COORDINACION DE SERVICIOS
DOCUMENTALES - BIBLIOTECA

UNIVERSIDAD AUTÓNOMA METROPOLITANA IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

SISTEMA AUXILIAR PARA EL PROCESAMIENTO DIGITAL DE SEÑALES EN TIEMPO REAL

TESIS PARA OBTENER EL GRADO DE
MAESTRÍA EN INGENIERÍA BIOMÉDICA

JOSÉ JOAQUÍN AZPIROZ LEEHAN

227469

ASESOR: DR. ADRIANO DE LUCA P.

México, D. F., octubre 1988

AGRADECIMIENTO

Al M. en C. Miguel Lindig Bös, por sus ideas iniciales para generar este proyecto.

INDICE

INDICE

INTRODUCCION

INTRODUCCION

La mención del desarrollo del algoritmo de la transformada rápida de Fourier por Cooley y Tukey y el impacto de éste en los avances del análisis de señales es ya un lugar común. Sin embargo, este impacto no es comparable al logrado por el desarrollo de las microcomputadoras (y muy especialmente las computadoras tipo PC) y la introducción de éstas a muchos laboratorios donde la adquisición y el uso de una computadora tenía un costo prohibitivo. Desde la aparición de la computadora personal de IBM en 1981 y todas las compatibles, se han desarrollado cientos de aplicaciones a su alrededor dentro del área de procesamiento digital de señales.

Dentro del área de ingeniería biomédica de la UAM-Iztapalapa, la adquisición de este tipo de computadoras ha permitido la constitución de un grupo de trabajo en análisis de señales biomédicas con un laboratorio que aunque modesto, ha permitido el desarrollo desde simples sistemas de graficación de archivos, con fines didácticos, hasta sistemas complejos de análisis de señales estocásticas, que requieren de grandes tiempos de procesamiento.

A pesar de las bondades que se presentan en el uso de las microcomputadoras, sobre todo por lo relacionado a su bajo costo, existen limitaciones que deben considerarse con algún cuidado. La experiencia en los proyectos desarrollados por el grupo de trabajo mencionado ha mostrado que existen limitaciones en cuanto a la velocidad de ejecución de instrucciones de las computadoras tipo PC. Los 2 últimos proyectos de este grupo: Análisis de señales de impedancimetría transtorácica (Charleston,1988) y Análisis espectral del electroencefalograma (Medina, 1986), tienen tiempos de procesamiento que van de 18 a 35 minutos. Este problema es una seria limitante para lograr aplicaciones

concretas a nivel industrial fuera de un laboratorio de investigación. Algunas soluciones propuestas como la utilización de procesadores numéricos o la adquisición de computadoras más poderosas tienen el inconveniente de incrementar los costos de desarrollo en una relación de costo/incremento de velocidad que no es aceptable.

Con el propósito fundamental de incrementar la velocidad de procesamiento digital de señales se plantea el objetivo de este proyecto como el desarrollo y evaluación de un sistema que permita llevar a cabo algunas de las rutinas para procesamiento de señales biomédicas en tiempo real. Más específicamente se trata de un conjunto de procedimientos escritos en los lenguajes Pascal y ensamblador para 8088 y una unidad auxiliar alambrada para procesamiento de señales que interaccionan dentro de un ambiente de procesamiento en paralelo.

ANTECEDENTES

ANTECEDENTES

Un sistema de procesamiento auxiliar tiene como característica fundamental el ejecutar funciones de procesamiento de señales que resultan ser de ejecución lenta en computadoras personales. El "estado del arte" actual permite tomar 3 caminos diferentes para resolver los problemas de velocidad de ejecución en microcomputadoras, a saber:

Procesadores Más Rápidos

Una de las opciones que se utilizan constantemente para aumentar la velocidad del procesamiento de señales es el utilizar computadoras cada vez más poderosas, junto con variaciones de los algoritmos fundamentales del procesamiento. Se han comparado diversos algoritmos y distintas computadoras para determinar cual es la mejor opción para el cálculo de la transformada de Fourier. Estos estudios comparativos se han hecho para operaciones en punto flotante y para longitudes de 504 a 2520 datos. Como ejemplo, mostramos la tabla 1, donde se puede ver que para 1024 datos, la computadora Cray1 toma 8.98 ms, comparado con 360 ms para una VAX 11/780 y 40,000 ms para una Cromemco. Los programas están escritos en Fortran, y los tiempos de ejecución dependen tanto del compilador como de la arquitectura de la computadora (Mehalic et.al.,1985; Morris, 1978; Blanken and Rustan, 1982).

