Diseño de un procesador analógico-digital para la implementación integrada de redes neuronales en sistemas de bajo consumo

Javier Alejandro Martínez Nieto[1], María Teresa Sanz Pascual[1], Nicolás Medrano Marqués[2], Belén Calvo López[2]

[1] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Departamento de Electrónica, Puebla, México

[2] Universidad de Zaragoza, Grupo de Diseño Electrónico (GDE), Zaragoza, España

{almartinez,materesa}@inaoep.mx, {nmedrano,becalvo}@unizar.es

Resumen. En este artículo se presenta el diseño electrónico en tecnología CMOS estándar de 0.18μm con alimentación $V_{DD} = 1{,}8V$, de los principales bloques que conforman una neurona: el multiplicador y la función de activación no-lineal. De igual forma, se presentan los resultados por simulación eléctrica en CADENCE, así como el modelado matemático en MATLAB de su comportamiento. Una comparación de ambos modelos presenta errores relativos $e_r < 1\,\%$ para las dos operaciones. Para su validación, los modelos matemáticos generados fueron aplicados a una estructura de red neuronal entrenada para resolver la operación lógica XOR.

Palabras clave: Red neuronal artificial, CMOS, calibración de sensores.

Design of an Analog-digital Processor for Integrated Implementation of Neural Networks in Low-power Systems

Abstract. The electronic design of the artificial neuron main characteristic blocks is presented in this paper. Both, the multiplier and the non-linear activation function, were designed in standard $0{,}18\mu m$ CMOS process with $1{,}8V$ power supply. CADENCE electrical simulation results and mathematical modeling in MATLAB are also presented. A comparison between the high level and electrical models shows a relative error below 1 % for both operations. In order to verify the correct operation, the generated models were applied to a trained neural network structure to solve the XOR logical operation.

Keywords: Artificial neural networks, CMOS, sensor calibration.

1. Introducción

Los sistemas de inteligencia artificial basados en redes neuronales han mostrado recientemente su potencialidad en espectaculares casos de éxito: las victorias a humanos en el juego del *GO*, sistemas que superan el test de Turing, o el desarrollo de arte psicodélico, son algunos de los ejemplos más actuales [**?**]. Sin embargo, los recursos computacionales requeridos, así como el consumo de potencia y tamaño del hardware necesario para este tipo de sistemas han impedido hasta ahora su incorporación en dispositivos comerciales.

Por otro lado, el desarrollo microelectrónico de sistemas de procesamiento de las últimas décadas ha supuesto su implantación ubicua, dando lugar incluso a una nueva disciplina como son los sistemas ciber-físicos. A esta proliferación no ha sido ajeno el desarrollo de módulos de procesamiento basados en redes neuronales, con diseños para su incorporación específica a dispositivos portátiles con alimentación por baterías, como los smartphones, o con recursos energéticos limitados, como los vehículos auto-dirigidos. En general estos módulos neuronales de propósito específico, pensados para procesamiento y reconocimiento de imágenes, muestran unos niveles de consumo reducidos, del orden de cientos de mili-watts.

La potencialidad de las redes neuronales artificiales, capaces de establecer relaciones funcionales entre conjuntos de patrones de estímulos-respuestas sin un conocimiento previo matemático del problema a resolver, así como su capacidad de modificar dicha relación funcional de manera dinámica ante cambios en los patrones, hacen de estas técnicas de procesado una herramienta de gran flexibilidad. Para su inclusión en módulos de bajo consumo, como los nodos en una red inalámbrica de sensores, donde la vida operativa debe contarse en meses, es preciso reducir aún más el consumo y el tamaño de los dispositivos, procurando mantener su adaptabilidad. En este sentido, existen dos aproximaciones para su implementación: mediante procesadores digitales o con el diseño de sistemas de procesamiento analógicos o mixtos [**?**]. La elección de uno u otro método vendrá condicionada por diversos parámetros, entre los que se incluyen las restricciones de consumo y área, así como la resolución requerida en la respuesta del sistema.

En general, si las restricciones vienen impuestas por el consumo y área, las implementaciones analógicas integradas de los procesadores neuronales ofrecen claras ventajas. Si además es preciso disponer de la flexibilidad que supone la reprogramación de los parámetros de la red neuronal, una implementación mixta que incluya procesadores analógicos con almacenamiento digital puede ser la solución óptima. Un ejemplo de este tipo de aplicaciones puede ser el diseño de módulos de acondicionado y calibrado integrados para *smart sensors* [**?**], donde un sistema neuronal analógico permitiría optimizar la respuesta del sensor antes de su digitalización, compensando derivas y linealizándola, con la capacidad de adecuar su respuesta ante cambios en el comportamiento del propio sensor asociados al envejecimiento.

