The Revised Digital Neuron:
Modelling, Implementation and Validation

Neurona Digital Optimizada:
Modelamiento , Implementación y Validadación

Alvaro Varela [1], Johann Osma[2], Fredy Segura[3], Antonio García[4]
[5]Centro de Microelectrónica de la Universidad de los Andes.
{[1]al-varel, [2]j-osma, [3]angarcia, [4]fsegura, [5]cmua}@uniandes.edu.co

Abstract

This project presents a hardware-based software modelling of the mathematical model of a artificial neuron or perceptron (also called the neuronal model) programmed in VHDL, to be used in a complete system of neural networks.

In this paper we will present the modelling, implementation and validation of the latest revision on the working prototype, which uses a fixed decimal point arquitecture for multiplication and addition.

This validation will take the form of an application in neural networks by recognizing characters typed by a seven by seven (7 X 7) keyboard.

Resumen

El proyecto presenta el modelamiento software, basado en hardware, de un modelo matemático de una neurona artificial o perceptrón en VHDL, para ser usado en un sistema de redes neuronales.

En el momento se presentará el modelamiento, implementación y validación hecha a los resultados de la última optimización de un prototipo, que maneja una arquitectura de punto fijo para la multiplicación y acumulación.

Esta validación se hará efectiva mediante una aplicación en reconocimiento de caracteres en una matriz de pulsadores de siete por siete (7 X 7).

Neurona Digital Optimizada:
Modelamiento , Implementación y Validadación.

Alvaro Varela [1], Johann Osma[2], Fredy Segura[3], Antonio García[4]
[5]Centro de Microelectrónica de la Universidad de los Andes.
{[1]al-varel, [2]j-osma, [3]angarcia, [4]fsegura, [5]cmua}@uniandes.edu.co

Resumen

El proyecto presenta el modelamiento y la implementación hardware del modelo matemático de una neurona artificial o perceptrón en VHDL, para ser usado en un sistema de redes neuronales.

En el momento se presentarán los resultados de la optimización sobre un prototipo desarrollado anteriormente por el equipo de Johann Osma y Alvaro Varela, que maneja una arquitectura de punto decimal fijo para la multiplicación y acumulación.

Se explica una aplicación validatoria del modelamiento en VHDL de la neurona y se exponen los resultados de ésta, comprobando el funcionamiento de la misma.

Palabras claves:
Redes neuronales, VHDL.

1. Introducción

Para generar industria tecnológica en una economía en crisis, se deben crear productos aplicados, que tengan la facilidad de una utilización global sin tener los costos de un producto completamente especializado, esto se logra creando prototipos genéricos que se puedan utilizar en varias aplicaciones. Este es el caso de las Redes Neuronales, que mediante un algoritmo matemático "simula" el criterio humano para tomar decisiones.

La toma de decisiones es parte de la vida humana, pero en esta revolución de la velocidad en los procesos hay ciertas decisiones que simplemente la capacidad humana no puede satisfacer las necesidades de un sistema en crecimiento, por tanto se han creado nociones de hacer que las máquinas hagan ciertas decisiones difíciles por nosotros, con un criterio aproximado o igual pero más veloz que el nuestro, este concepto es mejor conocido como Redes Neuronales, o en casos más complicados Inteligencia Artificial.

Estos funcionan mediante un algoritmo matemático que reconoce y clasifica patrones de datos, y por tanto según el patrón recibido hace una determinada acción, en otras palabras, al recibir los datos, según un criterio determinado toma una decisión, igual que los humanos.

Esto se ha logrado con el poder computacional de los programas (software), pero significa que para hacer un producto que necesite o haga decisiones propias se debe hacer un sistema que interprete un programa determinado, que resulta tedioso y más importante costoso para el productor. Por tanto hacer un circuito que se pueda programar para que haga una red con un criterio que el productor o el consumidor necesite y si quiere lo pueda cambiar, ocupando en un espacio determinado solamente por las necesidades físicas, de la tecnología utilizada para crear el circuito.

El objetivo principal del proyecto no es la creación de inteligencia artificial sino la implementación redes neuronales aplicadas, máquinas que hacen decisiones específicas dentro de una actividad o aplicación en donde el criterio humano no sea viable o factible **[4]**.

El presente estado del proyecto es un algoritmo en pruebas físicas mediante una implementación de una aplicación simple, en un arreglo de circuitos manejados por el sistema diseñado de la red neuronal. Este algoritmo maneja directamente números con punto fijo decimal, lo que lo hace en ocasiones inexacto, es un prototipo de lo que se quiere lograr al final del proyecto.

2. De Neurona a Neurona.

2.1. Especificación inicial

El concepto del manejo de redes neuronales mediante medios digitales es un campo explorado por varias compañías, que se especializan en crear sistemas con un alto manejo de software. El proyecto a largo plazo consiste en la creación de un sistema completo sin la necesidad de software mediante una implementación Hardware utilizando el lenguaje VHDL. En este momento el proyecto está en su etapa de prototipo, la cual se ha dividido en dos instancias para facilitar la creación y corrección y optimización de ella.

Los objetivos que se buscaron para la primera instancia fueron:

- Diseño Completamente Sintetizable.
- Diseño Portátil, que toda clase de FPGA pueda utilizarlo.
- Compacta.

Estos objetivos fueron cumplidos, mediante las siguientes estrategias:

- Un corte sistemático de elementos demasiado grandes que fueran reemplazables por elementos simples tales como memorias, como la sigmoide, que es una LUT (implementada fácilmente mediante una ROM[1]).
- La "no utilización" de librerías exclusivas de fabricantes de FPGAs.
- Una implementación utilizando lógica secuencial.
- 2 modalidades de funcionamiento: Carga de Datos/Pesos y cálculo y comparación de la neurona.

