# Diseño e Implementación de una Red Neuronal en un FPGA para Recuperación de Patrones

Ing. MSc. Aurelio Morales Villanueva

Facultad de Ingeniería Eléctrica y Electrónica
Universidad Nacional de Ingeniería

http://www.uni.edu.pe

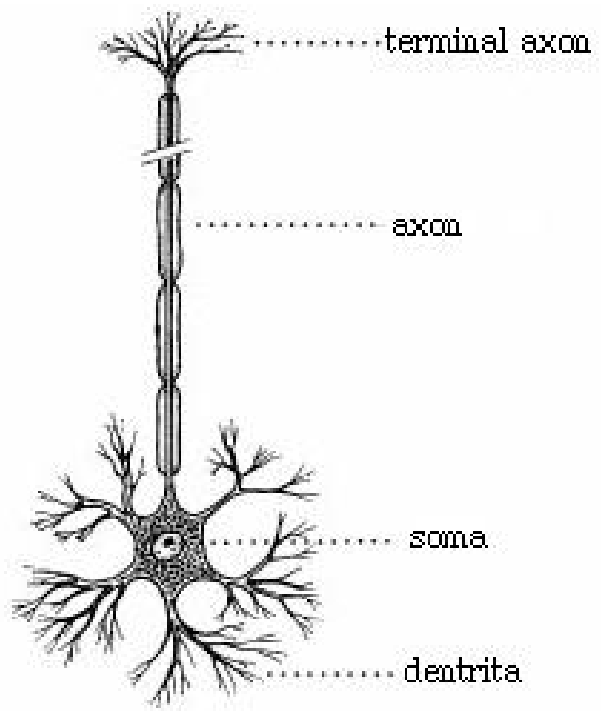
# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Red Neuronal Biológica




Neurona Biológica

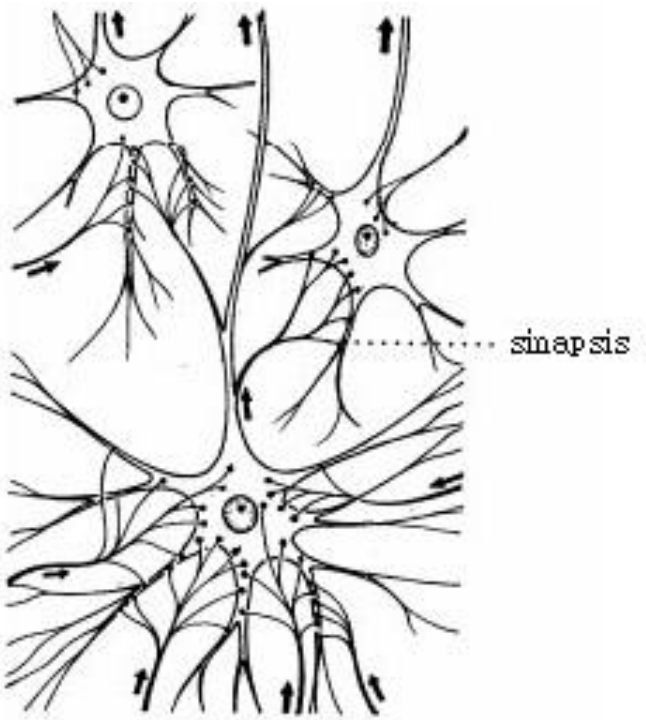
El elemento básico del sistema nervioso es la neurona.

El cuerpo de la neurona se llama *soma*. La neurona tiene canales de entrada y salida de información, conocidos como *dentritas* y *axón*, respectivamente.

Los terminales axón se conectan a otras dentritas a través de las *sinapsis*.

El cerebro humano contiene aproximadamente 100 billones ($10^{11}$) neuronas y 100 trillones ($10^{14}$) sinapsis.

# Red Neuronal Biológica (cont.)



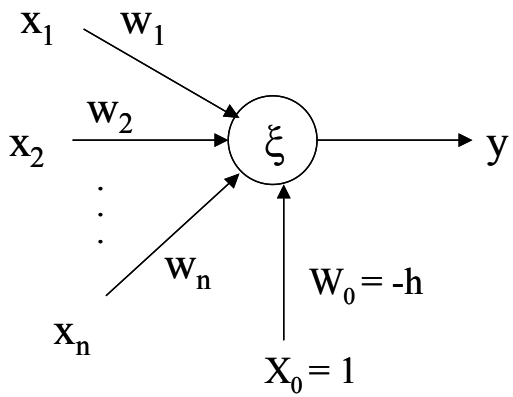

Red Neuronal

Conexiones sinápticas entre las neuronas establecen una red.
Sinapsis *excitadoras*: habilitan a un estímulo en el sistema nervioso.
Sinapsis *inhibidoras*: atenúan un estímulo en el sistema nervioso.

La comunicación entre neuronas es por lo general de manera química.

Las últimas neuronas de la red recolectan todas las señales de las otras neuronas y determinan si se excede de cierto límite para responder o no a un estímulo.

