Contenido

Editorial

En su afán por mantenerse como un medio de difusión para el desarrollo tecnológico en cómputo del IPN, accesible y de fácil consulta, la revista polibits **a buscado siempre encontrar mecanismos que le permitan una mayor presencia en el ámbito académico, ya sea dentro del Instituto como fuera del mismo. Como resultado de estos esfuerzos,** polibits **cuenta con una presencia nacional indiscutible y, en la medida de nuestras posibilidades, se ha logrado su distribución en algunos paises de Centroamérica.**

Sin embargo, estos esfuerzos no son suficientes, considerando que este medio debe llegar a todo aquel que considere de interés los temas que aquí se tratan, sin que debiera importar la distancia. Por ello, el CIDETEC se planteo la generación de una pagina en la Web, de tal forma que, sin costo alguno, todo aquel que cuente con la posibilidad de conectarse a la Internet tenga la oportunidad de consultar los diversos artículos que aparecen en la versión impresa de la revista. Además de poder consultar el número mas reciente de la misma, se podrá consultar un archivo histórico de todos los ejemplares impresos de polibits **desde su primer número editado en el ahora venerable y prácticamente desconocido editor microstar hasta su versión mas reciente, editado para Internet en el formato PDF de Adobe Acrobat. La dirección de la página del CIDETEC es www.cidetec.ipn.mx y dentro de ella se encuentra la liga a la página de** polibits**.**

Ahora bien, este número en particular lo integran artículos de diversa procedencia, con un equilibrio del 50% entre colaboraciones internas del Centro y colaboraciones externas al mismo, pero todas, salvo una, del

mismo Instituto. Esto es bastante satisfactorio, ya que indica un mayor interés del resto de la comunidad politécnica en emplear nuestra revista como un medio para la divulgación de sus investigaciones y desarrollos tecnológicos, pero desearíamos que fuera de nuestro Instituto también se considerara a polibits **como un medio académico válido para realizar divulgación tecnológica, por lo cual esperamos una participación más activa de la comunidad académica del país, relacionado con el desarrollo tecnológico en materia de cómputo y áreas afines, objetivo principal de este Centro y de la revista** polibits**.**

Zanya: Compositor Evolutivo de Música Virtual

Horacio Alberto García Salas
e-mail itztli@eudoramail.com

La Informática es una disciplina que ha permitido grandes avances en muy corto tiempo y hay que destacar que ha tomado una gran aceleración con la aparición de herramientas como las computadoras y una serie de dispositivos electrónicos que hacen que el mundo en el que vivimos se vuelva cada vez más pequeño.

Prácticamente, todas las áreas del conocimiento humano se han visto apoyadas con el uso de estas nuevas tecnologías, tal es el caso de las bellas artes y en particular de la Música.

De esta manera, tenemos por un lado a la *informática*, que ha revolucionado el mundo drásticamente y por otro lado la *música*, que es tan antigua como el mismo ser humano. Unidas ambas áreas con el paso del tiempo, han creado el marco dentro del cual fue desarrollado este trabajo, "La Música Informática".

En el área de desarrollo informático musical se encuentran diferentes corrientes de desarrollo, así pues encontramos a quienes se dedican a fabricar software de edición musical, reproductores de música, secuenciadores, etc.

Por otro lado, otra de las corrientes, hasta ahora más de tipo experimental, es la dedicada a la obtención de software para la composición musical automática, "el sueño" de la maquina con la capacidad de componer música por siempre exquisitamente diferente.

INTRODUCCIÓN

Con la aparición de los fractales[1] y la aplicación de las teorías de caos[8], el problema de crear una máquina capaz de hacer composición musical automática, se centró en lo natural. Y no es para menos, los hermosos paisajes generados por el científico Bernoit Mandelbrot, han mostrado que la mejor manera de hacer modelos es imitar el comportamiento de la Naturaleza[4].

Aplicando teorías fractales, el científico Richard F. Voss[2], ha hecho desarrollo musical informático, obteniendo compositores verdaderamente hermosos. En su trabajo ha desarrollado generadores de ruido browniano, ruido blanco y resalta uno que genera música muy agradable, que se denomina ruido 1/f[5], que de acuerdo a algunos estudios que ha realizado, música como el jazz, la clásica, el blues etc. se comportan de manera muy similar al ruido 1/f. La cuestión es un poco más trascendente cuando se descubre que muchos fenómenos naturales, como la aparición de manchas solares, el crecimiento de las poblaciones, la formación de nubes, etc. tienen un comportamiento similar al ruido 1/f.

Escuchar el algoritmo para la generación de ruido 1/f es bastante agradable.

Por otro lado, el investigador Fernando Galindo Soria ha desarrollado una herramienta informática que tiene una amplia gama de aplicaciones, a la que ha denominado "Sistemas Evolutivos" [3].

Los Sistemas Evolutivos modelan una de las principales características de la Naturaleza, la Evolución.

La Naturaleza ha mostrado desde hace largo tiempo, que los sistemas que evolucionan son los que tienen la capacidad de adaptarse al medio que los rodea. De tal forma que si lo que se pretende al hacer un modelo es imitar a la realidad, ¿porqué no desarrollar modelos o sistemas con la capacidad de adaptarse al medio que los rodea? En general, este tipo de sistemas son hermosos, basta con echar una mirada a nuestro alrededor para percatarnos de los bellos sistemas evolutivos que la Naturaleza tiene por gusto crear, y es que prácticamente todo a nuestro alrededor evoluciona, sean animales, plantas o minerales, en general sufren modificaciones del medio que los rodea y a su vez modifican al medio que los contiene, creándose procesos de coevolución[6], convirtiendo a la evolución en un proceso Universal.

De esta manera, cuando se desarrolla un sistema evolutivo, hay que pensar en un sistema que tenga la capacidad de interrelacionarse con el

medio ambiente y que aprenda a través de él.

Aplicando esta filosofía, se han desarrollado impresionantes Sistemas Evolutivos, algunos de ellos aplicados a la generación de paisajes.

Aplicando la misma filosofía, en este trabajo presentamos el desarrollo de Zanya, un Sistema Evolutivo que tiene la capacidad de aprender y evolucionar de manera permanente, con el fin de hacer composiciones musicales.

1. Sistema Compositor

1.1 Generación de las Notas Musicales

Como primer paso en el desarrollo de un sistema informático musical, como el que se presenta en este trabajo, es interesante hablar de las notas musicales, que representan la base sobre la cual está sustentada la música.

En términos de la Física, las notas musicales son ondas que están comprendidas en un rango de frecuencias, relativamente pequeño, para el cual el oído humano está perfectamente adaptado. El ser humano tiene la capacidad de escuchar las frecuencias de entre 20 y 20000 Hz[7] y sólo ciertas frecuencias se consideran como notas musicales puras. Cada instrumento musical emite las frecuencias puras de las notas musicales acompañadas por algunas otras frecuencias, denominadas armónicas, que permiten distinguir cuando se trata de un piano, un violín, un saxofón o cualquier otro de los instrumentos musicales que existen.

De esta forma lo primero que se necesita construir es una representación de las notas y se puede lograr almacenando en un arreglo los valores de las frecuencias de las notas, creándose de esta manera un "piano virtual" sobre el cual se pueden interpretar melodías y desde luego, hacer composiciones musicales.

Do	#Do	Re	#Re	Mi	Fa	#Fa	Sol	#Sol	La	#La	Si
65	69	73	77	82	87	92	98	103	110	116	123
130	138	146	154	164	174	184	196	206	220	232	246
260	276	292	308	328	348	368	392	412	440	464	492
520	552	584	616	656	696	736	784	824	880	928	984
1040	1104	1168	1232	1312	1392	1472	1568	1648	1760	1856	1968

Tabla 1 Frecuencias de las notas musicales en ciclos/seg.

Cuando se duplica el valor de una nota, *Do* por ejemplo, que tiene una frecuencia de 65 ciclos/seg, se obtiene una nota que se encuentra una octava o escala más aguda, sin embargo, esta nota también es *Do*, sólo que con una frecuencia de 130 c/s y si volvemos a duplicar este valor, se obtiene otro *Do*, pero aún más agudo, con una frecuencia de 260 c/s. De igual forma se puede hacer con todas las notas musicales. En la tabla 1 se pueden ver los valores de las frecuencias de 5 escalas cromáticas.

1.2 Mecanismo General de Composición

A continuación se describirá la forma en la que se generan las composiciones musicales.

Para representar la información musical se utilizan dos matrices una que se ha denominado Matriz Evolutiva Aleatoria y otra que se obtiene a partir de esta, llamada Matriz Evolutiva de Frecuencias Acumuladas. Dichas matrices son arreglos de 60 renglones por 60 columnas, sin embargo, para la explicación de su funcionamiento, utilizaremos matrices de 7 renglones por 7 columnas.

a) Construcción de la Matriz Evolutiva Aleatoria

Para ejemplificar el funcionamiento de estas matrices, primeramente, se construye un arreglo cuadrado llamado Matriz Evolutiva Aleatoria, utilizando como etiquetas de las columnas y renglones las notas musicales. Después, se llenarán algunas de las casillas con números aleatorios como se ve en el ejemplo de la figura 1.

b) Construcción de la Matriz Evolutiva de Frecuencias Acumuladas

A partir de la matriz evolutiva aleatoria se construye la Matriz Evolutiva de Frecuencias Acumuladas y se hace agregando una columna llamada totales, que originalmente está llena de ceros, a la derecha en la matriz evolutiva aleatoria. En cada renglón vamos a hacer un recorrido de izquierda a derecha ignorando las casillas con valor cero.

	Do	Re	Mi	Fa	Sol	La	Si
Do	0	0	90	0	30	40	0
Re	10	0	15	45	20	70	80
Mi	5	25	0	60	0	30	90
Fa	0	80	10	40	15	95	30
Sol	25	15	0	40	65	70	0
La	0	35	5	10	0	0	0
Si	20	30	0	0	60	0	70

Fig 1. Matriz Evolutiva Aleatoria

El primer número (más a la izquierda) del renglón, que se encuentre diferente de cero se suma a totales y el resultado sustituye a este primer número.

El segundo número diferente de cero se suma a totales y el resultado sustituye al segundo número.

Así sucesivamente con todos los valores diferentes de cero, el i-ésimo número diferente de cero se suma a totales y el resultado sustituye al i-ésimo número. Agotados estos, la suma debe quedar almacenada en la columna totales del renglón.

Por ejemplo, el renglón original es

	Do	Re	Mi	Fa	Sol	La	Si	Totales
Do	0	0	90	0	30	40	0	0

Y al aplicar el algoritmo queda:

			0+90		90 +30	120+40		90+30+40
	Do	Re	Mi	Fa	Sol	La	Si	Totales
Do	0	0	90	0	120	160	0	160

De esta manera, al recorrer de izquierda a derecha cada uno de los renglones de la matriz evolutiva aleatoria, en la columna totales de la matriz evolutiva de frecuencias acumuladas quedará almacenada la suma de los valores diferentes de cero de cada renglón y en cada cada casilla con valor diferente de cero se almacenará el valor acumulado durante el recorrido.

En esta forma se obtiene la matriz evolutiva de frecuencias acumuladas, como se ve en la figura 2.

c) Generación de la Música

Los números contenidos en la matriz anterior, representan la probabilidad de pasar de una nota a otra y se van a utilizar para hacer la composición musical mediante el siguiente proceso:

	Do	Re	Mi	Fa	Sol	La	Si	Totales
Do	0	0	90	0	120	160	0	160
Re	10	0	25	70	90	160	240	240
Mi	5	30	0	90	0	120	210	210
Fa	0	80	90	130	145	240	270	270
Sol	25	40	0	80	145	215	0	215
La	0	35	40	50	0	0	0	50
Si	20	50	0	0	110	0	180	180

Figura 2. Matriz Evolutiva de Frecuencias Acumuladas

- **Se escoge aleatoriamente uno de los renglones de la matriz evolutiva de frecuencias acumuladas y este representa la primer nota de la composición.**
- **Se utiliza el valor almacenado en la columna totales de ese renglón para generar un número aleatorio entre cero y dicho valor.**
- **El número aleatorio generado se empezará a comparar de izquierda a derecha con los valores distintos de cero de ese renglón, hasta que se encuentre uno que sea mayor o igual. La nota de la columna donde esté almacenado este número, indica la segunda nota de la composición y el siguiente renglón a procesar.**
- **Se escoge el renglón correspondiente a la nota anterior y nuevamente se genera un número aleatorio entre cero y el valor contenido en la columna totales de ese renglón. Se compara de izquierda a derecha con los valores diferentes de cero almacenados en las casillas de ese renglón, hasta que alguno de ellos sea mayor o igual al número aleatorio. La columna indica la siguiente nota de la composición y el renglón al que se le aplicará el mismo proceso, que se repite indefinidamente.**

Por ejemplo, tómese al azar una nota, Fa por ejemplo, será la primer nota de la melodía; se toma el valor de la columna totales del renglón Fa, en este caso 270 (véase la fig. 2, matriz de frecuencias acumuladas) y se genera un número aleatorio entre cero y este número. El número aleatorio así obtenido, se compara en orden de izquierda a derecha, con los valores del renglón Fa; al llegar a un valor que sea mayor o igual al número aleatorio, se toma la nota de la columna como la nota segunda de la melodía. Por ejemplo, si se obtiene el número 157, como este es mayor que 145 (nota Sol) y menor que 240 (nota La), se toma la nota La como la segunda nota de la melodía, llevando Fa, La,...

Se repite el procedimiento antes descrito, sólo que ahora se toma el renglón de la nota anterior, La, que tiene en la columna totales un valor de 50, por lo tanto, se genera un número aleatorio entre 0 y 50; por ejemplo el 18, 35 es mayor que 18, luego Re es la tercer nota de la

melodía y así va Fa, La, Re,…. Para obtener la 4ª, 5ª, 6ª y n-ésima nota, basta repetir el mismo procedimiento:

- **Se genera un número aleatorio entre cero y el valor de la columna totales del renglón de la última nota obtenida.**
- **Se compara el número así obtenido con los valores de ese renglón, hasta que alguno sea mayor o igual. La columna en la que se encuentre este valor, representa la siguiente nota de la melodía.**

2. Componente Evolutiva

A continuación se explicará el funcionamiento de la componente evolutiva del sistema, que le da la capacidad de componer música de cualquier tipo, lo que es una ventaja; ya qué, se pueden implementar diferentes tipos de algoritmos para la generación de música, sin embargo, habría que desarrollar algoritmos específicos para cada tipo de música y más complejo aún, habría que hacer un algoritmo específico que refleje las características propias de composición de cada autor, convirtiendo a esta en una tarea que sin duda alguna requeriría mucho de tiempo de programación.

Por lo tanto, la gran ventaja de utilizar las técnicas evolutivas, es que permiten que sea el mismo sistema quien encuentre las reglas de composición y de esta forma, para que el sistema aprenda a componer como algún autor en especial, basta con proporcionarle ejemplos de composiciones musicales de dicho autor, evitando el arduo trabajo de desarrollar miles de algoritmos.

Este sistema evoluciona en base a dos procesos, uno de aprendizaje, que le da la capacidad de aprender a través de ejemplos de música y por otro lado un proceso de aplicación que mientras el sistema genera alguna composición musical le permite irse modificando a si mismo en tiempo real, como si la música generada fueran nuevos ejemplos de música que se le estuvieran proporcionando.