COMPUTADORA	DATOS	TIEMPO(ms)
Cray1	1024	8.98
Cyber750	1024	24
IBM370	1024	404
VAX11/780	1024	360
PDP11/60	1024	566
PDP11/50	1024	1452
Cromemco	1024	4000

TABLA 1.
Tiempos de ejecución de la FFT en distintas computadoras.

fisiología, es raro encontrar una relación señal/ruido mejor a los 60 dB, lo cual sugiere que no es necesario utilizar cálculos con punto flotante o con grandes precisiones para procesar su rango dinámico.

Procesadores de Señales

Otra modalidad para mejorar la velocidad en el análisis de señales es el de utilizar procesadores especializados con estas funciones conjuntas con la microcomputadora. Los procesadores aritméticos de alta velocidad se conocen como procesadores de arreglos (array processors) o procesadores de señales (digital signal processors). En estas configuraciones de multiprocesamiento, los procesadores de arreglos llevan a cabo las tareas de procesamiento de los datos, mientras que la computadora huésped se dedica al manejo y supervisión de los dispositivos periféricos (Caspe, 1978). Los ejemplos más conocidos de éstos son los procesadores de la serie TMS320 de Texas Instruments, que están basados en la arquitectura Harvard de mapeo separado de memoria de programa y memoria de datos, de tal manera que las operaciones y la adquisición de instrucciones se ejecutan paralelamente. Estos procesadores contienen básicamente un multiplicador paralelo, una unidad aritmética y lógica y un registro de corrimientos. Típicamente, su ciclo de operación es de 200 ns (Texas Instruments, 1986), con su arquitectura que refleja el criterio de tener un conjunto de instrucciones reducido (arquitectura RISC), sin tener la versatilidad de un microprocesador convencional, pero es muy eficiente cuando se trata de ejecutar tareas de procesamiento numérico. Además del incremento de velocidad que se obtiene por el multiplicador paralelo interno, el procesador tiene la facilidad de manejar varias operaciones en paralelo, como por ejemplo, la transferencia de datos, el escalamiento y las operaciones matemáticas. El TMS32020, particularmente, es un procesador poderoso, con ciclos de 150 ns y con la facilidad de hacer una operación de

multiplicación-acumulación con corrimiento, en un solo ciclo. Tiene la facilidad de conectarse en un esquema de multiprocesamiento, con la capacidad de ejecutar una FFT radix 2 de 256 datos complejos en 5.06 ms (Essig, 1986).

Los procesadores WE DSP16 y WE DSP32 de AT&T son específicos para el procesamiento de señales. El DSP16 es un dispositivo programable construido con tecnología CMOS que tiene a su salida 84 patas en un encapsulado LCC. Es un circuito que se puede utilizar como bloque básico para desarrollar sistemas de procesamiento de señales. Tiene una alta velocidad de transferencia de información debido a su arquitectura con separación de líneas de instrucciones y datos y decodificación previa de instrucciones (arquitectura tipo "pipeline"); una unidad aritmética y lógica que es capaz de efectuar una multiplicación de 16 x 16 bts en 75ns; una memoria de programa interna tipo ROM de 2048 palabras y una memoria de datos RAM de 512 palabras, junto con un conjunto completo de instrucciones para el procesamiento de señales y la transferencia de información. El WE DSP32 es un producto similar pero con una palabra interna de 32 bits, con mayor cantidad de memoria y con la posibilidad de utilizar aritmética de punto flotante. Su arquitectura es la de una computadora típica RISC (Hays, 1985; AT&T, 1986).