# Red Neuronal Artificial

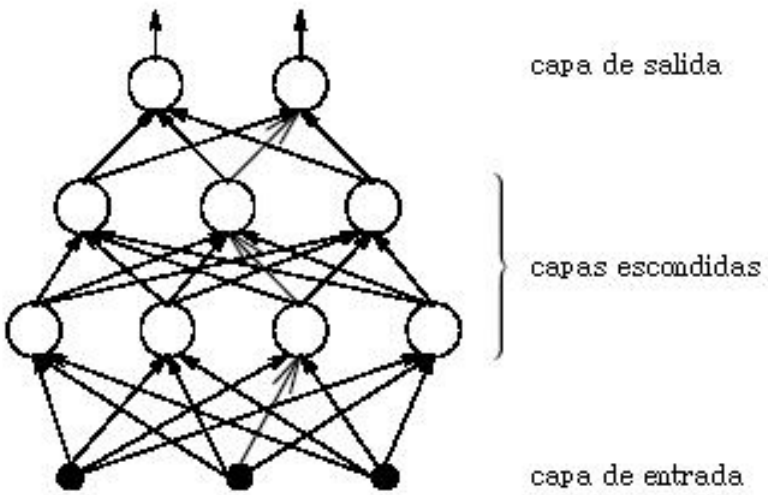



Neurona Artificial

$$\xi = \sum_{i=0}^{n} w_i x_i \quad ..... \ (1) \qquad y = \sigma(\xi) \ .... \ (2)$$

**σ( )** es en general una función contínua del tipo sigmoide, y **h** es el nivel de umbral. Para una función discreta unipolar (con h = 0):

$$y = \sigma(\xi) = \begin{cases} 1 & \text{if } \xi \geq 0 \\ 0 & \text{if } \xi < 0 \end{cases} \quad \text{donde } \xi = \sum_{i=0}^{n} w_i x_i \quad ..... \ (3)$$




La red neuronal se establece conectando las neuronas y organizándolas en capas.

Red Neuronal Artificial Multicapa

# Tipos de Redes Neuronales Artificiales

## Redes con aprendizaje supervisado

Pesos ***W*** deben ser adaptados en función del error entre la salida ideal y la obtenida por la red.

## Redes con aprendizaje sin supervisión

Pesos ***W*** se adaptan conforme se ingresan las entradas. No es necesario conocer la salida ideal.

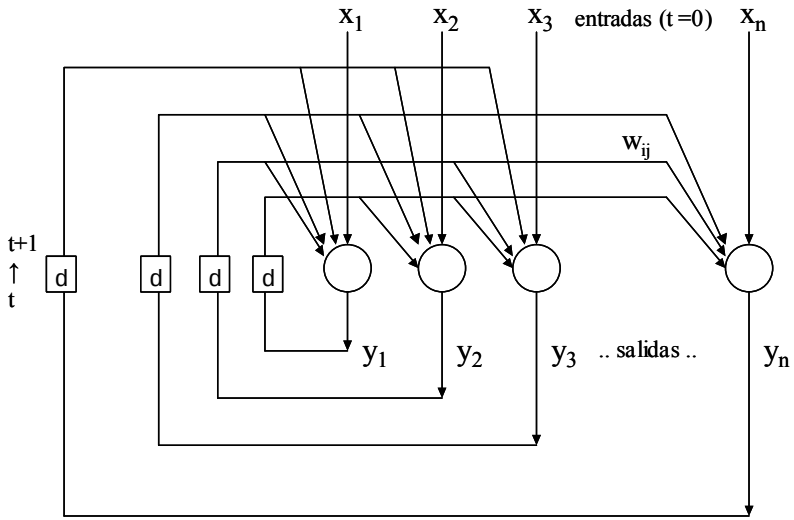
Ejemplos de Redes Neuronales:

- Single-Layer Neural Network
- Multilayer Feedforward Neural Network
- Recurrent Neural Network
- Sef-Organizing Neural Network
- Neuro-Fuzzy Network

# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Red Neuronal de Hopfield




Red Neuronal de Hopfield

La red de Hopfield es una red autosociada recurrente de una sola capa y sin supervisión.
Salida de cada neurona está en función de las conexiones sinápticas de todas las neuronas excepto ella misma.

$$net_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} y_i + x_i \quad \text{para } i = 1, 2, \ldots, n \quad \ldots\ldots (4)$$

$$\begin{pmatrix} net_1 \\ net_2 \\ net_3 \\ . \\ . \\ . \\ net_n \end{pmatrix} = \begin{pmatrix} 0 & w_{12} & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & w_{23} & \cdots & w_{2n} \\ w_{31} & w_{32} & 0 & \cdots & w_{3n} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ w_{n1} & w_{n2} & w_{n3} & \cdots & 0 \end{pmatrix} * \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{pmatrix} \quad \ldots\ldots (5)$$

# Red Neuronal de Hopfield (cont.)

Para el caso de la red de Hopfield Discreta, las salidas de las neuronas solo toman dos posibles valores. En el caso de una red de Hopfield discreta unipolar, tenemos:

$$\boldsymbol{net = W * Y + X} \qquad \text{.... (6)}$$

$$y = \begin{cases} 1 \text{ si } net_i \geq 0 \\ 0 \text{ si } net_i < 0 \end{cases} \qquad \text{.... (7)}$$

Los pesos ***W*** deben determinarse a través del proceso de aprendizaje, en función de las señales de entrada.