2.2. Funcionamiento Básico de la Red Neuronal Digital.

2.2.1. Funcionamiento de una Neurona.

Básicamente el funcionamiento de una neurona es describible en tres simples palabras: Adquisición, Criterio y por último Decisión. Adquisición se refiere a la adquisición de datos, Criterio se refiere a la importancia que se le da a cada uno de los datos adquiridos, esto genera un dato determinado para poder decidir, y Decisión dado el dato anterior lo comparamos con una tabla de valores y decidimos según eso.

El algoritmo matemático es muy similar:

Adquisición: Se tienen varios números que me determinan una situación.

Criterio: Se tienen preestablecido una importancia (también en forma de un número) diferente a cada uno de los datos anteriores, por tanto multiplico, y obtengo un factor que es una sumatoria de todos los datos y sus importancias.

Decisión: Este factor se compara con lo que se ha hecho con factores similares anteriores (miro el factor en una LUT), y decido.

[1] A diferencia del diseño en **[5]**.

Con una neurona podemos reconocer dos patrones diferentes, con una red neuronal que es la interconexión de varias neuronas podemos reconocer muchos más patrones más complicados de detectar.

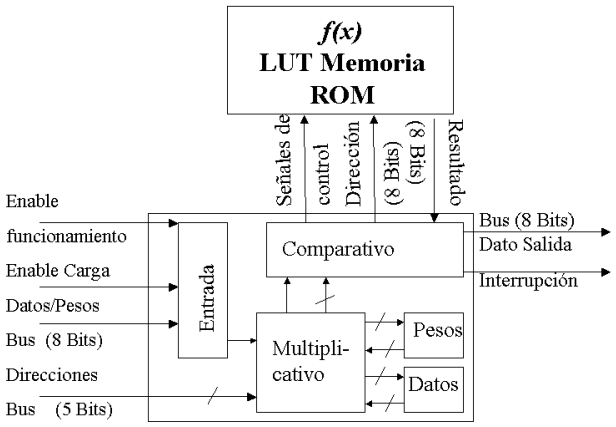
Por tanto para hacer una neurona en VHDL debemos seguir la misma metodología explicada anteriormente, que se simplifica a un módulo de adquisición de datos, un multiplicador / acumulativo y una ROM manejada con una maquina de control[2].

2.3. Funcionamiento de la primera instancia.

Como se describió anteriormente el proyecto original de la neurona tenía 2 modalidades de funcionamiento:

- Almacenamiento o carga de datos a la memoria temporal dentro del FPGA **(2)**: Mediante un bus de direcciones para la memoria interna a la neurona se le alimentan los datos y los pesos para su uso en la segunda modalidad.

Multiplicación y acumulación de los datos en memoria temporal **(3)**: Ya con los datos dentro del circuito, se multiplica el dato con su correspondiente peso, mediante un método secuencial de suma acumulada multiplexada llevando esta suma de multiplicaciones a una gran suma secuencial, que puede manejar directamente el resultado final de la acumulación. Y por último aproximarlo hasta tener otra vez unos datos de salida de la neurona de 8 bits, para interconectar con las otras neuronas en la red.

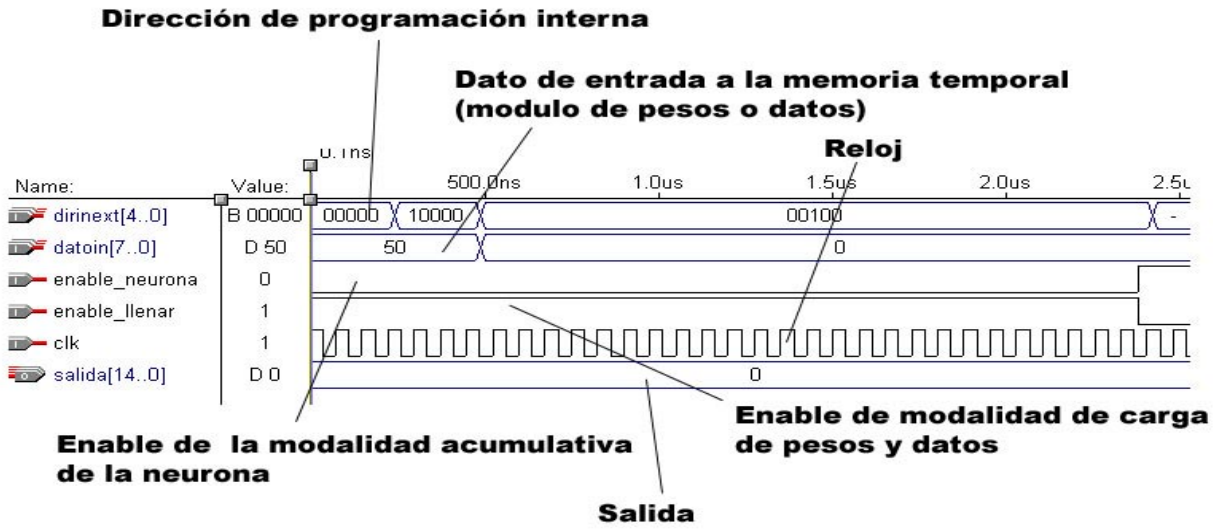


(1) Diagramas de bloques mostrando la arquitectura interna modular de la primera instancia.

[2] A diferencia del diseño en **[5]**.