Existen otros procesadores de señales que utilizan circuitos integrados específicos para el procesamiento digital de señales, tales como las tarjetas DF-1 de Data Flow Imaging, Inc. y el procesador vectorial (Vector Signal Processor) de Zoran. Las tarjetas DF-1son coprocesadores para computadoras tipo IBM-PC y que utilizan a 4 circuitos NEC uPD7281 (Image Pipelined Processors) y un uPD9305 (Memory Access and Bus Interface Chip) para poder llevar a cabo 20 MIPS (millones de instrucciones por segundo) dentro de una estructura de procesamiento en paralelo para datos en formato de punto fijo (16 bits mas signo), donde cada

uPD7281 puede efectuar 5 MIPS, ya que cuenta con un multiplicador interno de 200 ns. Sus aplicaciones pueden ser el procesamiento de vectores y matrices, , el procesamiento digital de señales, el procesamiento de imágenes y en redes neurales (Data Flow Imaging, 1986). El procesador de Zoran (ZR34161) cuenta con operaciones complejas como instrucciones para hacer procesamiento por vectores (FFT, Complejo Conjugado, Multiplica, Valor Absoluto, Almacena Vector). Es capaz de ejecutar una FFT de 1024 datos complejos en 2.4 ms con aritmética de precisión fija (Electronics, 1986).

Los circuitos y sistemas que se han descrito anteriormente ofrecen la posibilidad de llevar a cabo el procesamiento de señales de manera rápida y a un costo menor al de la adquisición de una minicomputadora. Sin embargo, el desarrollar sistemas alrededor de estos componentes es difícil para localidades o laboratorios con presupuestos limitados, ya que todos requieren de la adquisición de sistemas de desarrollo que pueden resultar ser costosos.

Un ejemplo de aplicación de estos elementos para el análisis de señales biomédicas es el descrito por Wunk, donde utilizan una tarjeta compatible con el bus de la computadora tipo IBM-PC (SKY320), desarrollada por Sky Computers. Esta tarjeta coprocesadora contiene 128 Kbytes de RAM para almacenamiento de datos y 8Kbytes de RAM para programa. Estas memorias utilizan los mismos espacios de direccionamiento de la computadora principal, de tal manera que este sistema se comporta como una expansión de memoria. En esta aplicación, se lleva a cabo la estimación de la densidad espectral de señales electroencefalográficas para 16 canales en tiempo real (Wunk, 1985). Este sistema sustituye a otros intentos de efectuar el análisis de señales en tiempo real que resultaron ser demasiado costosos tanto en términos económicos, como en tiempo de desarrollo (Quint et.al.,1984).

Circuitos Básicos

La tercera opción para levar a cabo el procesamiento digital de señales en tiempos breves con microcomputadoras, es el utilizar elementos discretos para construir procesadores especializados. En este caso se trata de resolver el problema de la velocidad de ejecución de las funciones de procesamiento de señales, por medio del diseño de sistemas específicos que son especialmente eficientes para ciertas operaciones.

La función de multiplicación es un factor fundamental en la efectividad de los sistemas de análisis de señales. En casi todos los aspectos del procesamiento de señales se requiere de la ejecución de muchas multiplicaciones. Desgraciadamente, esta función no se puede implantar de manera eficiente dentro de computadoras tradicionales. De hecho, el desarrollo de algoritmos que reducen el número de multiplicaciones ha sido uno de los factores que impulsó significativamente al estudio de las señales en el dominio de la frecuencia. Adicionalmente a este avance en los algoritmos, se diseñaron sistemas especializados para efectuar el procesamiento de señales en menores tiempos. Un ejemplo de este enfoque, es el diseño y construcción de multiplicadores de arreglos discretos, donde se utiliza un multiplicador rápido que se multiplexa para efectuar todas las multiplicaciones que se requieren. Sin embargo, estos diseños discretos eran muy costosos, ya que utilizaban 164 sumadores ECL para construir un multiplicador de 17 x 17 bits con tiempo de multiplicación de 40 ns (Pezaris, 1971). No fué sino hasta la aparición de los multiplicadores monolíticos, que este enfoque se puede considerar como práctico (Finn, 1980). En la actualidad, el diseño de procesadores de señales es una opción atractiva para el procesamiento, ya que ofrecen un alto grado de flexibilidad. Un procesador de señales debe poder ejecutar distintos algoritmos y debe ser fácilmente adaptable. Esta estructura se ha llevado a cabo a nivel de utlilzar