Aplicación tipica de la Red de Hopfield: Recuperación de patrones a partir de la presentación de patrones difusos. Para esto, debe crearse la Base de Datos de patrones, de tal forma que ante la presencia de un patrón con ruido, la red “recuerda” o “asocia” el patrón al que más se le asemeja.

# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Entrenamiento de la Red Neuronal

El algoritmo de entrenamiento de la red neuronal de Hopfield consiste en el almacenamiento de los pesos $w_{ij}$. Para esto, asumamos que se desea almacenar $p$ patrones en forma de vectores bipolares, $\mathbf{S}^{(m)}$ donde $m = 1, 2, ..., p$.

$$\boldsymbol{W} = \sum_{m=1}^{p} \mathbf{S}^{(m)} \mathbf{S}^{(m)t} - p\mathbf{I} \quad \text{para } m = 1, 2, \ldots, p \qquad \ldots. (8)$$

$$w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} S_i^{(m)} S_j^{(m)} \qquad \ldots. (9)$$

donde $\delta_{ij}$ es la función delta de Kronecker, tal que $\delta_{ij} = 1$ si $i = j$, y $\delta_{ij} = 0$ si $i \neq j$.

Para vectores unipolares, los valores "–1" se reemplazan por "0":

$$w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} (2S_i^{(m)} - 1)(2S_j^{(m)} - 1) \qquad \ldots. (10)$$

# Entrenamiento de la Red Neuronal (cont.)

Fórmula (10) fue obtenida aplicando la regla de Hebb.
Se aplica a redes cuyos patrones son mutuamente ortogonales.
Para un caso más general, se usa una regla conocida como el uso de la *matriz pseudoinversa*. Se permite realimentación a la misma neurona, $w_{ii} \neq 0$.

$$\mathrm{Sgn}\left[\boldsymbol{W} * \mathbf{S}^{(m)}\right] = \mathbf{S}^{(m)} \quad \text{para } m = 1, 2, \ldots, p \qquad \ldots. (11)$$

$$\boldsymbol{W} * \mathbf{S}^{(m)} = \mathbf{S}^{(m)} \qquad \ldots. (12)$$

$$\boldsymbol{W} * \boldsymbol{G} = \boldsymbol{G} \qquad \ldots. (13)$$

$$\boldsymbol{W} = \boldsymbol{G} * \boldsymbol{G}^{-1} \qquad \ldots. (14)$$

$$\boldsymbol{W} = \boldsymbol{G} * \boldsymbol{G}^{+} \qquad \ldots. (15)$$

colocamos los ***p*** patrones $\mathbf{S}^{(m)}$ a memorizar como columnas de una matriz ***G***. Si la matriz ***G*** fuera una matriz cuadrada de determinante no nulo, ésta sería invertible, de tal forma que la matriz ***W*** se puede expresar como en (14). Para el caso general, la matriz ***G*** no es cuadrada, con lo que la matriz ***W*** se obtiene de (15), donde $G^{+}$ es la matriz pseudo inversa de G.

# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Recuperación de Patrones

Dos posibles algoritmos:
a) Actualización dinámica síncrona o paralela
b) Actualización dinámica asíncrona

En a) todas las neuronas se actualizan a la vez
Es posible que la red "quede atrapada" entre dos mínimos de energía iguales.

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{pmatrix}^{k+1} = \sigma \left( \begin{pmatrix} 0 & w_{12} & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & w_{23} & \cdots & w_{2n} \\ w_{31} & w_{32} & 0 & \cdots & w_{3n} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ w_{n1} & w_{n2} & w_{n3} & \cdots & 0 \end{pmatrix} * \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{pmatrix}^{k} \right) \quad \text{para } k = 0, 1, \ldots \qquad \ldots. \ (16)$$

# Recuperación de Patrones (cont.)

En b) una neurona se actualiza a la vez. Las neuronas a actualizar se escogen de manera random, hasta que la red no cambie entre una iteración y la siguiente. Se garantiza que la red llegue a un mínimo de energía.

$$v_i^{k+1} = \text{sgn}\left( \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij}\, v_j^k \right) \quad \text{para } i = 1, 2, \ldots, n \quad \text{y} \quad k = 0, 1, \ldots \qquad \ldots. (17)$$

1ra. Actualización: $\boldsymbol{Y^1} = [y_1^0 \;\; y_2^0 \ldots y_m^1 \ldots y_p^0 \ldots y_q^0 \ldots y_n^0]^t$

2da. Actualización: $\boldsymbol{Y^2} = [y_1^0 \;\; y_2^0 \ldots y_m^1 \ldots y_p^2 \ldots y_q^0 \ldots y_n^0]^t$ .... (18)