El funcionamiento de la primera instancia se puede ver mejor con un ejemplo hecho en el software de Altera MAX-PLUS II, en donde veremos cómo interpretar las simulaciones en la neurona.

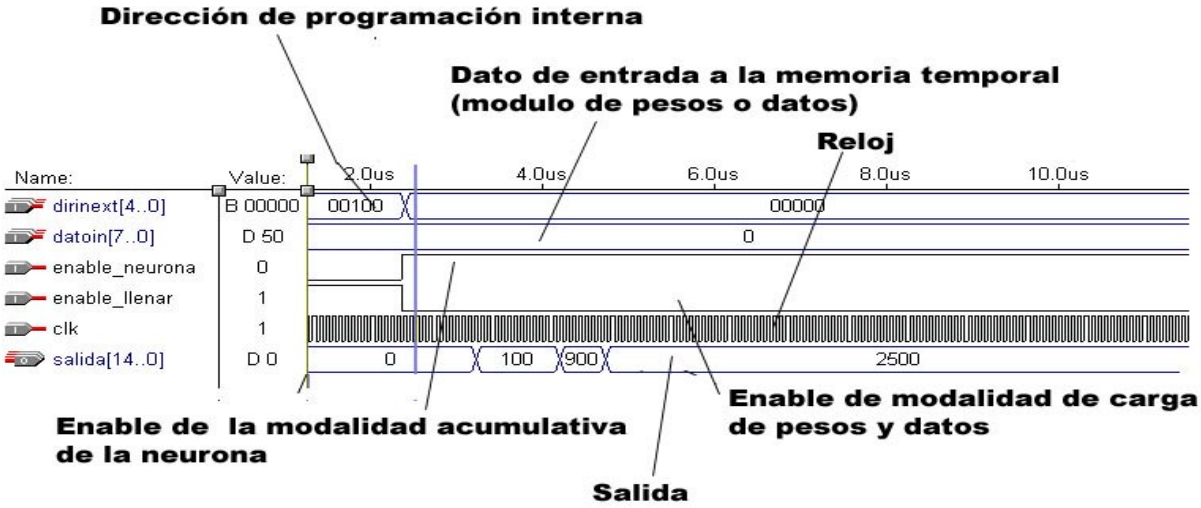
Ciclo de Carga en la primera instancia diseñada



(2) Ciclo de Carga de pesos y datos en la memoria interna de la FPGA

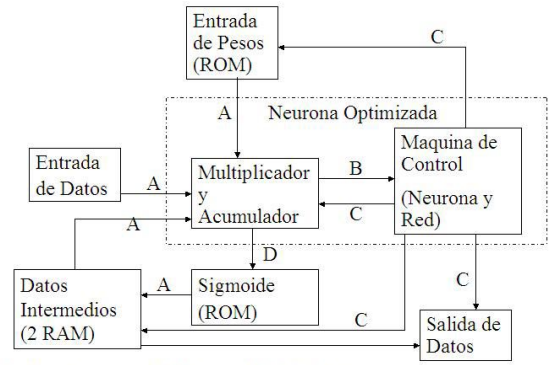
(3) Ciclo de Multiplicación y acumulación

Ciclo de acumulación en la primera instancia diseñada.



En esta primera simulación vemos la carga de los valores con una guía de interpretación de las simulaciones y en la segunda es la multiplicación y resultado del algoritmo:

3. La "Nueva" Neurona



(4) Arquitectura de la Nueva Neurona.

Con los resultados anteriores se plantearon nuevas especificaciones basándose en los defectos de la propuesta inicial.

Los defectos encontrados después de varias pruebas en la primera instancia fueron:

- Error para la acumulación de valores negativos, lo cual es fatal en la implementación de cualquier aplicación en redes neuronales.

- La lógica secuencial genera una necesidad de tener varias maquinas de control para la sincronización entera de los módulos, generado un déficit en recursos.

- Utilizar una parte específica de la neurona para únicamente guardar datos y pesos, implica que hay un límite de procesamiento de la neurona dado exclusivamente por el número de datos y pesos que es capaz de guardar (en este caso 12 pesos y 12 datos en un formato de 8bits, el MSB es el signo el resto es el número.).

Teniendo esto en cuenta, podemos mirar las posibles mejoras:

- Crear un algoritmo de multiplicación y acumulación basado en un sumador que diferencie magnitudes para lograr la acumulación de valores negativos.

- El algoritmo de multiplicación y acumulación, en vez de ser secuencial debe ser una suma multiplexada, manejada por una sola maquina de estado, para sincronización, salvaguardando recursos.

- Crear una maquina de estados que aparte de manejar las operaciones de la neurona, hiciera la red neuronal completa.

- Los parámetros operativos y el formato de los datos se mantienen. Dos modalidades de funcionamiento y el dato de 8 bits (en donde el MSB es el signo, y el resto es el número).

3.2. Modelamiento de la segunda instancia.

Durante los últimos meses se ha trabajado en los puntos que fueron mencionados anteriormente, y satisfactoriamente podemos decir que se tienen resultados concretos que muestran el funcionamiento en simulación de todas las mejoras propuestas.

Estas mejoras se han hecho mediante la creación de una nueva arquitectura de la neurona y por tanto una nueva arquitectura de la red.

3.2.1. Módulo Multiplicativo.

Funciona a través de una suma multiplexada de la siguiente forma:

```
  18              0110    6
 *10             *1010  *10
  00              0000    0
+180            +  01100  +60
 180            + 000000
                +0110000
                 0111100 = 60
```

Como podemos darnos cuenta cada vez que el multiplicando es '1' el vector corrido del multiplicador es sumado a lo que se tiene acumulado. Esto es lo que estoy llamando suma multiplexada.