Algunas características de las redes neuronales como el aprendizaje adaptativo, la auto-organización y la tolerancia a fallos [**?**], las hacen sumamente atractivas para la resolución de problemas relacionados con la predicción de eventos,

clasificación y ajuste de datos, entre otros. Además, las redes implementadas en hardware presentan robustez a variaciones de proceso debido al esquema de retroalimentación inherente proporcionado por el mecanismo de aprendizaje de retropropagación de errores, teniendo en cuenta las no-idealidades de los circuitos, como sucede con el enfoque de aprendizaje *chip-on-the-loop* [**?**], el cual lleva a cabo el entrenamiento de la red fuera del chip pero teniendo en cuenta toda la circuitería utilizada en la implementación de los bloques característicos de la neurona.

El objetivo de este trabajo es generar modelos de alto nivel de los principales bloques que conforman una neurona diseñada de forma analógica, para su posterior inserción y simulación en distintas estructuras de red. El artículo está dividido de la siguiente forma: en la Sección 2 se presenta la estructura básica de una neurona y los principales bloques que la conforman; además se describe su implementación electrónica y se presentan algunos resultados obtenidos mediante simulación. En la Sección 3 se describe el modelo matemático de alto nivel generado para cada uno de los bloques y se hace una comparación de éste con la respuesta eléctrica. En la Sección 4 se presenta una implementación sencilla de ANN para verificar los modelos y, finalmente, en la Sección 5 se presentan las conclusiones del trabajo.

2. Implementación analógica de la neurona

Como es sabido, las neuronas son las unidades básicas de procesamiento de este tipo de sistemas, ya que se encargan de llevar a cabo todo el procesamiento de las señales. Es por esto que se debe tener especial cuidado en su diseño y modelado, sobre todo si se trata de una implementación en hardware.

En la implementación de la neurona es posible distinguir dos bloques significativos: un bloque de multiplicación encargado de multiplicar la señales de entrada o bien las señales de las capas intermedias, por un conjunto de pesos o coeficientes de ajuste, que pueden seer representados de manera analógica o digital; y un bloque que genera la función de activación para implementar la

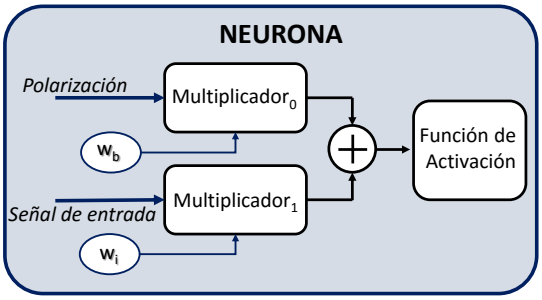


Fig. 1. Diagrama a bloques de la unidad básica de procesamiento

operación no-lineal. En la Fig. 1 está representado el esquema de esta unidad básica de procesamiento.

Es conveniente mencionar que para la implementación analógica de este tipo de sistemas, la tecnología CMOS es la opción más viable debido a que es una tecnología madura en cuanto a implementaciones VLSI, además de que ofrece la posibilidad de co-integración del sensor con la electrónica asociada a éste si fuera requerido. En el trabajo que se está desarrollando se utiliza una tecnología CMOS estándar de 0.18μm con voltaje de alimentación $V_{DD} = 1{,}8V$.

2.1. Multiplicador

Como ya se ha comentado, el multiplicador se encarga de ponderar una señal a partir de un coeficiente o peso de activación w. Éste sólo tomará valores comprendidos entre [-1,1], de tal forma que si $w = 0$ se entiende que la señal se inhibe y es cero, si $w = 1$, toda la señal se transmite, y si $w = -1$, la señal se invierte o cambia de signo.

El bloque implementado es un amplificador de ganancia programable (PGA) que actúa como multiplicador analógico-digital. Todo el procesamiento de la señal se hace en el dominio analógico, pero mediante un control digital se establece el factor de ponderación w. Este control permite definir de forma sencilla el valor del peso mediante una palabra digital.

La implementación del PGA se muestra en la Fig. 2. El circuito es un amplificador de voltaje que utiliza resistencias y transistores MOS para establecer la ganancia. Los dos transistores trabajan como como un divisor de corriente MOS (MCD) [**?**], en donde las corrientes I_1 e I_2 se relacionan de la siguiente forma:

$$I_1 = K_1[f_1(\psi_{sL_1}) - f_1(\psi_{s0_1})] = -V_{in}/R_1, \tag{1}$$

$$I_2 = K_2[f_2(\psi_{sL_2}) - f_2(\psi_{s0_2})] = -V_o/R_2. \tag{2}$$

Los potenciales de superficie de drenaje ψ_{sL_1} y ψ_{sL_2} son los mismos por conexión, mientras que los potenciales de superficie de fuente ψ_{s0_1} y ψ_{s0_2} son iguales por las condiciones de operación, ya que ambas fuentes están conectadas a la terminal de entrada de un amplificador, y los voltajes de compuerta y de