3ra. Actualización: $\boldsymbol{Y^3} = [y_1^0 \;\; y_2^0 \ldots y_m^1 \ldots y_p^2 \ldots y_q^3 \ldots y_n^0]^t$

# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

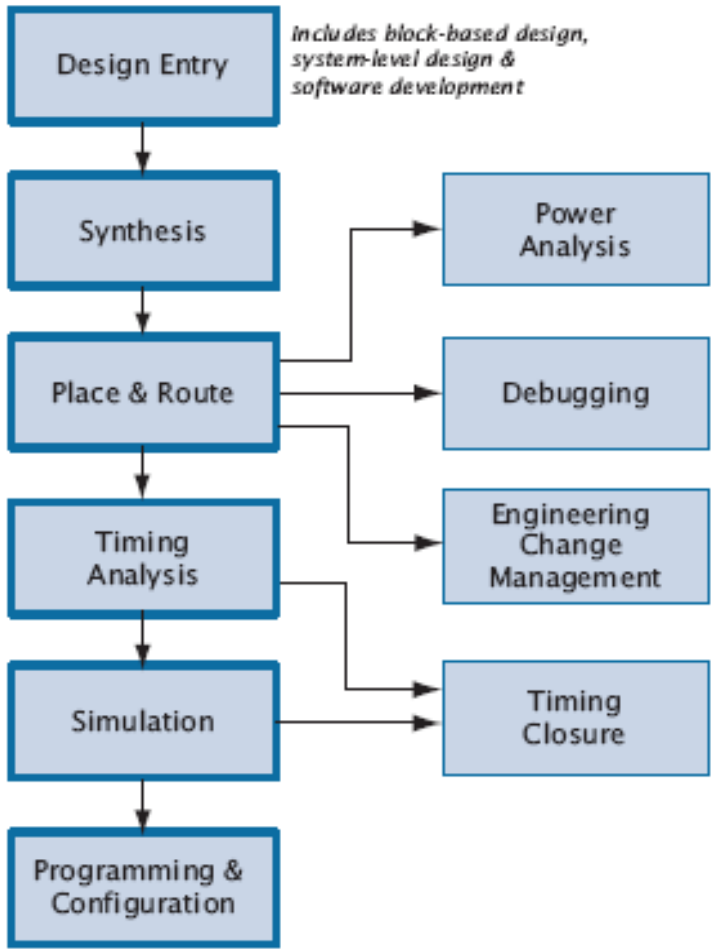
# Herramienta EDA de Diseño Digital

## Quartus II de Altera Corporation

- Herramienta para diseño e implementación de sistemas digitales.
- Múltiples formas para especificar el diseño:
  Gráfica, Texto, Diagrama de Bloques
- Posibilidad de generar sistemas completos en un chip, conocido como SoPC (System on Programmable Chip).
- Integración HW y SW usando el IDE (Integrated Development Environment).
- Muchas utilidades para depuración de proyectos.
- Soporte de amplia variedad de dispositivos (CPLDs y FPGAs).

# Herramienta EDA de Diseño Digital (cont.)

## Metodología de Diseño de Quartus II



# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Diseño de la Red de Hopfield en un FPGA

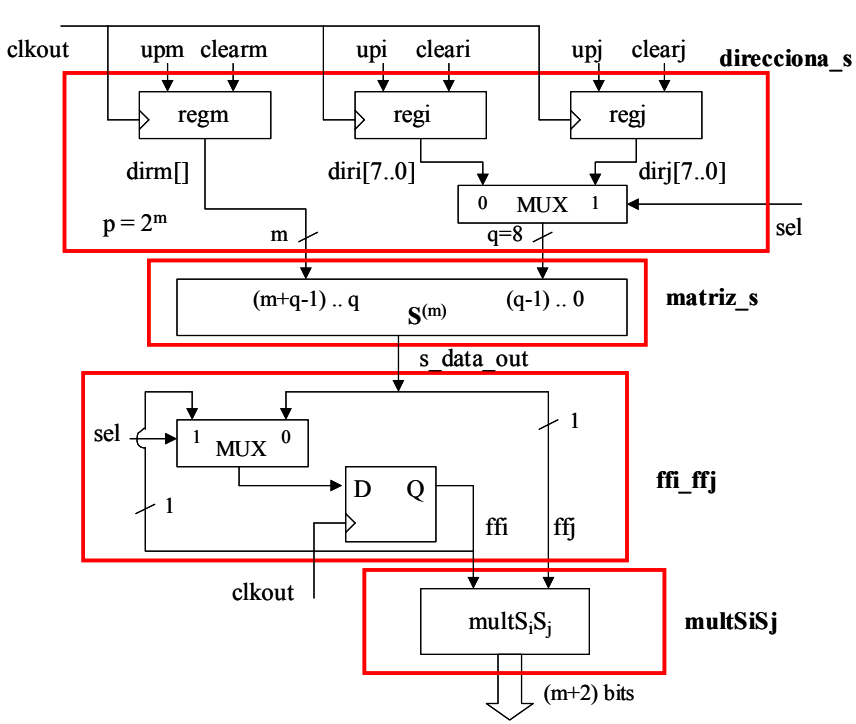
## Red Neuronal Básica (regla de Hebb):