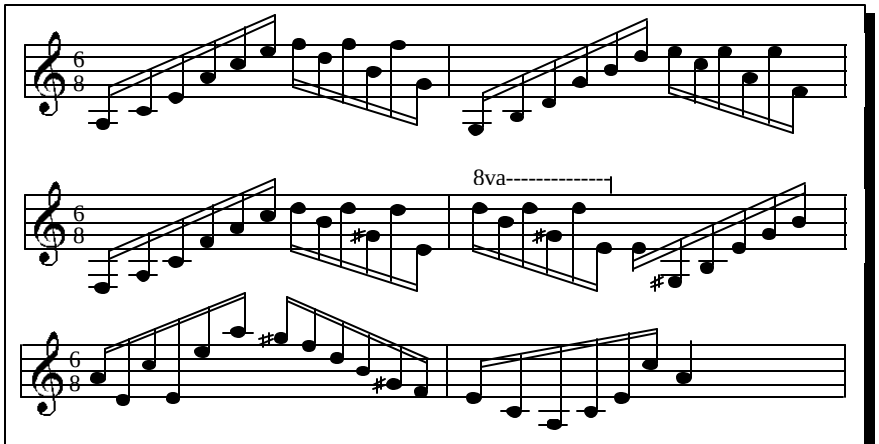


Fig. 3 Fragmento de la obra Los 24 Caprichos de Niccoló Paganini.

2.1 Proceso De Aprendizaje

La función que desempeña el proceso de aprendizaje es encontrar los valores apropiados con los que se han de llenar las matrices evolutivas, esto es, qué casillas deberán de contener valores y qué valores se deberán de almacenar en ellas, de manera tal que reflejen las características de algún autor o algún tipo de música en particular.

Para explicar el proceso de aprendizaje se utilizará un breve fragmento de la obra «Los 24 caprichos» del maestro Nicoló Paganini.Este fragmento se muestra en la figura 3.

Basándonos en el teclado de la figura 4, los números correspondientes a las notas de esta composición son los siguientes:

10(La), 13(Do), 17(Mi),
22(La), 25(Do), 29(Mi),
30(Fa), 27(Re), 30(Fa),
24(Si), 30(Fa), 20(Sol),
8(Sol), 12(Si), 15(Re),
20(Sol), 24(Si), 27(Re),
29(Mi), 25(Do), 29(Mi),
22(La), 29(Mi), 18(Fa),
6(Fa), 10(La), 13(Do),
18(Fa), 22(La), 25(Do),
27(Re), 24(Si), 27(Re),
21(#Sol), 27(Re), 17(Mi),
15(Re), 12(Si), 15(Re),
9(#Sol), 15(Re), 5(Mi),
17(Mi), 9(#Sol), 12(Si),
17(Mi), 21(#Sol), 24(Si),
22(La), 17(Mi), 25(Do),
17(Mi), 29(Mi), 34(La),
33(#Sol), 30(Fa), 27(Re),
24(Si), 20(Sol), 18(Fa),
17(Mi), 13(Do), 9(#Sol),
13(Do), 17(Mi), 25(Do),
22(La).

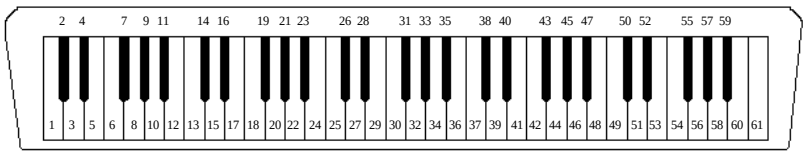


Figura 4. Teclado base

Conforme se vaya avanzando en el ejemplo se podrá observar que el aprendizaje consiste en encontrar las notas que ha utilizado el maestro Paganini, así como contar las veces que se repiten y saber que notas ha usado después de otras con mayor regularidad.

Al iniciar el proceso de aprendizaje, la matriz evolutiva aleatoria se encuentra vacía, es decir llena de ceros.

El primer renglón de la Matriz Evolutiva Aleatoria del sistema (ver figura 5) que sufrirá algún cambio, es el que corresponde a la primer nota del fragmento, es decir el renglón 10 equivalente a la nota La; recorriendo este renglón de izquierda a derecha, se encuentra la columna correspondiente a la nota Do, segunda nota del fragmento, que es la columna 13 de la matriz. Entonces, el valor de la casilla que se encuentra en la intersección del renglón 10 (La) con la columna 13 (Do) se incrementa en una unidad. Tomando en cuenta que el valor anterior era cero, queda almacenado el valor uno. De esta forma se ha incrementado la posibilidad de que la nota Do sea tocada después de la nota La.

Para continuar, se utiliza la nota Do como el número del siguiente renglón que se ha de modificar. Es decir, en el renglón 13 (Do), la segunda nota leída, se incrementará el valor de la casilla, ubicada bajo la columna que corresponde a la tercer nota leída 17 (Mi) en una unidad.

Se repite el mismo procedimiento, modificando en esta ocasión el valor de la casilla que se encuentre en la intersección del renglón 17 (Mi) con la columna 22 (La), que es la cuarta nota del ejemplo.

El mismo procedimiento se lleva a cabo con el resto de la melodía y de igual forma se procede con todos los ejemplos musicales que se le proporcionen al sistema. De esta manera, la matriz se va modificando conforme se le vayan presentando ejemplos. Esta matriz muestra como quedan los valores de las diferentes notas después de haber aprendido el fragmento de Paganini. (Debido a que la matriz que utiliza el sistema es de 60 renglones por 60 columnas, no fue posible incluir toda la información contenida en la matriz, sin embargo sólo se han quitado algunos renglones y columnas llenos de ceros).

Después de haber leído toda la partitura, se tiene la matriz lista para calcular la matriz de frecuencias acumuladas, la columna de totales y finalmente permitir que el sistema produzca su primera composición.

2.2 Proceso de Aplicación

La rutina generadora de música, se encarga de llevar a cabo el Proceso de Aplicación, dentro del cual, cada vez que se genera una nota, se modifican las matrices evolutivas, de manera tal que aumenta la probabilidad de que esa nota vuelva a ser tocada, tal y como sucede cuando un músico aprende a tocar.

Un ser humano, tiene que practicar durante "largas horas" para poder interpretar música en algún instrumento musical, repitiendo una y otra vez ejercicios y melodías, con el fin de aumentar su habilidad y de grabar en su memoria los ejercicios y melodías practicadas. De esta forma, las notas que más utiliza son de las que más se acuerda, de hecho, después de mucho practicar puede llegar a tocar sin tener que voltear a ver que nota está tocando.

De igual manera, lo que hace el generador de música cada vez que genera una nota, es aumentar el valor de la probabilidad de la casilla que co-

		5	6	8	9	10	12	13	15	17	18	20	21	22	24	25	27	29	30	33	34
		Mi	Fa	Sol	#Sol	La	Si	Do	Re	Mi	Fa	Sol	#Sol	La	Si	Do	Re	Mi	Fa	#Sol	La
5	Mi									1											
6	Fa					1															
8	Sol						1														
9	#Sol						1	1	1												
10	La							2													
12	Si								2	1											
13	Do				1					1	1										
15	Re	1			1		1			1		1									
17	Mi				1			1	1				1	1		2					2
18	Fa		1							1				1							
20	Sol			1							1				1						
21	#Sol														1		1				
22	La									1						2		1			
24	Si											1		1			1		1		
25	Do									1				1			1	2			
27	Re									1			1		1			1	1		
29	Mi										1			1		1			1		
30	Fa											1			1		2				
33	#Sol																		1		
34	La																			1	

Figura 5. Matriz Evolutiva Aleatoria que ha sido llenada con el fragmento de Paganini.

rresponde a esa nota, provocando que el sistema evolucione indefinidamente y que las notas que más toca sean de las que más se acuerde.

En la figura 6 se muestra el pseudocódigo del proceso de aplicación, donde se puede ver como se modifican las matrices mientras se genera la música, propiciando la evolución de ésta.

En la figura 7 se presenta la partitura de una composición producida por el sistema, después de haber aprendido 4 ejemplos de Paganini.

CONCLUSIÓN

Hoy día que se descubre que el Caos hace presa de todos los habitantes de este planeta y quizá de este Universo, resulta muy interesante utilizar herramientas como los Sistemas Evolutivos que generan caos de manera cotidiana y permiten modelar los fenómenos de manera maravillosa.

La versatilidad de los Sistemas Evolutivos ha permitido el desarrollo de Zanya, un sistema compositor que da la oportunidad de escuchar composiciones caóticas por siempre diferentes, pero similares a las de ejemplos musicales de los cuales tiene la capacidad de aprender. Esto provoca un efecto maravilloso, pues abre la posibilidad de escuchar como si estuvieran presentes a autores como Bethoven o Paganini.

Por otro lado, Zanya se puede usar como un apoyo para aprender a hacer composición musical, con la ventaja de contar con la asesoría directa de los grandes maestros o de cualquier autor que uno prefiera.

Queda abierto el camino, ya que este trabajo es parte de un proceso general de composición, en donde se busca mejorar los diferentes sonidos y efectos que se puedan obtener, además de utilizar matrices de 3, 4 ó n dimensiones en donde se puedan reflejar las tantas variables que intervienen en una obra maestra.

Este breve trabajo se verá completo si logra inspirar a alguien a hacer desarrollo musical-informático, para que pueda crecer esta área que se puede considerar nueva.

```
MusicaPorcentual()
{
 a1 = nota_siguiente                  //Genera un número aleatorio entre el valor almacenado en el
 zz = random( Bethoven [a1] [62] )+1; //(renglón a1, columna 62) y cero.

  a2 = 1;                             //encuentra la columna a2 del renglón a1, que contiene
 while( Bethoven[a1][a2] < zz && a2<=60 ) a2++;   //un valor mayor o igual al número aleatorio, el
 nota_siguiente = a2;                 // número de esa columna es la siguiente nota.

 Bethoven1[a1][a2] += 1;              //Actualiza (evoluciona) la Matriz Evolutiva Aleatoria.

 if( Bethoven1[a1][a2] > 500 )
 {
    for( a1 = 1; a1 <= 60; a1++ )     //Mecanismo de olvido, cada vez que algún valor de la
       for( a2=1; a2 <= 60; a2++ )    //matriz llega a 500, a todos los valores se les divide
          if( Bethoven1[a1][a2] > 1 ) //entre 2 y aquellos menores a 1 se igualan a 1.
             Bethoven1[a1][a2] /= 2;
           else
             Bethoven1[a1][a2] = 1;
   EscalaPorcentual();
 }
 while( a2 <= 62 )
  {
  if( Bethoven[a1][a2] != 0 )         //Actualiza (evoluciona) la Matriz Evolutiva de
  {                                   // Frecuencias Acumuladas
    Bethoven[a1][a2] += 1;
  }
  a2++;
  }

}
```

Figura 6. Pseudocódigo de la aplicación

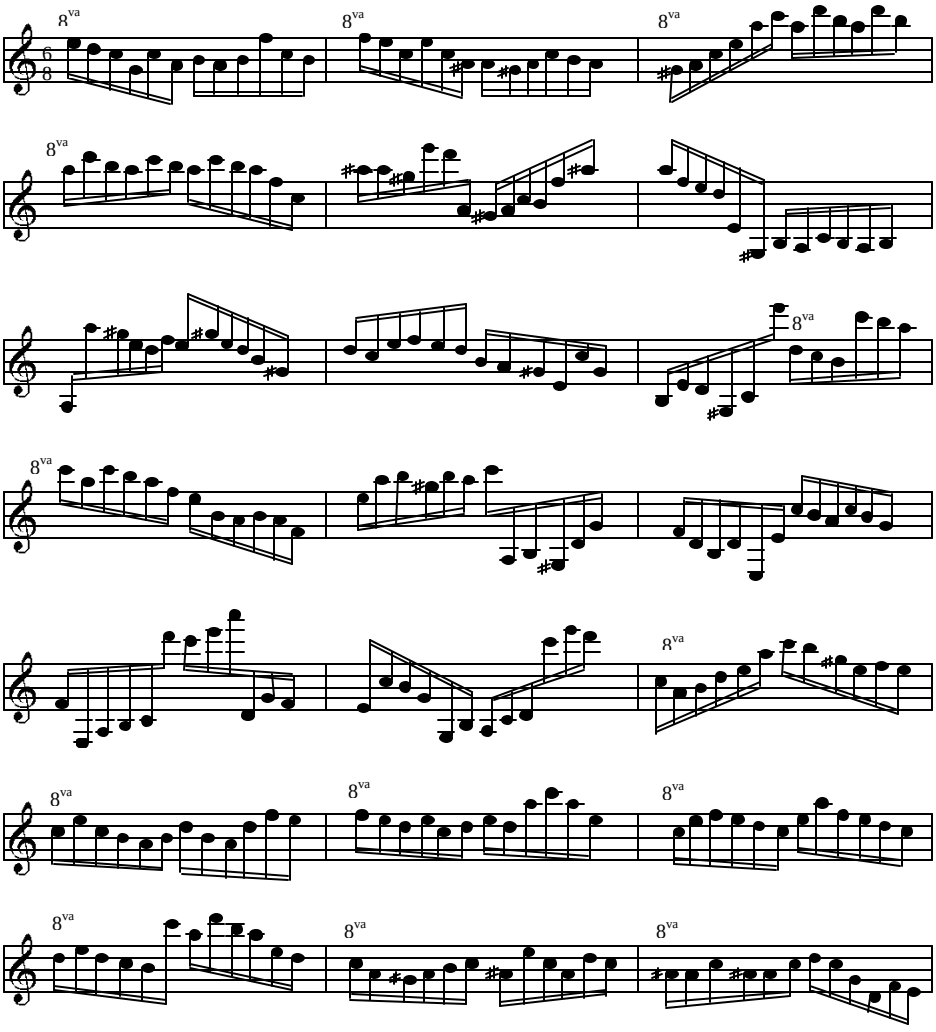


Figura 7. Partitura generada después de aprender 4 partituras dePaganini

BIBLIOGRAFÍA

[1] **Fractals in Nature. Benoit Mandelbrot.**

[2] **Música Blanca y Música Parda, Curvas Fractales y Fluctuaciones del tipo 1/f. Martin Gardner. Investigación y Ciencia, junio 1978.**

[3] **Evolución. Fernando Galindo Soria. Septiembre 1995.**

[4] **Del Azar Benigno al Azar Salvaje. Benoit Mandelbrot. Investigación y Ciencia, diciembre 1996.**

[5] **El Ruido 1/f. Edoardo Milotti. Investigación y Ciencia, noviembre 1996.**

[6] **El Caos, la nueva Física, las nuevas Matemáticas y sus Aplicaciones a las Ciencias Sociales. Miguel José Yacamán. Conferencia dictada en el Colegio de México. Noviembre 1993.**

[7] **Física General. Francis W. Sears y Mark W. Zemansky. Ed. Aguilar.**

[8] **El Orden Caótico. Monique Dubois, Pierre Aften y Pierre Bergé. Mundo Científico No 68. Vol 7.**

SISTEMA DE MULTIPROCESAMIENTO ACOPLADO A UN AMBIENTE DE COPROCESAMIENTO MEDIANTE LA TÉCNICA DEL MAPEO

M. en C. G. Abraham Mas Levario.
Centro de Investigación en Computación - I P N

El presente artículo considera una solución al problema de cómo relacionar un Host con un sistema multiprocesador ubicado en un circuito impreso. El circuito está conectado en el bus de expansión ISA del Host. Con una extrapolación de esa solución también se considera el problema de cómo relacionar internamente y externamente los nodos de un multiprocesador en un cómputo coordinado. Las soluciones se plantean en términos de la técnica de mapeo.

INTRODUCCIÓN

En la actualidad se cuentan con una gran cantidad de mecanismos computacionales para mejorar el rendimiento en uno o más de los atributos de las máquinas de arquitecturas abiertas, como lo es la Computadora Personal (PC). Existen con el objetivo de mejorar las velocidades nativas en diferentes tipos de procesamiento.