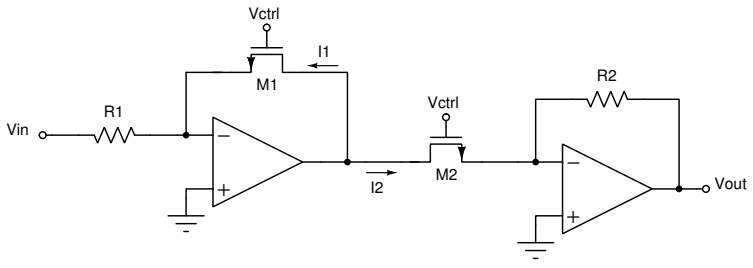


Fig. 2. PGA basado en divisores de corriente MOS

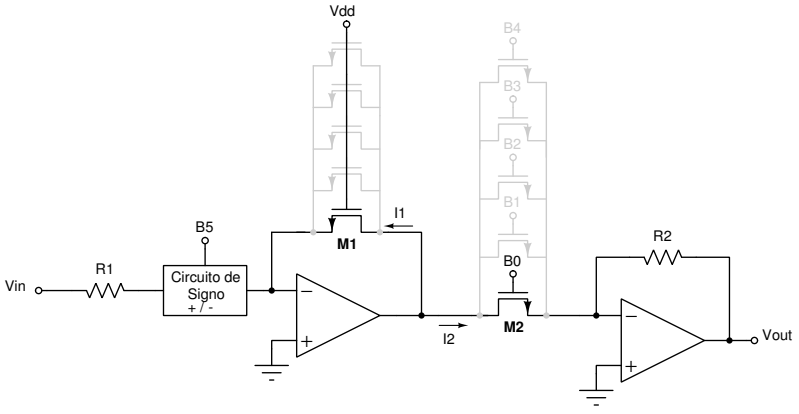


Fig. 3. Implementación eléctrica del bloque de multiplicación

cuerpo son iguales para ambos transistores. Por lo tanto, si M_1 y M_2 tienen un buen *matching*, la función de transferencia del amplificador va a estar dada por:

$$\frac{V_o}{V_{in}} = \frac{R_2}{R_1}\frac{K_2}{K_1} = \frac{R_2}{R_1}\frac{(W/L)_2}{(W/L)_1}. \tag{3}$$

A partir de esta expresión, el factor de proporcionalidad entre ambos transistores puede ser utilizado para definir la ganancia del amplificador y, por lo tanto, el factor de ponderación requerido. La manera más sencilla de hacerlo es utilizando un conjunto de transistores en paralelo como se aprecia en la Fig. 3, de tal forma que el transistor MOS equivalente generado tendrá una anchura igual a xW, donde x es el número de transistores en paralelo.

El esquema completo del multiplicador se muestra en la Fig. 3. La ponderación de la señal se controla mediante una palabra digital de 5 bits ($B_4...B_0$), haciendo posible tener 32 posibles valores entre 0 y 1 con un delta de variación entre cada bit igual a:

$$\Delta = \frac{1}{2^n - 1} = 32{,}258 \times 10^{-3}, \tag{4}$$

donde n es el número de bits. Ademas se ha añadido un bloque de signo, cuyo objetivo es cambiar el sentido de la señal que entra al PGA, de forma que sea posible tener también pesos negativos. Este circuito está controlado por el bit de signo B_5. Debido a que la máxima ganancia es 1, los transistores que conforman M_1 siempre están encendidos y operando en triodo, mientras que aquellos que conforman M_2 estarán controlados por la palabra digital. Así, si se desea la máxima ganancia $G = 1$, todos los bits de esta palabra digital deben ser V_{DD}.

Circuito de Signo. En la Fig. 4 se presenta el circuito de signo implementado. Este corresponde a un espejo de corriente clase AB [**?**]. En condiciones estáticas,

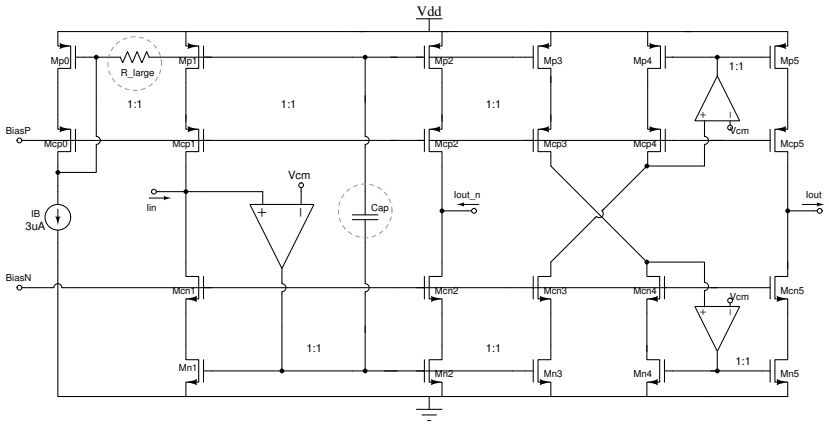


Fig. 4. Implementación eléctrica del bloque de signo

el capacitor se comporta como un circuito abierto y el esquema es equivalente a un espejo de corriente simple. Bajo condiciones dinámicas, la configuración opera en clase AB transformando los transistores M_{p1} y M_{p2} en fuentes de corriente dinámicas. El circuito cuenta con dos ramas de salida, una que pasa la corriente en la misma dirección en la que entra, y la otra que invierte su sentido. Mediante el bit B_5 es posible seleccionar qué rama se encuentra activa. El error relativo en la transferencia de corriente en ambas salidas es menor al 1 % considerando una corriente mínima de entrada $I_{in} = 10nA$; y la distorsión armónica total (THD) de la señal es menor a $-80dB$ teniendo en cuenta una señal máxima de entrada $I_{in} = \pm 15\mu A$ a una frecuencia de $1kHz$.