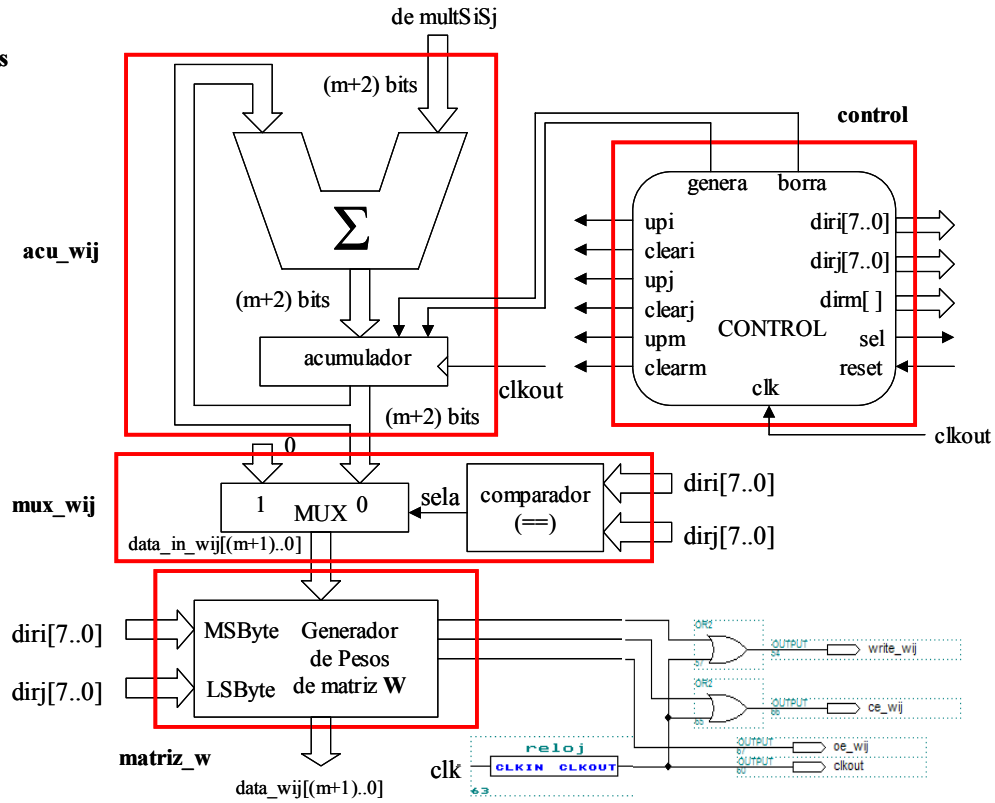
- Implementada en dispositivo EPF10K20RC240-4 de la familia FLEX10K de Altera.
- Se usó tarjeta de desarrollo UP-1 de Altera.
- Se usó herramienta EDA Max+Plus II de Altera.
- Uso de lenguaje de descripción de hardware.

## Red Neuronal Avanzada (regla de matriz Pseudo Inversa):

- Implementada en dispositivo EP2C35F672C6 de la familia Cyclone II de Altera.
- Se usó tarjeta de desarrollo DE2 de Altera.
- Se usó herramienta EDA Quartus II de Altera.
- Uso de lenguaje de descripción de hardware.

# Fase de Entrenamiento




Diagramas de Bloques para el almacenamiento de pesos en matriz W con el método de Hebb