Aquí se presenta una técnica alternativa que proporciona la conceptualización e implementación que es en sí un modo de direccionar un bloque de memoria a través de la proyección de un segmento del direccionamiento en modo real de la PC. El bloque de memoria no pertenece a la tarjeta matriz de la PC, sino que pertenece a una tarjeta de expansión conectada al Bus ISA, permitiendo así que la PC en ciertos momentos sea el Host de un coprocesamiento. Desde este punto de vista, se tiene una tarjeta coprocesadora que podría comportarse como una máquina paralela tipo MIMD (Multiple Instruction Multiple Data).

La máquina paralela es un sistema de multiprocesamiento de dos nodos o elementos de procesamiento con una red de interconexión. Los elementos de procesamiento están constituidos básicamente de un procesador de TI (DSP TMS320C50 de 40 Mhz), un bloque 32 Kbytes de memoria estática de 20 ns y dispositivos de lógica F con el objetivo de evitar colisiones de diferentes direccionamientos. La red de interconexión está construida con una red de conmutación y con 16 elementos de conmutación. Una FPGA de TI (TPC1010AFN-068C) proporciona la red de conmutación y 16 elementos transceptores 74F245 constituyen los elementos de conmutación. En la FPGA es donde se extrapola la técnica de tal manera que permite cómputos independientes entre los nodos, cómputos con recursos externos compartidos entre los nodos y cómputos con broadcasting entre los nodos.

MÉTODOS

Para conceptualizar la técnica se emplea el concepto del término map del Inglés cuya traducción apropiada en este caso es mapear. La palabra mapear es un verbo del tipo transitivo cuyo significado denotativo de "unir" se usa en dos puntos de vista para crear dos significados connotativos. Desde el punto de vista del software, mapear tiene la siguiente definición: Transferir un conjunto de objetos de un lugar a otro. Desde el hardware, tiene la siguiente definición: Relacionar un conjunto de objetos con otro. Con el verbo mapear se da origen a un par de adjetivos: mapeado o mapeada; y también a dos sustantivos: mapeador y mapeo.

En vista de lo anterior, se adjetivo al sustantivo memoria para crear el término compuesto memoria mapeada.

Una memoria mapeada es una unidad de direccionamiento lógico usada por dos o más entidades computacionales, de tal modo que a través de un objeto lógico sólo una unidad computacional pueda usarla en un determinado momento. Esta memoria mapeada es el bloque de memoria de la tarjeta coprocesadora que se alcanza al proyectar un segmento del modo real de la PC a ésta.

El término mapeador se refiere al objeto lógico que tiene la función de

mapear la memoria mapeada de la tarjeta coprocesadora con el Host. El mapeador también está localizado en la tarjeta y actúa directamente sobre la memoria mapeada. La circunstancia que se origina en la tarjeta coprocesadora cuando el mapeador está actuando sobre la memoria mapeada es lo que se llama mapeo. La técnica del mapeo intrínsecamente es una actividad de direccionamiento sin colisiones.

El Mapeador

Con la definición descriptiva del mapeo se establece el tipo de función a crear. Esta función organiza un conjunto de funciones lógicas del diseño digital que permiten visualizar el mapeador como una unidad computacional.

Para evidenciar con más detalle el fenómeno del mapeo de un mapeador, se presentan una relación de observaciones que esencialmente son los siguientes axiomas y teoremas.

Axiomas:

Cualquier idea o conjunto de ideas pueden ser representadas con matemáticas con la finalidad de poder calificar y cuantificar el objeto mental al que se refiere o refieren las ideas.

Cualquier objeto o conjunto de objetos computacionales pueden ser organizados y coordinados con la calificación y cuantificación del objeto mental al que se refieren las ideas.

Corolario:

En vista de lo anterior a cualquier objeto computacional se le puede atribuir cierto concepto cuando uno puede percibirlo.

A cualquier objeto computacional se le puede asignar un nombre según un criterio que tenga relación con la zona donde se aplique, por lo tanto a éste se le llama mapeador.

La primera acción creada en un mapeo es la comunicación entre el entorno de la PC y el mapeador.

La segunda acción creada en un mapeo es la comunicación entre el segmento D (elegido arbitrariamente y seleccionado entre los segmentos de propósito general) del entorno de la PC y la memoria mapeada.

El primer factor creado en una comunicación es el fenómeno de localización de espacios, llamado en este trabajo direccionamiento.

Teoremas:

Las líneas más significativas del direccionamiento del Bus ISA son los factores que contribuyen a que el mapeador establezca una correspondencia unívoca entre el segmento D y la memoria mapeada.

Las líneas menos significativas del direccionamiento del Bus ISA son parte de la correspondencia unívoca que se da entre el segmento D y la memoria mapeada.

Postulados:

El fenómeno del mapeo es esencialmente una circunstancia de localización por reflejo desde el punto de vista de la tarjeta coprocesadora, y además es una circunstancia de localización por correspondencia unívoca desde el punto de vista del mapeador.

La correspondencia unívoca en el caso del mapeo es una ecuación lineal de primer grado.

El mapeador es la solución de la ecuación lineal de primer grado. En el mapeo se distinguen dos conjuntos: el de las direcciones del segmento D y el de las direcciones de la memoria mapeada. Ambos conjuntos contienen 64 Kbytes de localidades. La representación esquemática y correspondencia postulada se observa en la figura 1.

La correspondencia que se da en el mapeo existe sólo para un par de

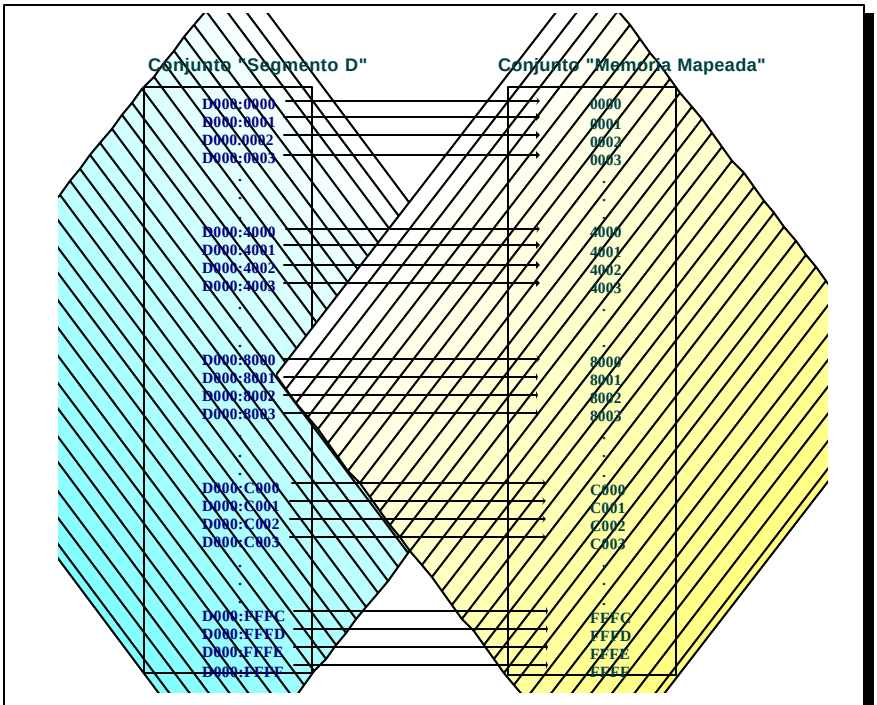


Figura 1. Correspondencia en el mapeo

elementos asociados en un determinado tiempo t_n, donde n es cualquier número mayor que cero. Considerando la figura anterior y que en un instante el mapeo es un par ordenado de números, se puede suponer que el mapeo en todos los instantes es un conjunto de pares ordenados de números, tales que ninguna pareja de ellos tiene el mismo primer número, y que los segundos números de las parejas pueden ser obtenidos mediante una relación que se puede simbolizar a su vez como la ecuación de la función que se expone a continuación:

$$f(N_{12}N_8N_4N_0) \rightarrow D N_{12}N_8N_4N_0 \rightarrow D0000 \qquad \text{Ec. 1}$$

En donde D0000 significa el inicio del segmento D, $N_{12}N_8N_4N_0$ significa el offset del segmento D y es cualquier valor entre $(0000)_{16}$ y $(FFFF)_{16}$, y el resultado que refiere $f(N_{12}N_8N_4N_0)$ es el segundo número hexadecimal de cierto par ordenado de un mapeo. Ahora, cada N se refiere a un número formado por cuatro magnitudes binarias de cuatro líneas de direcciones del bus, así que cada N se obtiene respectivamente como se indica a continuación:

$N_0=(A_3A_2A_1A_0)$,
$N_4=(A_7A_6A_5A_4)$,
$N_8=(A_{11}A_{10}A_9A_8)$ y
$N_{12}=(A_{15}A_{14}A_{13}A_{12})$.

La resta de la ecuación 1 es la solución y la operación algebraica que establece el mapeo en sí. Con la observación anterior se determina que para hacer un direccionamiento en la memoria mapeada se deben emplear las magnitudes de las líneas A_0 a la A_{15} del bus ISA. Además la resta de la ecuación es una aproximación y un modo implícito de conceptualizar el mapeador con las magnitudes de las líneas A_{16}, A_{17}, A_{18} y A_{19}.

Para hacer un análisis cualitativo estructural del mapeador, se puede seguir un camino y el camino es considerar al mapeador como parte de una red de interconexión. La idea es suponer que todas las tarjetas de expansión poseen un mapeador, así se puede visualizar que todo el conjunto de mapeadores de las tarjetas que interactúan con el bus ISA es una red de interconexión dinámica. Una red de interconexión dinámica desde el punto de vista de este trabajo es una estructura física de direccionamiento por etapas, cada una de las etapas se localizan en cada tarjeta y poseen componentes que se dedican a direccionar las memorias mapeadas de dicha tarjeta de expansión en el ambiente de las ranuras del bus ISA. Tales componentes son enlaces y conmutadores programables.

Si se considera que la PC es la única fuente de la red y las memorias mapeadas son los destinos, entonces se puede ver a la red como una estructura que permite hacer una sola conexión de una fuente a un destino a la vez. Cada uno de los destinos es uno de los 16 segmentos de memoria de la PC y cada uno de esos destinos es una memoria mapeada. Por lo tanto se puede deducir que las líneas A_{19}, A_{18}, A_{17} y A_{16} forman parte del mecanismo de direccionamiento de las etapas de la red, por la razón de que esas líneas indican uno de los 16 segmentos y así uno de los 16 destinos. Para establecer las características de la red se postulan una serie de suposiciones que se enlistan a continuación:

- Se usa una sola línea de las cuatro anteriores en cada etapa para direccionar gradualmente un destino.
- Los conmutadores de cada etapa deberán tener una sola entrada de programación, y además cada una de éstas deberán estar en corto circuito con la línea que llega a la etapa.
- Los conmutadores sólo pueden conmutar hacia uno de dos estados posibles debido a la naturaleza binaria de la línea que se conecta a sus entradas de programación. Por lo tanto, el conmutador programable hace dos correspondencias: la correspondencia de la magnitud del estado 0 y la del estado 1.
- Los conmutadores programables de cada etapa tienen sólo una entrada para permitir los dos tipos de correspondencias.
- Cada uno de los conmutadores necesitan un enlace a su entrada.
- La red debe poseer cuatro etapas para permitir un direccionamiento gradual según los dos tipos de correspondencia.
- Cada etapa de la red tiene un cierto número de salidas según la cantidad de conmutaciones (de 0 a 1 y de 0 a 1) de la línea que le corresponda.

Generalizando, la red consiste de n = N / A etapas, donde N es el número de salidas y A es el número de líneas usadas para establecer una conexión entre la PC y la memoria mapeada. Como N = 16 y A = 4, entonces n = 4 etapas. Cada etapa en la red consiste de un cierto número de enlaces y de conmutadores, en donde el número de enlaces y el número de conmutadores es el mismo de acuerdo a la posición de la etapa en la red. La etapa 1 que está controlada por el bit más significativo (A_{19}) de la dirección del destino tiene la posición cero, la etapa 2 que esta controlada por A_{18} tiene la posición uno, la que controla A_{17} es la etapa 3 y tiene la posición dos, y la que

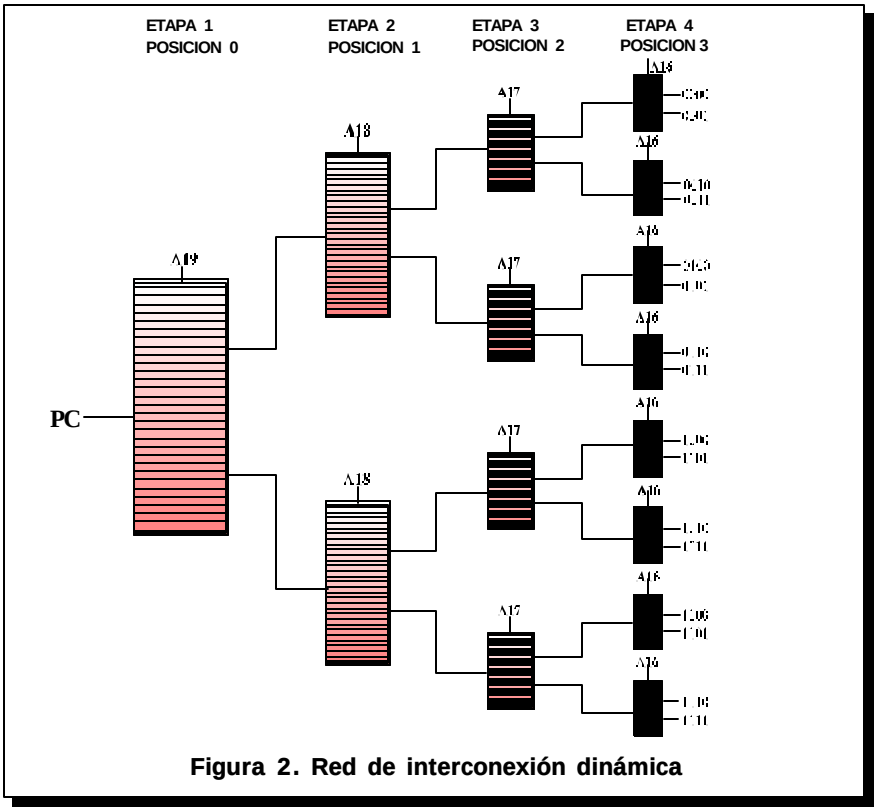


Figura 2. Red de interconexión dinámica

controla A_{16} es la etapa 4 y tiene la posición tres. Así que cada etapa consiste de $m = 2^p$ enlaces y conmutadores, en donde p es el número de la posición de la etapa en la red. Según lo anterior la etapa 1 consta de un enlace y de un conmutador ($m = 2^0 = 1$), la etapa 2 consta de dos enlaces y dos conmutadores ($m = 2^1 = 2$), la etapa 3 consta de cuatro enlaces y cuatro conmutadores ($m = 2^2 = 4$) y la etapa 4 consta de ocho enlaces y ocho conmutadores ($m = 2^3 = 8$). La figura 2 muestra la estructura de la red para el mapeo en cada uno de los segmentos.

En base a los estudios cuantitativos y cualitativos anteriores se concluye que el mapeador requiere un sistema de programación para programar a los conmutadores programables. Un sistema de programación en los sistemas digitales básicamente está integrado por expresiones Booleanas, las cuales determinan cómo debe operar el sistema según qué estimulo le llegue.

A partir de la figura 1 y del concepto de permutación para una red de interconexión se obtendrá para el sistema de programación una expresión Booleana. Una permutación refiere a la conexión de un conjunto de fuentes a un conjunto de destinos tales que cada fuente es conectada a un simple destino, al observar este concepto se ve que es congruente con el concepto que se aplicó anteriormente de la correspondencia unívoca. La representación simbólica de una conexión en este caso es la representación de un par ordenado. En vista de lo anterior y recordando que la red solo activa a la vez un mapeador, la permutación en este caso es [(S,D)] lo que significa que únicamente existe un mapeador en la red haciendo una conexión de una fuente "S" con un destino "D".