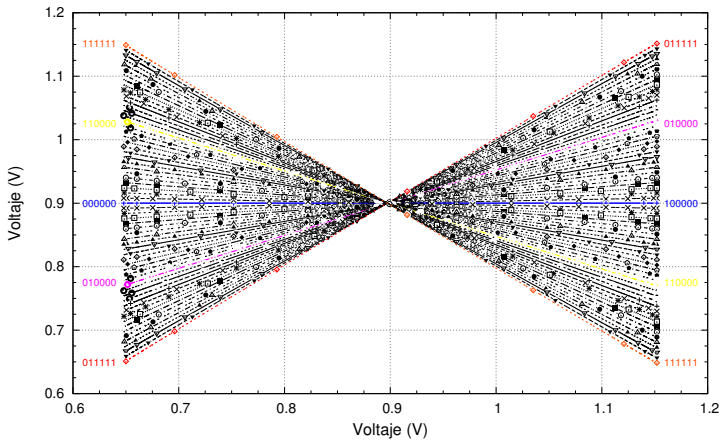
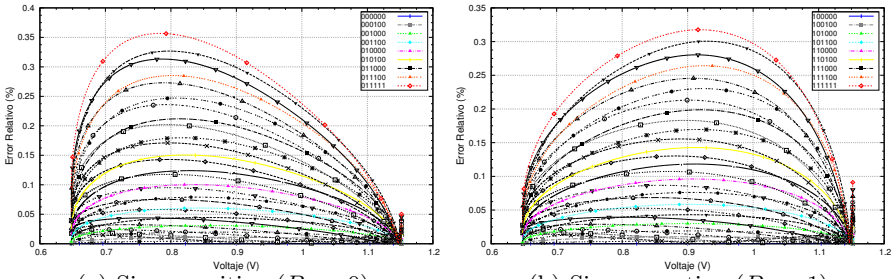


Fig. 5. Respuesta eléctrica del multiplicador en modo voltaje de 5 bits

Simulación del multiplicador. El voltaje de entrada considerado tiene un nivel de modo común $V_{cm} = 0{,}9V$. Señales de voltaje menores a este valor son consideradas negativas, mientras que por encima de $0{,}9V$ son positivas. Se simuló el multiplicador de la Fig. 3 considerando 5 bits más el bit de signo. El transistor correspondiente a cada bit está formado a su vez por un conjunto de transistores con una única dimensión. El amplificador utilizado en el esquema completo es un amplificador operacional de transconductancia (OTA) con etapa de salida clase AB con ganancia $G_{opam} = 76dB$ y resistencia de salida $R_{out} = 160\Omega$. Los valores de las resistencias son $R_1 = R_2 = 18k\Omega$.



(a) Signo positivo ($B_5 = 0$). (b) Signo negativo ($B_5 = 1$).

Fig. 6. Error relativo e_r con respecto a la respuesta calculada idealmente

La simulación se llevó a cabo utilizando el software de diseño electrónico CADENCE . En la Fig. 5 se representa el voltaje de salida con respecto al voltaje de entrada para las combinaciones posibles de bits. Se calculó el error relativo e_r que existe entre cada una de estas respuestas con respecto a la respuesta esperada o salida ideal. Para esto, se evaluó la expresión mostrada a continuación (Eq. 5) para calcular y graficar cada una de las respuestas ideales:

$$V_{out_{ideal}} = 0{,}9 - 0{,}9 \cdot w + V_{in} \cdot w, \tag{5}$$

donde V_{in} es el voltaje de entrada con un nivel de DC igual a $0{,}9V$, y w el peso.

En la Fig. 6 se presenta el error relativo calculado para cada palabra digital. El inciso (a) muestra el error relativo considerando sólo pesos positivos, es decir, sin inversión de la señal ($B_5 = 0$); mientras que en el inciso (b) se presenta el error relativo considerando inversión en la señal, y por lo tanto, pesos negativos ($B_5 = 1$). El máximo error relativo para el primer caso es $e_r = 0{,}3567\,\%$, y corresponde a la máxima ponderación de la señal ($w = 1$). Igualmente, tomando en cuenta sólo pesos negativos, el máximo error relativo calculado se tiene para la máxima ganancia y es $e_r = 0{,}3176\,\%$. En condiciones estáticas, el multiplicador presenta un consumo máximo de potencia $P_{max} = 42{,}8\mu W$, que se tiene cuando el bit de signo es '1'.