# Recuperación de Patrones

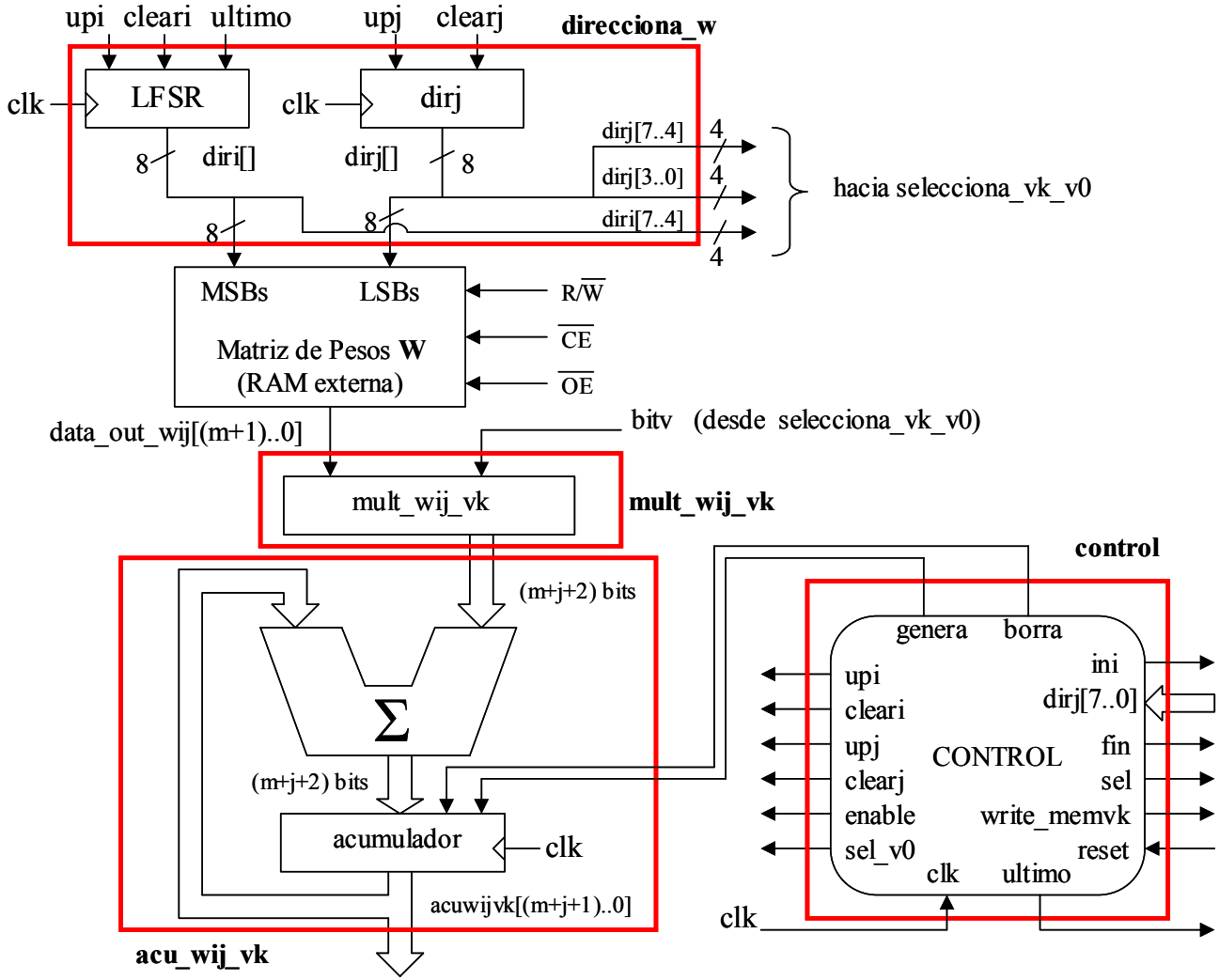



Diagrama de Bloques para recuperación de patrones (1 de 3)