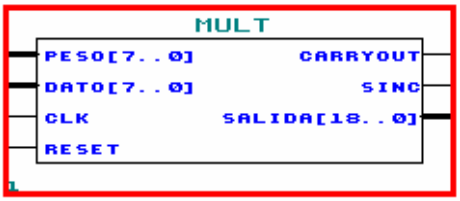
(5) Ejemplo de una suma multiplexada.

La suma en este caso se hace con un sumador basado en la arquitectura Carry look-ahead, para minimizar en lo posible el tiempo de cálculo y funciona de la siguiente

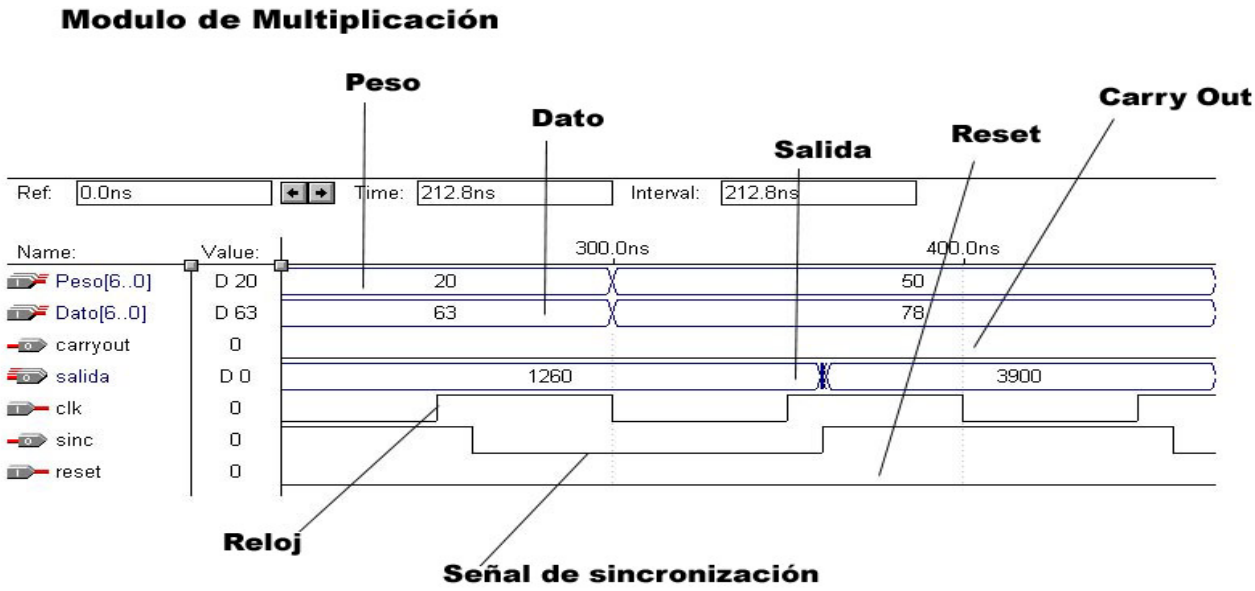
manera: Si el enable (bit del multiplicando) es '1' este suma al acumulado temporal el vector corrido del multiplicador, si es cero, no lo suma sino deja el acumulado como estaba.

Esta mejora tiene por consecuencia un menor uso de maquinas de estado para el manejo de la multiplicación y dado que las sumas son paralelas, sólo se sincroniza un tiempo máximo para obtener el resultado correcto.

3.5.1.1. Simulaciones del módulo.



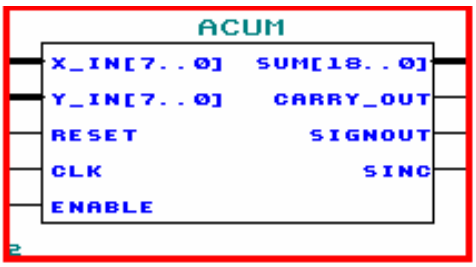
(6) Símbolo Gráfico del Módulo multiplicativo.



(7) Simulación del Módulo multiplicativo

3.5.2. Módulo Acumulador.

Un punto importante era eliminar el defecto de la resta acumulada en el algoritmo anterior, ésto se logró haciendo un sumador/restador que tomaba el acumulado completo (diferente al acumulado anteriormente mencionado es el acumulado parcial típico de cada multiplicación) y el nuevo factor multiplicado, compara sus magnitudes y hace la resta y suma de tal manera, que utilice el mismo algoritmo de la arquitectura de carry look-ahead, sin necesidad de crear casos extras, por lo tanto conservando recursos.

