Contenido

Editorial

Durante los últimos dos años, el CINTEC ha venido generando publicaciones en lo referente a las Tecnologías de Punta y su aplicación práctica. Para ello, nuestra revista ha intentado reflejar un equilibrio, desde el punto de vista del enfoque de los artículos, mediando entre la teoría y la práctica.

Desde su origen, el objetivo de **polibits** ha sido la divulgación científica y tecnológica, constituyéndose en el órgano de difusión del CINTEC. Sin embargo, no consideramos que dicho medio deba ser un escaparate exclusivo para los desarrollos propios del Centro; debe también incluir participaciones externas, de tal manera que su horizonte sea cada vez más amplio y diversificado.

Este afán de expansión no responde a un capricho o moda pasajera, sino a una necesidad real de ampliar nuestro universo de lectores, convirtiendo a la revista en algo más que un simple vehículo de información técnica, estableciendo una aportación permanente tanto a la comunidad científica y tecnológica como a la industria. De este modo, buscamos cumplir las metas originales de creación de este Centro, lo mismo para la formación de recursos humanos de alta calidad y competitividad que para el desarrollo de soluciones tecnológicas que impacten en corto plazo a la industria nacional, proporcionando soluciones viables (en costo y recursos) a problemas reales.

Por todo lo anterior, extendemos una invitación sincera a todos aquellos que deseen participar en este esfuerzo, para que nos envíen su material y comentarios, esperando sus aportaciones cfon el fin de elevar la calidad de nuestra revista.

Diseño de Circuitos Lógicos en base a la tecnología FPGA: Un ejemplo de aplicación (Compendio)

M. en C. Juan C. González Robles
Jefe del Departamento de Producción y Adecuación de Tecnologías del CINTEC-IPN.
Ing. Eduardo Vega Alvarado
Jefe del Departamento de Laboratorios Ligeros del CINTEC-IPN.

Continuando con la serie de artículos sobre el diseño en base a la lógica programable, el presente trabajo pretende describir una aplicación diseñada y construida con un arreglo de compuertas programables en campo (Field Programmable Gate Array, FPGA).

Introducción

Para el presente desarrollo servirá como base teórica el artículo "Arreglos de Compuertas Programables en Campo, FPGA's (compendio)" **[1]**, en el cual se describen las principales variantes comerciales en relación a estos dispositivos.

Si bien el dispositivo seleccionado es del tipo de Arreglo de Celdas Lógicas (Logic Cell Array, LCA), los criterios de diseño y las herramientas de programación aplicadas son, en lo general, similares a los correspondientes a otras familias de dispositivos FPGA. La razón de esta selección es que los LCA tienen una gran aceptación como alternativa práctica de los tradicionales PLDs, y en algunos casos, como reemplazo para arreglos de compuertas simples. Para este diseño, primeramente se hará una breve descripción de los dispositivos LCA, para posteriormente manejar el tema de implantación.

Arquitecturas LCA

Existen tres familias de LCAs de uso común, que son las series 2000, 3000 y 4000 de Xilinx Incorporation; así mismo, se cuenta con dispositivos similares provenientes de segundas fuentes , tales como AT&T.

La serie 2000 de Xilinx

El Xilinx XC2064 es el dispositivo más simple de esta serie; contiene un total de 64 CLBs distribuidos en una matriz de ocho por ocho. En los LCAs de la serie 2000, cada CLB contiene una sección de lógica combinacional que es capaz de producir cualquier función de hasta cuatro variables, contando para ello con cuatro entradas de propósito general: A, B, C, y D, y una entrada especial de reloj (K). La sección lógica combinacional del CLB se maneja como una tabla de consulta, similar a una PROM.

El retardo a través de la celda lógica es constante, sin importar la función lógica que se maneje. Además , el CLB puede ser configurado para usarse como dos funciones lógicas de tres entradas, o en aplicaciones de multinivel, incluyendo algunas funciones de hasta cinco entradas mediante una operación de multiplexaje; ambos modos se

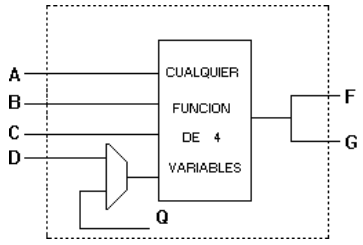


Figura 1
Función con una sola salida

muestran en las **figuras 1 y 2** :

El flip-flop de cada CLB puede

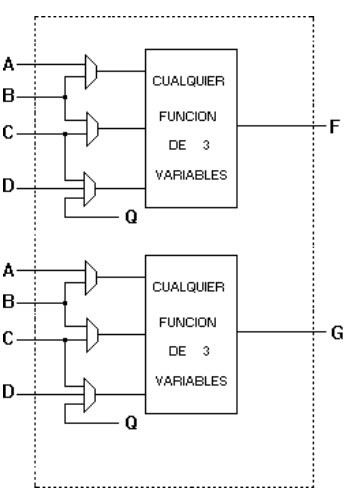


Figura 2
Función con dos salidas

simular un tipo D disparado por flanco o un latch por nivel; cuando no se usa como elemento de alma-

cenamiento se deshabilita. Cada elemento incluye funciones no sincrónicas de puesta y limpiado (set y reset, respectivamente).

También se cuenta con 58 bloques de Entrada/Salida ("Input/Output", I/O) para el XC2064, colocados en la periferia del dispositivo; estos bloques proveen la interface entre las terminales externas y la lógica interna. Cada bloque puede configurarse para operaciones como entrada, salida, salida en tercer estado, o buffer bidireccional. En la **figura 3** se muestra la arquitectura general de los bloques de I/O; la entrada contiene un circuito de detección de umbral para trasladar señales externas o niveles lógicos internos; este circuito permite

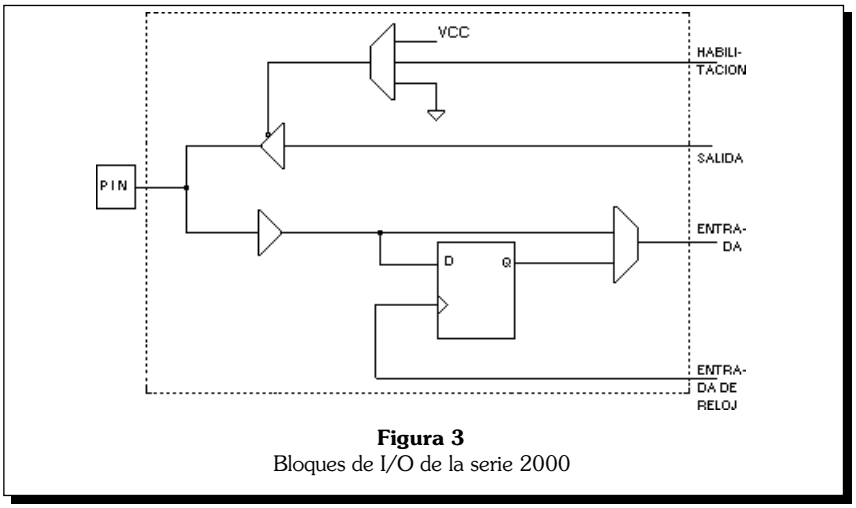


Figura 3
Bloques de I/O de la serie 2000

al dispositivo operaciones con niveles TTL o CMOS.

Los canales de interconexión programables del LCA proveen la trayectoria para conectar entradas y salidas de los bloques de I/O y las celdas lógicas en toda la configuración. Estos canales son programados en los puntos de cruce llamados *puntos de interconexión programable,* clasificándose en tres tipos: a) interconexiones de propósito general, b) líneas distantes, c) interconexiones directas.

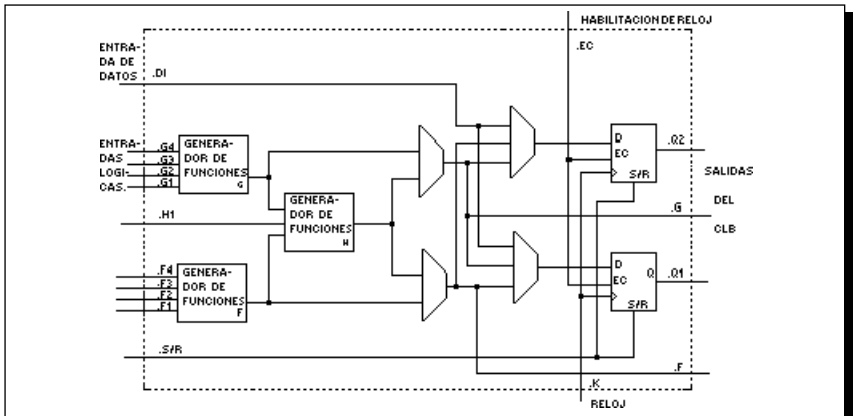


Figura 4
CLB de la serie 4000

La serie 3000 de LCAs.

Esta serie difiere de la serie 2000 en la complejidad de sus CLBs e IOBs, así como por la adición de un bus interno (especializado en líneas de gran longitud) para rutas de señales multiplexadas a través del dispositivo. El CLB contiene dos flip-flops, y tiene más líneas de entrada que las presentes en la serie 2000; cinco de estas entradas pueden aplicarse como entradas combinacionales, mientras las restantes se emplean para el reset y el reloj de los flip-flops, y para funciones de carga de datos. Los bloques de I/O incluyen flip-flops tipo D como registros de salida, que pueden trabajar alternativamente como latches de entrada.

La serie 4000 de LCAs.

Esta serie fue anunciada en 1989, está basada en RAM estática conteniendo CLBs y varios tipos de recursos de interconexión. Se considera que las mejoras en la arquitectura de ruteo y en el proceso incrementan la velocidad de operación de las serie 4000 al doble de la disponible en la serie 3000. El uso de un número significativamente mayor de recursos de interconexión contribuye, a su vez, a una mayor eficiencia en el ruteo de los dispositivos de esta serie.

Los CLBs en esta serie son similares a los de la serie 3000 (**figura 4**); la diferencia es que cuenta con dos bloques de lógica combinacional con entradas separadas, con lo que un solo CLB puede implementar funciones lógicas de hasta nueve entradas.

Proceso de diseño y programación del FPGA

Conceptualmente hablando, el transformar un diseño en un circui-

to basado en LCA que funcione apropiadamente es un proceso considerablemente más largo y laborioso que el desarrollo equivalente utilizando PLDs. La primera consideración es determinar que tan adaptable resulta el diseño inicial a una implementación con LCAs. Una vez tomada la determinación de aplicar esta tecnología, la siguiente etapa involucra el empleo de una herramienta de diseño proporcionada por el fabricante. Para el caso de Xilinx, este sistema recibe el nombre de Xact, llevándose a cabo los siguientes pasos (los cuales se muestran en la **figura 5**):

1.- A partir del diagrama esquemático, se obtienen los archivos de formato de la lista de conexiones (XNF). Este formato es una simple representación de los diseños y no involucra aún el uso de un dispositivo específico.

2.- El formato XNF generado se maneja para crear un circuito optimizado dirigido al LCA en particular que se desea utilizar. En esta etapa es posible combinar varios archivos XNF para generar un solo circuito, así como efectuar varias optimizaciones al mismo. Finalmente se obtiene un archivo denominado de diseño LCA, el cual en realidad consiste en la descripción detallada para el Xact de lo que se desea implementar.

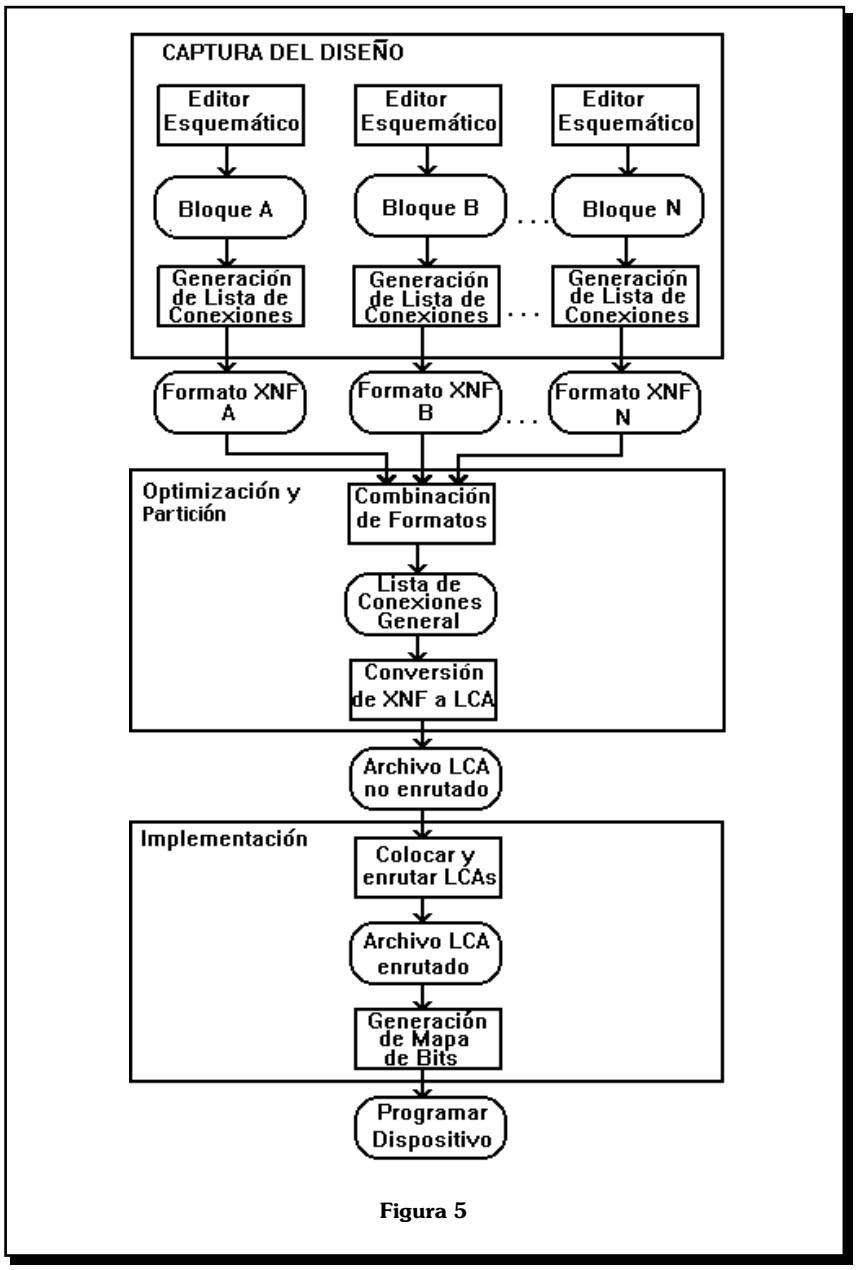


Figura 5

3.- Finalmente se pasa a la fase de implantación, en la cual el archivo de diseño para LCA se convierte en un conjunto de bits para configuración del dispositivo seleccionado.

La parte más importante en esta etapa es la colocación y enrutamiento de los elementos del circuito en el dispositivo. Para llevar a cabo la verificación final se cuenta con herramientas de simulación para comprobar tanto la operación del circuito LCA como su conexión física.

Como ya se mencionó, la primera etapa se puede desarrollar por medio de herramientas de propósito general, pero la segunda y tercera etapas (optimización e implementación) requieren manejar programación especial. Una vez descrito el proceso general, se describirá en detalle una aplicación real.

Descripción del Diseño

Para la descripción del diseño se pueden emplear herramientas para la generación de esquemáticos, tales como Orcad SDT, HiWire, y FutureNET, o bien programas que manejen descripciones de comportamiento, tales como el OPAL y PALASM2. Como ejemplo se tomará el esquemático mostrado en la **figura 6 [2]**, en la que

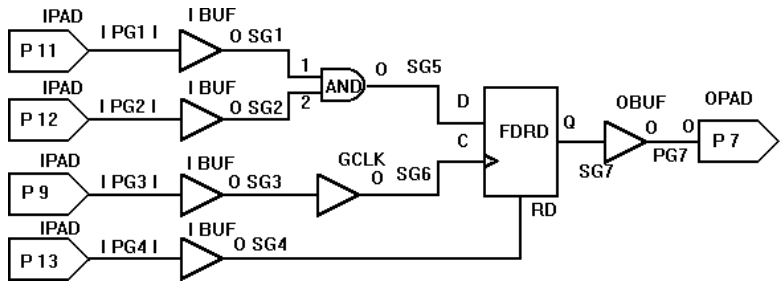


Figura 6

se tiene un flip-flop tipo D simple alimentado por una compuerta AND; como complemento, en el diagrama también se muestran los símbolos correspondientes al reloj interno del LCA y a los buffers de entrada y salida del mismo.

Una vez que se tiene el esquemático, se obtiene el archivo XNF. Para el ejemplo en desarrollo, dicho archivo es **[2]**:

```
LCANET, 1
PROG, PIN2XNF, 2.01, Created from ejemplo.pin
PART, 2064PC68-1
SYM, SG5, AND
PIN, O, O, SG5
PIN, 2, I, SG2
PIN, 1, I, SG1
END
SYM, SG1, IBUF
PIN, O, O, SG1
PIN, I, I, PG1
END
SYM, SG2, IBUF
PIN, O, O, SG2
PIN, I, I, PG2
END
SYM, SG4, IBUF
PIN, O, O, SG4
PIN, I, I, PG4
END
SYM, PG7, OBUF
PIN, O, O, PG7
PIN, I, I, SG7
END
SYM, SG3, IBUF
PIN, O, O, SG3
PIN, I, I, PG3
END
SYM, SG6, GCLK
PIN, O, O, SG6
PIN, I, I, SG3
END
SYM, SG7, DFF
PIN, RD, I, SG4
PIN, Q, O, SG7
PIN, D, I, SG5
PIN, C, I, SG6
END
EXT, PG1, I, , LOC=P11, BLKNM=PG1
EXT, PG2, I, , LOC=P12, BLKNM=PG2
EXT, PG3, I, , LOC=P9, BLKNM=PG3
EXT, PG4, I, , LOC=P13, BLKNM=PG4
EXT, PG7, O, , LOC=P7, BLKNM=PG7
EOF
```

A partir del archivo XNF, la siguiente etapa contempla una optimización del mismo, empleando para ello las herramientas adecuadas tales como el optimizador de circuitos de Xilinx (XNFOPT). Dado que el ejemplo es bastante simple no se hace necesario realizar esta etapa.