# Recuperación de Patrones (cont.)

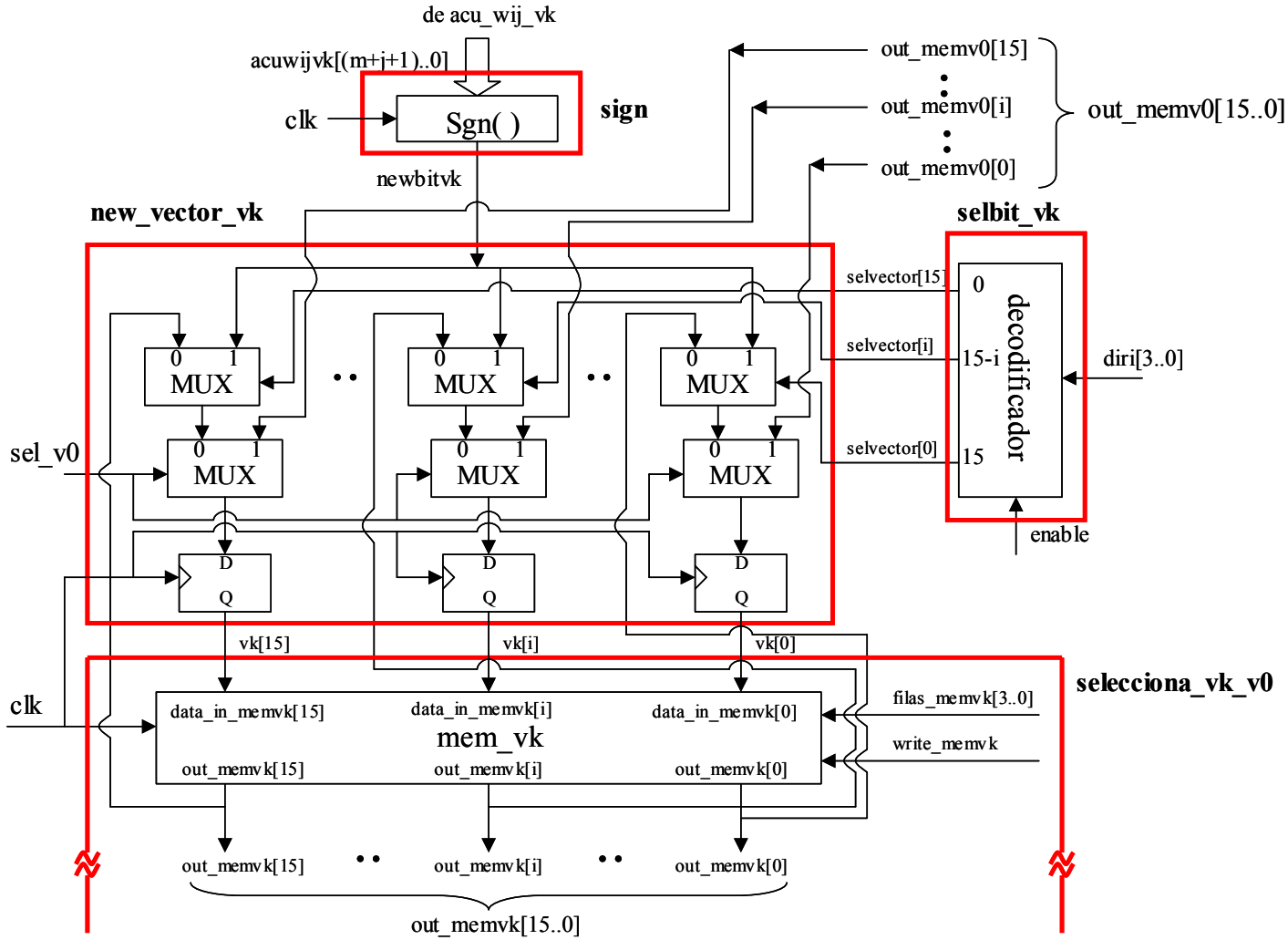



Diagrama de Bloques para recuperación de patrones (2 de 3)