Como la permutación es esencialmente un par ordenado se puede encontrar una regla de igualdad que establezca como determinar el segundo miembro a partir del primer miembro. Esta regla para el fin de este estudio puede ser representada como una función Booleana siguiendo la suposición de que si se esta hablando de una conexión, implícitamente se esta tratando con un espacio direccionado y según lo ya visto se puede establecer que S denota la representación binaria de una fuente, de modo que se puede establecer que:

$S = A_{19}A_{18}A_{17}A_{16}$

indica con cual destino se hace la conexión. Ahora, para tener consistencia, el destino D también debe denotar una representación binaria con cuatro números binarios. Así que la permutación que indica que existe mapeo en el segmento:

$D(A_{19}=1, A_{18}=1, A_{17}=0, A_{16}=1)$

es:

$[(1101, 1101)]$ Ec. 2

Si la función Booleana es F(S) y si además suponemos, en base a la figura 2 , que cada etapa de la red de una conexión es una función Booleana parcial de F(S), entonces :

$F(S) = f_{19}f_{18}f_{17}f_{16}$.

Estas funciones Booleanas parciales son básicamente funciones de conmutación; entonces, en general, cualquier permutación de una red puede ser representada en términos de un conjunto de funciones de conmutación. La determinación de F(S) en base a la magnitud de la fuente S, se encuentra a partir de una simple tabla de verdad como sigue:

S a F(S)

A_{19}	A_{18}	A_{17}	A_{16}	®	f_{19}	f_{18}	f_{17}	f_{16}
1	1	0	1	®	1	1	0	1

Cada una de las funciones de conmutación f_{19}, f_{18}, f_{17}, y f_{16}, pueden ser expresadas como:

$$f_{19} = A_{19},$$
$$f_{18} = A_{18},$$
$$f_{17} = \overline{A}_{17},$$
$$f_{16} = A_{16}.$$

La función Booleana, entonces, es

$$F(S) ? A_{19}A_{18}\overline{A}_{17}A_{16}$$

lo que indica que para encontrar el destino se hace que D = F(S). Esta función Boolena se cumple cada vez que se desea mapear memoria en el segmento D, o sea, se cumple cuando la permutación existente en la red de interconexión de mapeo es [(1101, 1101)], así que el mapeador para el segmento D debe integrar en su sistema de programación la implementación de la función a la cual se le puede nombrar como una función de mapeo de la red de interconexión.

En general, se puede decir que:

$$F(S) ? A_{19}A_{18}A_{17}A_{16}$$

es la función sintética de las dieciséis posibles funciones de la red de interconexión de mapeo. La función F(S) además de indicar que destino se conecta con la fuente, indica a través de los valores de las líneas A_{19}, A_{18}, A_{17} y A_{16} cómo programar los conmutadores de cada etapa para direccionar gradualmente un destino.

El Sistema De Multiprocesamiento

La tarjeta de coprocesamiento tiene una máquina construida con los conceptos del mapeo, que indica quienes desean comunicarse, y la técnica de enrutamiento de Wormhole que indica cómo deben comunicarse sin colisiones. El algoritmo para la técnica de enrutamiento está integrada en la red de conmutación de la red de interconexión. La red de conmutación posee una unidad computacional llamada Tabla de Consulta quien es responsable de un determinado mapeo según lo establece el algoritmo de Wormhole. Se puede considerar que el Wormhole es un caso particular de mapeo entre dos o más elementos de procesamiento. La figura 3 muestra la anatomía del sistema de multiprocesamiento.

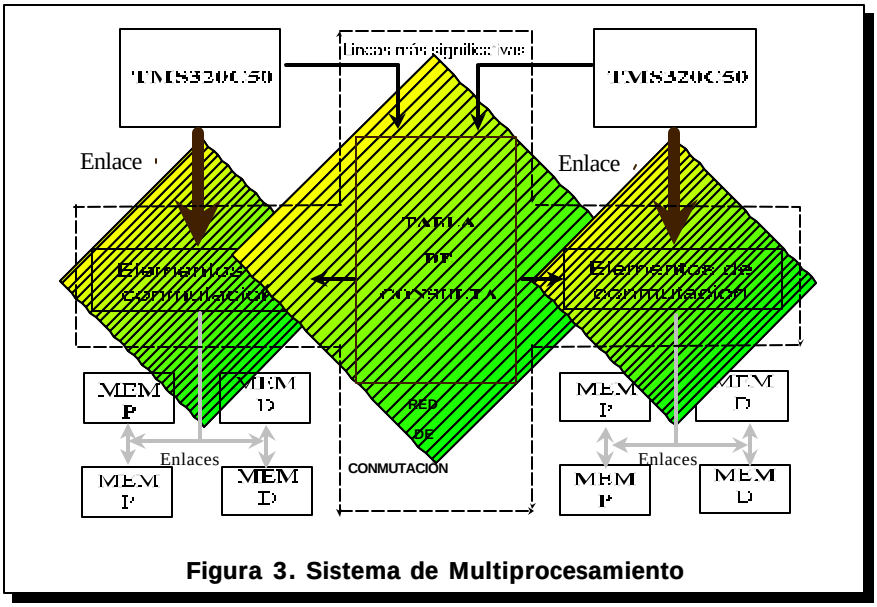


Figura 3. Sistema de Multiprocesamiento

La red de conmutación es una solución matemática para evitar colisiones de dos o más procesadores que necesitan actuar sobre recursos de comunicación compartidos. Cada nodo posee sus propios recursos, y cuyo comportamiento es controlado por la tabla de consulta a través de la petición del procesador del nodo. La tabla de consulta puede asumir en momentos distintos una de tres posibles soluciones según tres posibles problemas de comunicación en la red de conmutación.

El primer problema de comunicación en el sistema de multiprocesamiento que resuelve la tabla de consulta, es el que existe cuando los nodos funcionan independientemente uno del otro. El funcionamiento independiente puede ser síncrono o asíncrono, y además puede o no existir concurrencia. En la figura 4

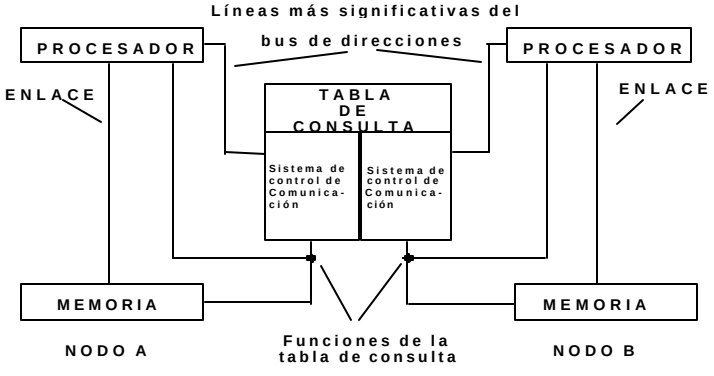


Figura 4. Comunicación independiente entre los nodos

se muestra el postulado de la representación aproximada de la disposición de los componentes funcionales de los nodos, y los enlaces que se establecen en este tipo de comunicación.

El segundo problema de comunicación en el sistema de multiprocesamiento que resuelve la tabla de consulta, es el que existe cuando lo nodos funcionan dependientemente uno del otro. Este funcionamiento puede ser síncrono o asíncrono, y además existe concurrencia. La dependencia se establece cuando los nodos direccionan sus memorias remotas. En la figura 5 se muestra el postulado de la representación aproximada de la disposición de los componentes funcionales de los nodos y los enlaces que se establecen entre ellos.

El tercer problema de comunicación en el sistema de multiprocesamiento que resuelve la tabla de consulta, es el que existe cuando los nodos direccionan la misma memoria. En este funcionamiento no existe la concurrencia. En las figuras 6 y 7 se muestra una representación aproximada para este tipo de comunicación. Para lograr la comunicación se usan dos técnicas anti-colisiones: la que genera estados de espera, y la que genera estados de retención en uno de los procesadores. Estas técnicas son mutuamente exclusivas y se activan dinámicamente según lo predisponga la aplicación.

Finalmente, los teoremas que fundamentan el diseño teórico de la tabla de consulta son esencialmente los teoremas del mapeo:

Teoremas

Las líneas más significativas del direccionamiento de un espacio de memoria crean la correspondencia entre el procesador y el espacio de memoria.

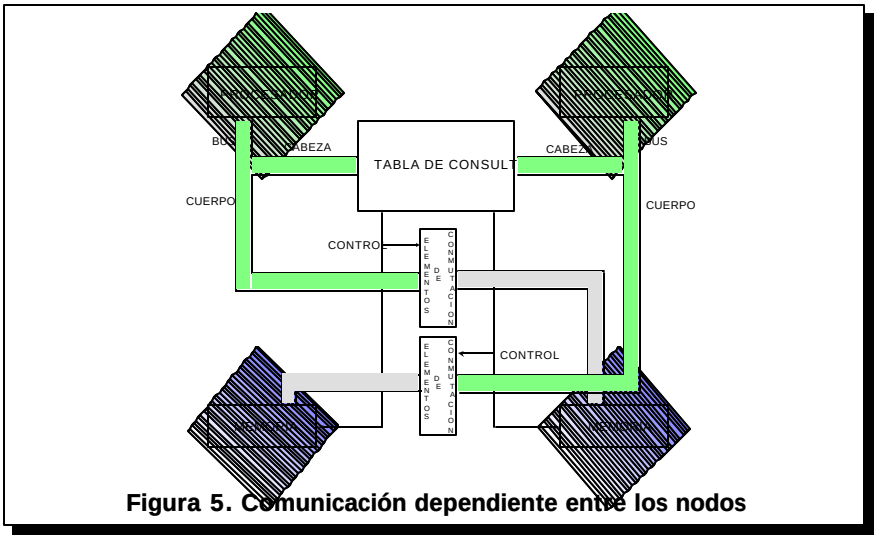


Figura 5. Comunicación dependiente entre los nodos

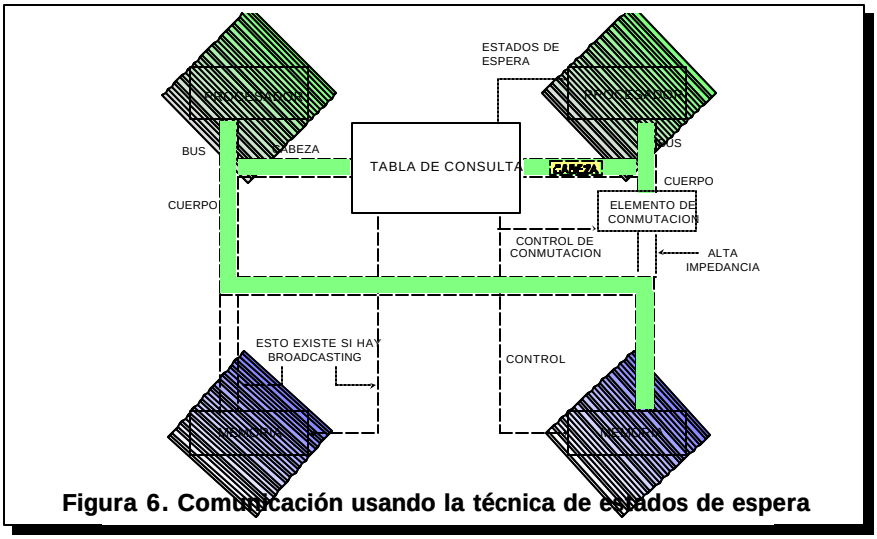


Figura 6. Comunicación usando la técnica de estados de espera

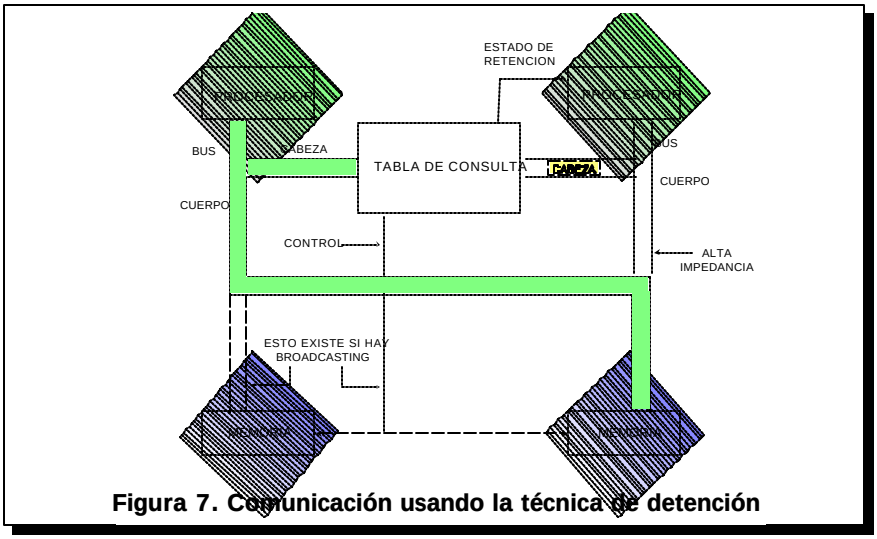


Figura 7. Comunicación usando la técnica de detención

Las líneas menos significativas del direccionamiento de un espacio de memoria están en la correspondencia del procesador y el espacio de memoria.

COROLARIO 1.1:

Las líneas más significativas del bus de direcciones del procesador son la cabeza del mensaje en la técnica de enrutamiento Wormhole.

COROLARIO 2.1:

La correspondencia entre el procesador y la memoria es también el enlace.

COROLARIO 2.2:

Las líneas menos significativas del bus de direcciones del procesador están en el cuerpo del mensaje en la técnica de enrutamiento Wormhole.

CONCLUSIÓN

La técnica del mapeo básicamente es un método de postulación que permite resolver conceptualmente los problemas del direccionamiento del tipo coplanar en los sistemas de coprocesamiento y en los sistemas multiprocesador.

RECONOCIMIENTOS

Se agradece al filósofo L. Ronald Hubbard por su Tecnología de Estudio y su filosofía aplicada en la vida, que es la piedra angular para el desarrollo de este diseño. Y al investigador Rodrigo Fernandez Mas por el conocimiento y el Know-How en el diseño digital de la técnica de mapeo; al investigador Miguel Lindig Bos por el conocimiento del diseño de máquinas paralelas y el diseño digital con FPGAs.

RESULTADOS

Este diseño es parte del proyecto: Diseño de un Chip Set de FPGAs para Implementar una Máquina Paralela tipo MIMD, de la línea de Investigación: *APLICACIONES DEDICADAS DE FPGAs* **del Instituto Politécnico Nacional. Este proyecto generó un grupo joven de investigadores, y con él tres tesis sustentadas y aprobadas en la Maestría de Sistemas Digitales:**

- *Diseño de un Sistema de Memoria Mapeada Implementado con FPGAs para un Arreglo Lineal de dos DSP TMS320C50***. Maestro en Ciencias: María Aurora Segura Corona.**
- *Integración de una Red de Conmutación basada en Wormhole Routing, utilizando la tecnología de FPGAs***. Maestro en Ciencias: Amadeo José Argüelles Cruz.**
- *Diseño y Construcción de un Filtro Digital en un Dispositivo FPGA***. Maestro en ciencias: Arguimiro Millán Alarcón.**

BIBLIOGRAFÍA

[1] **Tom Shanley / Don Anderson.** *"ISA System Architecture"***. Ed. Mindshare, Inc. 1995**

[2] *"Five Generation TMS320 User´s Guide".* **Texas Instruments 1994**

[3] **Daniel D. Gajski.** *"Principios de iseño Digital"***. Prentice Hall 1996**

[4] **M. Morris Mano.** *"Arquitectura de Computadoras"***. Prentice Hall. 3a Ed. 1997**

[5] *"Fast Advanced Schottky TTL Logic Data Book"***. National Semiconductor. 1990**

[6] **Hwang / Briggs.** *"Arquitectura de Computadoras y Procesamiento Paralelo"***. McGraw - Hill.**

Diseño de un Sistema de Codificación de Predicción Lineal (LPC)

M. en C. Pablo Manrique Ramírez
Profesor e Investigador del CIC-IPN
Ing. Miguel A. Meléndez Velázquez
Alumno del CIC-IPN

El área de procesamiento de señales de voz, desde hace tiempo ha sido tema de investigación en el desarrollo de proyectos. Sin embargo tras años de investigación y comenzando con los prototipos basados en resonadores acústicos hasta llegar a los modernos programas para computadora que sintetizan voz, todos estos productos han sido diseñados para el idioma inglés primordialmente; mientras que para el idioma español todavía se trabaja en una etapa de laboratorio. Además, muchos de los productos relacionados con el procesamiento de señales que ya son comercializados, solamente están implementados en software, el hardware empleado corresponde generalmente a tarjetas de sonido de las que ya se dispone en la mayoría de computadoras personales.