Posteriormente el archivo optimizado se convierte a un formato intermedio, conocido como MAP. Este archivo representa un circuito seccionado, donde cada una de las secciones corresponde a un CLB o a un IOB del dispositivo a emplear. Una vez que se obtiene el archivo MAP este puede convertirse a un archivo de diseño LCA. El archivo LCA no es otra cosa mas que una lista de instrucciones que dirigen al Xact durante los procesos de colocación y enrutamiento sobre el dispositivo físico. El siguiente listado muestra el archivo LCA del ejemplo **[2]**.

```
;: small.lca, MAP2LCA 2.20
Design 2064PC68
Speed -1
Addnet SG7  AA.X P7.O
Addnet SG6  AA.K CLK.AA.O
Addnet SG4  AA.D P13.I
Addnet SG3  P9.I CLK.AA.I
Addnet SG2  AA.A P12.I
Addnet SG1  AA.B P11.I
Nameblk AA SG7
Editblk AA
BASE FG
CONFIG X:Q Y:  Q:FF SET:
RES:D CLK:K:  F:A:B
EQUATE F=(A*B)
Endblk
Nameblk P9 PG3
Editblk P9
BASE IO
CONFIG I:PAD BUF:
Endblk
NameblkP7 PG7
Editblk P7
BASE IO
CONFIG I: BUF:ON
Endblk
Nameblk P13 PG4
Editblk P13
BASE IO
CONFIG I:PAD BUF:
Endblk
Nameblk P12 PG2
Editblk P12
BASE IO
CONFIG I:PAD BUF:
Endblk
Nameblk P11 PG1
Editblk P11
BASE IO
CONFIG I:PAD BUF:
Endblk
```

Implantación final

El archivo de diseño LCA se alimenta al programa APR ("Automatic Place and Route"), generándose así un nuevo archivo que incluye la colocación de los bloques lógicos y el enrutamiento de señales. Como su nombre lo indica, el APR calcula automáticamente la colocación y el enrutamiento, y si estos son correctos se obtiene otro archivo formado por un patrón de bits para programar al dispositivo; en caso de que se detecten errores en el archivo de diseño, Xact permite su edición para corregir las deficiencias.

```
:  0200  0002  0000  FC
:  1000  0000  FF04  400F  FBEE  FFFF  FFFF  FFFF  FFFF  B7FD  09
:  1000  1000  FDFB  FFFB  F7F7  FFBF  FFFF  FFFF  FFFF  FFFF  4A
:  1000  2000  FFFD  FFFF  FFFF  FFFF  FFFF  EFFF  FDFB  FBFB  00
:  1000  3000  F7F7  F77F  FFFF  FFFF  FFFF  FFFF  FFFB  FFFF  6C
:  1000  4000  FFFF  FFFF  FFFF  DFFF  FFFF  FFFF  FFFF  FFFF  E0
:  1000  5000  FEFF  FFFF  FFFF  FFFF  FFF7  FFFF  FFFF  FFFF  B9
:  1000  6000  FFFF  BFFF  FFFF  FFFF  FFFF  FFFF  FDFF  FFFF  E2
:  1000  7000  FFFF  FFFF  FFEF  FFFF  FFFF  FFFF  FFFF  7FFF  20
:  1000  8000  FFFF  FFFF  FFFF  FFFF  FBFF  FFFF  FFFF  FFFF  84
:  1000  9000  FFDF  FFFF  FFFF  FFFF  FFFF  FFFE  FFFF  FFFF  91
:  1000  A000  FFFF  FFFF  F7FF  FFFF  FFFF  FFFF  FF3F  FFFF  28
:  1000  B000  FFFF  FFFF  FFFF  DFFD  FFFF  FFFF  FFFF  FFFF  72
:  1000  C000  EFFF  FFFF  FFFF  FFFF  FF7F  FFFF  FFFF  FFFF  D0
:  1000  D000  FFFF  FFFB  FFFF  FFFF  FFFF  FFFF  9FFF  FFFF  94
:  1000  E000  FFFF  FFFF  FFEF  FCFF  FFFF  FFFF  FFFF  7FF7  BB
:  1000  F000  FFFF  FFFF  FFFF  FFFF  BFFF  FFFF  FFFF  FFFF  50
```

Contando con el patrón de bits, la configuración del LCA no presenta mayores dificultades; únicamente debe tomarse en cuenta que los LCA son dispositivos de tipo volátil, por lo que si se desea dejar permanente la programación es necesario almacenar ésta en una memoria ROM conectada al LCA en cuestión. A continuación se muestra el archivo del patrón de bits del ejemplo en formato codificado para una memoria ROM **[2]**.

Bibliografía

[1] González Robles Juan C., Vega Alvarado Eduardo. "Arreglos de compuertas programables en campo, FPGA's. Revista POLIBITS No. 12, Enero-Marzo 1994.

[2] David Pellerin/Michael Holley. PRACTICAL DESIGN USING PROGRAMMABLE LOGIC. . Prentice Hall, 1991.

[3] Edwards J. Mc. Cluskey. LOGIC DESIGN PRINCIPLES WHIT EMPHASIS ON TESTABLE SEMICUSTOM CIRCUITS. Prentice Hall, 1986.

[4] DESIGNING WHIT HIGH PERFORMANCE ASICS. Di Giacomo. Prentice Hall, 1991.

Simulación de Redes Neuronales. Memoria Bidireccional Bivalente Adaptiva: BAM

M.C. Miguel A. Partida Tapia
Subdirector Académico y de Investigación del CINTEC-IPN.
Ing. Rubén Peredo Valderrama e
Ing. Francisco F. Córdova Quiróz
Alumnos de la Maestría del CINTEC-IPN.

El presente artículo tiene como finalidad que el lector se familiarice con los conceptos sobre Redes Neuronales y como ejercicio simular el modelo básico propuesto por Bart Kosko [1988],una memoria bidireccional asociativa (**BAM**) en el lenguaje C++ orientado a objetos de Borland, versión 4.02 para Microsoft Windows 3.11.

Introducción

Tratando de superar los problemas y limitantes de los métodos y técnicas para simular Redes Neuronales, se han abordado y utilizado nuevos enfoques. Entre ellos están:

1. La Teoría de Conjuntos Difusos (Fuzzy Sets), desarrollada por Lofti Zadeh en 1965.

2. Los Autómatas Celulares (descendientes de la Teoría de Autómatas, pero aplicando muchos de los conocimientos de los cultivos celulares y su crecimiento).

3. Los Algoritmos Genéticos (desarrollados por John Holland en 1975, que consisten en una mímica de los procesos naturales de mutación, evolución, y selección natural, para la solución de un problema).

4. La Dinámica Matemática (también llamada *Ciencia del Caos* **[9]**, y que se encarga del estudio de la Dinámica no Lineal).

Sin embargo, el enfoque que ha resultado ser el más poderoso y prometedor, y que posee sólidas bases en la biología, en la fisiología y, particularmente, en las neurociencias, es el conocido como Redes Neuronales. Como su nombre lo indica, se trata de estructuras artificiales que simulan el comportamiento de las neuronas naturales.

Los inicios se dieron con los trabajos de Santiago Ramón y Cajal, realizados a finales del siglo XIX. Estos nos permitieron descubrir los elementos que componen al cerebro: las **neuronas**. Su funcionamiento fue descrito posteriormente por Sherrington, quien también identificó a las **sinapsis**.

En 1943, Warren McCulloch y Walter Pitts publicaron un modelo matemático sobre el comportamiento de las neuronas. El impacto que causó, desencadenó una gran actividad en el estudio de la lógica neuronal. Como resultado de todo esto surgieron los **Perceptrones**, y en él *demostraban* la incapacidad de estos sistemas para la solución de problemas lógicos sencillos (más tarde se comprobó que el problema no había sido estudiado tan a fondo como se pretendía y que los resultados de Minsky y Papert eran solo particulares). Debido a esto, durante mucho tiempo, las investigaciones en redes neuronales permanecieron en el olvido.

A principios de la década de los 80's, y gracias a los avances logrados en el procesamiento paralelo, surge el **Conexionismo**, como antecedente de la actividad actual en redes neuronales. El impulso inicial fue dado por los trabajos de John Hopfield quien retomó los arreglos neuronales demostrando que no habían sido estudiados lo suficiente por Minsky y Papert, y probó que estos sistemas tienden a generar procesos exponenciales que pueden ser fácilmente definidos y explicados por funciones no lineales.

Bastaron sólo 5 años para que se formara una sólida sociedad internacional para el avance de las redes neuronales. En este periodo salieron a la luz muchos trabajos de investigadores que habían trabajado en el anonimato o en la clandestinidad. Igualmente, los avances logrados se vieron enriquecidos por los conocimientos sobre el sistema nervioso central y el proceso de aprendizaje obtenidos por investigadores como Werbs, Widrow y Hebb.

Los conocimientos y técnicas aportados por la recién nacida Dinámica Matemática también han sido un factor decisivo para el análisis y formalización de los sistemas estudiados. Ahí, donde antes estaba el caos, se ha encontrado orden, en donde se veía aleatoriedad, se ha localizado regularidad. Se intuía la existencia de un determinado orden en el cerebro, más las caóticas señales de las neuronas hacían difícil su identificación . La *Ciencia del Caos* nos ha brindado conocimientos sobre los procesos concurrentes y recursivos; comportamiento no lineal, puntos de atracción y procesamiento caótico de señales.

La Inteligencia Artificial y las Redes Neuronales son dos campos diferentes con un mismo propósito: reproducir las habilidades cognoscitivas de humanos y animales mediante máquinas y dispositivos. Con todo y los logros obtenidos en Inteligencia Artificial, en la actualidad las aplicaciones de las redes neuronales son muy diversas y abarcan diferentes campos, usándose exitosamente en: diagnóstico médico, aprobación de crédito y seguros, control e inspección industrial, modelamiento económico, procesamiento de señales y filtrado de ruido, mercadotecnia, predicción bursátil y elaboración de presupuestos, detección de fraudes, reconocimiento de formas, generación de voz, identificación por radar o sonar, corrección ortográfica, desciframiento y recuperación de información.

Redes Neuronales

Descripción de las redes neuronales.

Las Redes Neuronales son modelos matemáticos que procesan en forma paralela información como lo realizan los sistemas biológicos mediante elementos simples llamados neuronas. La simulación de las Redes Neuronales es la experimentación con modelos, típicamente implementados en computadoras. La simulación por computadora de las redes neuronales tiene dos distintas aplicaciones:

1.- Estudio de modelos de sistemas biológicos.

2.- Estudio de redes neuronales artificiales usados para procesos computacionales y de control.

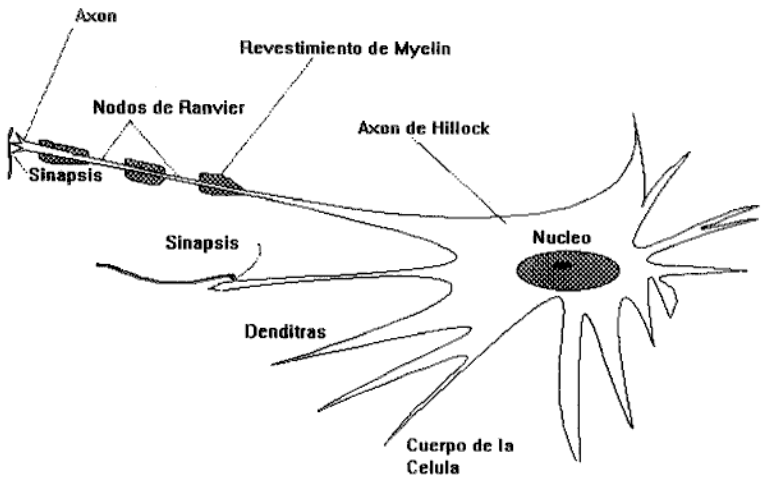


Figura 1

Anatomía de una neurona típica. La mayoría de las dendritas son del orden de 1mm de longitud, mientras que algunos axones encontrados han alcanzado hasta 1m. El axon muestra aquí un recubrimiento de Mylin, el cual previene los intercambios de iones excepto en el Nodo de Ranvier. La propagación del pulso puede alcanzar hasta 120 m/seg .

Circuitos neurales y su representación

Las estructura de las Redes Neuronales se define como una colección de *procesadores paralelos* conectados entre sí organizados como gráficas dirigidas, conformando una RED. Cada nodo representa un Elemento de Procesamiento (PE), conectado por medio de arcos. En la **figura 2** se muestran algunos tipos de redes.

Los elementos en las redes que quedan entre los PE de entrada y la salida se le llaman *niveles escondidos* (**hidden-layer**) y las conexiones de entrada y salida son definidas por *conexiones pesadas* (**weighted connections**).

En la **figura 2** las gráficas (a) y (b), ilustran los principios de divergencia y convergencia en un circuito neural. Cada neurona envía impulsos a muchas otras neuronas (*divergencia*), y recibe impulsos de muchas otras (*convergencia*). Las gráficas (b), (c), y (d) contienen retroalimentaciones. Las conexiones sinápticas pueden ser exitatorias e inhibitorias, estos circuitos facilitan sistemas de control ya que se pueden realizar retroalimentaciones positivas y negativas.

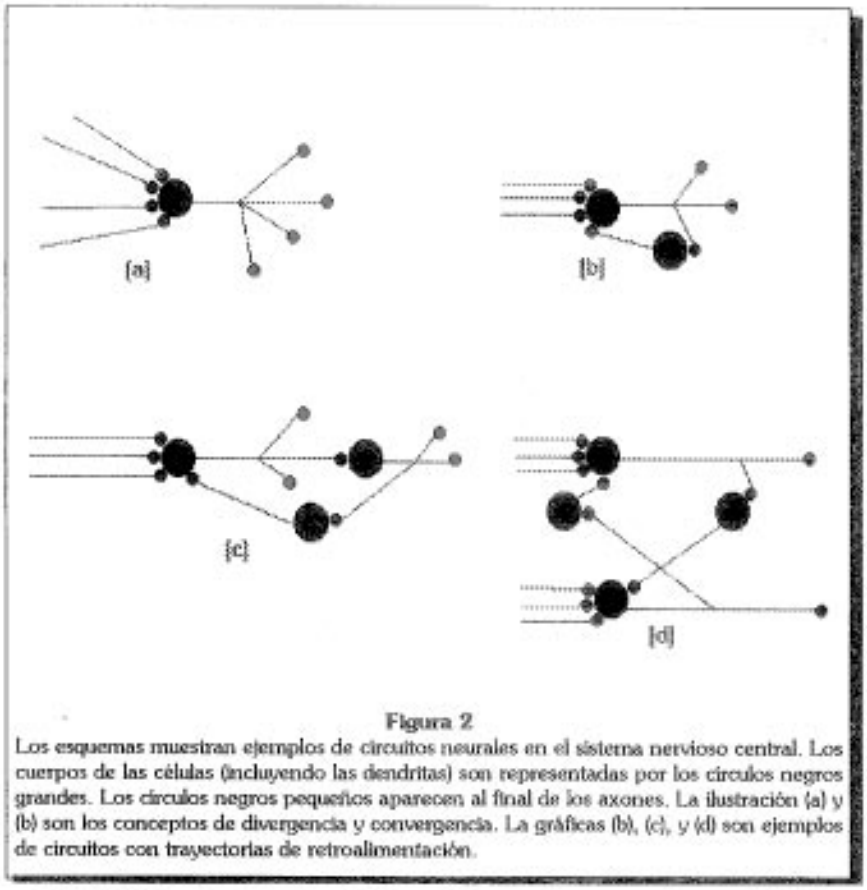


Figura 2

Los esquemas muestran ejemplos de circuitos neurales en el sistema nervioso central. Los cuerpos de las células (incluyendo las dendritas) son representadas por los círculos negros grandes. Los círculos negros pequeños aparecen al final de los axones. La ilustración (a) y (b) son los conceptos de divergencia y convergencia. La gráficas (b), (c), y (d) son ejemplos de circuitos con trayectorias de retroalimentación.

En 1943, los trabajos de McCulloch y Pitts establecieron cinco reglas fundamentales para el estudio de las redes neuronales como son:

1. La actividad de una neurona siempre está disponible en el proceso.

2. Un cierto número fijo de sinapsis (> 1) podrán ser exitadas en un período de latencia adicional para una neurona dada.

3. El único retardo significativo en un sistema nervioso es el retardo sináptico.

4. La actividad de cualquier sinapsis inhibitorio absolutamente previene excitación de una neurona en un tiempo dado.

5. La estructura de la interconexión de la red no cambia con el tiempo.

Asumiendo que un 1 identifica la neurona como un principio binario: Esto implica una razón *on/off*. Para definir la notación, indicaremos como $N_i(t)$, lo cual denota que la neurona i_{esimo} se dispara en un tiempo t. la notación, $!N_i(t)$, denota que la neurona i_{esimo} no se disparará en el tiempo t. Usando esta notación, nosotros podemos describir la acción de ciertas redes neuronales usando proposiciones lógicas.

En la **figura 3** (siguiente página) se muestran cinco redes simples. A continuación se describe la notación para cada uno de los ejemplos, en la figura (a) describe a dos neuronas, donde la neurona 2 se dispara después de la neurona 1.