elementos de "bit slice", tipo AM2900 y también utilizando multiplicadores en paralelo o multiplicadores/acumuladores, como los de la serie MPY de TRW. Un ejemplo de este tipo de diseño es el descrito por Steeger, donde se construyó un procesador de señales a partir de varios elementos AM2901 para formar una unidad de 16 bits que junto con un procesador de entrada /salida (basado en un 8085), un secuenciador 2909, también de bit-slice y un multiplicador MPY-16AJ, conforman un procesador de señales específico para neurología, donde se calcula la transformada de Fourier de 512 datos electroencefalográficos de 12 bits de resolución, en 11 ms, para determinar el estado de conciencia del paciente. (Steeger,1981; Kolb, 1980). Otro sistema similar es un procesador dedicado al análisis en frecuencia de mediciones ultrasónicas de velocidad del flujo sanguíneo (Pedersen, 1982), donde se calcula la transformada de Fourier de 256 datos de velocidad sanguínea en 4.5 ms y después se transfiere la información a una computadora de propósito general para almacenar y desplegar la información.

Otra opción posible dentro de este campo es la de utilizar circuitos periféricos especiales, dentro de microcomputadoras. La compañía American Microsystems Inc. (AMI) ofrece un circuito periférico para procesamiento de señales (S28211) y una versión preprogramada para transformada de Fourier, el S28214. Este circuito calcula la FFT y la FFT inversa por medio del algoritmo de decimación en frecuencia. Es capaz de ejecutar la FFT de 32 puntos complejos, utilizando coeficientes generados internamente, en 128 microsegundos. Se puede programar para hacer el cálculo de transformadas de arreglos mayores o para trabajar en paralelo con otros circuitos idénticos. Funcionalmente, estos circuitos son microcomputadoras para procesamiento específico, que tienen memoria de datos y memoria de instrucciones por separado, una unidad aritmética y lógica de 12 bits y un multiplicador paralelo (AMI, 1985). Desgraciadamente, aún cuando los tiempos de ejecución son

cortos, el sistema está diseñado para funcionar bajo el control de microprocesadores de 8/16 bits y la transferencia de información entre ambos procesadores hace que se reduzca efectivamente la velocidad del procesamiento. Esto, unido a la limitación de 12 bits, impide que este sistema sea considerado para muchas aplicaciones.

Hasta este momento, este tipo de arquitecturas han resultado ser útiles para el procesamiento rápido de señales dentro del campo de la medicina y la biología. Sin embargo, hasta ahora, el factor limitante ha sido el acoplamiento entre el procesador dedicado al análisis de señales y una computadora de propósito general, donde se almacenan los datos y donde se controla la operación del sistema completo.

Selección de la Arquitectura Básica del Sistema

El coprocesador, objeto de esta tesis es un producto de la última opción debido a las siguientes razones:

--El costo de los componentes principales que se han seleccionado es menor al de cualquier otra opción a corto plazo, si consideramos que casi todos los componentes se pueden adquirir dentro del mercado nacional.
--Se requiere de menor infraestructura para poder construir y probar al sistema.
--Las funciones que se deben llevar a cabo son muy específicas y no se necesita potencia computacional adicional de propósito general. Sin embargo, este diseño elimina el problema de la interfase con la computadora, ya que se utiliza un esquema similar al descrito por Wunk para el TMS32010, de tal manera que la comunicación entre el coprocesador y la microcomputadora es sencilla, a través del uso común de los bancos de memoria.

De las 3 tendencias expuestas anteriormente, es probablemente la opción de menor costo en términos monetarios. El diseño que se ha concluido permite el procesamiento de señales en tiempos significativamente menores a los posibles con el empleo de la microcomputadora por sí misma y es una mejor opción que el adquirir un sistema tipo 80386, ya que permite que se utilicen intensivamente los recursos de manejo de archivos y entrada y salida de datos que tienen las computadoras menos poderosas y sustituye adecuadamente las funciones de procesamiento de datos. Un sistema tipo 386 es muy costoso por su manejo intensivo de datos en precisión de 32 bits y en aritmética de punto flotante, características que no son de interés para el procesamiento de señales que no tienen un rango dinámico tan elevado.

A pesar de las consideraciones anteriores, la opción de utilizar un microprocesador especial para análisis de señales en conjunción con la microcomputadora debe estudiarse con cuidado, ya que ofrece la ventajas de tener varias o todas las funciones integradas en un solo encapsulado y un mayor soporte de programación y emulación. En este caso, se tendría un ahorro significativo en el tiempo de desarrollo de la "ferretería", pero con un incremento en los gastos de inversión, ya que se necesita de mayor infraestructura para poder desarrollar un sistema de este tipo.