El objetivo principal de este proyecto es realizar el procesamiento de señales de voz utilizando el método de Codificación de Predicción Lineal (LPC) y enfocándose en el idioma español; no solo aprovechando el desarrollo de software sino también explotar el uso de dispositivos digitales para el diseño de una interfaz, por lo que no se empleará una tarjeta de sonido de propósito general sino una tarjeta sintetizadora de voz con un microprocesador de propósito específico, esto con el fin de lograr una calidad aceptable en la señal de voz sintetizada.

Introducción

El modelo de Codificación de Predicción Lineal es considerado uno de los modelos más próximos (analógicamente hablando) al sistema vocal humano. Por esta característica se ha seleccionado como base para el diseño de una interfaz digital que sea capaz de sintetizar señales de voz. Para esto, es importante considerar cómo está constituido el Sistema Vocal Humano para realizar una analogía con un Sistema Digital.

Generalidades del Sistema Vocal Humano

La voz es un sonido producido en la laringe debido a la salida del aire que, al atravesar las cuerdas vocales, las hace vibrar. Uno de los parámetros que definen a la voz es su tono. El tono depende de cada individuo y está determinado por la longitud y masa de las cuerdas vocales. Por lo tanto, el tono puede alterarse variando la presión del aire exhalado y la tensión sobre las cuerdas vocales. Esta combinación determina la frecuencia a la que vibran las cuerdas: a mayor frecuencia de vibración, más alto es el tono.

Otro aspecto de la voz es la resonancia. Una vez que ésta se origina, resuena en el pecho, garganta y cavidad bucal. Finalmente, otro parámetro importante de la voz es su intensidad o fuerza, que depende de la resonancia y de la fuerza de vibración de las cuerdas vocales.

Para que la voz sirva como parte de un sistema de comunicación, se requiere también la articulación. La articulación se refiere a los sonidos del habla que se producen y combinan para formar las palabras del lenguaje. Los instrumentos de la articulación son: los labios, la lengua, los dientes, las mandíbulas y el paladar. El habla se articula mediante la interrupción o modelación de los flujos de aire, vocalizados y no vocalizados, a través del movimiento de la lengua, los labios la mandíbula inferior y el paladar. Los dientes se usan para producir algunos sonidos específicos.

Otro elemento para conformar un sistema de comunicación por medio del habla es el lenguaje. El lenguaje es un sistema de símbolos abstractos reconocido por un grupo de personas que sirve para comunicar sus pensamientos y sentimientos. Los símbolos pueden ser verbales o no verbales, es decir, hablados o escritos, además, los símbolos no verbales pueden ser gestos y movimientos

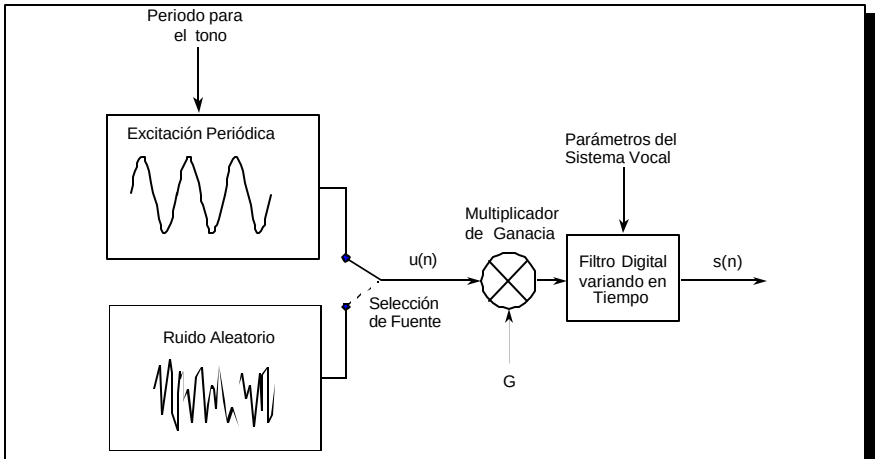


Figura 1. Modelo LPC para síntesis de voz

corporales. En el lenguaje hablado se utiliza la capacidad de articular sonidos y en el lenguaje escrito ésta se sustituye por la ortografía. Las capacidades auditivas y visuales son esenciales para la comprensión y expresión del lenguaje.

El Modelo LPC

El modelo LPC para señales de voz es una buena aproximación al sistema vocal humano. El LPC, matemáticamente es preciso [1] y además no es muy complejo llevarlo a la práctica a través de software y/o hardware, comparado con otros métodos digitales.

La idea fundamental del modelo LPC es representar a la señal de voz como una función de excitación constituida por un tren de pulsos cuasiperiódicos (para sonidos vocalizados) o una fuente de ruido aleatorio (para sonidos no vocalizados) [1]; el modelo de síntesis para voz con LPC [3] es mostrado en la figura 1.

En este modelo la fuente de excitación es seleccionada por un interruptor de posición controlado por un caracter vocalizado / no vocalizado de la voz. La ganancia G apropiada es estimada de la señal de voz, lo que es utilizado como entrada para un filtro digital que tiene la función de transferencia H(z). Este filtro digital es controlado por los parámetros característicos del sistema vocal de la voz que se está sintetizando. De esta manera, los parámetros para este modelo son:

- **Selección de la fuente (periódica o de ruido aleatorio).**
- **Periodo de tono (para sonidos vocalizados).**
- **Parámetro G de ganancia.**
- **Los coeficientes $\{a_i\}$ para el filtro digital.**

Estos parámetros varían lentamente en el tiempo y son los que hay que considerar en el diseño de un sintetizador digital de voz que utilice el modelo LPC.

El Sintetizador de Voz

En lo que corresponde al hardware, el sistema digital está basado en el microprocesador TSP53C30, de Texas Instruments [3]; que es un microprocesador de propósito específico que puede ser programado para seguir el modelo LPC. El TSP53C30 trabaja como un dispositivo esclavo a otro microprocesador, por lo que puede ser parte de la interfaz de una tarjeta de expansión o como un sistema independiente de una PC si se incluye en el diseño un microprocesador que realice todo el control, así como bancos de memoria para datos de voz.

El funcionamiento general del TSP53C30 es análogo al modelo LPC, ya mencionado anteriormente. Este modelo incorpora elementos análogos a los existentes en el sistema vocal humano. Tiene un generador de funciones periódicas y aleatorias (para producir sonido vocalizado y no vocalizado, respectivamente), un multiplicador de ganancia (realiza la función de la presión de aire que define la intensidad de la voz) y un

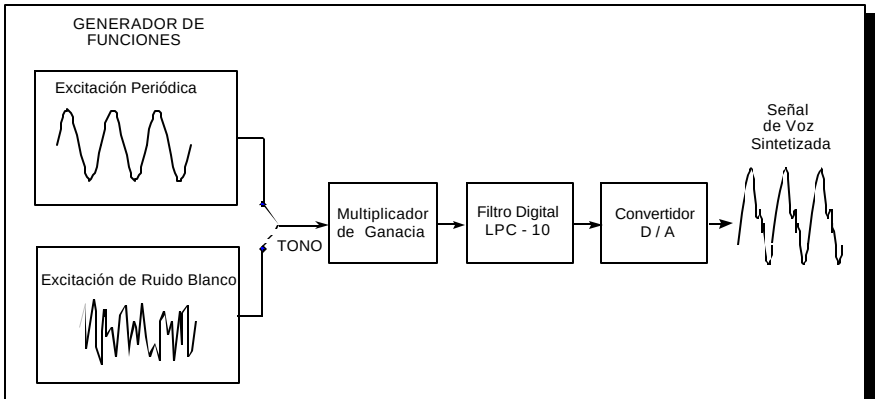


Figura 2. Esquema del modelo LPC basado en el TSP53C30

filtro digital de 10 polos (modela la resonancia de la cavidad oral).

En la figura 2 se muestra un esquema del modelo LPC basado en el TSP53C30. En dicho modelo el generador de funciones requiere como parámetro el periodo para producir un tono similar al obtenido con la vibración de las cuerdas vocales (sonido vocalizado). Este generador también produce ruido blanco, correspondiente al sonido no vocalizado. La función de salida del generador, es entonces multiplicada por un factor de energía que representa la presión de los pulmones. Finalmente, la señal es pasada a un filtro digital que modela la forma de la cavidad oral. Este filtro tiene 10 polos, razón por la cual la síntesis es referida como LPC-10.

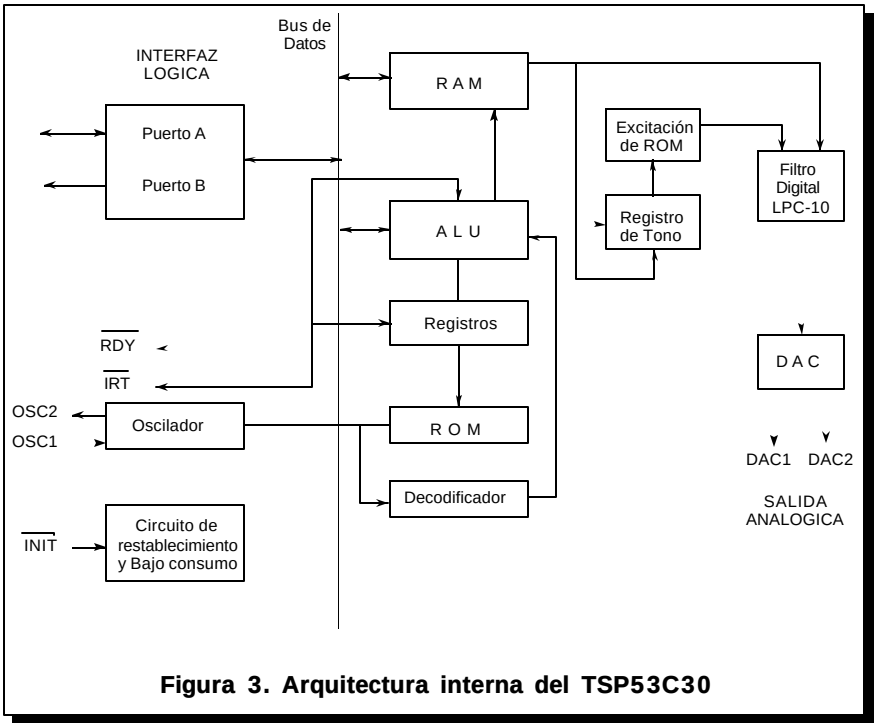


Figura 3. Arquitectura interna del TSP53C30

Arquitectura del TSP53C30

En la figura 3 se muestran a bloques los componentes internos del TSP53C30.

Básicamente se tienen los siguientes componentes: Un microprocesador de 8 bits, una ROM interna de 8K y una interfaz lógica de E/S. Las instrucciones son buscadas en la ROM cada 9 ms y se usan para controlar la acción que desarrolla el TSP53C30. Para producir voz, el TSP53C30 accesa datos de voz de una memoria externa o los que recibe del procesador al cual está conectado como esclavo. Una vez que el dato es leído, el procesador debe "descompactar" los parámetros individuales de voz y guardar los resultados en una sección de la RAM interna. De esta RAM interna se obtienen los parámetros de voz cada vez que son requeridos.

La sección de E/S, está formada por un puerto A bidireccional de 8 bits y un puerto B de 8 bits como interfaz a una memoria externa.

Buffer de datos de entrada

Todos los datos de entrada para la síntesis de voz son almacenados en una sección de la memoria RAM del TSP53C30, excepto cuando es inicializado para usar el formato PCM. En este caso, los datos son pasados directamente a la sección del sintetizador.

Software de Control

Para que el TSP53C30 realice sus funciones, es necesario que un programa controle los diferentes bloques de su arquitectura. Las instrucciones no son detalladas, pero el diagrama de flujo general del software de control se muestra en la figura 4.

Selección del modo de operación

El procesador debe inicializar al TSP53C30 para seleccionar un modo de operación del dispositivo y el formato de datos a manejar. El modo de operación determina si el TSP53C30 aceptará datos para la sintesis de voz o las direcciones en donde residen esos datos. Una vez inicializado el TSP53C30, este opera en uno de los siguientes tres modos de operación:

Modo "host": En este modo, el procesador envía directamente al TSP53C30 los datos para la síntesis de voz.

Modo Directo: El TSP53C30 recibe las direcciones de memoria en donde se encuentran los datos para la síntesis de voz.

Modo Indirecto: En este modo, el TSP53C30 recibe la dirección de una tabla de búsqueda que contiene la dirección de los datos para la síntesis.

En cuanto al formato de los datos utilizados por el TSP53C30, estos formatos pueden ser:

5220: Este formato es utilizado desde el sintetizador TSP5220C.

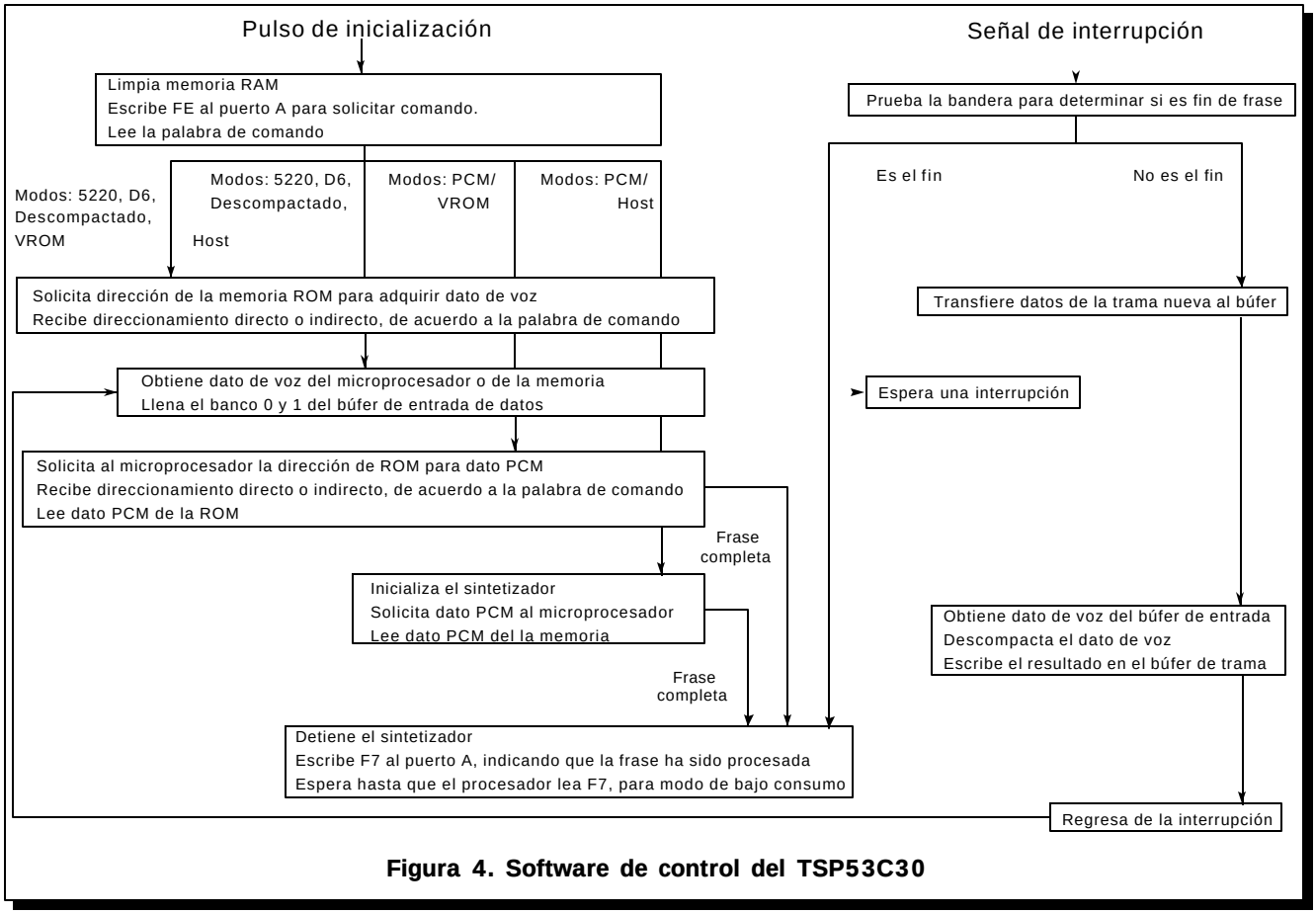


Figura 4. Software de control del TSP53C30

D6: Con este formato, los datos tienen ligeramente una mayor velocidad de transferencia y un control de tono más fino que el formato 5220.