Las expresiones quedan como sigue:

$N2(t) = N1(t-1)$ (a)

$N3(t) = N1(t-1) \vee N2(t-1)$ (b)
(Disyunción)

$N3(t) = N1(t-1) \,\&\, N2(t-1)$ (c)
(Conjunción)

$N3(t) = N1(t-1) \,\&\, !N2(t-1)$ (d)
(Conjunción negada)

Una prueba palpable de esta teoría fue que cualquier red que no contenga conexiones de retroalimentación puede describirse en términos de combinación de estas simples expresiones, y viceversa.

Elemento de procesamiento general

Los elementos computacionales simples que representan a un sistema neural son llamados *Neuronas Artificiales*, estos están referidos a nodos, unidades, o elementos de procesamiento (PE). En la **figura 4** (siguiente página) se muestra un modelo de PE. Cada uno de los PE es numerado, del 1 hasta i_{esimo}

De igual manera que en las neuronas biológicas, los PE tienen muchas entrada y una sola salida. Cada entrada es representada como x_j, y a cada una de estas se le asocia un ***peso*** o fuerza de ***conexión***. Los pesos o conexiones definen de donde provienen j_{esimo} nodo hasta el nodo i_{esimo} y es representado como w_{ij}. La salida PE corresponde a la frecuencia de disparo de la neurona, y los pesos corresponden a la fuerza de las conexiones sinápticas entre neuronas. En este modelo, estas cantidades pueden ser representadas como números reales.

Un PE puede aceptar varios tipos de entradas, dado que cada una de ellas realizan diversos efectos. Una conexión de entrada puede ser exitatoria o inhibitoria, por ejemplo. Para nuestro propósito de estudio, definiremos a las conexiones exitatorias con ***pesos positivos*** y a las conexiones inhibitorias con ***pesos negativos***. Otros tipos pueden ser posibles. Cada PE determina una

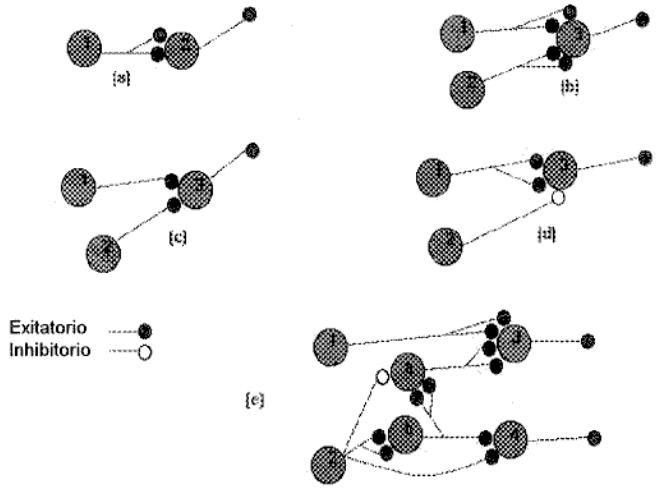


Figura 3

En la figura se muestran ejemplos de redes simples expuestas por McCulloch-Pitts que pueden ser definidas en términos de la notación proposicional lógica. Los círculos grandes indican células nerviosas, los círculos pequeños llenos representan conexiones exitatorias, los vacíos conexiones inhibitorias. La red (a) ilustra precedencia, (b) disyunción, (c) conjunción, y (d) conjunción negada. La red (e) es una combinación de las anteriores redes (a) - (d).

red de entradas en base a todas las conexiones de entrada. En ausencia de conexiones especiales, nosotros calculamos típicamente la red de entradas por suma de todos los valores de entrada, multiplicado cada uno de ellos por su correspondiente peso. En otras palabras, la red de entradas de i_{esima} describe la siguiente expresión:

$$net_i = \sum_j x_j w_{ij} \qquad (1)$$

donde el índice, *j*, recorre todas las conexiones del PE.

Note que la excitación y la inhibición son limitadas automáticamente por el signo de los pesos. El cálculo de la suma de productos juega una regla importante en la simulación de las redes neuronales. Por el gran número de interconexiones en las redes, la velocidad del cálculo determinará el rendimiento de la simulación de una red neuronal dada.

Cada una de las redes de entrada es calculada, y convertida a un *valor de activación* o simplemente *activación* para PE. Podemos, entonces, escribir el valor de activación como:

lo cual denota que la activación está implícita en la función de la red de entrada.

Modelos Aditivos Bivalentes

La activación de modelos aditivos discretos corresponden a neuronas con funciones de señales de umbral. Las neuronas pueden asumir únicamente dos valores, **ON** y **OFF**. **ON** representa al valor de la señal +1. **OFF** representa 0 ó -1. Estos valores, **bivalentes**, corresponden a modelos de redes neuronas contenidas en el modelo clásico de McCulloch y Pitts [1943].

Los modelos bivalentes pueden representar comportamientos asíncronos y estocásticos. Estas propiedades ofrecen valores prácticamente pequeños en simulaciones a pequeña escala. Existen dos tipos de redes neuronales bivalentes aditivos: la no adaptiva, la memoria asociativa adaptiva bivalente bidireccional (**BAM**) y la autoasociativa o modelo de **Hopfield**.

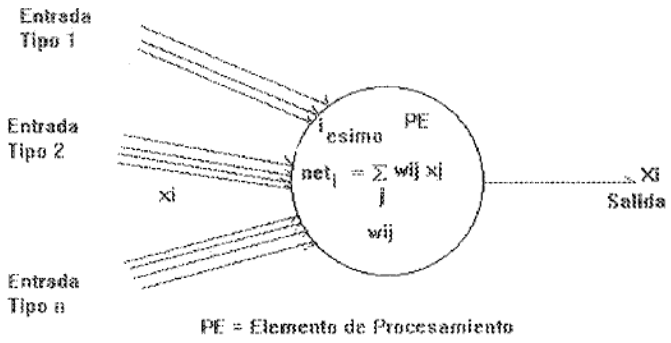


Figura 4

Esta estructura representa un elemento simple de procesamiento (PE) en una red. Las conexiones son representadas por flechas desde otros elementos de procesamiento. Cada una de las conexiones de entrada están asociadas con una cantidad, $\mathbf{w_{ij}}$, llamada **pesos**. Estos elementos tienen una única salida, la cual puede ser conectada en **abanico** con otras unidades.

Modelos Existentes

Existen varios tipos de redes neuronales; en la actualidad su número llega a sobrepasar los cuarenta modelos. Algunos de estos son sencillos y muy limitados, mientras que otros son sumamente flexibles y poderosos. Para cada rama en la clasificación mostrada existe un modelo que la caracteriza. Como se describen a continuación.

- **Modelo de Hopfield.** Con base a los trabajos de Shunichi Amari, de 1977, Jonh Hopfield presentó en 1982 un modelo sumamente sencillo de una red recurrente de tipo discreto, y que se ha constituido como modelo canónico de las de las redes neuronales con retroalimentación. Este modelo asocia el patrón presentado consigo mismo, y permite su posterior recuperación al presentar un patrón completo o impreciso. En estudios subsecuentes de Hopfield y otros investigadores se han presentado variantes continuas, estadísticas y generalizadas. Las principales características de este modelo son: generalización, estabilidad, convergencia de estados y autoorganización.
- **Memoria Asociativa Bidireccional.** Este modelo, identificado comúnmente como **BAM**, por sus siglas en inglés, constituye una extensión generalizada de las redes de Hopfield. Fue presentada por Bart Kosko en 1988. El modelo permite la asociación de patrones diferentes, a manera de una memoria hetereoasociativa. Para la **BAM**, las redes de Hopfield constituyen un caso especial dentro del conjunto de datos con los que se trabaja, por lo que ésta exhibe características de este modelo y otras adicionales.
- **Modelo ART.** (Adaptive Resonance Theory). Este modelo es muy complejo y se emplea más como objeto de investigación. Surgió como resultado de los trabajos de Gail Carpenter Stephen Grossberg realizados entre 1986 y 1987 sobre modelación de los procesos macroscópicos de la actividad cerebral. Su funcionamiento parte de la convergencia de las señales hacia patrones estables por la reverberación de las señales en los elementos de procesamiento. El modelo es capaz de proporcionar abstracción y generalización.
- **Asociador Lineal.** Utiliza la regla de aprendizaje de Hebb. El modelo sólo funciona si los patrones son ortogonales y su capacidad es de apenas un 10% a 20% con respecto al número de neuronas presentes.
- **Propagación Inversa.** (Backpropagation). El modelo parte del algoritmo publicado por Rumelhart, Hilton y Williams en 1985, aunque mucho de su trabajo ya había sido anticipado por Parker en 1982 y anteriormente por Werbos en 1974. Las redes neuronales de propagación inversa son netamente multiestrato; éstas se modifican a sí mismas en el mismo instante en que generan patrones de salida, por lo que resultan sumamente flexibles y de gran poder de generalización. Su entrenamiento es cíclico y consume demasiado tiempo y recursos de cómputo.
- **Neocognitron.** Esta es la versión de 1980 del Cognitron, ambos desarrollados por Kunihiko Fukushima, quien también publicó una versión supervisada en 1983. Este modelo es el resultado de las investigaciones de Fukushima sobre los órganos visuales en gatos y macacos. El Neocognitron es una red neuronal cuyo funcionamiento radica en su algoritmo de aprendizaje, estructura de las capas y los diversos tipos de conexiones manejados. En conjunto, la organización del modelo se asemeja mucho a la del sistema visual de pequeños mamíferos. El Neocognitron provee las capacidades de su antecesor y muchas más, puede reconocer patrones eliminando translación, rotación, distorsión, cambios de escala, y puede seleccionar entre dos patrones presentados simultáneamente, proporcionando una gran capacidad de generación. Computacionalmente hablando, el modelo resulta excesivamente costoso en tiempo y espacio.
- **Modelo de Kohonen.** Teuvo Kohonen ha propuesto a través de su obra un modelo no supervisado de memoria asociativa. En este caso el grupo de neuronas que conforman la red se comportan en la forma de "El ganador toma todo", en una arquitectura de un solo estrato.

Estructura del la BAM

La estructura de la **BAM** se compone de 2 estratos, identificados como los grupos de neuronas $\{a_1,a_2,\ldots,a_n\}$ y $\{b_1,b_2,\ldots,b_n\}$. En cada estrato, cada neurona **a** se encuentra conectada con cada neurona **b** y viceversa. Las **n x p** conexiones generadas conforman una matriz denominada ***matriz de correlación***, representada por $\boldsymbol{W}$. Por el método utilizado en la construcción de $\boldsymbol{W}$ todas las conexiones sinápticas son consideradas de la misma longitud y de tipo exitatorio. Estas conexiones satisfacen nuestras necesidades de proceso y de almacenamiento de información siendo suficientes para explotar las características de estabilidad y bidireccionalidad de la matriz de conexiones. Por no ser necesarias, se

omiten otro tipo de conexiones (de retroalimentación inhibitorias y de retardo) y de funciones para el procesamiento de las señales.

Por la sencillez del modelo, el comportamiento de la neurona de la **BAM** es síncrono. Este no corresponde del todo al comportamiento de las neuronas biológicas, asíncrono por naturaleza. Aunque, si se desea hacer una comparación, puede llegar a equiparse con el comportamiento de las neuronas cuando, en grupos pequeños, son capaces de sincronizar sus impulsos.

Funcionamiento y construcción de la BAM

La agrupación de los valores que asumen en un mismo instante todas las neuronas en cada estrato se puede tratar como una ***cadena de bits*** representando un ***patrón***, o como un ***vector*** indicando el ***estado*** del sistema. Las combinaciones posibles de vectores o cadenas forman dos conjuntos; $\{A_1, A_2, ..., A_n\}$, para las neuronas **a**, y $\{B_1, B_2, ..., B_n\}$, para las neuronas **b**. Así, los elementos A_i y B_i provienen de los espacios $\{0,1\}^n$ y $\{0,1\}^p$, respectivamente.

La ***asociación*** de dos vectores se establece durante su almacenamiento, y se representa como el par vectores, $A_i^T B_i$, lo cual dará por resultado una matriz que será agregada a W mediante una operación de adición que se hará para todos y cada uno de los pares a asociar.

$$W = \sum_k^m A_k^T B_k \qquad (2)$$

El algoritmo de almacenamiento por correlación constituye una aproximación discreta del aprendizaje Hebbiano. Podemos ver que cada neurona constituye una memoria a corto plazo, vigente en el momento de presentación o recuperación de la información. La memoria a largo plazo es formada por las conexiones, que son reforzadas o debilitadas por las neuronas involucradas.

Si los valores en la matriz de conexiones representa la resistencia sináptica entre las neuronas, entonces es necesario tener en cuenta los valores inhibitorios de cada neurona para la correspondiente generación de conexiones inhibitorias. Por el carácter aditivo de las operaciones de almacenamiento de los 0's en los patrones binarios son ignorados; sin embargo, si empleáramos un código ***(bivalente)*** los estados inhibitorios sí serían tomados en cuenta. La razón principal reside en el hecho de que 1+0=1, pero 1+(-1)=0. Para esto deberemos transformar los vectores binarios a

$$Y_i = fc_B(B_i) \qquad (4)$$

$$fc_B = \left.\begin{matrix} y_k = 1, \rightarrow b_k = 1 \\ y_k = -1, \rightarrow b_k = 0 \end{matrix}\right\} \forall k = 1,2,....,p$$

y los principios de creación W continúan siendo válidos al utilizar X_i y Y_i, por lo tanto la ecuación de matriz de pesos queda como sigue:

$$W = \sum_k^m X_k^T Y_k \qquad (5)$$

El carácter bipolar de X_i y Y_i agrega características especiales al almacenamiento de las asociaciones. Primero, tenemos que para cada vector X_i existe un ***complemento*** X_i^c, tal que $X_i^c = -X_i$, y similarmente, $Y_i^c = -Y_i$. Si consideramos que $X_i^T Y_i = (-X_i^T)(Y_i) = (X_i^c)^T (Y_i^c)$, veremos que tanto los vectores como sus complementos crean la misma matriz de conexiones. Entonces el codificar (A_i, B_i), también guarda en la memoria a (A_i^c, B_i^c) y viceversa. Nótese que (A_i, B_i) puede ***borrarse*** de W almacenando $X_i^T Y_i^c = -X_i^T Y_i$, dado el carácter aditivo de W.

Una **BAM** se comportará como una red de Hopfield cuando $A = B$ y, siendo la **BAM** una generalización de este modelo, la presentación y recuperación deberá igualmente hacerse mediante procesos recu-

$$\frac{\Delta E}{\Delta b_k} = -AW^k \qquad (8)$$

$\Delta \mathbf{b}_k > \mathbf{0}$ sólo es posible si la entrada en la neurona $\mathbf{b}_k > \mathbf{0}$ ($\mathbf{AW}^k > 0$), y $\Delta \mathbf{b}_k < \mathbf{0}$ si y sólo si $\mathbf{AW}^k < \mathbf{0}$; así, en cualquier caso $\Delta \mathbf{E} = -\Delta \mathbf{b}_k (\mathbf{AW}^k) < 0$, cuando $\Delta \mathbf{a}_k = \mathbf{0}$, $\Delta \mathbf{b}_k = \mathbf{0}$, $\Delta \mathbf{E}_k = \mathbf{0}$.

Resumiendo, cualquier cambio en el vector **A** o en el vector **B**, en la recuperación, resulta en un decremento de **E** dada la concordancia de signos del producto ***vector-matriz*** y el cambio de estado en las neuronas y, todo cambio en **E** es por una cantidad finita, a través de trayectorias discretas en $\{0,1\}^n$ **X** $\{0,1\}^p$. Entonces la convergencia a una solución, y que consiste en determinar el límite de los decrementos en **E**. La máxima variación estará dada por un cambio de estado en todas las neuronas del estrato, dado en cantidades discretas de 1 ó -1, donde:

$$E(A,B) \geq -\sum_i^n \sum_j^p \left| w_{ij} \right| \quad \forall \mathbf{A},\mathbf{B} \qquad (9)$$

Dado que ***W*** se mantiene invariante en la recuperación, **E** se encuentra acotada, lo que nos asegura que eventualmente los vectores **A** y **B** dejarán de cambiar alcanzando un punto fijo ($\mathbf{A}_f$, $\mathbf{B}_f$) que es un ***mínimo local*** o un ***punto de atracción***, esto demostrará que el sistema es ***estable***.

Capacidad. Teorema de Saturación

Este teorema establece que los modelos aditivos se saturan mientras que esto no ocurre en los modelos multiplicativos. De acuerdo con esto, y al ser un modelo aditivo, la **BAM** debe presentar un límite en su almacenamiento al saturar los valores de sus conexiones.

La saturación y los límites en la capacidad de la **BAM** dependen de su dimensionalidad, y aquélla sólo es perceptible hasta que se efectúe la recuperación de uno de los patrones almacenados, ya que durante el almacenamiento no tenemos forma de saber si la adición de una asociación altera significativamente o borra otra. La recuperación de un vector está determinado por el respectivo producto ***vector-matriz*** (7). Dicho producto puede expanderse para recuperar a un $\mathbf{B}_i$, como:

$$A_i W = (A_i X_i^T) Y_i + \sum_{j \neq i}^{m} (A_i X_j^T) Y_j \qquad (10)$$

Así tomando (2,3 y 4), tendremos:

$$X_i W = (X_i X_i^T) Y_i + \sum_{j \neq i}^{m} (X_i X_j^T) Y_j$$

$$= nY_i + \sum_{j \neq i}^{m} (X_i X_j^T) Y_j$$

$$= \sum_j c_{ij} Y_j \qquad (11)$$

El factor ***n*** es el valor máximo que se puede obtener para amplificar las características de $\mathbf{Y}_i$. Los coeficientes de ***ruido***, dados por $\mathbf{X}_i \mathbf{X}_i^T$, pueden ayudar a corregir las distorsiones producidas por $\mathbf{Y}_i$ en grado proporcional en la similitud de $\mathbf{X}_i$ con $\mathbf{X}_j$. De manera similar podemos concluir que el factor de ***corrección de ruido*** en el caso de la recuperación de $\mathbf{A}_i$ es **p**. El factor de ruido definido en (11) está determinado por la cantidad de ***m*** patrones almacenados en ***W***, por lo que debemos esperar recuperaciones fallidas si $\mathbf{m} > \mathbf{n}$ o $\mathbf{m} > \mathbf{p}$. De ocurrir esto, la suma de los factores de ruido excedería al factor de corrección, de aquí que la capacidad de la **BAM** sea estimada en :

$$m < min(n, p)$$

Bibliografía

1.- García-Pelayo y Gross, Ramón. Diccionario Pequeño Larousse en Color, 1981.