De acuerdo al diagrama de bloques de la neurona (Fig. 1), es necesario hacer una sumatoria de las señales ponderadas. Utilizando el mismo esquema

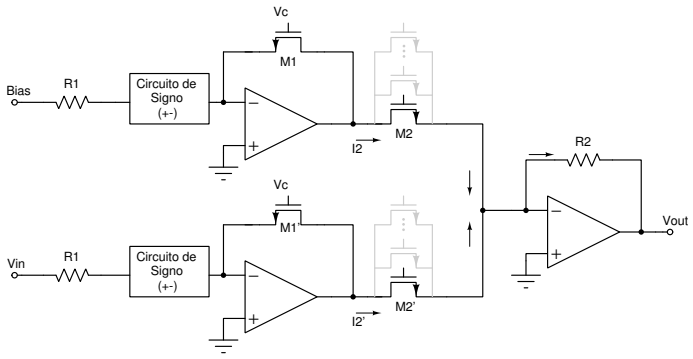


Fig. 7. Esquema para realizar la suma de las señales ponderadas

de multiplicación que se ha descrito, es posible realizar la suma de las señales en un nodo común, tal como se muestra en la Fig. 7. Las señales en corriente son sumadas en el nodo negativo del segundo amplificador, donde se tiene una tierra virtual debido a la configuración del circuito.

2.2. Función de activación

La función de activación calcula el estado de actividad de una neurona. Transforma la entrada global en un valor o estado de activación, cuyo rango normalmente es de [0,1] o [–1,1], ya que una neurona puede estar totalmente inactiva o activa. Las funciones de activación más utilizadas son: la función lineal, función escalón y función sigmoidal.

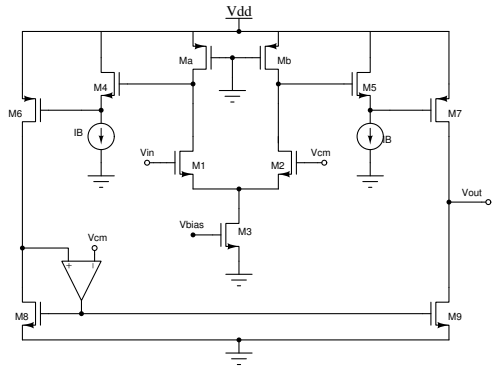
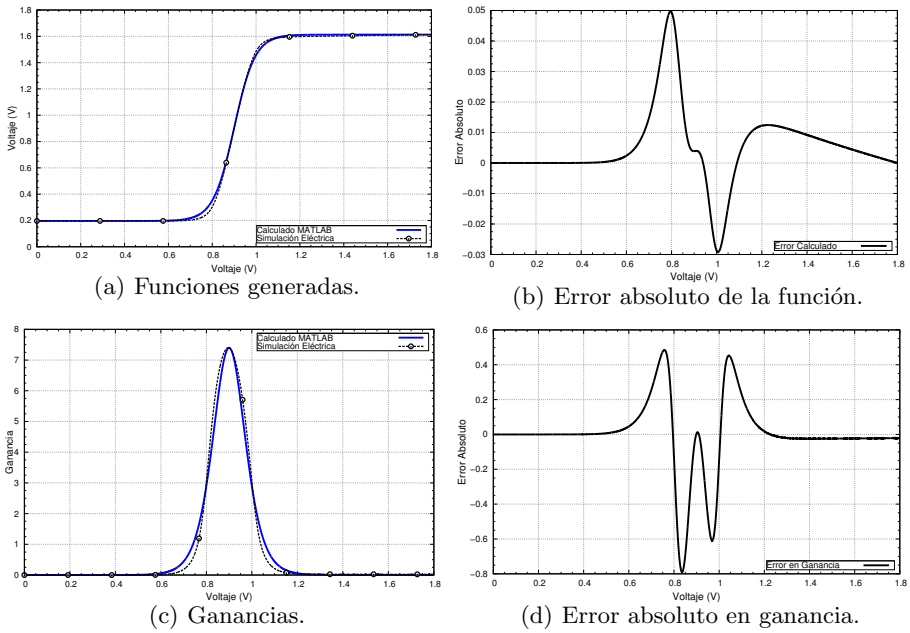


Fig. 8. Implementación eléctrica del bloque que genera la función de activación

En el trabajo se ha implementado una función sigmoide debido a que es diferenciable y presenta una característica no-lineal, propiedades importantes



(a) Funciones generadas.

(b) Error absoluto de la función.

(c) Ganancias.

(d) Error absoluto en ganancia.

Fig. 9. Comparación de la función de activación eléctrica con la función calculada

para el entrenamiento de redes multi-capa con retropropagación. La función implementada presenta una característica similar a una tangente hiperbólica, la cual tiene forma de 'S' con umbrales de saturación en -1 y $+1$, y pendiente no-lineal en la parte central. Sin embargo, en el circuito implementado estos niveles de saturación están definidos entre $0{,}2V$ y $1{,}6V$ con el mismo nivel de modo común que el multiplicador.