# Recuperación de Patrones (cont.)

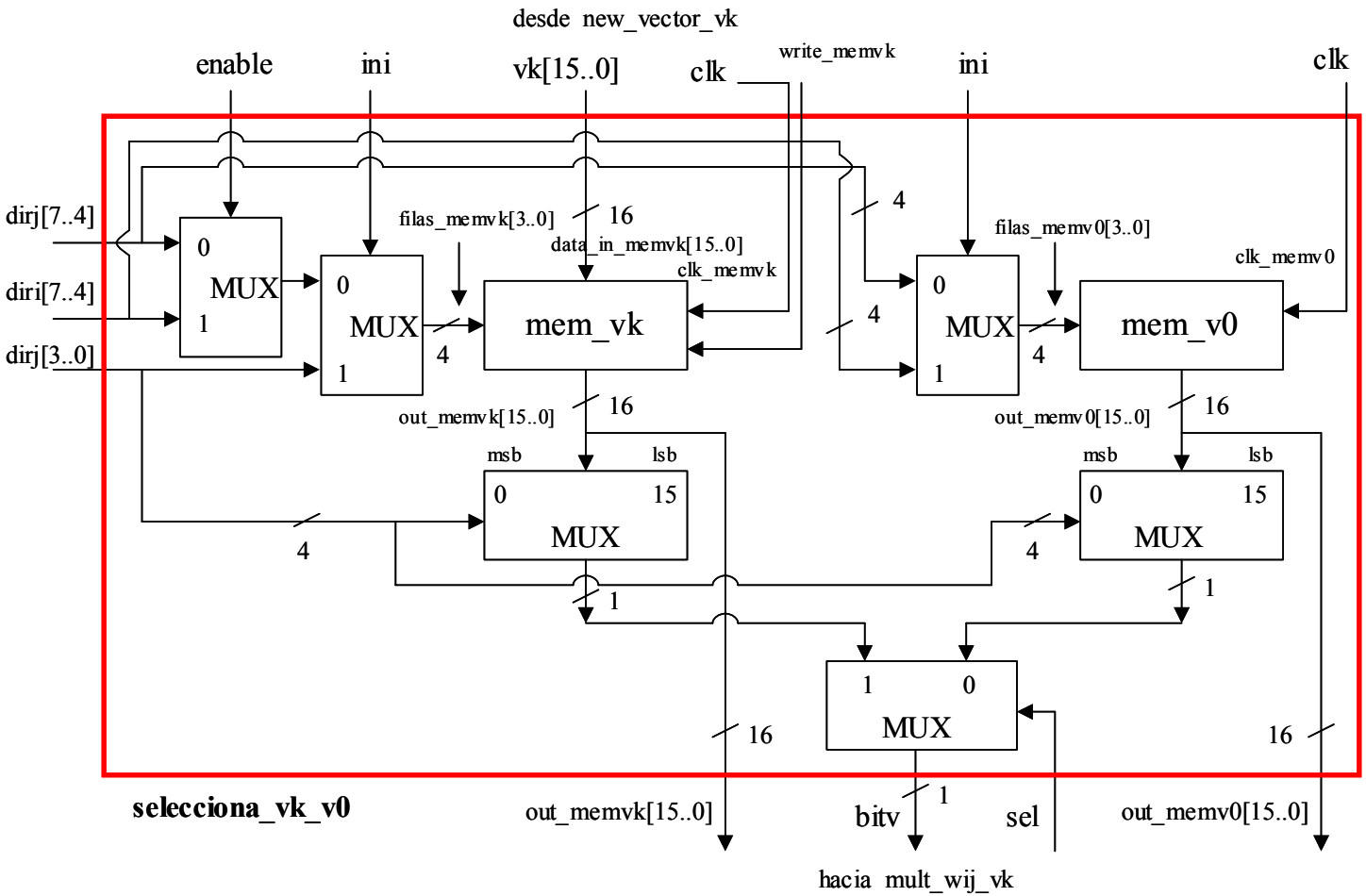


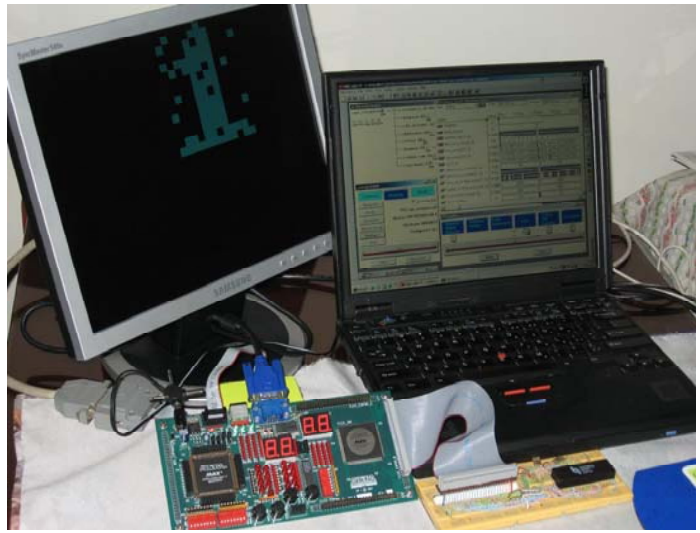

Diagrama de Bloques para recuperación de patrones (3 de 3)

# Agenda

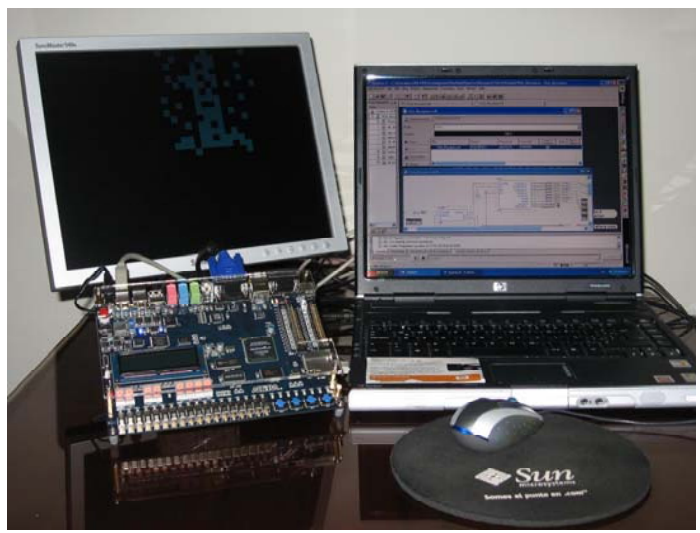
- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

# Demostración

- Red Neuronal Avanzada de Hopfield (regla de matriz pseudo inversa).
- Aplicación de recuperación de imágenes bidimensionales del tipo blanco y negro. Las imágenes están basadas en un arreglo de 16x16 pixeles que representan dígitos de “1” al “9”.
- Demostración con Matlab.
- Demostración con Quartus II.



Red Neuronal Básica




Red Neuronal Avanzada

# Desempeño de la Red Neuronal

## Función de Energía:

$$E = -(1/2)\, V^T * W * V \quad \ldots(19)$$

$$E = -(1/2)\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij}\, v_i v_j \quad \ldots (20)$$

donde ***V*** es el vector columna que contiene el valor de todas las neuronas.
La función de energía ***E*** debe ser decreciente conforme la red itera.

$$E(V^{(t)}) \geq E(V^{(t+1)}) \quad \ldots (21)$$

## Red Neuronal Básica:

- Solo se almacenan 4 patrones (dígitos “1”, “2”, “3”, y “4”)
- Actualización de cada neurona en 39 us
- Actualización de 256 neuronas en 9.98 ms
- Poco inmune al ruido

## Red Neuronal Avanzada:

- Se almacenan los 9 patrones (dígitos del “1” al “9”)
- Actualización de cada neurona en 12.9 us
- Actualización de 256 neuronas en 3.30 ms
- Red más robusta, soporta más ruido.

# Agenda

- Red Neuronal Biológica y Artificial
- Red Neuronal de Hopfield
- Fase de Entrenamiento de la Red
- Recuperación de Patrones
- Herramienta EDA de Diseño Digital
- Diseño e Implementación en un FPGA
- Demostración
- Resumen

## Resumen

- Revisión de Redes Neuronales
- Implementación de una Red Neuronal en un FPGA
- Demostración

## Referencias


- http://en.wikipedia.org/wiki/Neural_network
- http://en.wikipedia.org/wiki/Artificial_neural_network
- http://es.wikipedia.org/FPGA
- http://www.altera.com