2.- Standart Dictionary of the English Languaje. Funk & Wagnalls, 1967.

3.- Wienderhold, Gio. File Organization for Data Base Design; McGraw-Hill, 1967.

4.- Shaft, Adam; Historia y Verdad. Teoría y Praxis.

5.- Rodríguez René. 3EBAM: Un Sistema de Memorias Asociativas Bidireccionales, Tesis Maestría 1994.

6.- Villee, Claude A. Biología, 1986.

7.- Van Gigch, Jonh P. Teoría General de Sistemas, 1990.

8.- Hofstadter, Douglas R. Godel, Escher, Bach: Una Eterna Trenza Dorada; CONACYT, 1982.

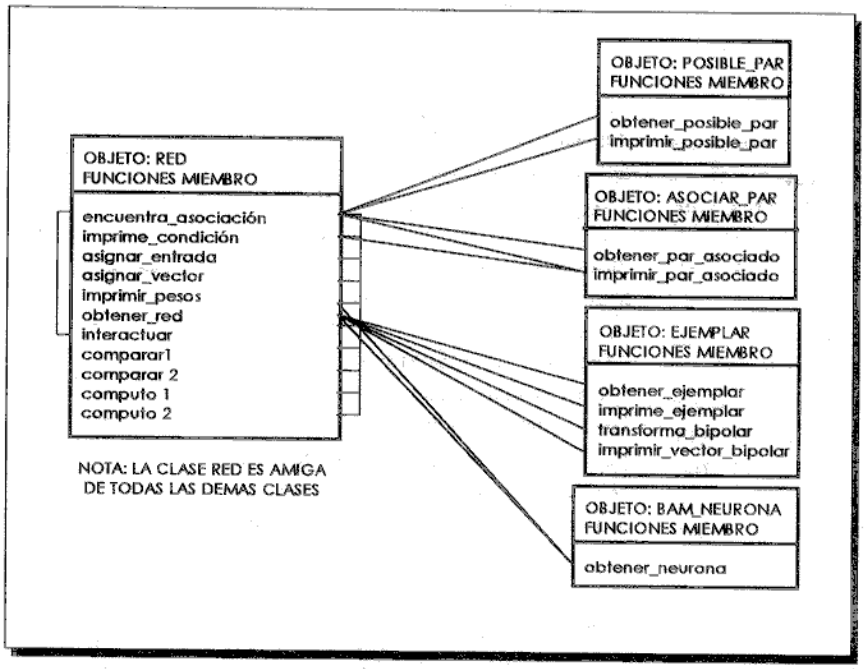
9.- Flores, Edmundo. La Creacion de la Nueva Ciencia del Caos y el Ocaso de la Metereología y de la Econometría; El BUHO, No. 253, EXCELSIOR, 1990.

10.- Freeman, James. Neural Networks, Algorithms, Applications, and Programming Techniques, Addison Wesley, 1991.

11.- Blum, Adam. Neural Networks in C++, Wiley, 1992.

12.- Kosko Bart. Neural Networks and Fuzzy Systems, Prentice Hall 1992.

13.- Freeman & Skapura. Neural Networks, Addison Wesley 1992.




```
//Programa para una BAM (Memoria asociativa
bidireccional)
//Elaborado por:

//M. en C. Miguel Angel Partida Tapia
//Alumno de la Maestría: Rubén Peredo
Valderrama
//Alumno de la Maestría: Francisco Flávio Córdova
Q.

//Fila de encabezado para el programa de la BAM
//CINTEC 1995

#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#define MAX_TAMANO 10

class bam_neurona
{
//Declaración de la clase bam_neurona
//Sección protegida y pública
 protected:
   int nnbr;
   int entrada_neurona,salida_neurona;
   int salida;
   int activacion;
   int peso_salida[MAX_TAMANO];
   char *nombre;
   friend class red;
   //Declaración de clase red como amiga

public:
   bam_neurona() { }; //Constructor
   void obtener_neurona(int,int,int,char *);
};

class ejemplar
{
//Declaración de la clase ejemplar
//Sección protegida y pública

protected:
   int dimension_x,dimension_y;
   int v1[MAX_TAMANO], v2[MAX_ TAMANO];
   int u1[MAX_TAMANO], u2[MAX_TAMANO];
   friend class red;
   //Declaración de la clase red como amiga
   friend class matriz;
   //Declaración de la clase matriz como amiga

public:
   ejemplar() { }; //Constructor
   void obtener_ejemplar(int,int,int *,int *);
   void imprime_ejemplar();
   void transforma_bipolar();
   void imprimir_vector_bipolar();
};

class asociar_par
{
//Declaración de la clase asociar_par
//Sección protegida y pública

protected:
   int dimension_x,dimension_y,idn,o,p;
   int v1[MAX_TAMANO],v2[MAX_TAMANO];
   friend class red;
//Declaración de la clase red como amiga

public:
   asociar_par() { }; //Constructor
   void obtener_par_asociado(int,int,int);
   void imprimir_par_asociado();
};
class posible_par
{
//Declaración de la clase posible_par
//Sección protegida y pública

protected:
   int dimension_x,dimension_y;
   int v1[MAX_TAMANO],v2[MAX_TAMANO];
   friend class red;
   //Declaración de la clase red como amiga

public:
   posible_par() { }; //Constructor
   void obtener_posible_par(int,int);
   void imprimir_posible_par();
};

class red
{
//Declaración de la clase red
//Sección pública
public:
   int   anmbr, bnmbr, bandera, nexmplr, nasspr,
         nentra;
   bam_neurona   (anrn) [MAX_TAMANO],
                 (bnrn)[MAX_TAMANO];
   ejemplar (e)[MAX_TAMANO];
   asociar_par (as)[MAX_TAMANO];
   posible_par (pp)[MAX_TAMANO];
   int   salidas1[MAX_TAMANO],
         salidas2[MAX_TAMANO];
   int  matriz1[MAX_TAMANO][MAX_TAMANO],
         matriz2
MAX_TAMANO][MAX_TAMANO];

   red() { }; //Constructor
   void obtener_red(int,int,int,int [][6],int [][5]);
   void comparar1(int,int);
   void comparar2(int,int);
   void imprimir_pesos();
   void interactuar();
   void encuentra_asociacion(int *);
   void asignar_entrada(int *);
   void asignar_vector(int,int *,int *);
   void computo1();
   void computo2();
   void imprime_condicion();
```

```
//Programa para una BAM (Memoria asociativa
bidireccional)
//Elaborado por:

//M. en C. Miguel Angel Partida Tapia
//Alumno de la Maestría: Rubén Peredo
Valderrama
//Alumno de la Maestría: Francisco Flávio Córdova
Q.

//Fila fuente para el programa de la BAM
//CINTEC 1995

#include <c:\bc4\bin\bamwin.h>
#include <conio.h>


void bam_neurona::obtener_neurona(int m1,int
m2,int m3,char *y)
{
int i;
nombre = y;
nnbr = m1;
salida_neurona = m2;
entrada_neurona  = m3;

for(i=0;i<salida_neurona;++i){
   peso_salida[i] = 0 ;
   }

salida = 0;
activacion = 0;
}

void  ejemplar::obtener_ejemplar(int  k,int  l,int
*b1,int *b2)
{
// Función miembro ejemplar::obtener_ejemplar
// Método para obtener el vector

int i2;
dimension_x = k;
dimension_y = l;

for(i2=0;i2<dimension_x;++i2){
   v1[i2] = b1[i2]; }

for(i2=0;i2<dimension_y;++i2){
   v2[i2] = b2[i2]; }
}

void ejemplar::imprime_ejemplar()
{
// Función miembro ejemplar::imprime_ejemplar
// Método para imprimir el vector

int i;
cout<<"\nEl vector X es:"<<"                El vector
Y es:\n";

for(i=0;i<dimension_x;++i){
   cout<<v1[i]<<"  ";}
   cout<<"                ";
for(i=0;i<dimension_y;++i){
   cout<<v2[i]<<"  ";}

cout<<"\n";
}

void ejemplar::transforma_bipolar()
{
// Función miembro ejemplar::transforma_bipolar
// Método para transformar el vector a su forma
bipolar
```

```
int i;

for(i=0;i<dimension_x;++i){
   u1[i] = 2*v1[i] -1;}

for(i=0;i<dimension_y;++i){
   u2[i] = 2*v2[i] - 1;}

}

void ejemplar::imprimir_vector_bipolar()
{
// Función miembro ejemplar:: imprimir_vector_
   bipolar
// Método para imprimir el vector bipolar

int i;
cout<<"\nForma bipolar de los vectores\n";

cout<<"El vector X bipolar"<<"              El vector
Y bipolar\n";
for(i=0;i<dimension_x;++i){
   cout<<u1[i]<<"  ";}

cout<<"                ";
for(i=0;i<dimension_y;++i){
   cout<<u2[i]<<"  ";}

cout<<"\n";
}


void asociar_par::obtener_par_asociado(int i, int
j, int k)
{
// Función miembro asociar_par :: obtener_par_
   asociado
// Método para obtener el par asociado

idn = i;
dimension_x = j;
dimension_y = k;
}

void asociar_par::imprimir_par_asociado()
{
// Función miembro asociar_par::imprimir_par_
   asociado
// Método para imprimir el par asociado

int i;

cout<<"\nEl vector X par asociado # "<<idn<<"
es: \n";

for(i=0;i<dimension_x;++i){
   cout<<v1[i]<<"    ";}
   cout<<"\nPresione una tecla para continuar\n";
   getch();

cout<<"\nEl vector Y par asociado # "<<idn<<"
es: \n";

for(i=0;i<dimension_y;++i){
   cout<<v2[i]<<"    ";}
   cout<<"\nPresione una tecla para continuar";
   getch();


cout<<"\n";
}
```

```
void posible_par::obtener_posible_par(int k,int j)
{
// Función miembro posible_par:: obtener_
   posible_ par
// Método para obtener el posible par

dimension_x = k;
dimension_y = j;

}

void posible_par::imprimir_posible_par()
{
// Función miembro posible_par::imprimir_
   posible_par
// Método para imprimir el posible par

int i;
cout<<"\n";
cout<<"\nEl vector X en posible asociación par
es: \n";

for(i=0;i<dimension_x;++i){
   cout<<v1[i]<<"    ";}

cout<<"\nEl  vector Y en posible asociación par
es: \n";

for(i=0;i<dimension_y;++i){
   cout<<v2[i]<<"    ";}

cout<<"\n";
}

void red::obtener_red(int k,int l,int k1,int b1[][6],int
b2[][5])
{
// Función miembro red::obtener_red
// Método para obtener la red de trabajo

anmbr = k;
bnmbr = l;
nexmplr = k1;
nasspr = 0;
nentra = 0;
int i,j,i2,d;
bandera =0;
char *y1="ANEURON", *y2="BNEURON" ;

for(i=0;i<nexmplr;++i){

e[i].obtener_ejemplar(anmbr,bnmbr,b1[i],b2[i]);
   e[i].imprime_ejemplar();
   e[i].transforma_bipolar();
   e[i].imprimir_vector_bipolar();
   }
   cout<<"\nPulse una tecla para continuar\n";
   getch();

for(i=0;i<anmbr;++i){

anm[i].bam_neurona::obtener_neurona(i,bnmbr,0,y1);}

for(i=0;i<bnmbr;++i){

bnm[i].bam_neurona::obtener_neurona(i,0,anmbr,y2);}

for(i=0;i<anmbr;++i){

   for(j=0;j<bnmbr;++j){
      matriz1[i][j]  = 0;

      for(i2=0;i2<nexmplr;++i2){
         matriz1[i][j]  += e[i2].u1[i]*e[i2].u2[j];}
```

```
   matriz2[j][i] = matriz1[i][j];
   anrn[i].peso_salida[j] = matriz1[i][j];
   bnrn[j].peso_salida[i] = matriz2[j][i];
}


primir_pesos();
ut<<"\n";


id red::asignar_entrada(int *b)

unción miembro red::asignar_entrada
étodo para asignar entrada

 i;
ut<<"\n";

(i=0;i<anmbr;++i){
anrn[i].salida = b[i];
salidas1[i] = b[i];
}



id red::comparar1(int j,int k)

unción miembro red::comparar1
étodo de comparación 1

 i;

(i=0;i<anmbr;++i){

if(pp[j].v1[i] != pp[k].v1[i]) bandera = 1;
break;
}



id red::comparar2(int j,int k)

unción miembro red::comparar2
étodo de comparación 2

 i;

(i=0;i<anmbr;++i){

if(pp[j].v2[i] != pp[k].v2[i]) bandera = 1;
break;}



id red::computo1()

unción miembro red::computo1
étodo para el primer cálculo

 j;

(j=0;j<bnmbr;++j){
int ii1;
int c1 =0,d1;
cout<<"\n";

for(ii1=0;ii1<anmbr;++ii1){
  d1 = salidas1[ii1] * matriz1[ii1][j];
  c1 += d1;
  }

bnrn[j].activacion = c1;
cout<<"\nsalida nivel neurona "<<j<<" la acti-
ción es "<<c1<<"\n";
if(bnrn[j].activacion <0) {
```

```
    bnrn[j].salida = 0;
    salidas2[j] = 0;}

  else

    if(bnrn[j].activacion>0) {

      bnrn[j].salida = 1;
      salidas2[j] = 1;}

      else

      {cout<<"\nA 0 es obtenido usando previo
valor de salida \n";

      if(nentra<=nexmplr){

        bnrn[j].salida = e[nentra-1].v2[j];
        cout<<"\nPresione una tecla para
continuar\n";
        getch();
        cout<<"\n";
        }

      else

        { bnrn[j].salida = pp[0].v2[j];}
        salidas2[j] = bnrn[j].salida; }

    cout<<"\nsalida nivel neurona "<<j<<" la sa-
lida es "<<bnrn[j].salida<<"\n";
    cout<<"\nPresione una tecla para
continuar\n";
    getch();
    }
}


void red::computo2()
{
// Función miembro red::computo1
// Método para el segundo cálculo

int i;
cout<<"\n";

for(i=0;i<anmbr;++i){
  int ii1;
  int c1=0;

  for(ii1=0;ii1<bnmbr;++ii1){
    c1 += salidas2[ii1] * matriz2[ii1][i]; }

  anrn[i].activacion = c1;
  cout<<"\nla entrada nivel neurona "<<i<<" la
activación es "<<c1<<"\n";

  if(anrn[i].activacion <0 ){anrn[i].salida = 0;
          salidas1[i] = 0;}


  else

    if(anrn[i].activacion >0 ) {

      anrn[i].salida = 1;
      salidas1[i] = 1;
      }

    else

    { cout<<"\nA 0 es obtenida, usando valor
previo si es posible\n";
```

```
    if(nentra<=nexmplr) {anrn[i].salida = e[nentra-
1].v1[i];
          cout<<"\nPresione una tecla para
continuar\n";
          getch();
          cout<<"\n";
          }

    else {anrn[i].salida = pp[0].v1[i];}

    salidas1[i] = anrn[i].salida;}
    cout<<"\nla entrada nivel neurona "<<i<<" la
salida es "<<anrn[i].salida<<"\n";
    cout<<"\nPresione una tecla para
continuar\n";
    getch();
    cout<<"\n";
    }
}

void red::asignar_vector(int j1,int *b1,int *b2)
{
// Función miembro red::asignar_vector
// Método para la asignación del vector

int  j2;

for(j2=0;j2<j1;++j2){
  b2[j2] = b1[j2];}

}

void red::imprimir_pesos()
{
int i3,i4;
clrscr();
cout<<"\nMatriz de pesos--Nivel de entrada a
Nivel de salida: \n\n";

for(i3=0;i3<anmbr;++i3){

  for(i4=0;i4<bnmbr;++i4){
    cout<<anrn[i3].peso_salida[i4]<<"     ";}

  cout<<"\n"; }

cout<<"\n";
cout<<"\Presione una tecla para continuar\n";
getch();
clrscr();

cout<<"\nMatriz de pesos--Nivel de salida a Nivel
de entrada: \n\n";

for(i3=0;i3<bnmbr;++i3){

  for(i4=0;i4<anmbr;++i4){
    cout<<bnrn[i3].peso_salida[i4]<<"     ";}

  cout<<"\n"; }

cout<<"\n";
cout<<"\nPresione una tecla para continuar\n";
getch();
clrscr();
}

void red::interactuar()
{
int i1;

for(i1=0;i1<nexmplr;++i1){
  encuentra_asociacion(e[i1].v1);
  }
```

```
}

void red::encuentra_asociacion(int *b)
{
// Función miembro red::encuentra_asociacion
// Método para la busqueda de asociación

int j;
bandera = 0;
   asignar_entrada(b);
   nentra ++;
   cout<<"\nVector de entrada:\n" ;

   for(j=0;j<6;++j){
      cout<<b[j]<<"     ";}
   cout<<"\nPresione una tecla para continuar\n";
   getch();
   cout<<"\n";
   pp[0].obtener_posible_par(anmbr,bnmbr);
   asignar_vector(anmbr,salidas1,pp[0].v1);

   computo1();

   if(bandera>=0){
      asignar_vector(bnmbr,salidas2,pp[0].v2);

      cout<<"\n";
      pp[0].imprimir_posible_par();
      cout<<"\n";

      computo2(); }

   for(j=1;j<MAX_TAMANO;++j){
      pp[j].obtener_posible_par(anmbr,bnmbr);
      asignar_vector(anmbr,salidas1,pp[j].v1);

      computo1();

      asignar_vector(bnmbr,salidas2,pp[j].v2);

      pp[j].imprimir_posible_par();
      cout<<"\n";

      comparar1(j,j-1);
      comparar2(j,j-1);

      if(bandera == 0) {

         int j2;
         nasspr += 1;
         j2 = nasspr;


as[j2].obtener_par_asociado(j2,anmbr,bnmbr);
         asignar_vector(anmbr,pp[j].v1,as[j2].v1);
         asignar_vector(bnmbr,pp[j].v2,as[j2].v2);

         cout<<"\nPATRONES ASOCIADOS:\n";
         as[j2].imprimir_par_asociado();
         j = MAX_TAMANO ;
      }


      else

         if(bandera == 1)

            {
            bandera = 0;
            computo1();
            }

   }
```

```
}

void red::imprime_condicion()
{
// Función miembro red::imprime_condicion
// Método para la impresión de los resultados
finales
// Encontrados por la BAM

int j;
cout<<"\n";
cout<<"\nLOS PARES ASOCIADOS SIGUIEN-
TES FUERON ENCONTRADOS POR LA
BAM\n\n";

for(j=1;j<=nasspr;++j){
   as[j].imprimir_par_asociado();
   cout<<"\n";}

}


void main()
{
//Función principal

int ar = 6, br = 5, nex = 3,d;
int vector_entradas[][6] = {1, 0, 1, 0, 1, 0, 1, 1, 1,
0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
1};
int vector_salidas[][5] = {1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
1, 0, 0, 1, 0};

clrscr();
cout<<"\n\nESTE PROGRAMA ES UNA RED
DE MEMORIA ASOCIATIVA BIDIRECIONAL
(BAM).\n\n";
cout<<"\n\nLA RED ES ACTUALIZADA POR
ILUSTRACION CON:\n\n";
cout<<ar<<" NEURONAS DE ENTRADA, Y\n";
cout<<br<<" NEURONAS DE SALIDA.\n ";
cout<<"\n\n\n";
cout<<nex<<" VECTORES USADOS AL
DECODIFICAR \n ";
cout<<"\n\nPresione una tecla para continuar";
getch();
clrscr();

static red bamn;
bamn.obtener_red(ar,br,nex,vector_entradas,
vector_salidas) ;
bamn.interactuar();
bamn.encuentra_asociacion(vector_entradas[3]);
bamn.encuentra_asociacion(vector_entradas[4]);
bamn.imprime_condicion();
}
```