En la Fig. 8 se muestra el circuito eléctrico implementado. Está formado por un par diferencial con transistores NMOS como entradas, y cargas resistivas implementadas con los transistores M_a y M_b operando en la región lineal. El amplificador utilizado en el esquema fija el modo común $V_{cm} = 0{,}9V$ a la salida del circuito, y está implementado mediante un par diferencial sencillo. La resistencia equivalente de los transistores M_a y M_b es $R_{on} = 4{,}6k\Omega$, y la ganancia total del circuito es $G = 7{,}4$.

Simulación del Circuito. El circuito se simuló variando el voltaje de entrada entre 0 y $1{,}8V$. En la Fig. 9(a), con línea negra punteada, se muestra la respuesta obtenida, acompañada también, en continua azul, por una función tangente hiperbólica generada matemáticamente en MATLAB para que tuviera los mismos valores de ganancia ($G = 7{,}4$) y niveles de saturación ($V_{max} = 1{,}6$ y $V_{min} = 0{,}2$) que la implementación eléctrica.

En la misma figura, pero inciso (b), está representado el error absoluto obtenido al comparar ambas funciones. Se aprecia un máximo error absoluto $e_{abs(f)} = 0{,}04V$ que corresponde a la secciones con más no-linealidad de la respuesta (codos de la función). Por otro lado, en en inciso (c) se presenta el comportamiento de la ganancia (derivada de la función) considerando el rango completo del voltaje de entrada, mientras que en el inciso (d) está representado el error absoluto en ganancia calculado. Se aprecia que el máximo error absoluto en ganancia es de $e_{abs(g)} = 0{,}79$. Finalmente, el consumo total de potencia estática es de $P_{max} = 27{,}88\mu W$.

3. Modelo de alto nivel de los circuitos eléctricos

3.1. Multiplicador

Con los datos obtenidos de las simulaciones eléctricas se generó un vector Y con todos los valores de salida posibles y una matriz X que incluía en una columna los valores de entrada y en otra, los pesos de activación. A partir del arreglo de datos generado, y mediante MATLAB se obtuvo un modelo matemático que fuera válido para todos los casos. Esto se llevó a cabo con ayuda de la función *regstats*, la cual realiza una regresión multilineal de las respuestas en Y.

El modelo matemático que describe la respuesta del multiplicador de 6 bits (5 bits y el signo) es el siguiente:

$$V_{out} = 0{,}90001 - (1{,}0298 \times 10^{-5}) \cdot V_{in} - (0{,}8969) \cdot w + (0{,}9983) \cdot V_{in} \cdot w. \quad (6)$$

Como se aprecia en esta ecuación, el modelo es muy parecido al modelo ideal establecido en la ecuación (5). Se evaluó el modelo obtenido y se comparó con los resultados eléctricos. En la Fig. 10 se presenta la gráfica obtenida para cada palabra digital considerando el mismo rango de voltaje de entrada que en la simulación eléctrica. Los resultados muestran que se tiene un error cuadrático medio $mse = 2{,}0641 \times 10^{-7}$ entre los valores esperados (obtenidos por simulación) y los calculados a partir del modelo matemático.

3.2. Función de activación

Al igual que el multiplicador, se requiere un modelo matemático que represente el comportamiento eléctrico del circuito. Aunque hay algunas aproximaciones basadas en ecuaciones racionales de polinomios de orden mayor que se pueden obtener en MATLAB, se decidió utilizar una tabla de valores, ya que la primera opción no era práctica, pues los umbrales no saturaban y el tiempo de procesamiento era elevado. Se verificó que el modelo descrito con la tabla de valores es 100 veces más rápido que el implementado con la expresión polinomial.

Cabe mencionar que a partir del conjunto de valores, se obtuvo la derivada de la función y, de igual forma, se generó una tabla con estos valores.

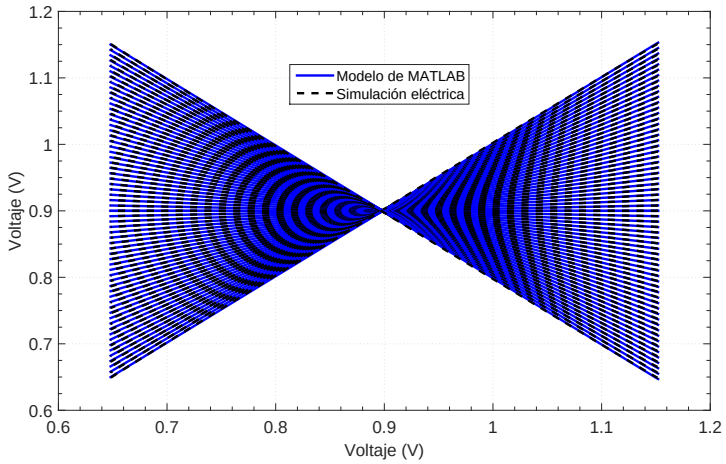


Fig. 10. Comparación de la respuesta obtenida con el modelo y la respuesta eléctrica

4. Resultados preliminares

El objetivo del trabajo, además de modelar el comportamiento eléctrico de los bloques característicos de la neurona, es comprobar que dichos modelos son funcionales al introducirlos en una red neuronal. En esta sección se presenta un ejemplo sencillo y práctico para verificar esto.

