuntos aleatorios de pesos iniciales, pueden mejorar las oportunidades de alcanzar un mínimo global.

El entrenamiento converge de una manera muy rápida, alcanzando un porcentaje de error con tendencia a 0, lo cual indica que todas las entradas fueron correctamente clasificadas.

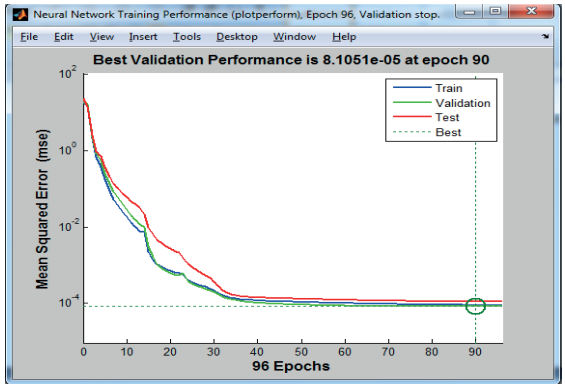

Figura3. Curva característica del error en la RNA

El diagrama de bloques de simulink, generado en NNTool, se puede observar en la Figura 4.

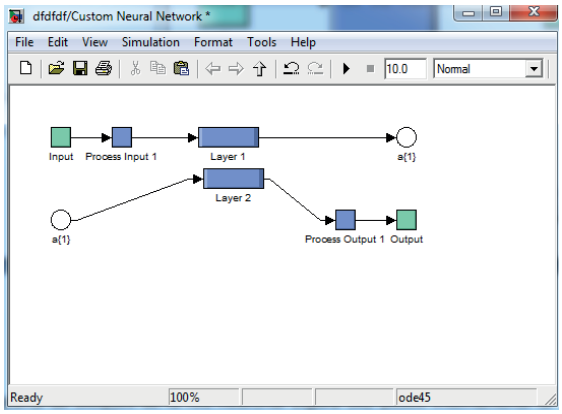

Figura 4. Diagrama de bloques de la RNA exportada en simulink

Desarrollo del Modelo en xsg de la RNA

La estructura de red neuronal mostrada en la Figura 3 se implementa en XSG por medio de bloques sumadores, multiplicadores, bloques de ganancia, constantes y bloques MCode. Estos últimos se utilizan para desarrollar la función de transferencia de activación, como se muestra en las figuras 5 y 6, que representan la implementación de una sola neurona de la capa inicial y la neurona de la capa de salida, respectivamente. Cada una de las entradas es multiplicada por el peso calculado en el entrenamiento y una vez realizada la suma, se adiciona el valor de Bias y se le aplica la función de transferencia para obtener finalmente la salida de la RNA.

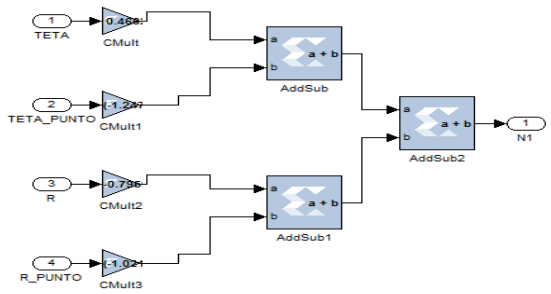

Figura.5. Diagrama de bloques de una neurona (capa inicial)

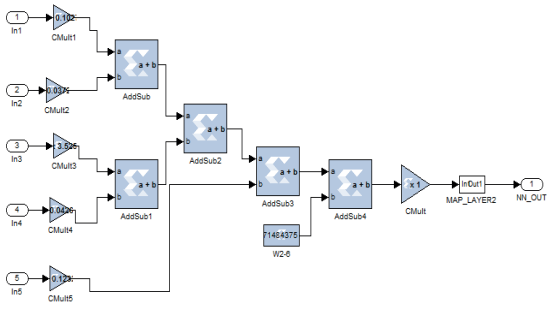

Figura 6. Diagrama de bloques neurona final (capa de salida)

Los parámetros de configuración de estos bloques permiten obtener un funcionamiento óptimo sin problemas de sobre-flujo o saturación, con la consecuencia de un mayor consumo de recursos de la FPGA [12]. Si en un diseño no se tienen limitaciones en recursos, resulta práctico utilizar esta opción, debido a que ahorra tiempo durante la puesta en marcha del diseño cuando se tienen un número grande de estos elementos. Pero si, por el contrario, el ahorro de recursos es fundamental o los recursos disponibles son limitados, se debe seleccionar debidamente el tipo de salida, especificando el numero de bits, la posición del punto binario, el tipo de cuantización y el tipo de sobre flujo, que permitan representar correctamente las cantidades.

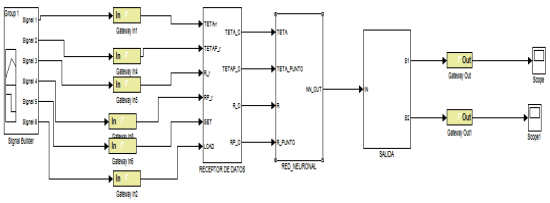

Figura 7. Modelo en XSG de la red neuronal

El diagrama mostrado en la figura anterior representa el bloque completo generado en el software system generator. Se debe tener en cuenta que XSG genera el tipo de archivo necesario para exportar la rutina de control a XiLinx ISE. El tipo de archivo es .bit, y una vez obtenido se puede implementar la rutina de control en la FPGA a utilizar.