CALCULO DE LA TRANSFORMADA RAPIDA DE FOURIER

CALCULO DE LA TRANSFORMADA RAPIDA DE FOURIER

Introducción

Una transformación es cualquier función que se aplique a un conjunto de datos para facilitar su análisis. El ejemplo más común de una transformación útil es el logaritmo de una función, que nos permite transformar las operaciones multiplicación y división, un tanto complejas, en simples sumas y restas. La transformada de Fourier es la representación de una señal por medio de la suma de funciones senoidales de diferentes frecuencias; es decir, es la representación en el dominio de la frecuencia de una señal. Aunque inicialmente esta transformación fue definida por Fourier para la solución de un problema de transferencia de calor, sus aplicaciones se han extendido a prácticamente todos los campos de la ciencia y la tecnología: sistemas lineales, óptica, física cuántica, procesos aleatorios, probabilidad, problemas de valor en la frontera, etc.

La transformada de Fourier es una transformación lineal de una función y se define como:

$$F(W) = \int_{+\infty}^{-\infty} F(t)\, e^{-jwt}\, dt$$

Para aquellas funciones en que no es posible obtener una expresión analítica de la transformada, se define la transformada discreta de Fourier (TDF), que representa muestras de la función F(w) y que es una función periódica de w, de la siguiente manera:

$$F(K)=\sum_{n=0}^{N-1} f(n)\, e^{-jkn/N} \qquad K=0,1,...n-1$$

donde f(n) son también muestras tomadas de la señal original f(t) a un cierto intervalo de muestreo. La naturaleza discreta de ambas señales les permite ser analizadas mediante una computadora. Sin embargo, el número de operaciones, y por lo tanto el tiempo necesario para calcular una transformada discreta de Fourier de la forma definida anteriormente, es muy grande (aproximadamente N^2 multiplicaciones complejas y N^2 sumas complejas, donde N es el número de datos); esto limitó durante muchos años el uso de la transformada de Fourier. En 1965 John W. Tukey y James W. Cooley publicaron un algoritmo eficiente para el cálculo de la transformada discreta de Fourier, que redujo el número de operaciones considerablemente (Cooley, Tukey, 1965). Posteriormente, se dieron a conocer los trabajos de diversos autores sobre algoritmos y técnicas similares para reducir el tiempo de cálculo de una transformación (Rudnick, 1966; Good, 1968). El conjunto de todas estas técnicas se conoce como transformada rápida de Fourier (TRF) y ha permitido extender las aplicaciones del análisis en el dominio de la frecuencia a varios tipos de señales.

El principio fundamental de reducción de estos algoritmos se basa en la periodicidad y simetría del factor e^{jwn}. Estas características pueden explotarse arreglando los coeficientes de la secuencia, ya sea en el tiempo o en la frecuencia, de tal manera de descomponerla en subsecuencias más pequeñas; es decir, en el cálculo de transformadas discretas de menor longitud. La manera en

que se efectúa esta reducción conduce a diferentes algoritmos, que disminuyen el tiempo de cálculo mas o menos en la misma proporción. Los algoritmos principales pueden dividirse en dos grupos:

- Decimación en tiempo. Consiste en arreglar la secuencia de entrada x(n) en secuencias más pequeñas, considerando todos los puntos de ésta.

- Decimación en frecuencia. Consiste en arreglar la secuencia de los coeficientes de la transformada discreta de Fourier X(k) en secuencias de longitud menor.

Ambos métodos conducen a resultados similares en cuanto al número de operaciones (aproximadamente N log N). Aunque estos algoritmos pueden aplicarse a cualquier número de datos, su eficiencia aumenta si se consideran secuencias cuya longitud sea una potencia de 2.

Para el desarrollo de este paquete, se empleo el método de decimación en tiempo, en el que se entrega la secuencia de entrada, que ha sido previamente ordenada de acuerdo al algoritmo, y se obtiene la secuencia ordenada de los coeficientes de la transformada discreta. En la siguiente secciòn se describe el algoritmo de decimación en tiempo, tal como aparece en la literatura (Oppenheim, Schafer 1975).

Descripción del método de decimación en tiempo.