4.1. Solución del operador XOR

Una tarea sencilla, pero que requiere el uso de redes neuronales multicapa es la solución del operador OR exclusivo. Como se sabe, esta compuerta funciona de acuerdo a la lógica mostrada en la Tabla 1, donde la salida es igual a '1' cuando uno de los operandos es '1', pero no ambos.

Tabla 1. Tabla de verdad de la compuerta XOR

B	A	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Con la herramienta *nntool* de MATLAB es posible crear, entrenar y simular redes neuronales de manera sencilla, y por esta razón es el software que se utiliza en el trabajo. Se creó una red neuronal multicapa de 2 entradas, 1 capa oculta con 4 neuronas y la capa de salida, tal como se muestra en la Fig. 11. En

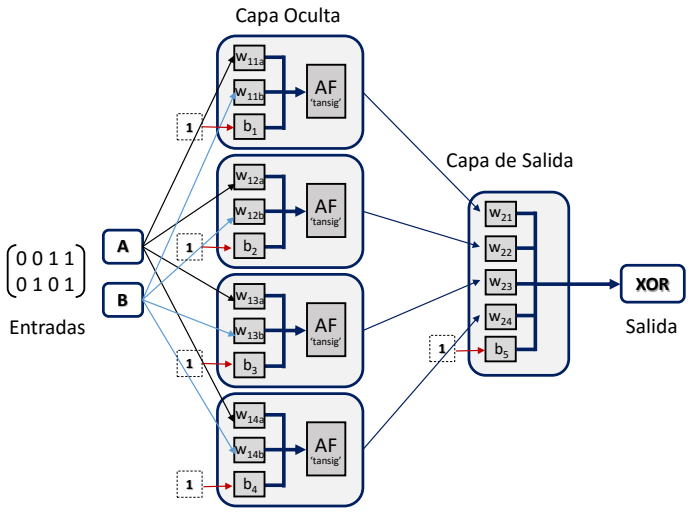


Fig. 11. Red neuronal creada en MATLAB para resolver el operador lógico XOR

esta se observa que hay 17 señales que necesitan ser ponderadas: en la capa oculta hay conexión de 8 señales que provienen de las entradas 'A' y 'B' y 4 señales de polarización o *bias*. Luego, en la última capa están conectadas las señales de salida de las 4 neuronas de la capa anterior y su respectiva señal de polarización. La función de activación en cada neurona de la capa oculta es la tangente hiperbólica.

El algoritmo de entrenamiento de la red está definido en la función 'TRAINLM'. Ésta corresponde al algoritmo *Levenberg-Marquardt* de retropropagación, el cual realiza las iteraciones y actualiza los pesos empleando el método de optimización por descenso de gradiente [**?**]. Aunque es posible utilizar otros algoritmos, éste es el más rápido y eficiente para una red de este tamaño.

Entrenamiento de la red con los modelos obtenidos. Es conveniente mencionar, que para el entrenamiento de la red, se estableció que el '1' lógico corresponde a un nivel de voltaje de $1{,}6V$, mientras que el '0' está definido como $0{,}2V$, ya que son los voltajes máximo y mínimo que maneja el bloque de la función de activación.

Entrenando la red con las funciones predefinidas de la herramienta, el sistema resuelve el problema en 5 iteraciones, y se observa que los valores de los pesos de activación resultantes toman cualquier valor positivo o negativo.

Sin embargo, como se ha establecido en el trabajo, el rango de variación de los coeficientes de activación debe estar limitado entre -1 y 1, ya que así se ha establecido en el multiplicador diseñado. Además, debido a que la ganancia del PGA se controla con 5 bits, y recordando la ecuación (4), la resolución que puede manejar también está limitada a $32{,}258 \times 10^{-3}$. Para llevar a cabo el

entrenamiento tomando en cuenta los bloques diseñados y sus condiciones de operación, es necesario introducir sus respectivos modelos en el esquema de red generado y modificar el algoritmo de entrenamiento para que considere estas condiciones.

En el caso del multiplicador, normalmente al definir la red la estructura emplea la función *producto*, que está definida como `z=w*p`, donde w hace referencia a la matriz de pesos y p a las señales. Para utilizar el bloque diseñado, se requiere cambiar esta función por el modelo matemático generado. También es necesario obtener la derivada de la expresión con respecto a w y la derivada con respecto a p (señal o entrada), pues son necesarias en el algoritmo de entrenamiento. De esta forma, dentro del toolkit *nnet*, se generó una nueva función definiendo la multiplicación y las derivadas de la siguiente forma:

```
z     = 0.900013675640531-(1.02981131896217e-05)*p
        -(0.896976394158687)*w + (0.998246762810866)*w*p;
dz_dw = - 0.896976394158687 + 0.998246762810866*p;
dz_dp = - 1.02981131896217e-05 + 0.998246762810866*w.
```

La función de activación predefinida en la estructura de red generada es la tangente hiperbólica o *tansig*. De igual forma que el multiplicador, se generó una nueva función en donde se definió el modelo matemático del circuito eléctrico. Al tratarse de una tabla, simplemente se programó para que fuera capaz de direccionar a la posición adecuada. Para el caso de la derivada, el procedimiento es similar.