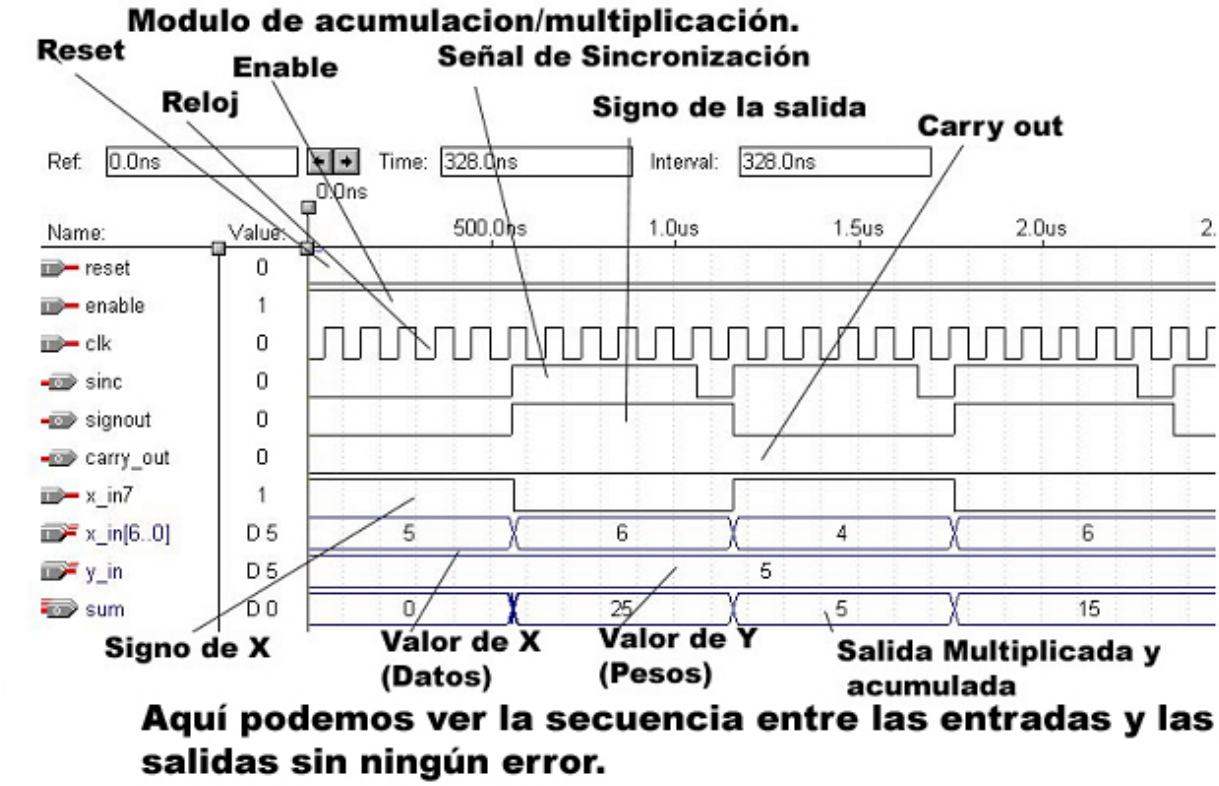


(8) Símbolo gráfico del módulo de acumulación.

3.5.2.1. Simulaciones del módulo (Ver figura 9).

3.5.3. Módulo de Datos y Pesos.

En la segunda instancia para preservar recursos utilizados en estos módulos que únicamente hacen la función de una memoria se decidió eliminarlos y guardar los datos y pesos por fuera de la neurona, en periféricos. Aumentando así la capacidad de la red neuronal a implementar sin tener la barrera de un limite de almacenamiento dado directamente por la FPGA.



(9) Simulación del Módulo de Acumulación.

3.5.4. Maquina de Control

Con lo anterior comprobado, necesitamos una maquina de control que no sólo maneje la operación de la neurona sino que maneje la red neuronal completa. Para ello se hizo un algoritmo que satisfaciera las necesidades (en términos de señales de control) de los módulos ya implementados y de los periféricos a utilizar. Así dirigiendo las interconexiones necesarias de los demás módulos por sus señales respectivas de control y sincronización. En otras palabras, hará el manejo del flujo de datos que permita mediante una sola neurona, hacer toda una red neuronal, mediante las señales de control.

El algoritmo de control en si, consiste en la comparación y conteo constante de los tres datos más importantes en una red neuronal:

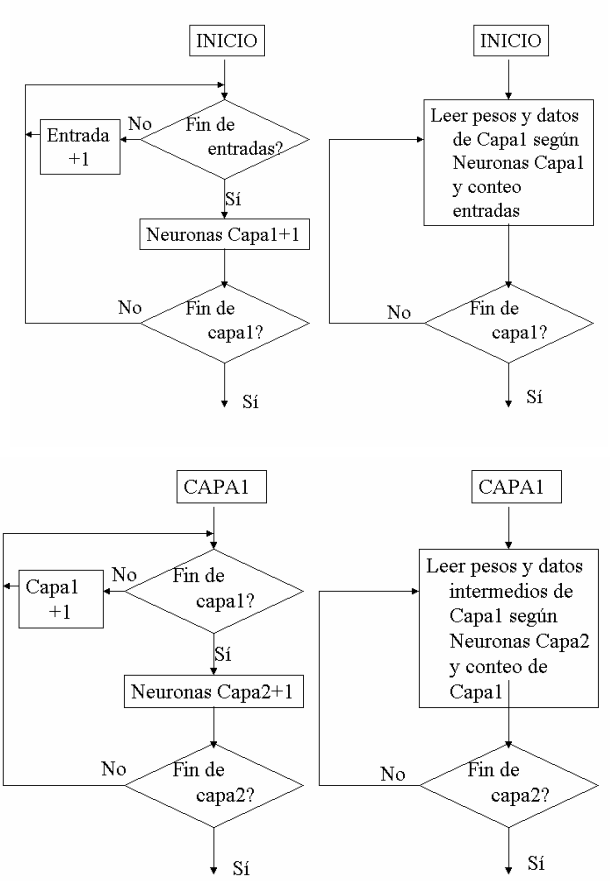
- Número de Entradas.
- Número de Neuronas Intermedias.
- Número de Neuronas de Salida.

(Ver figura 10 y 11).

Podemos establecer, que mediante el algoritmo diseñado con una neurona puede hacer toda una red neuronal.