Percepción Computacional

Fis. Mat. Pablo Manrique Ramírez
Profesor e Investigador del CINTEC-IPN.
Mat. Ricardo Barrón Fernández
Jefe del Departamento de Matemáticas Aplicadas del CINTEC-IPN.

La percepción computacional (percepción artificial) tiene la finalidad de la reproducción artificial (que engloba subsistemas no informáticos) del sentido de la vista. Se trata de un objetivo excepcionalmente ambicioso y complejo, que se encuentra actualmente en sus inicios, pero que avanza excesivamente rápido. Parece natural pensar que el objetivo de dotar a las máquinas del sentido de la vista supondrá un salto cualitativo en sus capacidades de actuación.

Una manera rudimentaria de definir **percepción artificial** o **percepción computacional**: es la que se basa en un computador. En estos tiempos no se ha podido llegar a dotar a una máquina del sentido de la vista en su totalidad, se han logrado pequeños avances desde un punto de vista sensorial, gracias a la tecnología de los microprocesadores, pero aún existe una gran complejidad de procesamiento.

Haciendo una analogía entre el sistema artificial de procesamiento de imágenes y el sistema visual humano, se tendría el **diagrama 1**.

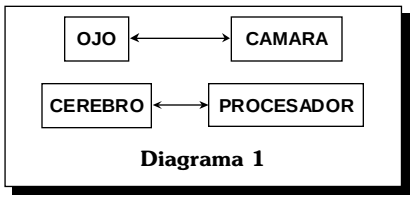


Diagrama 1

Desarrollo

Antes de abordar el tema, hablaremos acerca de una señal eléctrica (un voltaje de tensión) que varía en forma continua en el tiempo denominada señal de video. En ella existen dos tipos de impulsos de sincronía, la vertical y la horizontal. En la práctica se manejan dos formatos de señal de video: el americano y el europeo, cuyas comparaciones entre sí son:

- Difieren en los límites de los tiempos de sincronización (Esto se debe a las diferentes frecuencias de la red eléctrica a la que trabajan; 60Hz para la red americana y 50Hz para la red europea).
- Formatos iguales (Ambos tratan de codificar líneas y cuadros de imagen).

Una vez que se hizo una breve aclaración de la señal de video, para lograr un análisis automático es necesario definir las etapas que se muestran en el **diagrama 2**.

o El pre-procesamiento es la adquisición de imágenes el cual consiste en digitalizar una señal de video, con la información luminosa de la imagen a interpretar. Naturalmente, la señal continua de video deberá ser:

- muestreada y,
- digitalizada bináriamente para su tratamiento en el computador.

El efecto combinado del número de líneas (consustancial a la cámara de video) y el de la frecuencia de muestreo (es decir, el número de muestras de la señal de video que se

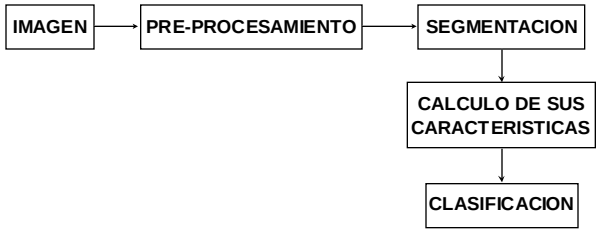


Diagrama 2

toma por cada línea) da lugar a lo que se conoce como **nivel de resolución espacial de la imagen**.

Con el número de muestras que se toman de la señal de video por cada línea se forma una matriz de NxM elementos (N = número de líneas, M = número de muestras por línea); cada elemento de esta matriz va conformar un pixel que incorpora el nivel de luminosidad del punto correspondiente de la imagen digital. Por lo general se trabaja con matrices cuadradas M=N.

Antes de enviar la imagen electrónica al computador, se codifica en binario los niveles de intensidad de cada pixel. El número de bits empleados por pixel se conoce como la **resolución en niveles de gris**; cabe destacar que con los efectos combinados de las resoluciones espacial y de los niveles de gris se pueden apreciar las imágenes.

o La segmentación es un proceso orientado a seccionar la imagen digital en zonas disjuntas con significado propio. Este significado dependerá de la imagen, así como de los objetivos que se persigan con su interpretación. La operación es común en las dos segmentaciones; lo que varía es aquello que se desea segmentar. En un caso, zonas; en el otro, ya dentro de un zona concreta, objetos individuales.

Las técnicas de segmentación están orientadas a objetos o a agrupaciones de objetos, y se pueden dividir en dos grandes grupos:

Aplicación de umbrales de nivel de gris, mediante el empleo de histogramas.

Aplicación de rasgos comunes, en base al reconocimiento de formas.

o El cálculo de las características consiste en la extracción de rasgos (o bordes); es decir, dado un objeto físico dentro de una imagen se puede obtener, a partir de su contorno (resultado de la segmentación), un conjunto de parámetros característicos como: perímetro, grado de circularidad, momentos de orden, etc. Estos rasgos hacen posible distinguir objetos mediante el uso de un número reducido de propiedades. Los rasgos de un objeto forman un vector $\mathbf{X}=(X_1,X_2,\ldots,X_n)^T$, que particularizando en ciertos valores numéricos (los del objeto a clasificar) determina un punto en el espacio n-dimensional de las características.

o La clasificación de los objetos, se realiza al definirlos en la memoria una vez que se segmentaron y se extrajeron ciertas características propias para reconocer un objeto.

Configuración de un sistema mínimo de Percepción Artificial

El siguiente diagrama (**Diagrama 3**) muestra la organización básica de un sistema mínimo de Percepción Artificial, teniéndose a continuación una breve descripción de cada bloque funcional.

o Sensor óptico. Normalmente entrega una señal de video estandarizada para su posterior digitalización. El conjunto sensor y óptica se integra en una cámara de video.

La etapa sensorial es la adquisición digital de la imagen, iluminación, sensores ópticos, formato de la imagen digital, así como la geometría de la formación de imágenes digitales.

o Conversor analógico/digital; permite digitalizar la señal de video entregada por el subsistema anterior. Aparecen dos parámetros esenciales:

Definición espacial: es decir, número de puntos de digitalización por imagen o cuadro.

Resolución digital: número de bits para codificar los niveles de intensidad luminosa de cada punto de digitalización (que suele denominarse pixel, es decir, elemento de imagen).

o Una señal de video estandarizada, una vez digitalizada, se transforma en una matriz cuadrada (que contiene la información visual digi-

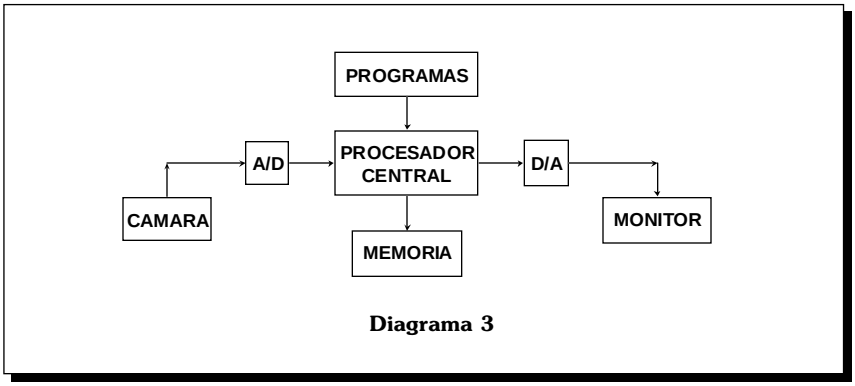


Diagrama 3

talizada) y se almacena en la memoria central de un procesador, para que posteriormente se procese y manipule a través de programas.

o Monitor de video; permite visualizar tanto las imágenes captadas como los resultados del procesamiento de imágenes. En realidad, se requiere un conversor D/A y, finalmente un monitor de video, ya que este último trabaja con una señal analógica.

Hasta ahora se ha hablado a grosso modo de la etapa sensorial de la percepción artificial. A partir de aquí se habla del procesamiento y de los verdaderos problemas, pues involucra los algoritmos, el tipo de lenguaje de programación y el tipo de procesador que se este utilizando. Uno de los microprocesadores que se utiliza para la adquisición y procesamiento de imágenes es el DSP (Digital Signal Processor).

El análisis automático (informático) propiamente dicho de una imagen digital es aplicado a ciertas operaciones orientadas a mejorar la calidad de la misma; ésto nos lleva ha aplicar técnicas y algoritmos de procesamiento, como por ejemplo:

1) Conversión de los niveles de gris. (Transformación adecuada de los niveles de gris para mejorar la sensibilidad de la imagen).

2) Transformaciones geométricas:
   -Homogéneas
   -Translación (desplazamiento)
   -Escalamiento
   -Rotación (Se utilizan en la corrección de la perspectiva, y reconstrucción tridimensional de los objetos de una imagen).

3) Transformación del histograma. (Es una función donde su dominio son los niveles de gris y su contradominio el número de pixeles de cada nivel. La función es un diagrama de barras de la propia imagen, que frecuentemente se normaliza entre 0 y 1; por tanto, se emplea la frecuencia relativa de cada nivel de gris).

4) Filtrado espacial. (Actúa sobre los niveles de gris de los pixeles de la imagen).

5) Filtrado frecuencial. (Es costoso en tiempo y en memoria y se lleva acabo utilizando la transformada rápida de Fourier)

Para aplicar estas técnicas y algoritmos es necesario definir formalmente una imagen digital. Una imagen digital es una función discreta bidimensional: **f**(x,y) donde $0 \le x, y \le N$ que se denota como sigue:

$$\mathbf{f}(x,y) = \begin{matrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & & & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,N-1) \end{matrix} \quad (1)$$

Una imagen digital es una matriz cuadrada NxN binaria, salvo en casos excepcionales, conteniendo cada elemento de la matriz un valor discreto que cuantifica el nivel de intensidad luminosa (también denominado nivel de escalas de grises) del correspondiente elemento de imagen o pixel. Esta matriz binaria es accesible y manejable por un programa ordenador, con lo que se puede en principio realizar cualquier operación sobre ella.

La coordenada (x,y) denota una coordenada espacial de una imagen digital, y el conjunto de puntos en el plano x-y forman el espacio de posición. La resolución espacial NxN viene determinada por la calidad que se persigue en las imágenes digitales. A mayor N, mayor calidad, aunque esto obliga a una carga computacional mayor. Por necesidades del formato binario, N es siempre múltiplo de 2, es decir, $N=2^m$ donde **m** es un entero positivo. En cuanto a la resolución en niveles de gris también es un múltiplo de 2, ya que la cuantificación de los niveles de intensidad luminosa se realizan en código binario.

Por lo tanto, si q es el número de bits empleados para codificar los niveles de gris, entonces:

$$0 \le f(x,y) \le 2^q \quad (2)$$

El convenio habitual es codificar los niveles oscuros o negros con valores binarios más bajos, mientras que los niveles claros o blancos con los valores binarios más altos.

Se observa que con la definición anterior de una imagen digital ya se puede realizar un tratamiento matemático discreto, que nos facilitará el tratamiento del procesamiento de las imágenes por computadora.

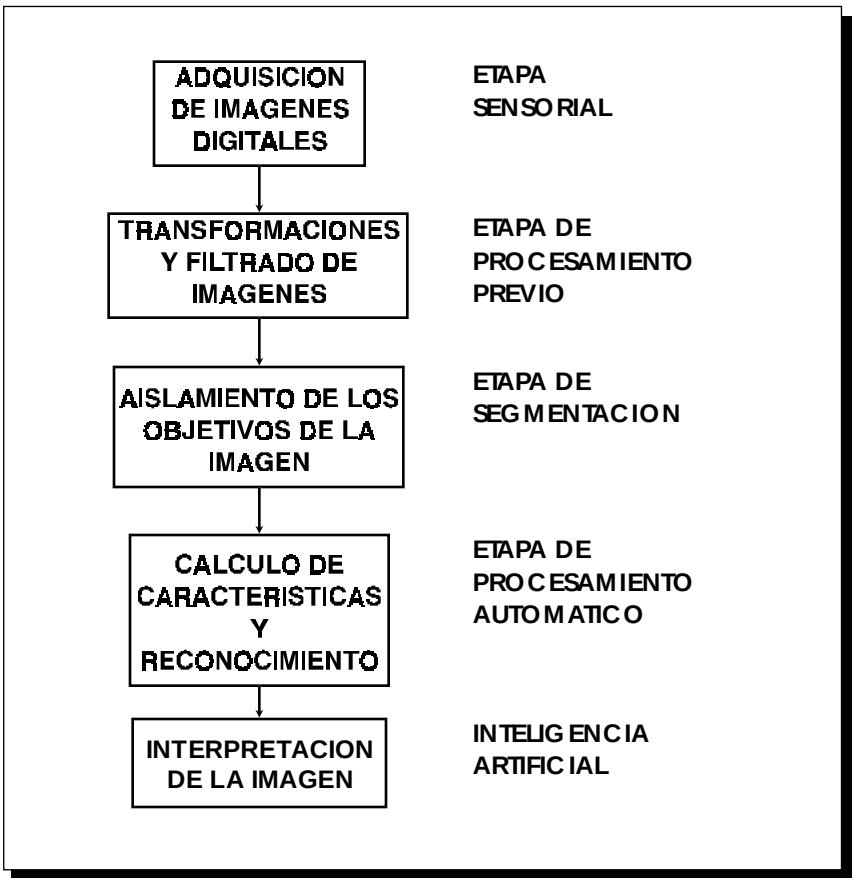
El método en el dominio del espacio, maneja procesamientos definidos por medio de una transformación **T**, cuyo dominio es el conjunto de pixeles f(x,y) que generan la imagen, es decir:

$$t(x,y) = \mathbf{T}(f(x,y)) \quad (3)$$

donde, esta transformación es lineal, homogénea e invariante en la posición.

Un ejemplo de estas transformaciones es: Translación del procesamiento de las imágenes, al dominio de las frecuencias espacial con la ayuda de las Transformada de Fourier.

$$F\{f(x,y)\} = \mathbf{F}(u,v)$$



$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) e^{-j2\pi(ux+vy)} dxdy \qquad (4)$$

El siguiente diagrama muestra las etapas en que puede organizarse un sistema visual artificial.

Conclusiones

De lo precedente se desprende que el desarrollo inicial de un sistema de procesamiento de imágenes resulta viable, si se conocen de antemano sus características y necesidades, para lograr un ahorro en espacio de memoria y tiempo de ejecución; para esto se eligen adecuadamente las técnicas y algoritmos que se deben de implementar para satisfacer los requerimientos de dicho sistema. La digitalización de la señal de video permite que su información visual pueda ser introducida y tratada mediante un ordenador. No obstante, el formato de las imágenes digitales es distinto al de la señal de video, entre otros motivos porque se trata de una información discreta, tanto en los valores que puede tomar la propia imagen digital, como en sus argumentos o variables independientes, que son igualmente discretos.

Bibliografía

- Computer Techniques in Image Processing. H. C. Adresws Academic Press, 1970.
- Digital Image Processing. W. Pratt. John Willey & Sons, 1978.
- Fundamental of Digital Image Processing. A. K. Jain. Prentice-Hall, 1989.

Multiplicador Digital de Frecuencia Programable Adaptivo de Alta Precisión.

M, en C. Miguel A. Partida Tapia
Subdirector Académico y de Investigación del CINTEC-IPN.
Dr. Adriano de Luca
UAM I.
Dr. John Goddard
UAM I.