PCM: Debido a la codificación, provee una mayor velocidad de transferencia para los datos.

Descompactados: A diferencia de los formatos anteriores, los datos no son "compactados", lo que permite un control mucho más fino sobre los parámetros de voz. Este es el formato ideal para el modelo LPC.

Las velocidades de Transferencia para estos formatos son:

Formato de Datos	*Bytes por Segundo*
5220	225
D6	250
Descompactado	1000
PCM	10 000

Procedimiento de operación

La tabla 1 muestra los comandos del TSP53C30 para sus diferentes modos de operación.

Para establecer un protocolo de comunicación entre el TSP53C30 y el procesador principal, también es necesario un conjunto de palabras de estado reportadas por el TSP53C30 hacia el procesador. Esto puede apreciarse en la tabla 2.

El TSP52C30 es inicializado al principio de cada palabra o frase, llevando /INIT a estado bajo. El dispositivo permanece en modo de bajo consumo mientras esta terminal permanezca en estado bajo. Una vez que /INIT se vaya a estado alto, el protocolo de comunicación es esta-

Tabla 1. Comandos del TSP53C30

Entrada del puerto P1 durante la inicialización								Formato de Datos	Fuente de Datos de Voz
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	PCM	Procesador principal
n1	n0	s	0	0	1	1	0	5220	Procesador principal
n2	n0	s	0	0	1	0	0	D6	Procesador principal
p2	p1	p0	0	0	0	1	0	Descompactado	Procesador principal
0	0	0	0	i	0	0	1	PCM	Memoria Externa
0	0	s	0	i	1	1	1	5220	Memoria Externa
0	0	s	0	i	1	0	1	D6	Memoria Externa
0	0	0	0	i	0	1	1	Descompactado	Memoria Externa

Tabla 2. Palabras de estado para el TSP53C30

Palabra de Estado								Código	Interpretación
7	6	5	4	3	2	1	0	Hexadecimal	del Estado
1	1	1	1	1	1	1	0	FE	Esperando comando de modo de operación.
1	1	1	1	1	1	0	1	FD	Esperando dirección de dato de voz.
1	1	1	1	1	0	1	1	FB	Esperando dato de voz.
1	1	1	1	0	1	1	1	F7	Fin de síntesis. Se detectó código final.
1	1	1	0	1	1	1	1	EF	Terminación anormal de síntesis.

blecido entre el microprocesador y el sintetizador. Este protocolo se muestra en la tabla 3.

MODO "HOST" CON DATOS DESCOMPRIMIDOS

En este modo, el microprocesador principal envía los comandos y los datos de síntesis, los cuales son:

No. de byte	Dato
1	Longitud de la trama
2	Tono
3*	tono S/Fraccional
4	Energía
5	Energía fraccional
6	Parámetro K1
7	Parámetro K2
8	Parámetro K3
9	Parámetro K4
10	Parámetro K5
11	Parámetro K6
12	Parámetro K7
13	Parámetro K8
14	Parámetro K9
15	Parámetro K10
16	Parámetro K1 fraccional
17	Parámetro K2 fraccional
18	Parámetro K3 fraccional
19	Parámetro K4 fraccional
20	Parámetro K5 fraccional
21	Parámetro K6 fraccional

*El bit más significativo corresponde al código de parada. Los cuatro bits menos significativos son el dato de tono.

La secuencia de la operación a seguir se enlista a continuación:

1. **Permitir a /INIT realizar una transición a estado bajo.**
2. **El TSP53C30 lleva /RDY a estado bajo.**
3. **Lectura de dato del puerto A. Debe ser "FE" (Esperando comando de modo de operación).**
4. **Escribir uno de los siguientes comandos al puerto A (/RDY será puesta en estado alto para escribir el dato de operación).**

Comando	Longitud de la trama	Coeficiente de Reflexión Fraccional
02	15 bytes	Ninguno
22	16 bytes	K1
42	17 bytes	K1, K2
62	18 bytes	K1, K2, K3
82	19 bytes	K1, K2, K3, K4
A2	20 bytes	K1, K2, K3, K4, K5
C2	21 bytes	K1, K2, K3, K4, K5, K6

5. **El TSP53C30 lleva la terminal /RDY a estado bajo.**
6. **Lectura del dato en el puerto A. Si es "FB" (Esperando dato de voz), entonces se realiza el siguiente paso, de otra manera con "F7" (Fin de síntesis) se detecta el código final.**
7. **Se escribe el dato de síntesis de voz al puerto A. /RDY se lleva de estado bajo a estado alto. Se repite desde el paso 5.**

TSP53C30	Procesador Principal
a) Pone "FE" en el latch del puerto A y la linea /RDY en estado bajo	b) Detecta /RDY en estado bajo. Lee el código de petición. escribe el comando de operaciónal puerto A. (Cuando /ENA2 es puesto en estado bajo, /RDY se va a estado alto)
c) Pone "FD"(petición de dirección de dato) o "FB" (petición de dato de voz) en el latch del puerto A y la líne/RDY en estado bajo	d) Detecta /RDY en estado bajo. Lee el código de petición. Escribe la dirección o el dato de voz en el puerto A
e) Se repiten los pasos (c) y (d) hasta que el TSP53C30 detecte el código de finalización al término de una palabra o frase	
f) Código de finalización detectado. Pone "F7" en el latch del puerto A y la línea /RDY en estado bajo	g) Detecta /RDY en estado bajo. Lee el código de estado
h) Permanece en estado de bajo consumo mientras el procesador lee el código de estado "F7"	

Tabla 3. Protocolo de comunicación

COMPRESIÓN DE DATOS EN EL MODELO *LPC*

El modelo LPC toma ventajas de las características de la voz para "ahorrar" información redundante. La señal de voz cambia lentamente y la cavidad oral tiende a caer dentro de ciertas áreas de resonancia más que en otras. La voz es analizada en periodos de 10 a 25 milisegundos. En este periodo considerado, la señal de voz es interpolada de tal manera que no existan cambios abruptos respecto a la siguiente muestra. Además, con el modelo LPC los coeficientes del filtro [5] son constantes (para un periodo repetido de señal de voz) por lo que solamente son requeridos los valores del tono y el factor de energía. Los coeficientes del filtro son mantenidos con los valores anteriores. Sumado a esto, todos los coeficientes son codificados de 7 a 3

bits por cada coeficiente. Esta codificación está hecha de tal forma que se tomen los valores de los coeficientes que ocurren con más frecuencia.

Resultados

Actualmente, el sistema aún se encuentra en una etapa de pruebas de laboratorio, en lo que respecta a la programación de las rutinas de síntesis. Sin embargo, la construcción del prototipo de la tarjeta en su versión para expansión de PC ha sido completada. En la imagen 1 se muestra el prototipo de la tarjeta vista desde la cara de componentes.

Conclusiones

El desarrollo de este proyecto muestra la aplicación de los Sistemas Digitales dentro de la tecnología de voz. Como la mayoría de las aplicaciones de procesamiento de señales, la producción de voz consume muchos recursos informáticos, sin embargo, el uso de un procesador dedicado (el TSP53C30) a la síntesis de voz "libera" de mucho trabajo al sistema, pudiendo enfocarse el diseño y la implementación en el software de control para mejorar la calidad de la voz producida, así como rutinas para edición y reproducción de voz.

El prototipo continuará como auxiliar experimental para la síntesis de voz, permitiendo la investigación más a fondo dentro de esta área. Esto es muy importante, ya que sirve como base para el desarrollo de aplicaciones que exploten la tecnología de la síntesis de voz a través de un sistema dedicado a este fin.

Bibliografía

[1] **Lawrence Rabiner, Biing-Hwang J.** *"Fundamentals of Speech Recognition"*. **Prentice Hall, 1993.**

[3] *"TSP53C30 Speech Synthesizer"*. **Texas Instruments, 1990.**

[4] *"TSP53C0x Family Speech Synthesizer, Design Manual".* **Texas Instruments, 1994.**

[5] **Grice, Donald G. Rensselaer.** *"Adaptive bandpass filtering and its relationship to techniques used in speech compression, synthesis, and recognition"*. **Polytechnic Inst.**

[6] **F. J. Owens.** *"Signal Processing of Speech".* **McGraw-Hill, 1993.**

Imágen 1. Cara de componentes de la tarjeta para Síntesis de Voz con el TSP53C30

Diseño de una Interfaz PCI para una Tarjeta Coprocesadora Basada en el DSP TMS320C40-40

Ing. Israel Rivera Zárate.
Profesor e Investigador del CIDETEC-IPN
M en C. Héctor Samuel García Salas.
Profesor e Investigador del CIDETEC-IPN
M. en C. Antulio Morgado Valle.
Profesor e Investigador del CIC-IPN

En este artículo se presenta el diseño de una interfaz orientada al bus PCI basada en el controlador S5933 de AMCC y lógica programable. La interfaz está integrada en una tarjeta de coprocesamiento basada en el DSP TMS 320C40-40 de Texas Instruments. La interfaz permite transferencias de datos en modos esclavo y maestro del bus PCI. La tarjeta coprocesadora está destinada al tratamiento de gráficos tridimensionales en tiempo real.

INTRODUCCIÓN

El empleo de gráficos tridimensionales en tiempo real se explota ampliamente en aplicaciones de simulación de eventos y procesos, diseño de instrumentos y piezas, y en la Realidad Virtual. Gráficamente un escenario tridimensional se integra por objetos, un objeto es una construcción poligonal, esto es, un conjunto de cuerpos geométricos básicos (por ejemplo prismas, cubos, esferas, planos, etc.), que se definen a su vez por otros más elementales (polígonos), éstos por sus vértices y finalmente cada vértice por sus coordenadas espaciales. Esta información es representada en la computadora como una serie de estructuras de datos referenciadas. Las rutinas principales de tratamiento de gráficos tridimensionales son: escalamiento, rotación, translación, eliminación de superficies ocultas, iluminación, generación de color y despliegue

DESCRIPCIÓN

La presentación gráfica en pantalla en tiempo real exige una frecuencia de exposición de 30 cuadros/seg., lo que conduce a un intervalo de tiempo de 33 mseg. Esto es, se dispone de 30 mseg para la ejecución de las rutinas de tratamiento gráfico y su presentación en pantalla. La resolución gráfica a emplear corresponde a los 320x200 pixeles y paleta de 256 colores (64 KB de datos). Conforme aumenta la complejidad de los escenarios, es decir, aumenta el número de objetos o bien de polígonos, la restricción temporal adquiere mayor importancia. Se sabe además, de la literatura relacionada [1], que el tiempo establecido para la transferencia de cada cuadro no debe exceder una décima parte del intervalo con el fin de no monopolizar el bus, esto es, no debe exceder 3.3 mseg. Lo anterior establece que la tarjeta debe ser capaz de transferir 64 KB en 3.3 mseg, esto es; la tarjeta debe proporcionar una taza de transferencia de 19 MB/s.

Atendiendo a las necesidades de desempeño se seleccionó trabajar con el bus PCI (Peripheral Component Interconect - Interconexión de Componentes Periféricos) por ser un bus local de alto rendimiento, que además se está volviendo un estándar para los sistemas gráficos. La razón de su alto desempeño se debe a que conecta a los dispositivos periféricos directamente al bus del procesador del sistema, proporciona un ancho de datos de 32 bits y opera a una frecuencia de 33 Mhz que le permite desarrollar un ancho de banda de 132 MB/seg. Existe un bus local competitivo denominado VESA o VL-Bus (Video Electronics Standars Asociation - Asociación de Estándares Electrónicos de Vídeo) que opera con una ruta de datos de 32 bits a una frecuencia de 50 Mhz pero que desarrolla una ancho de banda de tan sólo 107 MB/seg; debido a que requiere el empleo de varios ciclos de reloj por dato a tranaferir. Aunado a esto, los fabricantes de equipo de cómputo proporcionan de facto cada vez más, ranuras de expansión (slots) PCI en vez de VL-bus. Respecto a otros buses, se tiene por ejemplo el ISA (Industry Standard Architecture - Arquitectura Estándar de la Industria) cuyo ancho de banda no resuta adecuado a las necesidades ya que alcanza tan sólo los 8 MB/seg; debido a que opera con un ancho de datos de 8 o 16 bits a una frecuencia máxima de 8 Mhz, además que fue diseñado para la operación de periféricos lentos. Por su parte los buses EISA (Extended Industry Standard Architecture - Arqui-

tectura Extendida Estándar de la Industria) y MCA de IBM (Microchannel - Microcanal) aunque posen un ancho de datos de 32 bits no desarrollan una alta taza de transferencia por operar a frecuencias de 8.22 y 10 Mhz respectivamente.

La tarjeta coprocesadora se aloja en una computadora basada en el procesador Pentium de Intel @120 Mhz con un requerimiento mínimo de memoria de 16 MB y monitor Super VGA. El objetivo del diseño estriba en transferir cada cuadro gráfico confeccionado en la tarjeta hacia la memoria de cuadro (frame buffer) de la Adaptadora de Vídeo, mediante un acceso directo de memoria (DMA) en modo maestro del bus PCI sin intervención de la CPU anfitriona. Para llevar acabo este propósito, se eligió el empleo del controlador de bus PCI S5933 de AMCC debido a sus capacidades de comunicación en modos esclavo y maestro del bus PCI, además de su gran flexibilidad para el diseño de interfaces con sistemas basados en micrprocesadores.

Interfaz con el Bus PCI

El controlador de bus PCI S5933 de AMCC cuenta con tres interfaces de bus físicas: una hacia el bus PCI, otra hacia el bus Add-On, y una última hacia una memoria externa no volátil (donde se almacena el contenido del encabezado de configuración). Las transferencias de datos se realizan entre el bus PCI y el bus Add-On donde se interfaza el DSP C40. Las transferencias en el bus Add-On pueden realizarse a través de tres secciones funcionales integradas en el dispositivo: los registros tipo buzón (mailbox registers), las pilas tipo FIFO (first input first output – lo que entra primero sale primero), o bien, la trayectoria directa de datos (Pass-Thru data path) [2].