Al hacer esto se le dio la capacidad de versatilidad de la red, pues la máquina de control al ser manejada por números de entradas, números de neuronas intermedias y números de neuronas de salida, es completamente programable, mediante un formato de 8 bits, lo que implica un máximo de neuronas por capa de 254, de tal forma que podemos tener una red neuronal diferente sin necesidad de hacer una nueva implementación, únicamente se necesita programar los números de entradas, neuronas intermedias y neuronas de salida, dando así un poder de universalidad en aplicaciones, en otras palabras, se puede utilizar la misma implementación para varias aplicaciones sin necesidad de cambiar el circuito para hacerlo.

El funcionamiento verdadero de este algoritmo es demasiado complicado para demostrar en una simple simulación, por lo tanto fue probado directamente con la implementación de una red neuronal completa aplicada como un reconocedor de caracteres.



(10 y 11) Algoritmo de Flujo de Datos y de control de la red neuronal digital.

3.5.5. Red Completa. (13)

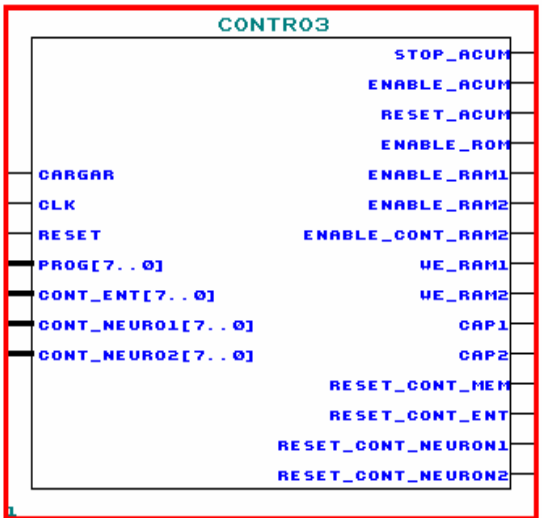
Con esta maquina de control completa tenemos una red neuronal completa y podemos hacer la implementación en circuito, utilizando la Altera UP1 Board con el FPGA EPF10K20RC240-4.

3.6. Reconocimiento de caracteres

La aplicación validatoria del prototipo existente esta basado en los resultados de un proyecto trabajado en conjunto en software para el estudio de las redes neuronales de ésta misma índole.

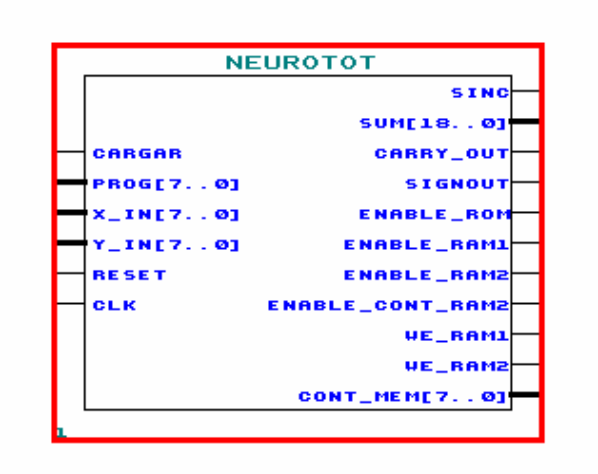
Los pesos como la mayoría de redes neuronales fueron encontrados mediante algoritmos de aprendizaje hechos en C++. La forma de entrada fue mediante una matriz de botones programado en JAVA **(Ver Figura 14)**.

Teniendo esto se creo una implementación similar, mediante una matriz de botones reales de siete por siete (7 X 7) y la implementación se hizo utilizando la Altera UP1 Board con el FPGA EPF10K20RC240-4. **(Ver Figura 14).**

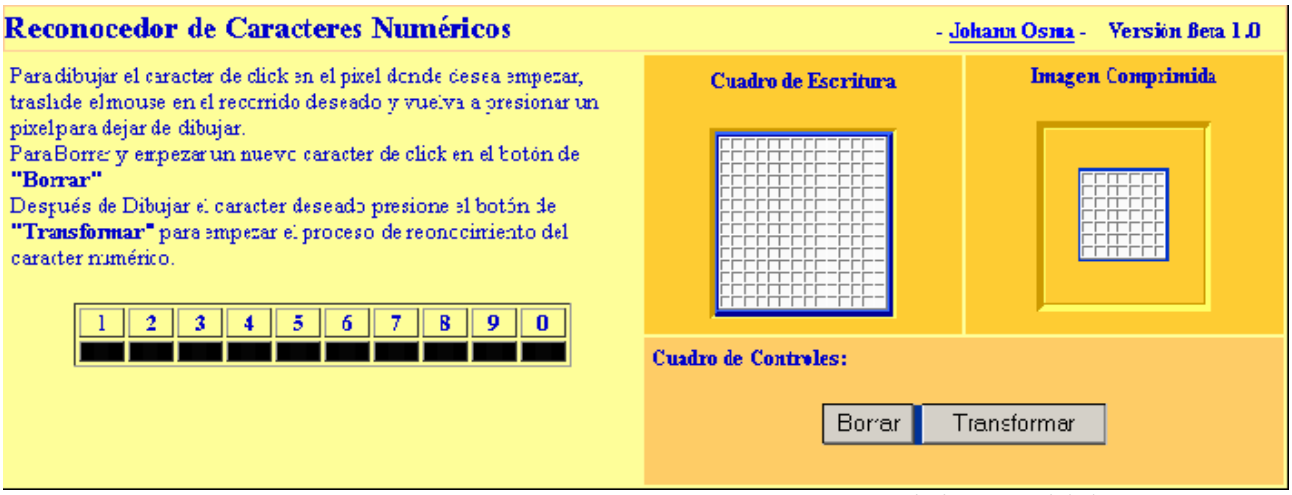


(12) Símbolo gráfico del módulo de control.