El uso de los multiplicadores digitales de frecuencia tiene su principal importancia en el campo del Procesamiento Digital de Señales, por la necesidad de acercar la frecuencia de muestreo a la frecuencia armónica que se interesa estudiar. Muchos tipos de multiplicadores digitales de frecuencia han sido planteados en las referencias **[1]** - **[10]**, donde los descritos en **[1]** y **[3]** son los que han aportado significativamente un avance en estos últimos años. Estas dos propuestas se han tomado como base para realizar un multiplicador digital de frecuencia programable y adaptivo. Haciendo un análisis de estas dos últimas se han encontrado diferencias objetivas en la manera de resolver la minimización del error generado en la frecuencia de salida; esto sucede cuando la relación de la frecuencia del reloj maestro y la frecuencia de entrada no entregan un cociente entero. El modelo que en este documento se describe, modifica la interpretación del reloj maestro como base de referencia a los contadores, esto es, se deriva una frecuencia de reloj que resulta de dividir el reloj maestro entre el factor de multiplicación, lográndose con esto mejorar por un factor importante la disminución del error en la frecuencia de salida y evitando el uso de circuitos de corrección adicional al circuito de multiplicación. En los dos casos mencionados se requiere de un circuito de corrección asociado a la lógica del multiplicador para lograr un mínimo error, aumentando así el tiempo de disponibilidad de la frecuencia de salida; en este modelo no se requiere del circuito de corrección, limitando con esto, también, el número de componentes. En resumen, las ventajas son:

a).- Disminuye de manera notable el factor de error, relativo a la frecuencia de entrada y salida.

b).- No requiere de un circuito de corrección.

c).- Dada las características de diseño permite la integración en un solo dispositivo FPGA.

Principios de operación del Multiplicador de Frecuencia

Idealmente, un multiplicador digital de frecuencia está definido como un generador de ***N*** pulsos equidistantes de salida durante un período de una señal de entrada. Generalmente se usa un contador para medir el período de la señal de entrada en términos de los ciclos de reloj maestro.

Todo multiplicador digital de frecuencia opera de la siguiente manera: Después de una transición *bajo-alto* de la señal de entrada *fin*, el contador ascendente *upcounter #1* es puesto en *reset* y cuenta ***M*** pulsos del reloj maestro *fc*, desde el punto donde la transición *bajo-alto* del reloj de entrada ocurrió. En este momento el contador ascendente tiene el valor de ***M***, y esta cuenta es transferida al *buffer #1* antes de inicializar el contador ascendente. El número ***M*** es igual al número de pulsos generados por el reloj maestro a una frecuencia *fc* en un ciclo de la señal de entrada *fin* . ***M*** es entonces un indicador de la relación entre el reloj maestro *fc* y la señal de entrada *fin*, quedando expresado por [3]:

$$M \cong \frac{f_c}{f_{in}} \qquad (1)$$

Entonces el multiplicador de frecuencia tendrá que generar ***N*** pulsos de salida durante un ciclo de la señal de entrada; así tendrá que generar un pulso de salida aproximadamente cada ***M/N*** ciclos de reloj maestro. Tomando en consideración a (1), se tiene:

$$\frac{M}{N} = \frac{f_c}{N * f_{in}} \qquad (1.1)$$

donde

$$f_{div} = \frac{f_c}{N} \qquad (1.2)$$

M se representa por un número binario de *m-bits;* el número ***M*** es entonces dividido por ***N***, antes de ser aplicado a la entrada del contador descendente previamente puesto en *reset*, donde ***N*** es el factor de multiplicación por la que la frecuencia de entrada *fin* será multiplicada para obtener la frecuencia de salida *fout*. En **[3]** la división se implementa tomando como base a una potencia de base dos, ***N*** = 2n; sin embargo, en **[1]** y en este modelo esta restricción no existe. La programación del factor de multiplicación en las propuestas **[1]** y **[3]** se realiza por *interruptores*, en la nuestra se escribe a un registro de *n* bits contenido en el dispositivo programable.

Al tener la cuenta solo se tiene que transferir el contenido del contador descendente *downcounter#1* (*m-n*) los *bits* más significativos de ***M***, donde *n* es el número de *bits* representados por ***N;*** los bits menos significativos en las propuestas **[1]** y **[3]** se toman para ser usados en un circuito de corrección que ajusta la cuenta de salida para disminuir su error. En la propuesta **[1]**, la cuenta resultante es definida como un cociente entero ***Q*** y un residuo ***R*** , donde este último pasa a un circuito de corrección. Con un pulso del reloj maestro *fc* el contador descendente *downcounter#1* disminuirá su cuenta en uno. En este punto se emitirá un pulso de salida y el contador descendente *downcounter#1* será puesto en *reset*. La repetición de este pulso estará dada por:

$$f_{out} = \frac{f_c}{M/N} \qquad (2)$$

$$f_{out} = \frac{f_{in} * f_c}{f_{div}} = \left(\frac{f_{in}}{f_{div}}\right) * f_c \qquad (2.1)$$

Una variante importante entre las propuestas **[1]** y **[3]** reside fundamentalmente en las expresiones (1.1) y (2.1). Esta variante consiste en la descripción de la variable *fdiv*, que define a la frecuencia resultante de dividir la frecuencia del reloj maestro y el factor de multiplicación, donde fdiv alimenta a los contadores ascendentes *upcounter#1*.

Sustituyendo (1) en (2) , se tiene que la salida *fout* se define como la frecuencia de entrada fin repetida ***N*** veces, quedando la expresión como sigue:

$$f_{out} = N * f_{in} \qquad (3)$$

Un elemento importante a considerar es que el error relativo de la frecuencia de salida *fout* depende de manera directa de un valor alto de la frecuencia del reloj maestro. En **[3]** el error relativo se genera en el caso en que los *n-bit* menos significativo (LSB's) de ***M*** no son tomados en la cuenta (como se muestra en la figura 1). El valor transferido al contador descendente *downcounter#1* es entonces menor o igual a la relación exacta ***M/N,*** pero cercano a la relación (***N-1/N***). Tomando (2) se tendrá la frecuencia de salida como:

$$f'_{out} = \frac{f_c}{\frac{M}{N} - \left(\frac{N-1}{N}\right)} \qquad (4)$$

así, el error máximo posible estará definido por la siguiente expresión, como se expone en **[3]**, donde *f'out* es la frecuencia de salida con un factor de error.

$$E_{rmax} = \left(\frac{f'_{out} - f_{out}}{f_{out}}\right) = \left(\frac{N-1}{M-(N-1)}\right) \qquad (5)$$

Comprobación:

de (2) y (4) tenemos:

$$f_{out} = \frac{f_c}{M/N} = \frac{N * f_c}{M} \qquad (5.1)$$

de (5.1) y (4) tenemos:

$$\left(\frac{M(M * f'_{out} - N * f_c)}{N * M * f_c}\right) =$$

$$E_{rmax} = \frac{M * f'_{out} - N * f_c}{N * f_c} = \left(\frac{M * f'_{out}}{N * f_c}\right) - 1 =$$

$$= M\left(\frac{\frac{N * f_c}{M-(N-1)}}{N * f_c}\right) - 1 = \left(\frac{\frac{M * N * f_c}{M-(N-1)}}{\frac{N * f_c}{1}}\right) - 1 = \left(\frac{M}{M-(N-1)}\right) - 1 =$$

$$= \left(\frac{M}{M-(N-1)}\right) - \left(\frac{M-(N-1)}{M-(N-1)}\right) =$$

$$E_{rmax} = \frac{N-1}{M-(N-1)}$$

Si el error máximo aceptable de pulsos de salida está dado por ***N***, entonces el error máximo estará por debajo de 1/***N***. Esto implica que:

$$E_{rmax} = \left(\frac{f'_{out} - f_{out}}{f_{out}}\right) = \left(\frac{N-1}{M-(N-1)}\right) < 1/N$$

invirtiendo se tiene :

$$\frac{M-(N-1)}{N-1} > N$$

$$M-(N-1) > N*(N-1)$$

$$M > N^2 - N + (N-1)$$

$$M > N^2 - 1 \qquad (7)$$

Sustituyendo (1) en (7)

$$f_c > \left(N^2 - 1\right) * f_{in} \qquad (8)$$

donde esto implica que la frecuencia del reloj maestro tendrá una dependencia del cuadrado del factor de multiplicación ***N*** . Sin embargo, esta relación puede ser disminuida utilizando para las propuestas **[3]** y **[1]** un circuito de corrección (**figuras 1**,**3** y **4**) o, en este modelo un

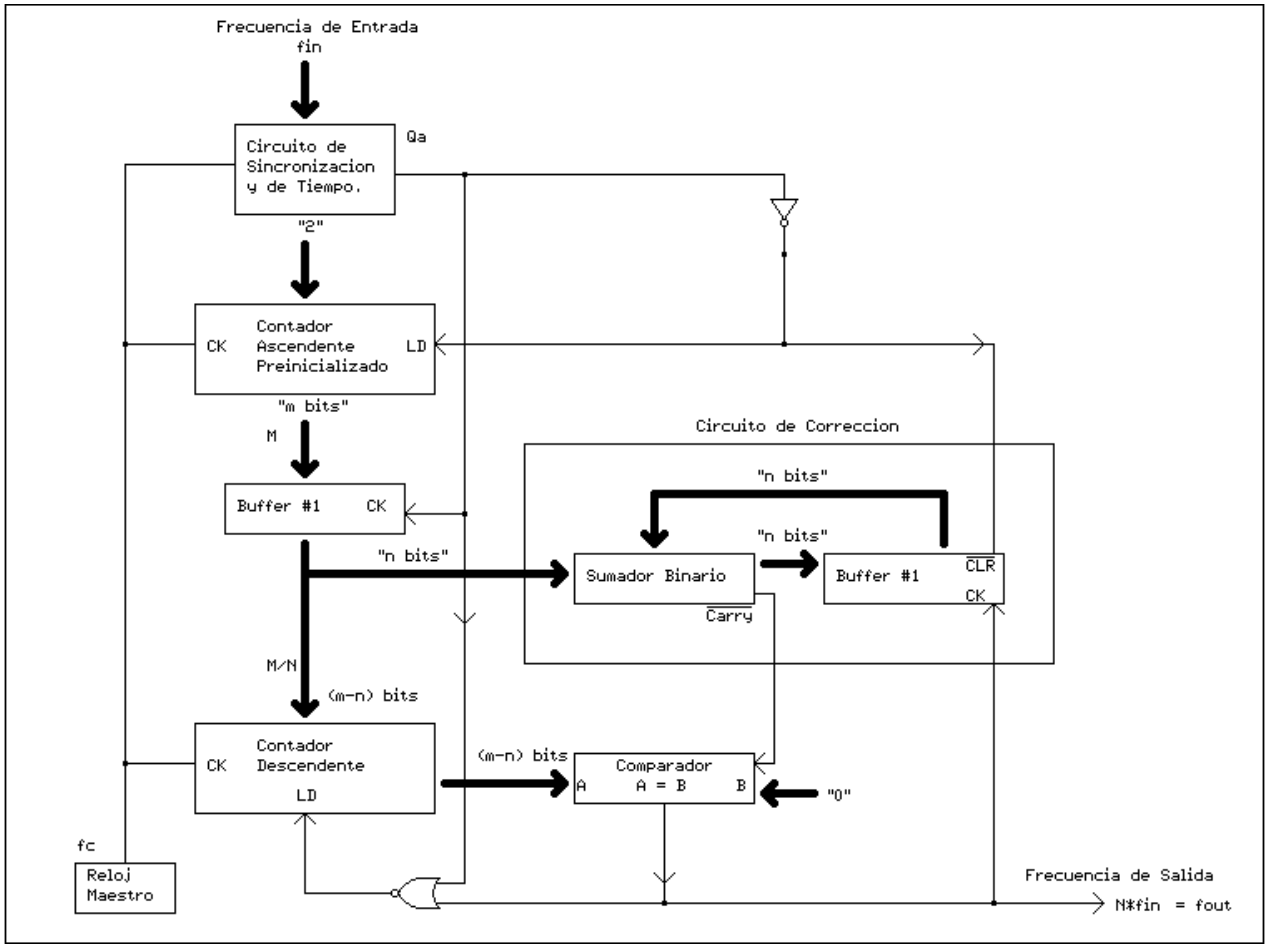


Figura 1
Diagrama a Bloques del Multiplicador de Frecuencia Ref. [3].

divisor en cascada, (**figura 4** y **5**), donde en ambos casos el error máximo admisible no sobrepasará la relación 1/ ***N***; entonces la expresión (4) quedará de la siguiente manera:

$$f'_{out} = \frac{f_c}{\frac{M}{N} - \frac{1}{N}} \qquad (9)$$

y (5) quedaría entonces como:

$$E_{rmax} = \left(\frac{f'_{out} - f_{out}}{f_{out}} \right) < 1/N$$

Desarrollando (9) se tiene:

$$f'_{out} = \frac{N * f_c}{M - 1} \qquad (9.1)$$

En la figura (5) se puede apreciar que el error asociado de la relación fc/***N*** estará dado por ±1 pulso del reloj maestro *fc* , pudiendo ser compensado en el factor de multiplicación ***N*** . Esta compensación es posible debido a que el factor de multiplicación se pasa en binario a través de *m Latch's,* donde ***N*** = (***N***)$_2$, dicha cuenta es pasada a *m contadores*, los cuales tienen la particularidad de contar de manera ascendente o descendente, y al concluir la cuenta determinan el valor de *fdiv* = *fc*/***N*** .

La cuenta a transferir a los contadores podrá estar dada por:

Para modo *Upcounter*

[(2m - 1) - (***N***)2] -1 ;

y para modo *Downcounter*

(***N***)2 +1 ;

El período de fdiv estará definido por la expresión:

1/ fdiv = tdiv
tdiv = ***N*** /fc
tdiv = ***N*** * tc

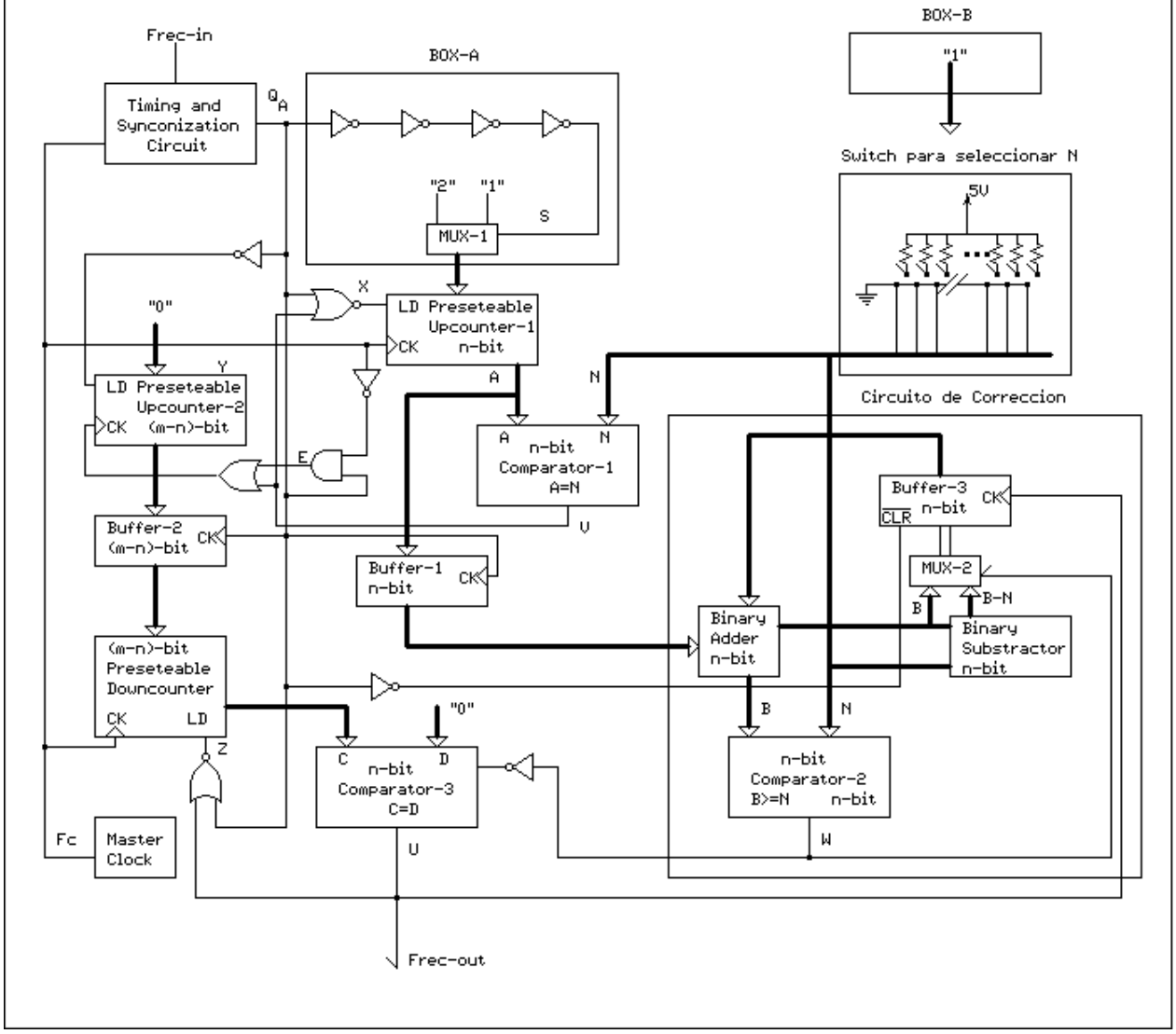


Figura 2
Diagrama a Bloques del Multiplicador de Frecuencia Ref. [1].

donde tdiv y tc son los períodos de el divisor en cascada y del reloj maestro, respectivamente.