Se deben incluir las características propias de la FPGA donde se desea generar la compilación. Asimismo, se debe seleccionar el lenguaje de descripción de la

rutina a compilar, la dirección donde se desea guardar y el tipo de compilación. Para el caso particular de una FPGA, el archivo .bit representa la opción de Bitstream. Una vez configurado, el software se encarga de finalizar la tarea, mediante la síntesis propia.

Conclusiones

La implementación de redes neuronales artificiales en dispositivos lógicos programables permite el desarrollo de sistemas de control en tiempo real para un sistema de equilibrio, asumiendo las no linealidades presentes en el modelo del péndulo, ya que la arquitectura física de los dispositivos presentes en la tarjeta de desarrollo Nexys 2 brinda la posibilidad de implementar sistemas de control inteligente.

No existen métodos establecidos para el diseño de una RNA; por tanto, en este trabajo el criterio utilizado fue la experiencia adquirida y las investigaciones de otros autores, permitiendo por medio de la práctica de ensayo y error la adecuación de la red que mejor se ajustara a la solución del problema.

Se utilizó un sistema de desarrollo Nexys 2, implementándose la rutina de control inteligente. En la recepción y transmisión de información para y desde la RNA establecida, también fueron utilizados los puertos pmod de la misma, adecuándose la recepción de información mediante el protocolo de comunicación I2c, característico del sensor de medida inercial MPU 6050.

En el desarrollo de la rutina de control del sistema de equilibrio, fue necesario utilizar herramientas como: NNTool para la selección de topología de red, Simulink para el desarrollo del sistema mediante diagramas de bloques y, finalmente, XiInx system generator, para transportar la rutina de control inteligente desarrollada y llevarla a lenguaje VHDL característico del sistema de desarrollo Nexys 2. Todas estas pertenecientes al software Matlab 2012.

La implementación de la estrategia de control se realizó por medio de XSG en la Nexys 2. Esta herramienta brinda la posibilidad de programar mediante diagramas de bloques, evitándose así las extensas y complejas líneas de códigos que resultarían del desarrollo de la rutina de control por medio de códigos VHDL, pro-

Referencias Bibliográficas

1. M. Santos. “Un enfoque aplicado del control inteligente”, Revista Iberoamericana de Automática e Informática industrial, vol. 8 p. 283–296, 2011.

2. D. Matich. Redes neuronales: conceptos básicos y aplicaciones. Argentina: Universidad Tecnológica Nacional. Grupo de investigación aplicada a la ingeniería química (GIAIQ). Rosario, 2011.

3. H. Vega. Tesis doctoral Redes neuronales para el reconocimiento de la calidad morfológica de mangos exportables para la empresa Biofruit del Perú S.A.C. Universidad nacional Federico Villarreal. Lima. Perú, 2011.

4. R. Salas. Redes neuronales artificiales. Universidad de Valparaíso. Departamento de computación, 2010.

5. J. Medina, Ferreira J., y O. Gualdron,"Redes neuronales recurrentes en dispositivos lógicos programables para el control de desplazamiento de un robot móvil", Revista Colombiana de Tecnologías de Avanzada, vol. 1, 2009.

6. A. Ramos, P. Sanchez, J. Ferrer, Barquin J., y P. Linares. Modelos matemáticos de optimización. Universidad pontificia de Madrid, Escuela técnica superior de ingeniería, 2010.

7. A. Nacelle, Las redes neuronales: de la biología a los algoritmos de clasificación. Universidad de la república. Montevideo. Uruguay: Núcleo de ingeniería biomédica, 2009.

8. National Instruments. Introducción a la tecnología fpga: los cinco beneficios principales, 2004. Disponible: http://zone.ni.com/derzone/cda/tut/p/id/8289.

9. M. Valencia, L. Sanchez., y Y. Cornelio. Algoritmo Backpropagation para Redes Neuronales: conceptos y aplicaciones. México DF: Instituto politécnico nacional centro de investigación en computación, 2006.

10.A. Hernández, M. Legaspi., y J. Pelaez. Control inteligente del péndulo invertido. Madrid: Universidad Complutense de Madrid, 2012.

11. D. Melo y R. Molina. Desarrollo de un sistema de control de equilibrio para un móvil con dos ruedas de apoyo ubicadas lateralmente "Robox 1.0". Universidad Pedagógica Nacional- Bogotá D.C., 2011.

12. J. Medina. Metodología para la implementación de redes neuronales en dispositivos lógicos programables aplicadas en el control de desplazamiento de un robot móvil. Universidad de Pamplona. Departamento de maestría y doctorados, 2009.

Este artículo se cita:

Citación estilo

J. Medina, N. Chinchilla, R. Vargas, y Y. Restrepo, "Control neuronal de un sistema de equilibrio (péndulo invertido) en dispositivos lógicos programables", Investigación e Innovación en Ingenierias, vol. 4, n°. 2, pp. Xx-xx, 2016.