En esta implementación se tuvieron que re-entrenar los pesos para que tuvieran en cuenta las aproximaciones que conllevan el diseño y el manejo del punto fijo, que no es el mismo que en la aplicación final.



(13) Símbolo gráfico de la red neuronal completa.



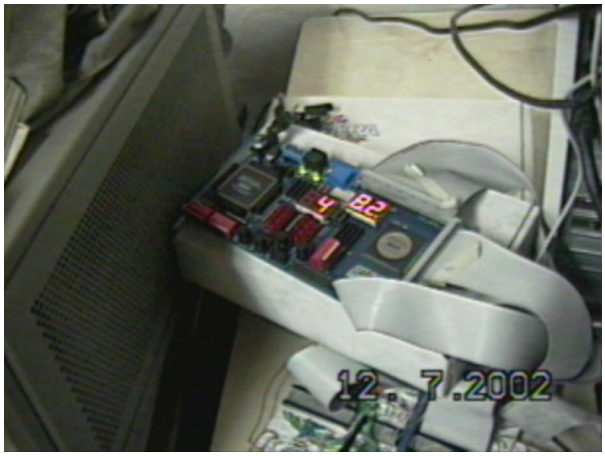
(14) Reconocedor de Caracteres Numéricos hecho en JAVA[1].

(15) Foto del teclado hecho para asemejar el teclado en la aplicación JAVA.

3.7 Resultados del Diseño

Los resultados del diseño son muy variados, dado que hay muchas conclusiones por cada uno de los pasos hechos y descritos dentro del documento. Dividiremos los resultados según el objetivo que cumplió.

El objetivo principal fue la implementación de una red neuronal digital esto fue logrado. La comprobación de ello fue la aplicación de reconocimiento de caracteres que tuvo un reconocimiento parcial de caracteres como el 2, 5, 0 y 8 con la salida de máximos de un porcentaje de 40% y en la salida de rango se encontró el carácter a reconocer entre otros en un 100% de las pruebas[3]. Aunque los resultados no son favorables en el sentido de un reconocimiento veraz de caracteres, debemos tomar en cuenta que la implementación es una aproximación de un algoritmo en software que utiliza punto flotante que dá resultados más finos, encontramos que el hecho que si llega a reconocer un carácter en un factor repetitivo la generación de la red neuronal es correcta por lo tanto valida el diseño hecho como correcto, cumpliendo el papel al cual se implementó la aplicación.



(16) Foto de la interconexión entre los periféricos y la Fpga.

[1] Autor: Johann Osma. j-osma@uniandes.edu.co

[3] Numero de pruebas 100, con caracteres diferentes de 0 a 9.

3.8. Resultados según los objetivos principales.

3.8.1 Portabilidad (Universalidad). Este es el objetivo más simple y complicado de demostrar, dado que la universalidad se dá con las simples reglas mencionadas en las bases del diseño, pero la comprobación de universalidad únicamente se da mediante la prueba constante de compilación del diseño hecho en diferentes programas de fabricantes de Fpgas.

Entre los programas de diferentes fabricantes en los que se trabajaron están:

3.8.1.1 Altera Corporation. Altera ofrece una opción muy buena entre los fabricantes con MAX-PLUS II con una interfase gráfica amigable que demuestra con facilidad los errores cometidos en lenguaje, pero en el momento que el error es muy grande o complejo no da ningún indicio de posible corrección. Esto es contrarrestado con la facilidad de un programador integrado UP1, con el cual se puede programar directamente el Fpga y es disponible en la Universidad de Los Andes. Las simulaciones y la implementación se hicieron en este software dado a su fácil manejo y programación de señales, proyección de resultados y capacidad de Fpga disponible en la universidad.

3.8.1.2 Xilinx. La compañía Xilinx ofrece un software versátil y complejo llamado Foundation. Este da información detallada de todas las señales a utilizar y emplea un sistema de chequeo paso a paso antes de la generación de un programa para el Fpga, aunque la simulación es muy complicada de manejar y no maneja retroalimentación de señales (un requisito necesarios para el diseño hecho). Los trabajos en este fueron pocos, se hizo pruebas de compilación pero el trabajo no continuó debido a lo anterior y el hecho que el fpga de la Universidad de los Andes no tenía la capacidad suficiente para el diseño hecho según el mismo software.

3.8.1.3 Cypress. Cypress ofrece Warp, una herramienta poderosa con la que se logró la programación de varios de los módulos dentro de los Fpga de la misma marca disponibles en la universidad, más la dificultad de capacidad no fue superada, este software tiene capacidades de programar también PLD* en VHDL, una de las debilidades es su forma de programar las salidas y entradas del circuito.

3.8.1.4 Cadence. Cadence es tal vez la plataforma de trabajo más poderosa en circuitos integrados hasta el momento, es un parte integral del proyecto mayor, pues esta herramienta será la encargada de traducir el VHDL hecho a términos de un circuito impreso. Para lograr una transición fácil se hicieron pruebas de compilación más no de simulación.

3.8.2 Diseño Eficiente. Este es un logro constante que se hecho mediante un compromisos de espacio físico y parámetros operativos, como se ha podido ver a lo largo de este documento. Una fiel prueba de ello es la utilización de la implementación de las instancias en donde en el espacio de la primera instancia, que es una neurona simple, se hizo toda una toda red neuronal en la segunda instancia.