De la figura (4), se tiene que la expresión

$$f_{out} = \frac{f_{in} * f_c}{f_{div}} = \left(\frac{f_{in}}{f_{div}}\right) * f_c$$

la expresión se deriva en:

$$\frac{1}{f_{out}} = \left(\frac{1/f_{div}}{1/f_{in}}\right) * \frac{1}{f_c}$$

$$t_{out} = \left(\frac{t_{in}}{t_{div}}\right) * t_c \qquad (10)$$

Sí *Cin* es la cuenta que entregan los contadores y pasa al *buffer#1* de la **figura 4**, entonces:

$$C_{in} = \frac{t_{in}}{t_{div}} = \frac{t_{in}}{N * t_c}$$

$$f_{out} = \frac{1}{C_{in} * t_c} \qquad (11)$$

y la frecuencia de salida se tendrá pasando la cuenta a los contadores

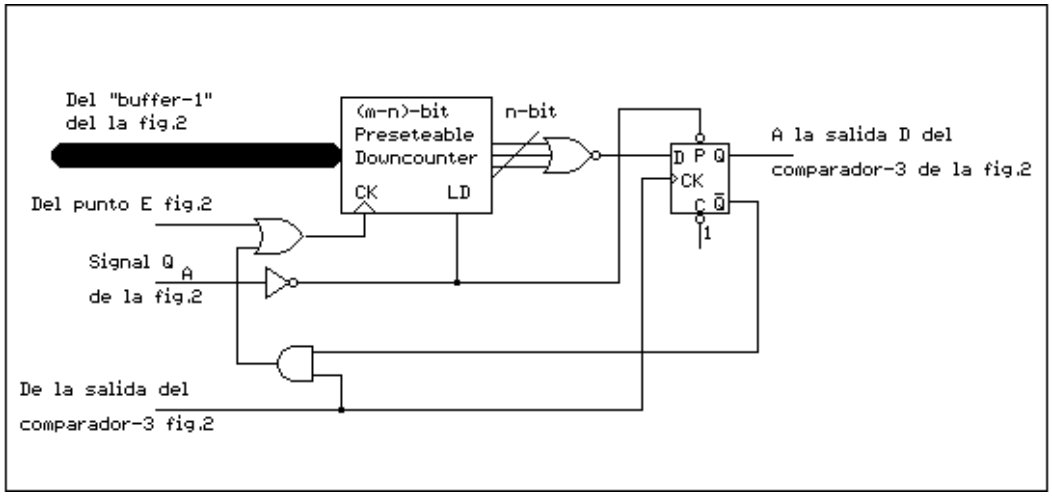


Figura 3
Diagrama a Bloques del Circuito de Corrección del Multiplicador de Frecuencia Ref. [1].

descendentes *downcounter#1* a una frecuencia de reloj **fc**, como se define en la expresión (10) y (11). El error máximo asociado a esta propuesta estará definido de igual manera como para las propuestas **[1]** y **[3]**, quedando como sigue:

$$E_{rmax} = \left(\frac{f'_{out} - f_{out}}{f_{out}} \right) < 1/N$$

tomando las expresiones (9.1) y (2.1) se tiene:

$$E_{rmax} = \frac{1}{M-1} \qquad (12)$$

donde:

$$E_{rmax} \cong \frac{1}{M}$$

lo cual permite establecer que:

$$\frac{1}{M} < \frac{1}{N}$$

donde se fija que el número aceptable de pulsos de salida por período de entrada es igual a ***N*** , entonces la desigualdad deberá satisfacer:

$$M > N \qquad (13)$$

lo cual implica que la frecuencia de reloj maestro estará dado por:

$$\frac{f_c}{f_{in}} > N$$

$$f_c > N * f_{in} \qquad (14)$$

Comparando con la expresión (8) se tiene que el tamaño de la frecuencia de reloj fc se reduce por un factor de ***N*** y así mismo el error para una frecuencia dada junto con el error relativo.

De la expresión (11) se define entonces que la frecuencia del reloj maestro :

$$f_{out} = \frac{f_c}{C_{in}}$$

donde:

$$C_{in} = \frac{f_c}{f_{out}}$$

y

$$t_{div} = \frac{t_{in}}{C_{in}}$$

teniéndose que:

$$f_{div} = \frac{C_{in}}{t_{in}} \qquad (15)$$

$$f_{div} = C_{in} * f_{in}$$

$$C_{in} = \frac{f_{div}}{f_{in}}$$

resultando:

$$\frac{f_{div}}{f_{in}} = \frac{f_c}{f_{out}} \qquad (16)$$

En la expresión (16) se establece la relación entre los dos principales contadores y la transferencia de la cuenta de la etapa de entrada a la de (13) salida.

De lo anterior se puede determinar el tamaño de la cuenta y como esta influye en la definición del reloj maestro y la frecuencia de entrada.

De la expresión (15), se tiene:

$$C_{in} * f_{in} = \frac{f_c}{N}$$

reordenando se tiene:

$$N * C_{in} = \frac{f_c}{f_{in}}$$

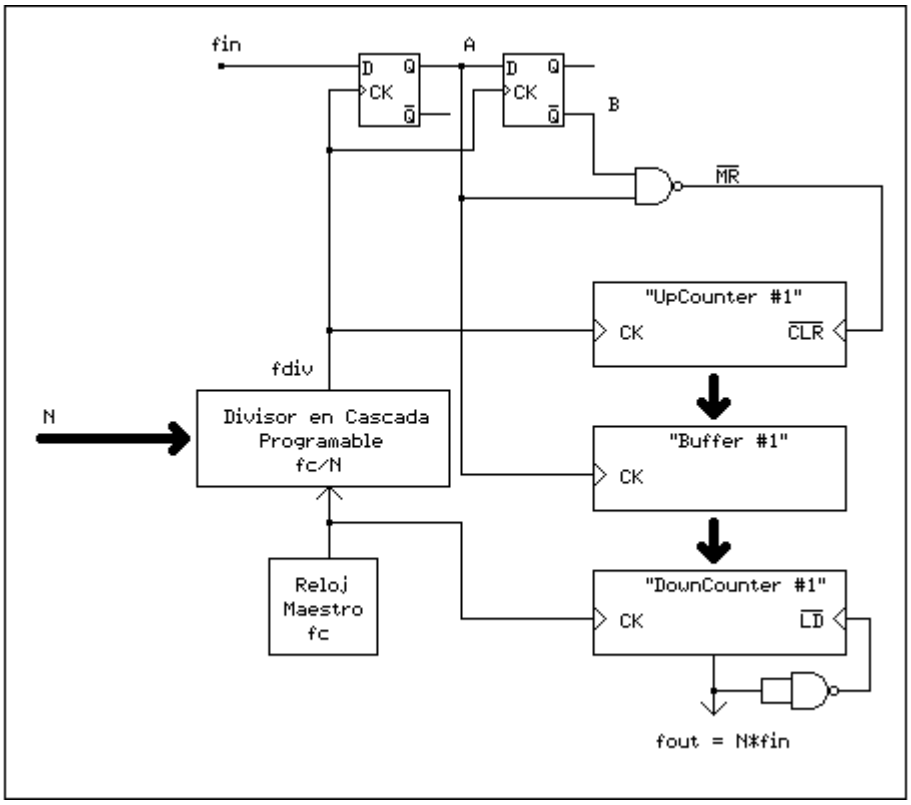


Figura 4
Diagrama a Bloques del Circuito del Multiplicador de Frecuencia de la presente propuesta de este documento.

de (1)

$$N * C_{in} = M$$

resultando

$$C_{in} = \frac{M}{N} \qquad (17)$$

La resolución del multiplicador digital de frecuencia reside precisamente en la cuenta que se tiene de la expresión (17) y de la cual se deriva la limitación de este mismo. Lo anterior establece que si la cuenta esta definida por *m* bits, la máxima cuenta a alcanzar será 2^m -1, por tanto los contadores y registros tendrán la siguiente expresión:

M = ***N*** * (2m-1) (18)

donde

fc = ***N*** * (2m-1) * fin

y

fdiv = (2m-1) * fin (19)

La particularidad de esta propuesta se vuelve a mostrar en la expresión (19), donde el uso de un divisor en cascada del reloj maestro permite que el registro de los contadores se limite a solo el tamaño de la cuenta *Cin,* a diferencia de lo expresado en (18).

Implementación en Hardware

Como se muestra en la **figura 6**, la implementación del multiplicador digital de frecuencia adaptivo se realizó en principio con lógica discreta utilizando 6 contadores de 8 *bits* , 2 *buffers* y un oscilador de 24 Mhz como reloj maestro, sin embargo, la alimentación del reloj maestro se puede realizar por medio de un generador de pulsos. La GAL22V10 se utilizó para realizar una división entre 2,4,8 y 16 a la frecuencia del reloj maestro, de tal forma que la alimentación de los contadores usados como divisores pudiese alcanzar la cuenta en los 16 *bits* disponibles; este subsistema constituye el divisor en cascada del reloj maestro definido en la expresión (1.2). La frecuencia resultante *fdiv* se obtiene de ubicar una cuenta inicial del factor de multiplicación en los divisores a través de *dip-switchs* (16), tomando en cuenta la división previa seleccionada en la GAL22V10 por medio de los *puentes* JMP1 (véase Apéndice

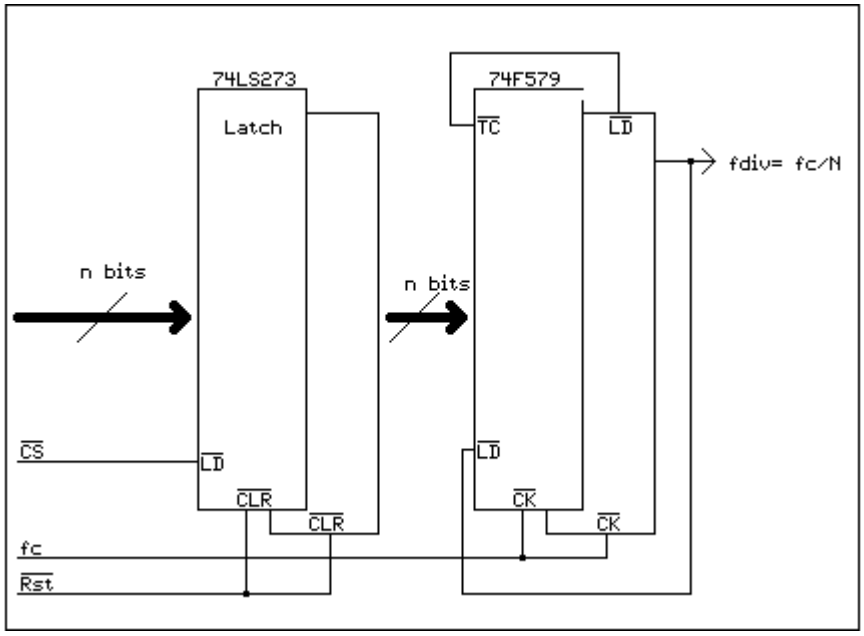


Figura 5
Diagrama a Bloques del Circuito Divisor en Cascada Programable del Multiplicador de Frecuencia de la propuesta de este documento.

A), lo cual implica que la cuenta resultante será el resultado de ***N/2, N/4, N/8 y N/16*** . Para aplicaciones acotadas a 16 *bits*, se puede omitir la división previa y utilizar los divisores a su máxima resolución.

La frecuencia *fdiv* es alimentada a los contadores ascendentes *up-counter #1* , los que iniciaran su cuenta con una transición de *fin* de *alto-a-bajo* y concluirán la cuenta con una transición de *bajo-a-alto* de *fin*, un tiempo después se producirá una *puesta-a-cero* de los contadores para reiniciar la cuenta.

La cuenta generada ***Cin*** se transfiere a los *buffers* y un tiempo después será transferida a los contadores descendentes *downcounter #1*, que a una frecuencia de reloj *fc*, son decrementados en uno en cada ciclo, alcanzando la frecuencia de salida *fout*.

La aplicación aquí propuesta es factible a ser implementada en un dispositivo FPGA, lo cual permitirá reducir la solución final a solo 2 dispositivos, por otro lado permitirá incrementar la frecuencia de operación de los dispositivos de 40 Mhz a 80 Mhz.

EMAIL: dlap @ xamum , uam , mx

Referencias.

[1] Masud Mahmud, Syed. "A Progammable Self-Adaptive Digital Frequency Multiplier," IEEE Trans. Instrum. Means., vol. 37. no. 2, pp237-230, June 1988.

[2] Lo, H.Y. and Lu, J.H. "A simple design for a digital programmable frequency multiplier, " Int. J. Electron., vol. 46, no. 5, pp 535-542, Dec. 1979.

[3] Boutin, N. and A. Boucher, "A novel digital frequency multiplier," IEEE Trans. Instrum. Means., vol. IM-35, no. 4, pp. 556-570, Dec. 1986.

[4] Bilgic,O. "A synchronous frequency multiplier using phase-locked loop," Int. J. Electron., vol. 52, pp. 569-573, 1982.

[5] Muniappan, K. and Kitai, R. "Digital frequency multiplier for spectrum measurement of periodic signal," IEEE Trans. Instrum. Meas., vol. IM-29, no. 3, pp. 195-198, Sept. 1980.

[6] Gimmel, B.A. "A digital method of frequency multiplication and its application to numeric spectrum analysis of periodic signal," IEEE Trans. Instrum. Meas., vol. IM-26, no. 2, pp. 181-183, June 1977.

[7] Krishnamurthy, K.A., G.K. Dubey, and G.N. Revankar, "A simple frequency multiplier," Int. J. Electron. Vol. 43, no. 2, pp. 201-205, 1977.

[8] Even, R.K. "A modified novel frequency multiplication technique." IEEE Trans. Circuits Syst., vol. CAS-22, no. 12, pp. 954-959, Dec. 1975.

[9] Seth, S. et al., "A low-cost pulse frequency multiplier," Proc. IEEE, vol. 66, no. 11, pp. 1578-1580, Nov. 1978.

[10] Boutin, N. et al., "A bit-rate multiplier for digital data processors, " in Proc. Electronicom '85 (Toronto, Canada), Oct. 7-9, 1985, pp. 470-473.

Computación de Usuario Final: ¡ Cambiar o morir ! Definiendo el nuevo paradigma

Benito Piñeiro, Omar Huerta y Juan Carlos Corral
Alumnos del ITESM
Colaboración externa.

Desde el momento en que el fenómeno de la Computación de Usuario Final (CUF) apareció en las organizaciones debido al problema de backlog (retroalimentación de aplicaciones), al crecimiento de la familiaridad con las computadoras o a que el concepto se encuentra en su segunda fase (proliferación), comenzó a crecer y a ser utilizado por más profesionistas, ejecutivos, administradores, oficinistas y vendedores. A partir de su aparición y a través de su desarrollo la computación de usuario final ha sufrido diversos cambios y mutaciones en su forma y fondo que a últimas fechas han llevado a los encargados de la función de información a un punto sin retorno, en el que deben de tomar una decisión con respecto a la CUF, de la cual depende que la función de los Sistemas de Información no se convierta en un lastre para la empresa, llevándola fuera del mercado para siempre.

¿Qué es la CUF?

La computación de usuario final se puede definir como la adopción y uso de la tecnología de información por personal que se encuentra fuera del departamento de sistemas de información para agilizar sus funciones y tareas y de manera más importante para ***desarrollar*** aplicaciones de software encaminadas a apoyar las tareas organizacionales. Las aplicaciones de software pueden ser usadas tanto por los mismos desarrolladores como por sus compañeros de trabajo; estas aplicaciones se pueden clasificar en diferentes tipos, desde muy sencillas como modelos financieros hasta complejos sistemas de información. La computación de usuario final toma lugar mayormente en los tres niveles inferiores de la organización:

Nivel departamental
Nivel de trabajo de grupo
Nivel individual

Existen 12 categorías de uso (ver cuadro uno) para la CUF, algunas de estas no son todavía muy comunes, pero se irá incrementando su uso conforme los usuarios se familiaricen con las actividades comunes y desarrollen más aplicaciones creativas.

1. **Como una ayuda de contabilidad, reportes y cálculos.**
2. **Como una ayuda para escritura.**
3. **Como ayuda de búsqueda y recuperación.**
4. **Como una ayuda para comunicaciones.**
5. **Como una ayuda para presentación.**
6. **Como una ayuda para planeación, calendarización y monitoreo.**
7. **Como una ayuda de memoria.**
8. **Como un ayuda de procesamiento.**
9. **Como una ayuda de aprendizaje.**
10. **Como una ayuda en el desarrollo de nuevos programas.**
11. **Como una ayuda en la toma de decisiones.**
12. **Como una ayuda de análisis.**

Cuadro uno.
Categorías en el uso de CUF

Algunos ejemplos de las actividades específicas de cada categoría pueden ser:

Como una ayuda de contabilidad, reportes y cálculos.

Se pude usar una hoja de cálculo para realizar la planeación financiera, estimar presupuesto, llenar reportes de gastos y ventas; calcular amortizaciones y distribución de costos.

Como una ayuda para escritura.

Para preparar memorándums, notas, minutas de juntas e incorporar datos obtenidos de una búsque-

da de archivos; para utilizar un corrector de escritura, para editar y revisar documentos propios o escritos por otras personas.

Como ayuda de búsqueda y recuperación.

Para búsqueda en bases de datos o en archivos muy grandes y la recuperación de la información seleccionada, búsqueda en archivos de correspondencia y recuperación de cartas y memorándums y búsqueda de información comercial y recuperación de la información deseada.

Como ayuda para comunicaciones.

Envío, recepción y contestación de mensajes computacionales, realizar arreglos de viajes utilizando los servicios de una red, uso de conferencias por computadora. Envío, recepción y contestación de mensajes de voz digitalizada.

Como ayuda para presentación.

Uso de las gráficas computacionales para preparar una presentación o animación de una propuesta.

Como ayuda para planeación, calendarización y monitoreo.

Desarrollo y revisión de agendas, planeación de juntas y asistencias, desarrollo y revisión de planes de trabajo, uso de paquetes de control y administración de proyectos, uso de paquetes para la planeación financiera y administrativa; monitoreo de ordenes de clientes y sus pagos.