Con los nuevos modelos del multiplicador y de la función de activación, dentro de la estructura de red se cambiaron las funciones predefinidas por los nuevos modelos, de forma que en todos los puntos donde se utiliza el operador producto, ahora se va a utilizar la nueva función (igual para la función no-lineal). Además, se incluyeron algunas líneas de código en la función que realiza el entrenamiento (*TRAINLM*) para delimitar el rango de variación de los pesos y también para discretizarlos, de manera que sólo tome valores que puedan ser generados con los 5 bits considerados.

Con las modificaciones mencionadas se entrenó la red neuronal. Después de 8 iteraciones, la red llegó a la solución y estableció los valores de los 17 pesos. En el circuito eléctrico, éstos se interpretan como las ganancias que debe tener cada multiplicador (PGA). En la Tabla **??** se presenta el valor de cada uno de los coeficientes junto con su identificador (definido en la Fig. 11), así como su correspondiente palabra digital.

Finalmente, al simular la red con estos coeficientes, la salida corresponde con los valores esperados:

```
out = [0.26682 1.5262 1.5352 0.25703].
```

Tabla 2. Pesos de activación obtenidos al finalizar el entrenamiento de la red

Peso (w)	Valor	Palabra Digital	Peso (w)	Valor	Palabra Digital	Peso (w)	Valor	Palabra Digital
w_{11a}	-0.29032	101001	w_{11b}	0.41935	001101	b_1	-0.48387	101111
w_{12a}	-0.77419	111000	w_{12b}	-0.45161	101110	b_2	-0.29032	101001
w_{13a}	-0.32258	101010	w_{13b}	0.80645	011001	b_3	0.3871	001100
w_{14a}	-0.48387	101111	w_{14b}	0.48387	001111	b_4	0.67742	010101
w_{21}	0.74194	010111	w_{22}	1	011111	b_5	-0.3871	101100
w_{23}	1	011111	w_{24}	-1	111111	–	–	–

5. Conclusiones

En este trabajo se ha presentado la implementación analógica de los bloques característicos que conforman la unidad de procesamiento básica de una red neuronal, así como su modelado matemático en MATLAB.

El diseño electrónico del multiplicador y del bloque que genera la función no-lineal se realizó en tecnología CMOS de 0.18μm con alimentación de 1.8V. Las simulaciones eléctricas de ambos bloques muestran buenos resultados al compararlas con funciones ideales implementadas numéricamente, ya que, para el caso del multiplicador de 6 bits (5 para definir el peso y 1 para controlar el signo) implementado, el error relativo e_r se mantiene por debajo del 1 %, mientras que para el caso del bloque no-lineal con característica de tangente hiperbólica, el máximo error absoluto es $e_{abs} = 0{,}04V$.

Además, con los datos de simulación, se modeló en alto nivel el comportamiento eléctrico de ambos bloques y se verificó, resolviendo el operador lógico XOR, que es posible entrenar una red neuronal introduciendo dichos modelos en la estructura de ésta. Así mismo, modificando el algoritmo de entrenamiento es posible discretizar y delimitar los valores posibles de los coeficientes de activación.

Agradecimientos. Este trabajo ha sido financiado por CONACYT con el número de beca de doctorado 362674, así como por el proyecto de investigación MINECO-FEDER con número TEC2015-65750-R.

Referencias

1. Bourzac, Katherine: Neural Networks on the Go. IEEE Spectrum (2016)
2. Draghici, S.: Neural Networks in Analog Hardware - Design and Implementation Issues. International Journal of Neural Systems, vol. 10, no. 1, 19–42 (2000)
3. Horn, G.van der J., Huijsing, L.: Integrated Smart Sensors: Design and Calibration. Kluwer Academic Publishers (1998)
4. Graude, D.: Principles of Artificial Neural Networks. Advanced Series on Circuits and Systems, vol. 6., 2nd. Edition. World Scientific Publishing Company (2007)

5. Sanz, M. T., Celma, S., Calvo, B.: Using MOS Current Dividers for Linearization of Programmable Gain Amplifiers. International Journal of Circuit Theory and Applications, vol. 36, no. 4, 397–408 (2008)
6. Lopez-Martin, A.J., Ramirez-Angulo, J., Carvajal, R.G., Algueta, J.M.: Compact Class AB CMOS Current Mirror. Electronics Letters, vol. 44, no. 23, 1335–1336 (2008)
7. Beale, M. H., Hagan, M. T., Demuth, H. B.: MATLAB Neural Network Toolbox: User's Guide. The MathWorks, Inc. (2015)