3.8.3 Independencia Parcial de Software. Para este diseño se utilizo mucho software tanto en el aprendizaje como en las pruebas de compilación pero al momento de la aplicación física sólo se necesito la simple programación del Fpga y un circuito que emitiera los datos de entrada y los recibiera los datos de salida, para un funcionamiento completo del prototipo. La independencia parcial fue lograda, con éxito al mantener el sistema de redes neuronales operando sin necesidad de tener un computador conectado al circuito.

4. Proyecciones

Dado que el proyecto en esta etapa forma parte de un proyecto mayor, que necesita esta clase de planeamiento y comprobación de resultados se tienen proyecciones para darle un sentido de continuidad y de posible mejora en el futuro.

Dado que los resultados demuestran un correcto funcionamiento de la red hecha se pueden recomendar dos mejoras disponibles a un alcance de un año.

- Cambio de arquitectura de punto fijo, a punto decimal flotante, dado un estándar IEEE.
- Implementación del algoritmo de aprendizaje para la neurona, existente.

* PLD: “Programmable Logic Device”. Circuitos de lógica programmable.

La prueba de estas dos posibles mejoras será o debería ser la implementación de alguna aplicación, al igual que se hizo hasta en este punto, para asegurar la validez de las mejoras hechas.

5. Conclusiones

Por el momento se ha logrado implementar con éxito la red neuronal completa con una aplicación validatoria, con la cual se tienen resultados buenos, en el hecho que aparte de reconocer caracteres hasta cierto grado, cumple con los algoritmos propuestos, de operación y control de la red.

Con esto se concluye que el siguiente paso a dar no sólo es posible sino comparable con los resultados del proyecto presente.

Una conclusión interesante que se llego del proyecto era que la libertad que nos otorga un lenguaje como VHDL, que nos deja hacer aplicaciones utilizando un lenguaje de alta descripción, es contrarrestado al estar atados por la configuración de programación que tenga el fabricante para su chip, en otras palabras, ganamos en libertad de diseño, pero perdemos el control propio de los recursos del chip.

En este momento haciendo un simple trabajo en software para entrenamiento de datos de constantes de punto fijo, podemos hacer otras aplicaciones con mayor grado de funcionamiento que utilicen la arquitectura presente.

Con estas proyecciones y logros podemos tener una visión más clara del trabajo a realizar en el proyecto mayor, cuyo objetivo es obtener un sistema con una neurona que se puede utilizar como individual, o como red neuronal secuencial programable, con un nuevo módulo de manejo de memoria que habilite el aprendizaje.
Esto es aporte muy significativo al estudio de redes neuronales pues se le dará un acceso directo a los estudiantes para trabajar directamente en hardware de redes neuronales sin que tenga una función de polarización (sigmoide) determinada, y ellos puedan trabajar en la función, descubriendo diferentes comportamientos de las redes neuronales y encontrando posibles aplicaciones, en todos los campos: comunicaciones, control de maquinas, lógica difusa, economía, entre muchos otros.

4.1. Autor

Alvaro Varela es un estudiante de Ingeniería Electrónica de X semestre en la Universidad de Los Andes. Ha trabajado en el tema de redes neuronales durante el último año de su carrera. Ha asistido a varios eventos de la comunidad internacional de ingenieros, como el Asiem (2001) en Colombia, como asistente, y al IberCHIP 2002 como participante.

Actualmente trabaja en su tesis (Neurona Optimizada). Además de ser un asistente de investigación en el Centro de Microelectrónica de la Universidad de Los Andes.

5. Referencias.

[1] Ranjeet Ranade, Sanjay Bhandari, A. N. Chandorkar, *VLSI Implementation of Artificial Neural Network Based Digital Multiplier and Adder*.1063-9667/95, 1995 IEEE.

[2] Rafael Gadea, Joaquín Cerdá, Francisco Ballester, Antonio Mocholí., *Artificial Neural Network Implementation on a Single FPGA of Pipelined On-line Backpropagation.*1080-1820/00, 2000 IEEE.

[3] Mario Porrmann, Ulf Witkowski, Heiko Kalte, Ulrich Rückert, *Implementation of Artificial Neural Networks on a Reconfigurable Hardware Accelerator.* 1066-6192/02, 2002 IEEE.

[4] Kuno Köllmann, Karl Ragmar Riemschneider, Hans Christoph Zeidler, *On-Chip Backpropagation Trainins Using Parallel Stochastic Bit Streams.* 1086-1947/96, 1996 IEEE.

[5] B.Girau, A. Tisserand, *On-line Arithmetic-Based Reprogrammable Hardware Implementation of Multilayer Perceptron Back-Propagation.* 1086-1947/96, 1996 IEEE.

[6] Restrepo, Hoffmann Perez-Uribe, Teuscher, Sánchez, *A Networked FPGA-Based Hardware Implementation of a Neural Network Application.* 0-7695-0871-5/00, 2000 IEEE.

5.1. Referencias Virtuales:

[1] Michael John Sebastian Smith, *ASICs... the website: This website is about ASICs or Application-Specific Integrated Circuits, which are a type of silicon integrated circuit.* http://www-ee.eng.hawaii.edu/~msmith/ASICs/HTML/ASICs.htm

[2] Ian Elliott, *Advances Electronic Design Automation, Examples of VHDL Descriptions.*

http://www.ami.bolton.ac.uk/courseware/adveda/vhdl/vhdlexmp.html

[3] Altera Corporation Home Page:
http://www.altera.com