Como ayuda a memoria.

Utilización de un calendario electrónico, mantener un archivo de datos generales de clientes y proveedores en línea.

Como una ayuda de procesamiento.

Para realizar rutinas y transacciones de entradas, realizar transacciones de procesamiento completas, no limitándose a las transacciones de entrada.

Como ayuda de aprendizaje.

Utilización de paquetes de entrenamiento basados en computadoras.

Como ayuda en el desarrollo de nuevos programas.

Desarrollo de sistemas de aplicaciones, desarrollo de sistemas de soporte a las decisiones.

Como ayuda en la toma de decisiones.

Para predicción, análisis, búsqueda y simulación de paquetes que ayuden a la comprobación de una hipótesis, así como la ayuda de paquetes de sistemas expertos que agilicen la toma de decisiones

Como ayuda de análisis.

Para analizar la información de la venta a clientes y definir patrones de compra; para agilizar la labor de los vendedores, para mejorar las políticas de crédito y analizar información de compras para detectar tendencias en los costos; mejorar entrega de los proveedores y analizar la varianza del plan inicial o presupuesto inicial.

Es difícil decir que estos usos, en general, se estén llevando a cabo hoy en día. Pero hay que mencionar que a principios de los años 90's, surgieron nuevos e importantes usos de la computación de usuario final y se ha dado y continuará dándose un acelerado crecimiento de la CUF.

Crecimiento de la CUF

Uno de los aspectos de mayor importancia de la CUF es su increíble tasa de crecimiento, considerándose el segmento de mayor rapidez de crecimiento de las actividades de sistemas de información en un gran número de organizaciones. Un estudio de Nolan y Norton Corp., llevado a cabo en E.U.A. reveló que un 55% de los empleados de las Compañías utilizan una microcomputadora, de ese 55% un 70% está conectado a una red local; este estudio también revelo que desde 1989 a 1993 el uso de la computadora se duplicó y el de las redes se cuadruplicó. Existen diversas teorías que explican este crecimiento:

La creciente familiarización con las computadoras

Nuestra vida diaria es bombardeada cada vez más con términos computacionales. Los automóviles tienen *tableros computarizados*, las VCR's pueden ser *programadas* para grabar en días posteriores programas de TV, los relojes y los aparatos de sonido son *digitales* y los juegos *computarizados* son parte de la vida de nuestros jóvenes.

Esta familiarización con aparatos programables, así como el bajo nivel de habilidad necesario para operar una terminal o computadora, e inclusive al hecho de que las computadoras se han convertido en símbolos de status ha contribuido a que secretarias, profesionistas, administradores y ejecutivos utilicen cada vez más la tecnología de información y desarrollen sus propias aplicaciones.

Más computadoras en las escuelas

A mediados de los 80's se introdujo a los planes de estudio de las preparatorias y universidades (y recientemente en las secundarias de nuestro país), una educación computacional, debido a esto los jóvenes empresarios y ejecutivos que actualmente forman parte de la población económicamente activa, requieren y promueven el uso de la computación en sus centros de trabajo. Esta es una tendencia que seguramente se irá incrementando con el paso del tiempo.

Teorías del crecimiento

Según diversos teóricos (entre ellos Nolan,) después de una fase introductoria el uso de la tecnología de información (y con ello la CUF) se encuentra en una segunda fase que se caracteriza por la experimentación, aprendizaje y proliferación de la misma. Es durante esta fase que el uso de la tecnología parece expandirse muchísimo más que lo esperado. El cambio repentino de la renuencia a la utilización de esta tecnología, se debe principalmente a la aceptación y uso por parte de los líderes organizacionales que promueven su uso en los niveles inferiores.

El problema de Backlog

La necesidad de la CUF se convierte en una realidad desde que satisface la producción de información tan rápido como las líneas organizacionales la van necesitando y por entender la naturaleza del negocio; por ejemplo, un contador desarrolla sus propias aplicaciones mejor de lo que lo haría un especialista de sistemas, porque conoce mejor sus funciones, además de que está comprometido con su aplicación y por ésto le da mayor uso del que le daría si la aplicación hubiera sido realizada por cualquier otra persona.

Incremento en la velocidad del hardware y la proliferación de las redes locales

El incremento en la velocidad del hardware propició que mucha gente se ocupara de sus aplicaciones y obtuviera respuesta inmediata de éstas, en lugar de que otra persona la ejecutara y después de un tiempo le entregara un reporte con los resultados de la aplicación. De la misma forma propicio la proliferación de las redes locales, pues estas se hicieron mas eficientes, con la información disponible en red se podía tener acceso a toda la información necesaria en cualquier escritorio de cualquier edificio de una empresa. Esto proporcionaba gran ventaja y por lo mismo todo mundo quería tener acceso a la información de manera inmediata, motivándose en gran medida que el usuario final buscara, consultara, y procesara la información necesaria para desempeñar sus funciones.

Una gráfica publicada en el libro «Local and Metropolitan Area Network», sobre un estudio realizado en E.U.A. muestra que de la información total generada por un departamento el 50% de la misma se usa dentro de ese departamento, el 25% se usa en un área de 600 pies, el 15% se usa dentro del mismo establecimiento pero por otros departamentos de otras divisiones, y el 10% salía a otro edificio. Esto nos hace pensar que sería muy problemático dar a todo el mundo la información que necesita aun estando dentro del mismo departamento; es por eso la necesidad de que la información este disponible en red y que la pueda accesar quien la necesite por si mismo sin necesidad de solicitarla al departamento de sistemas.

Otras razones para el crecimiento actual y futuro de la CUF son el abaratamiento del equipo, lo que permite que las empresas adquieran capacidad computacional con una característica muy importante: la llegada de software extremadamente fácil de usar (como los lenguajes de cuarta generación). Un estudio publicado en el libro «Fourth-Generation Languages» estima que si no se hubieran creado los lenguajes de cuarta generación, para 1986 se hubieran necesitado 148 millones de programadores solo en E.U.A.; para crear las aplicaciones existentes a esa fecha, cada persona que trabajará necesitaría emplear un programador, esto nos da una idea de la importancia de que cada usuario desarrolle sus propias aplicaciones, objetivo que se ve apoyado por el software antes mencionado.

El viejo paradigma

El uso de la tecnología de información se había manejado de una forma centralizada, es decir, todas las aplicaciones y programas se desarrollaban por un grupo de especialistas que diseñaban, creaban e implantaban el software en equipo que pertenecía a su propio departamento; podían existir algunas terminales, pero en ellas se limitaban las acciones a teclear datos de entrada y recibir los resultados del procesamiento de esos datos. Este tipo de administración trajo como consecuencia una falta de entendimiento entre los usuarios (clientes) y los usados (equipo de desarrollo); este distanciamiento provocó que la imagen del departamento de sistemas fuera negativa con respecto a la satisfacción y productividad.

Como ya se mencionó, la tecnología computacional ha avanzado a pasos agigantados, lo que dio lugar al surgimiento de la CUF, pero toda-

vía el control era llevado por el departamento de sistemas de información que algunas veces dictaba que paquetes se podían usar y cuales no, así como que tipo de aplicaciones eran las más convenientes para ser desarrolladas por los usuarios. Este medio de administrar la CUF mutilaba en gran medida la creatividad y el entendimiento de hasta que punto se podían desarrollar sistemas complejos por los mismos usuarios; otras tantas veces se iniciaba la generación de software sin el consentimiento del departamento de SI, provocando un caos computacional. Esta última forma es la que prevalece actualmente en la mayoría de las empresas mexicanas que utilizan consciente o inconscientemente el concepto de CUF. Sin embargo CUF en México, como muchos otros aspectos de la tecnología de información, es un área de controversia, ya que hay empresas muy adelantadas en el manejo del concepto y muchas otras que están más rezagadas. Así encontramos organizaciones mexicanas cuya CUF se encuentra en la etapa de control; pero la gran mayoría se encuentra en la etapa de contagio.

La mayoría de los autores como McKenney, McFarlan y Rockart, coinciden en que hay dos formas de implementar la CUF dentro de las organizaciones. La primera hace énfasis en tener un control muy fuerte en la iniciación de la CUF, siendo ésta la relevancia principal del departamento de sistemas; es decir, se debe decir hasta dónde y cómo pueden llegar los usuarios con sus aplicaciones, dando un soporte técnico bajo, que se irá incrementando con el tiempo y madurez de los usuarios. El otro enfoque es contrastante con el primero, enfocándose en un amplio soporte técnico, dándole oportunidad a los usuarios de compenetrarse con las distintas posibilidades que tienen para el desarrollo de sus trabajos y aplicaciones, dándose el control en las etapas finales o de maduración. Este último es el enfoque más cercano a lo que ha sucedido en las empresas mexicanas, ya que el origen de la CUF en éstas se llevo a cabo a través del surgimiento de islas de tecnología, es decir del desarrollo de aplicaciones aisladas por parte de los departamentos de los usuarios, podemos decir que la CUF en las empresas mexicanas surgió al azar, sin ningún tipo de control, por lo que ahora urge que se establezcan los controles adecuados y se realicen los cambios necesarios en la administración del departamento de SI para que CUF se convierta en una verdadera fuente de productividad para la empresa.

Nuevos retos, nuevo rol

Tomando en cuenta lo mencionado anteriormente consideramos que el reto para los departamentos de SI de las empresas mexicanas consiste en promover la CUF, integrarla a la planeación de sistemas y llevar la administración del departamento de SI con un enfoque hacia el usuario final.

Entonces, el nuevo rol que el departamento de SI tiene que tomar para lograr sobrevivir a esta nueva época de cambios, está englobado en los siguientes aspectos:

Un fenómeno que debe darse en los procesos de información es que los usuarios de las empresas tomen el control del desarrollo de sistemas, es decir que el departamento de sistemas esté al servicio del usuario final, tomando en cuenta las necesidades de este.

Los usuarios deben tener una participación activa definiendo las necesidades de los sistemas, también deben participar en la decisión de donde colocar los datos; los usuarios indicarán como tiene que trabajar el sistema.

El desarrollo de sistemas cambiará de un cálculo multianual a un ciclo de meses, pues al realizar cada quien sus aplicaciones, éstas podrán iniciar inmediatamente sin necesidad de esperar a que lo atienda el departamento de sistemas.

El objetivo principal de los sistemas de información ha sido siempre asegurar el crecimiento de los recursos de información para la empresa. Por lo tanto es productivo tomar al usuario como un productor de ideas y preguntarle que es lo que SI puede hacer para que los recursos de información crezcan como un bien para la empresa.

Un punto muy importante es que el departamento de sistemas debe apoyar y asesorar a los usuarios finales sobre la aplicaciones que se realicen. El rendimiento es un reto, si SI está involucrado como parte de su trabajo puede enfocarse en el "benchmarketing" y evaluar el rendimiento; si no se encuentra involucrado es común tener problemas de rendimiento.

Un área en la que puede necesitar asistencia el usuario es en los sistemas de integración, ya que no tiene la apreciación en definir los datos de interface. Una vez integrados los sistemas se debe determinar quien tiene acceso y a que parte tiene acceso.

El usuario debe de estar asistido por SI en la fase de pruebas de una aplicación, en este punto es importante la experiencia de SI.

Le compete a SI tomar las decisiones fundamentales de la arquitec-

tura técnica que utilizará la empresa. SI debe dar opciones sobre qué tipo de software se debe de usar en la empresa, el usuario debe de escoger la opción u opciones (máximo 2) que más se le faciliten o con las que tenga mas afinidad; se recomienda que se use un solo tipo de software por motivos de estandarización y de gastos de capacitación. Hay que enfatizar que estos cambios, orientados a la CUF, en la forma de trabajar del departamento de SI son hoy en día necesarios; pero en un futuro próximo serán indispensables para que la empresa logre mantenerse en la lucha en el ambiente agresivamente competitivo de la globalización.

Aspectos administrativos

Guiar y proporcionar soporte a la computación de usuario final representa el nuevo reto para los encargados de sistemas de información. Hay cinco áreas que necesitan políticas y soporte:

1. Hardware
2. Software
3. Datos
4. Comunicaciones y
5. Entrenamiento.

La administración tendrá que pisar la delgada línea divisoria entre control y soporte en el momento de establecer las políticas de la CUF. El control puede ahogar la creatividad y experimentación del usuario final, lo cual no es muy deseable. Por otra parte, la ausencia de guías en el uso de las computadoras puede ser dañina y originar incompatibilidad, que como ya se mencionó es lo que esta sucediendo en muchas empresas mexicanas; incompatibilidad que se traduce en un gasto excesivo para la organización. Además de esto los administradores de sistemas de información deben tener en mente lo que puede pasar en el futuro y anticipar que es lo que van a querer los usuarios, considerando que la mayoría de los empleados se convertirán en usuarios computacionales directos eventualmente. ¿Qué políticas se deben establecer para enfrentar un futuro inminente como el aumento de la CUF y la necesidad del cambio en la forma de manejarla?. Nosotros sugerimos algunas para manejar el software en la CUF.

Software

¿Quién va a desarrollar las aplicaciones? Un criterio para determinar cuando los usuarios finales desarrollen determinadas aplicaciones es la *sensibilidad* de la aplicación, o sea el riesgo de fraude o la revelación de información importante a personas no autorizadas. Por ejemplo, un departamento se propone desarrollar una aplicación para monitorear sus compras, dinero o inventario. En tal caso, el departamento de SI y el grupo de auditoría querrán investigar la *sensibilidad* de la aplicación. Si ésta va a ser desarrollada, operada y utilizada por uno o pocos individuos, la compañía va a exponerse a problemas. Siendo así, probablemente la administración decidirá que la aplicación la desarrolle el departamento de sistemas. Podemos decir que las aplicaciones sensitivas no deben ser desarrolladas por los usuarios finales y menos si van a correrse en sus propias computadoras.

¿Quién es el dueño del software? Se debe establecer una política para determinar la propiedad del software desarrollado por los usuarios, la política más recomendable es que si los usuarios finales desarrollan sus aplicaciones en sus horas de trabajo y en el equipo de la empresa, entonces esas aplicaciones pertenecen a la empresa.

¿Cómo obtener el mayor provecho de la CUF?

Gary Gulden, de Index Systems, Inc., identifica una estrategia de tres elementos para administrar la CUF que es utilizada por la mayoría de los clientes de su compañía y que les ha proporcionado grandes beneficios:

1. Proveer suficiente poder de mainframes y microcomputadoras para satisfacer la demanda de los usuarios, mientras se establecen límites de proliferación de tecnologías.

2. Hacer la tecnología de usuario final tan accesible y fácil de usar como sea posible.

3. Marcar criterios tipo costo/beneficio que provean protección contra el uso indebido de la computadora, y protejan a los usuarios finales de sus propios errores.

Existe otro elemento importante para obtener el mayor provecho de la CUF: Buscar usos computacionales de gran apalancamiento (que proporcionen un valor agregado a la empresa y respeten los controles establecidos); si los usuarios aceptan esta meta, entonces las submetas de tener hardware y software compatibles, integridad y seguridad de datos y la efectividad de las telecomunicaciones serán más fáciles de conseguir.

Conclusiones.

Es necesario que en México las empresas se enfoquen a la CUF, para lograr ser competitivas ante los nuevos cambios políticos y económicos de nuestro país.

En un artículo publicado en 1992 por Daniel Cohen y Jorge Ramírez, en la revista «Personal Computing México», se muestra un estudio realizado en México que revela que las empresas mexicanas tienen la siguiente experiencia con la idea de promover el desarrollo por usuario final:

El 11% no lo ha experimentado ni lo considera factible.
El 38% si lo considera factible pero no lo ha hecho
El 5% lo ha experimentado con escasos resultados
El 46% lo ha experimentado con buenos resultados.

Con ésto podemos darnos cuenta que las empresas mexicanas tienen conciencia de la importancia de la CUF y de la falta de competitividad que le puede ocasionar el carecer de esta forma de trabajar el area de SI. Sin embargo es importante que las empresas mexicanas adopten el nuevo rol de SI, recordando que el reto está en el aire y son dos las opciones: CAMBIAR O MORIR…

Referencias

(1) Delligatta, Ann y Umbaugh, Robert E. EUC becomes enterprise computing. Information systems management. Fall 1993

(2) Ryan, Hugh W. User-driven systems development. Information systems management. Summer 1993

(3) Tayntor, Christine B. Customer-driven long-range planning. Information systems management. Fall 1993

(4) Halloran John P. Achieving world-class end-user computing. Information systems management. Fall 1993

(5) Cohen, Karen Daniel y Ramírez Vargas Jorge A. Alternativas para la reducción de costos de informática en México. Personal Computing México. No. 55 1992

(6) Gray, Paul. The end-user paradigm. Information systems management.

(7) Tayntor, Christine B. New challenges or the end of EUC. Information systems management.

(8) Brancheau, James C. & Brown Carol V. The management of end-user computing: status and directions.

Bibliografía

[1] Martin, James & Leben Joe. Fourth-generation languages. Volume II. Prentice-Hall. 1986

[2] Stallings, William. Local and Metropolitan Area Networks. Fourth Edition, Macmillan. 1993.

[3] Wysocki, Robert K. Information Systems Management Principles in Action. Wiley. 1990.

OBSERVACIONES

Si te interesa contribuir con algún comentario o sugerencia, el Consejo Editorial de la Revista Polibits agradecerá utilices este espacio.

Y lo remitas a :

Centro de Investigación Tecnológica en Computación
Consejo Editorial Revista POLIBITS
Av. Té No. 950, Edificio de Graduados 2o. piso.
Colonia Granjas México. C.P. 08400
México D. F.
☎ 6-54-39-32, 6-49-50-36, 6-49-03-66 ext. 327. Fax : 6-57-74-53.

Si tu contribución es en forma de un artículo, favor de anexar los siguientes datos: Nombre del autor, Grado Académico e Institución de procedencia.