La comunicación hacia el bus PCI se realiza mediante lógica interconstruida en el controlador, lo que resta es determinar su forma de operar en el caso de la aplicación particular. La operación del controlador es definida en registros de control que pueden maperase en el espacio de puertos o de memoria (definido en los registros del encabezado de configuarción del dispositivo). Por lo tanto deben reservarse estos recursos del sistema.

La memoria ROM es la 24C02 de National Semiconductor, es del tipo serial, aunque puede utilizarse una de 8 bits; pero es condición necesaria para el manejo de otras terminales que permiten la operación en modo maestro del bus PCI. Esta memoria resguarda la información del encabezado de configuración y de otros parámetros de programación del controlador.

Los registros tipo buzón del S5933 ofecen una ruta bidireccional de datos de 32 bits y son establecidos principalmente para la transferencia de datos de control de programa, comandos, e información del estado de la ejecución de aplicaciones. Además, se pueden configurar interrupciones dirigidas al bus PCI o al bus Add-On basadas en la ocurrencia de eventos específicos en los buzones.

La interfaz que establece una ruta directa entre el bus PCI y el bus de la aplicación Add-On permite mapear memoria integrada en la tarjeta, esto es: el procesador anfitrión accesa a la memoria de la tarjeta considerándola parte del sistema. Desde el punto de vista del procesador anfitrión la memoria está ubicada en direcciones de su mapa de memoria.

Por su parte, el DSP C40 cuenta con su propio espacio de direcciones y define la ubicación de la memoria donde le sea conveniente. La correspondencia entre la dirección generada por el procesador anfitrión y el C40 recibe el nombre de mapeo, y en el caso particular, es realizado por el propio C40. La dirección de la memoria de la tarjeta en el mapa de memoria del anfitrión es asignada por el Bios o el sistema operativo durante la fase de inicialización de los dispositivos instalados en los buses de expansión, y que cumplen con el estándar plug and play [2],[3]. A través de esta interfaz el anfitrión puede enviar las estructuras de datos de nuevos escenarios a procesar.

El controlador S5933 cuenta con dos pilas separadas tipo FIFO. Una de ellas maneja el movimiento de datos del bus PCI hacia el bus Add-On y la otra lo hace en dirección opuesta. Ambas se integran por 8 Dwords (palabras de 32 bits). El bus PCI requiere que la transferencia de datos se haga en forma de ráfaga de datos (burst), es decir; una fase de direcciones y fases sucesivas de datos. El S5933 permite la comunicación de datos de manera síncrona o asíncrona a la señal de reloj del bus PCI de 33 Mhz. Un sistema logrará máxima transferencia sólo si es capaz de efectuar accesos en modo ráfaga síncronos.

En el caso particular, un DSP C40 desarrolla ciclos de bus de la mitad de la frecuencia de la señal de reloj, es decir; para el C40-40 son de 20Mhz. La diferencia de frecuencias entre el procesador y el controlador de bus impiden establecer de antemano un sistema síncrono a la frecuencia del bus PCI.

El análisis de diagramas de tiempo relacionados con la taza máxima de transferencia posible a través de las interfaces Pass-Thru y FIFO, revelan que debido a su comportamiento asíncrono respecto al bus PCI y a la necesidad de un estado de espera debido a la lógica programable, ambas son capaces de desarro-

llar como máximo 20 MB por segundo. Por lo tanto en operación como maestro del bus, la interfaz diseñada satisface adecuadamente la taza de transferencia requerida. Como esclavo, el controlador S5933 puede emplear cualquiera de las dos interfaces. Cabe mencionar que el controlador restringe la operación en modo maestro únicamente a la FIFO.

Por su parte, el DSP C40-40, cuenta con dos buses externos denominados como Local y Global respectivamente, cada uno consiste de un bus de datos de 32 bits, un bus de direcciones de 31 bits y de dos conjuntos de señales de control. El C40 ejecuta instrucciones en un solo ciclo de bus, centra su desempeño en una pipeline multietapa y en buses internos individuales para manejo de direcciones y datos para acceso a memoria de programa y memoria de datos, lo que le permite la ejecución de hasta 11 operaciones por ciclo. Ofrece un desempeño de 20 MIPS [4].

El C40 disminuye su rendimiento cuando en sus accesos a memoria opera con estados de espera, lo anterior repercute en la ejecución de la pipeline debido a que debe esperar antes de continuar en la siguiente etapa. Para explotar su máximo desempeño se determinó el empleo de memorias de cero estados de espera (tiempo de acceso necesario=20 ns) tanto para almacenar el código, como para datos. Debido a que el C40 posee un bus de datos de 32 bits, su código está representado en formato de 32 bits, es decir; las instrucciones de programa tienen longitud de 32 bits. Por lo anterior se consideró el empleo de memorias con capacidad de 128Kx32.

Para lograr esto, para el almacenamiento del código se descartó el empleo de ROMs debido a su baja densidad y su alto tiempo de acceso. Para memorias de datos, se descartó el empleo de memorias dinámicas debido a la necesidad del empleo de controladores.También se descartaron las memorias cache por la necesidad de un controlador y su baja densidad.

El C40 cuenta con un programa almacenado en su ROM interna (Boot Loader Program – Programa Cargador de Arranque), que le permite copiar el contenido de una memoria ROM externa en una memoria RAM externa y pasarle el control. De esta forma arranca con una memoria lenta que requiere varios ciclos de espera pero ejecuta finalmente su código a máxima velocidad desde la RAM. Las memorias elegidas son SIMMs de Cypress con densidad de 128Kx32 y con un Tacc=20ns.

Interfaz con el Controlador de Bus PCI S5933

Atendiendo a las necesidades de comunicación asíncrona entre el controlador de bus PCI S5933 y el DSP C40, se realizó un estudio de los ciclos de bus de ambos dispositivos, llegando al establecimiento de una lógica programable basada en tres PLDs GAL22V10 (10ns). La programación de estos dispositivos permite la lectura y escritura a los registros del controlador del lado de la interfaz Add-On (donde se interfaza el DSP C40), el mapeo de memoria integrada en la tarjeta a través de la interfaz Pass-Thru, también la lectura y escritura a las FIFOs respectivamente, así mismo las acciones de control necesarias para volverse maestro del bus PCI y la lógica de generación de estados de espera del C40. Ver figura 1.

Decodificador de Direcciones

En esta aplicación, se destinó una memoria de datos y una memoria de

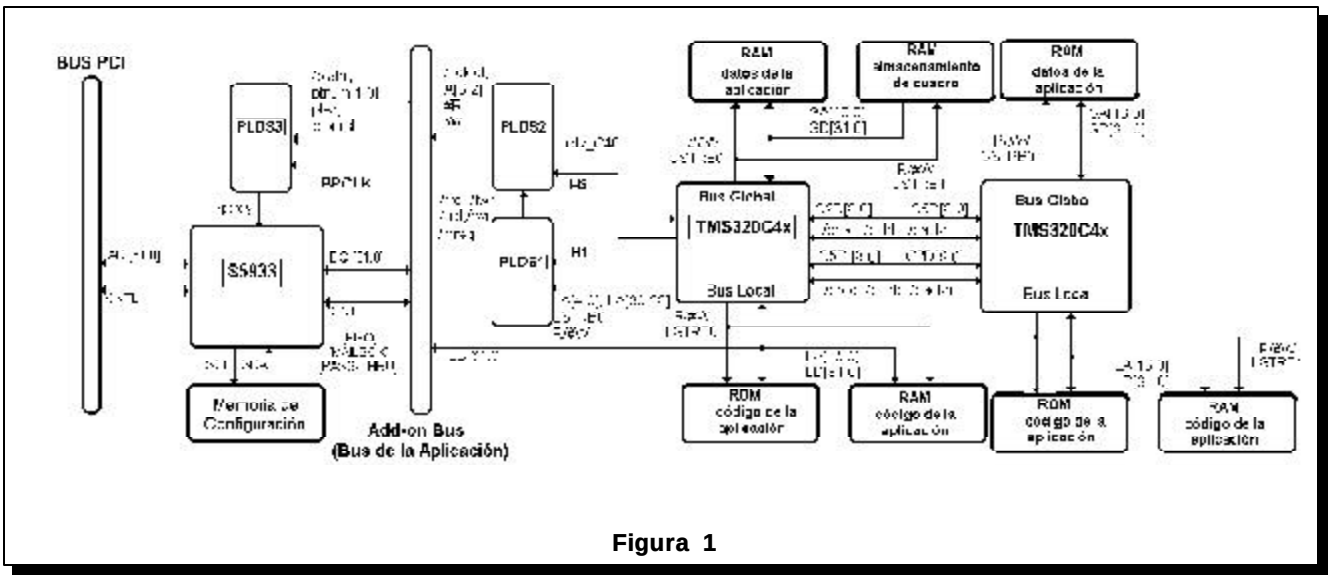


Figura 1

cuadro (para el almacenamiento del cuadro a transmitir vía DMA) del lado del bus global, en tanto que el controlador S5933 es accesado como un dispositivo mapeado dentro del espacio de direcciones del bus local. El espacio de direcciones del bus local se dividió en dos regiones principales mediante la escritura al campo STRB ACTIVE del registro de control de memoria local; una abarca la ROM de código de la aplicación y al controlador, y la otra a la RAM donde se respalda la ROM de código para ejecución a máxima velocidad. La GAL22V10 etiquetada como PLDS1 decodifica los ciclos de bus local del C40 empleando la fase de reloj H1 del C40. Además, mediante la señal de control STRB0 y las líneas de direcciones LA30-LA28 y LA4-LA0 (LA31 es implícitamente un 0 lógico), divide la región en 4 secciones correspondientes con los registros de control e interfaces del bus Add-On del controlador S5933. El mapa de memoria del C40 decodificado por el PLDS1 aparece en la tabla 1 .

Dirección	Unidad Direccionada
0000 0000h	ROM interna (Programa de Carga para Arranque)
3000 0000h	RAM código (Respaldo de la ROM de código)
4000 0000h	Solicitud Maestro PCI
5000 0000h	Mailboxes
5000 0008h	FIFO
5000 000Ah	Direcciones Pass-Thru
5000 000Bh	Datos Pass-Thru
5000 000Ch	Control
6000 0000h	ROM externa (Código de la aplicación)
8000 0000h	RAM datos (Datos de la aplicación)

Tabla 1. Asignación del espacio de direcciones del DSP C40.

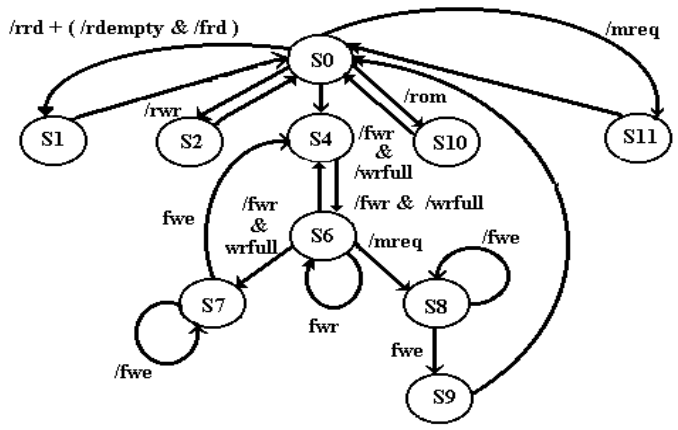


Figura 2. Diagrama de estados del generador de ciclo Maestro y Esclavo del bus PCI.

Generación del Ciclo Maestro del Bus PCI

El DSP C40 genera ciclos como maestro del bus mediante accesos de escritura a la interfaz FIFO.

Para identificar esta condición la salida del PLDS1 es conducida a un segundo PLD, etiquetado como PLDS2. Cuando la FIFO alcanza su máxima profundidad (8 Dwords), la lógica integrada en el controlador para el manejo de la FIFO genera en el bus Add-On la señal WRFULL. El PLDS2 es una máquina de estados sincronizada con la fase de reloj H3 del C40, y ante la presencia de la señal WRFULL genera la señal de control AMWEN requerida por el controlador para solicitar la poseción del bus. Para garantizar que el C40 no accese a la pila bajo esta condición, genera los estados de espera necesarios manipulando adecuadamente la señal LRDY0 del C40. La razón del empleo de la fase de reloj H3 se justifica en términos de un menor empleo de estados de espera atribuidos a los retardos de propagación de la lógica programable. El PLDS2 genera además las señales de control para el acceso a la ROM de código, el acceso asíncrono a los registros de control y a los registros tipo buzón. El diagrama de estados de este PLD aparece en la figura 2 .

Generación del Ciclo Esclavo del Bus PCI

El C40 acepta accesos como dispositivo esclavo del bus PCI a través de la FIFO o la Pass-Thru. El PLDS2 controla los accesos basados en la FIFO. Para realizar accesos mediante la Pass-Thru se hace necesario guardar sincronía con el reloj del bus PCI. Por esto, el PLD etiquetado como PLDS3 constituye una máquina de estados sincronizada con la señal BPCLK generada por el controlador S5933. La señal BPCLK es la versión reforzada de la señal de reloj PCI.

La interfaz Pass-Thru cuenta con una terminal de entrada para la generación de estados de espera, el PLDS3 maneja esta terminal tomando en cuenta el estado de la señal LRDY0 que le es suministrada por el PLDS2. El diagrama de estados aparece en la figura 3.

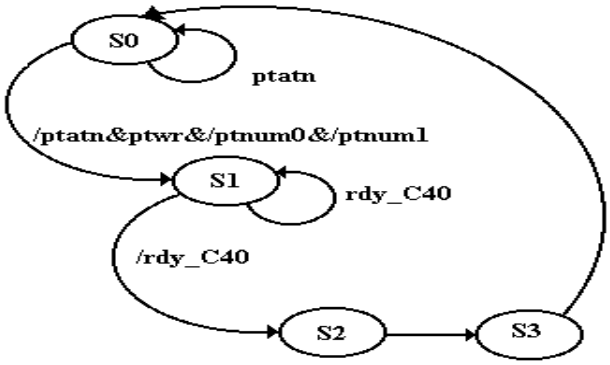


Figura 3. Diagrama de estados para acceso a la Pass-Thru.

CONCLUSIONES

Actualmente la tarjeta se encuentra en fase de prueba de las rutinas de comunicación con la tarjeta y se basan en las bibliotecas escritas en lenguaje C proporcionadas por AMCC.El código de tratamiento gráfico se elaboró en lenguaje C y actualmente se está depurando en el programa de simulación del DSP C40 de Texas Inst. La integración del sistema está muy próxima y dará paso a pruebas y caracterización del prototipo.

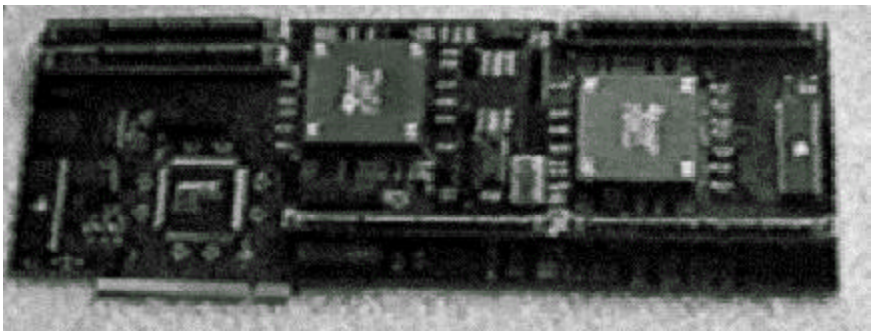
BIBLIOGRAFÍA

[1] *"Virtual Reality Systems"***. R. A. Earn Shaw. Academic Press, 1993.**

[2] AMCC S5933 PCI Controller Data Book,1997.

[3] *"PCI System Arquitecture"***. Tom Shanley. Addison Wesley, 1994.**

[4] TMS320C40 User's Guide. 1993.



Implementación de Manejadores de Dispositivo (Device Drivers)

Ing. José Ortega Bernal
Catedrático de la U. V. M.
Alumno de la Maestría del CIC-IPN
M. en C. Eduardo Vega Alvarado
Profesor del CIDETEC-IPN

Indudablemente, MS-DOS® (MicroSoft Disk Operating System) ha sido el sistema operativo de mayor éxito en la historia, estando presente en una amplia base de equipos de tipo personal. Si bien se le considera obsoleto después de la aparición de nuevos sistemas tales como Windows, DOS aún representa una herramienta flexible y sumamente poderosa como monitor para equipos dedicados, principalmente para labores de control. Así, mediante el desarrollo de la interfaz (circuitería) para control y un manejador de dispositivos adecuado, una computadora PC trabajando en DOS puede desempeñar funciones que difícilmente se programarían en Windows, dado el paradigma de programación de espera de eventos de este último.

INTRODUCCIÓN

Los manejadores o controladores de dispositivos (*device drivers*) constituyen uno de los aspectos más fascinantes (a la vez que complejos) de la programación de sistemas. Estos pequeños programas controlan el acceso y la comunicación con los dispositivos de entrada/salida externos, representando el nivel más bajo del sistema operativo; dadas sus características, estas rutinas normalmente deben programarse en lenguaje ensamblador.

En el caso especial del DOS, sus manejadores tienen la característica de incorporarse al kernel al momento de inicio de sesión, lo que proporciona una gran flexibilidad. Así, a diferencia de sistemas operativos como UNIX, donde la carga de nuevos controladores o la modificación de los ya existentes implica una recompilación del núcleo, en MS-DOS simplemente se requiere cambiar algunas líneas en el archivo de configuración CONFIG.SYS y reiniciar el equipo.

ESTRUCTURA BÁSICA

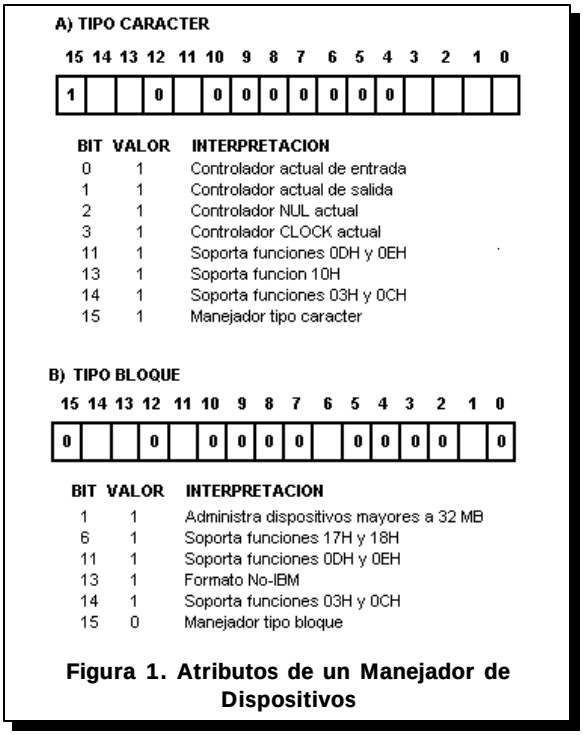
En MS-DOS se dispone de dos tipos de manejadores, correspondiendo a los dispositivos de carácter (*character*) y de bloque (*block*), respectivamente; los dispositivos de carácter transmiten información byte por byte, mientras que los de bloque comunican conjuntos de información, obviamente denominados bloques. Aunque los dos tipos difieren en varios aspectos, su estructura general es la misma. Cada controlador de dispositivos consiste básicamente de información de estado (*status*), y de una serie de rutinas para controlar al dispositivo; la comunicación se basa entonces en un protocolo de llamadas a dichas rutinas.

La estructura básica incluye 6 módulos principales:

1. Encabezado (*header*), área de datos con una longitud de 18 bytes, que proporciona la información necesaria para la activación del *driver*. Las 2 primeras palabras del encabezado (bytes 00H a 03H) indican la dirección relativa (desplazamiento - segmento) del controlador siguiente en la cadena interna del sistema. Al escribir el *driver*, el programador debe inicializar estos datos con el valor -1 (FFFFh); cuando se carga el manejador, DOS actualiza automáticamente estas referencias. La siguiente palabra corresponde al atributo, y en el se describe el tipo de manejador y sus características. Así, se tiene una distribución de los bits del atributo para los dispositivos de carácter, y otra para los de bloque; ambas interpretaciones se muestran en la figura 1, en la siguiente página.

Las dos palabras a continuación son apuntadores a los desplazamientos (*offset*) donde se encuentran las rutinas de estrategia y de interrupción, respectivamente. Finalmente, los últimos 8 bytes contienen el nombre del controlador, si este es tipo carácter, o el número de dispositivos lógicos soportados, para un manejador tipo bloque.

2. Tabla de salto a las funciones. Esta tabla contiene referencias al *offset* (dirección relativa) de cada



A) TIPO CARACTER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1			0		0	0	0	0	0	0	0				

BIT	VALOR	INTERPRETACION
0	1	Controlador actual de entrada
1	1	Controlador actual de salida
2	1	Controlador NUL actual
3	1	Controlador CLOCK actual
11	1	Soporta funciones 0DH y 0EH
13	1	Soporta funcion 10H
14	1	Soporta funciones 03H y 0CH
15	1	Manejador tipo caracter

B) TIPO BLOQUE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			0		0	0	0	0		0	0	0	0		0

BIT	VALOR	INTERPRETACION
1	1	Administra dispositivos mayores a 32 MB
6	1	Soporta funciones 17H y 18H
11	1	Soporta funciones 0DH y 0EH
13	1	Formato No-IBM
14	1	Soporta funciones 03H y 0CH
15	0	Manejador tipo bloque

Figura 1. Atributos de un Manejador de Dispositivos

una de las funciones (código) implantadas en el manejador.

3. Rutina de estrategia. Esta rutina, a pesar de su nombre, no plantea la forma de operar del controlador. Su función es establecer la comunicación entre DOS y el manejador, para inicializar a este último. Cuando DOS va a invocar a un controlador, se construye un encabezado de petición (*request header*) en un área reservada de memoria, y su dirección se transmite a la rutina de estrategia por medio de los registros ES:BX. El encabezado de petición sirve como espacio de comunicación, en donde DOS indica al controlador que debe hacer, y el manejador responde con el resultado de la operación. La figura 2 muestra una estructura típica del encabezado de petición.

+00H	Longitud en bytes del bloque de datos	1 word
+01H	Número de comunicación del dispositivo	1 word
+02H	Código de comando	1 word
+05H	Reservado	8 bytes
+0DH	Descriptor del medio	1 byte
+0EH	Dirección del buffer (offset)	1 word
+10H	Dirección del buffer (segmento)	1 word
+12H	Número	1 word
+14H	Sector de inicio	8 bytes

Figura 2. Estructura del Encabezado de Petición

4. Rutina de interrupción. Esta porción del controlador realiza todo el trabajo de control del dispositivo. Primeramente debe guardar el estado de todos los registros y banderas del procesador, para evitar su modificación, restituyéndolos al final de la operación. Posteriormente obtiene, del encabezado de petición, el código de comando a ejecutarse, verificando si este código es válido para ejecución con las rutinas del manejador. Si la operación es permitida, transfiere el flujo del programa a la rutina especializada y la procesa. En caso contrario, se detiene el curso del programa, desplegando un mensaje de error.

5. Funciones soportadas por el manejador. En esta zona se implementan todas las rutinas de operación del controlador. Por ejemplo, un controlador de consola debe contener rutinas de lectura y almacenamiento del teclado, carácter por carácter, así como de despliegue de esta información. Todo manejador debe soportar al menos 13 funciones básicas, numeradas de 00H a 0CH, aún y cuando su única acción sea activar la bandera de REALIZADO, en la palabra de estado. Además, se cuenta con funciones adicionales que corresponden a alguna versión específica de DOS o al tipo de controlador (bloque o carácter); la siguiente lista muestra las funciones principales:

FUNCION	SIGNIFICADO
00H	Inicialización del controlador
01H	Verficación del medio
02H	Construir bloque de parámetros de BIOS
03H	Lectura IOCTL
04H	Lectura
05H	Lectura no destructiva
06H	Estado (*status*) de entrada
07H	Buffers de flujo de entrada
08H	Escritura
09H	Escritura con verificación
0AH	Estado de salida
0BH	Buffers de flujo de salida
0CH	Escritura IOCTL
0DH	Abrir
0EH	Cerrar dispositivo
0FH	Medio removible
10H	Salida hasta ocupado
11H	IOCTL genérico
13H	IOCTL genérico
17H	Obtener dispositivo lógico
18H	Determinar dispositivo lógico

Todas estas funciones reciben sus argumentos del encabezado de petición, y almacenan sus resultados dentro del mismo. Este tipo de mecanismo libera al sistema operativo de la necesidad de conocer la dirección de cada función del manejador; el DOS simplemente utiliza las rutinas de estrategia y de interrupción como interface a las diversas funciones implementadas en el controlador.

Para entender mejor lo anterior, a continuación se describe a los parámetros de mayor relevancia solicitados por el controlador, para procesar la Función 01h (verificación del medio, *media check*); en la figura 3 , se observa la estructura correspondiente al encabezado de petición para esta función.

Esta operación la utilizan los controladores de dispositivos de bloque, para determinar cuando el medio (disco) ha cambiado. Estos cambios se determinan comparando la información actual en memoria después del último acceso a disco, con la correspondiente al acceso anterior. Si hubo cambios, el sistema operativo debe releer la información del

PARAMETROS DE ENTRADA

OFFSET	CONTENIDO	TIPO
01H	Número del dispositivo	Byte
02H	Número de función	Byte
13H	Descriptor del medio	Byte

PARAMETROS DE RETORNO

OFFSET	CONTENIDO	TIPO
03H	Palabra de estado	Word
14H	Cambio en el medio? FFH=si, 00H=No sé, 01H=No	Byte
15H	Dirección del buffer con el nombre del volúmen (solo si el apuntador indica cambio del medio)	Apuntador

Figura 3. Función 01H: Verificación del Medio

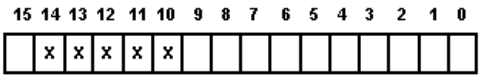
disco, lo cuál retardará la ejecución; en caso contrario, el contenido de memoria es correcto. Sin embargo, en medios tales como el disco flexible (*floppy disk*), la contestación a la pregunta *¿Hay cambios en el medio?* **es más complicada, por lo que DOS permite a la función contestar** *si* **o** *no*, **pero también** *no sé*.

Las posibles contestaciones dependen del manejo que realiza el sistema operativo con la información. Cuando el medio no ha cambiado, el acceso al mismo es inmediato; si hay cambios, DOS cierra los buffers relacionados con el dispositivo lógico actual, ocasionando la pérdida de los datos que debían ser transmitidos. Si la función contesta *no sé*, **la operación que realice el sistema operativo depende del estado del buffer relacionados con el dispositivo lógico actual; si está vacío asume que el medio ha cambiado y contesta entonces** *si*; **pero si contiene datos que debían ser transmitidos al medio, MS-DOS considera al medio como intacto y escribe en él, pudiendo ocasionar daños a la estructura del medio nuevo. En la figura 4 , se observa la estructura de la Palabra de Estado dentro del encabezado de petición, en la zona relativa a los códigos de error.**

6. Identificador. El nombre del controlador tiene doble uso, ya que además de proporcionar una identificación sirve para que, en caso de tener el nombre duplicado (esto es, otro manejador tenga el mismo nombre), el último en ser dado de alta sea el que se ejecute; lo anterior es sumamente útil cuando se requiere sustituir a un manejador por otro propio. En caso contrario, de tratarse de un controlador autónomo nuevo, éste se anexa a la cola de controladores del sistema.

IMPLEMENTACIÓN

Una vez diseñado el controlador de dispositivos, su implementación se basa en el manejo de herramientas de programación en lenguaje ensamblador, tales como TurboAssembler (Borland) y MacroAssembler (Microsoft). A continuación se incluye el listado de un manejador ejemplo, el cual reemplaza al controlador de consola estándar (CON); en el se indica claramente cada uno de los componentes de cualquier controlador.



BIT	VALOR	INTERPRETACION
8	1	Llamada ejecutada (ready)
9	1	Manejador ocupado (busy)
10-14	X	No usados
15	1	Sin error
15	0	Error

Si se presenta error, los bits 0 a 9 indican el tipo de fallo:

CODIGO	ERROR
0	Medio protegido contra escritura
1	Dispositivo desconocido
2	Manejador no listo
3	Comando desconocido
4	Error de lectura del CRC
5	Bloque de datos del parámetro
6	Error de búsqueda
7	Medio desconocido
8	Sector no encontrado
9	Impresora sin papel
10	Error de escritura
11	Error de lectura
12	Error común
15	Cambio ilegal de medio

Figura 4. Códigos de Error en la Palabra de Estado

Para probar algún manejador recién programado, lo recomendable es incluirlo en un disco flexible de arranque, para evitar daños en el sistema instalado en las unidades de disco duro. Como ya se mencionó, el archivo CONFIG.SYS constituye la vía de inclusión para los controladores, por lo que se debe agregar la línea

```
Device = X
```

donde X representa tanto la trayectoria donde está situado el archivo del controlador como la especificación del mismo (nombre y extensión, normalmente SYS).

CONSIDERACIONES FINALES

Los controladores de DOS tienen un inconveniente: dada sus estructura especial, solo pueden escribirse en lenguaje ensamblador. Sin embargo, esta aparente deficiencia se compensa con la mayor velocidad de operación de las rutinas resultantes, comparadas con programas equivalentes desarrollados en lenguajes de alto nivel. Las características de este tipo de programas los asemejan (en reglas de estructura) con los programas con extensión COM, pero se diferencian en que los controladores inician su código en el desplazamiento 0h, mientras que los programas COM lo hacen en el *offset* **100h.**

También debe mencionarse que en ocasiones la comunicación entre el sistema y un manejador puede resultar sumamente complicada para el programa-

dor, debido a la falta de documentación sobre las diversas funciones soportadas para un dispositivo. Este problema puede ser aún más importante si se considera la existencia de muchas características de MS-DOS no documentadas, por lo que su aplicaciòn puede producir resultados inesperados.

A pesar de estos aparentes inconvenientes, los manejadores de MS-DOS constituyen una alternativa flexible y poderosa para implementaciones de control por computadora. La facilidad para integrarlos al sistema permite una actualización permanente, sin efectos negativos sobre el rendimiento.

BIBLIOGRAFÍA

[1] **Abel, Peter.** *"IBM PC Assembly Language and Programming"***. Prentice-Hall, 1995.**

[2] **Tischer, Michael.** *"PC Intern"***. ABACUS 1995.**

[3] **Barkakati; Nabajyoti & Hyde, Randall.** *"Microsoft Macro Assembler Bible"***. SAMS 1992.**

[4] **Dettmann, Terry.** *"DOS Programmer´s Reference"***. QUE 1990.**

[5] **Schulman, Andrew.** *"El DOS no Documentado"***. Addison Wesley 1995.**

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 Innovación y Desarrollo Tecnológico en Cómputo
Consejo Editorial Revista POLIBITS
Av. Té No. 950, Edificio de Graduados 2o. piso.
Colonia Granjas México. C.P. 08400
México D. F.
☎ 5624-2019, 5624-2015, 5624-2000 ext. 70256
FAX 5624-2019 **e-mail** *erodrig@redipn.ipn.mx*

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