La disminución del número de operaciones en este método se logra partiendo la longitud de la secuencia inicial en secuencias más pequeñas, agrupando los términos en función de las características de periodicidad y simetría de la exponencial compleja:

$$e^{-j\,(2\pi/N)\,k\,(N-n)} = e^{j\,(2\pi/N)\,kn}$$

$$e^{-j\,(2\pi/N)\,kn} = e^{-j\,(2\pi/N)\,k\,(n+N)} = e^{-j\,(2\pi/N)\,(k+N)\,n}$$

Considese el caso particular cuando N es una potencia entera de la base 2, es decir:

$$N = 2^{v}, \quad \text{donde } v \text{ es un número entero.}$$

La transformada discreta de Fourier (X(k)) de la secuencia x(n) se define por la siguiente relación:

$$X(k)=\frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{-j(2\pi/N)nk} \qquad k=0,1,\ldots N-1$$

La relación inversa, es decir, la antitransformada de la función X(k) se define de la siguiente manera:

$$X(k)=\frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j(2\pi/N)nk} \qquad n=0,1,\ldots N-1$$

y nos permite recuperar la secuencia x(n) a partir de su transformada. Ambas secuencias x(n) y X(k) son periódicas, con período N, por las características del par de transformación discreta de Fourier (Brigham, 1979).

El cálculo de las ecuaciones anteriores es idéntico, exceptuando el signo de la exponencial y el factor 1/N. Por esto, basta con analizar el caso de la transformada directa. Partiendo la secuencia de entrada x(n) en dos secuencias, construidas con los puntos pares e impares de x(n), tenemos:

$$X(k)= \sum_{n\ par} X(n)\ e^{-j(2\pi/N)nk} + \sum_{n\ impar} X(n)\ e^{-j(2\pi/N)nk}$$

Sustituyendo las variables n = 2r para n par y n = 2r+1 para n impar, se tiene:

$$X(k)= \sum_{r=0}^{N/2-1} x(2r)\ e^{-j(2\pi/N)2rk} + \sum_{r=0}^{N/2-1} x(2r+1)\ e^{-j(2\pi/N)(2r+1)k}$$

$$= \underbrace{\sum_{r=0}^{N/2-1} x(2r)\ e^{-j(4\pi/N)2rk}}_{G(k)} + e^{-j(2\pi/N)k} \underbrace{\sum_{r=0}^{N/2-1} x(2r+1)\ e^{-j(4\pi/N)(2r+1)k}}_{H(k)}$$

G(k) y H(k) son dos transformaciones de N/2 puntos cada una, correspondientes a las secuencias de puntos pares e impares, respectivamente, de x(n). Nótese que cada una de ellas requiere de (N/2) multiplicaciones y sumas, mas las N operaciones correspondientes a la combinación de ambas transformadas; es decir, un total de N + (N /2) multiplicaciones y sumas complejas. Esta cantidad es menor que el número de operaciones iniciales N^2 ,para cualquier N mayor que 2.

Llamemos ahora g(r) a la secuencia formada por los datos pares de x(n) y h(r) a la formada por las muestras impares. El mismo procedimiento se sigue para obtener las transformadas G(k) y H(k), de tal forma que se tiene:

$$G(k) = \sum_{r=0}^{N/2-1} g(r)\; e^{-j(4\pi/N)2rk} \qquad k=0,1,\ldots N/2-1$$

$$= \sum_{l=0}^{N/4-1} g(2l) e^{-j(8\pi/N)lk} + e^{-j(4\pi/N)k} \sum_{l=0}^{N/4-1} x(2l+1)\; e^{-j(8\pi/N)(2r+1)lk}$$

Las divisiones se continuan hasta que llegamos a tener la transformada más pequeña, de dos puntos, definida así:

$$M(k) = m(0) + e^{-j\ k}\ m(1) \qquad k = 0,1$$

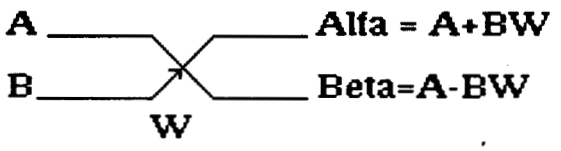
o sea:

$$M(0) = m(0) + m(1)$$

$$M(1) = m(0) - m(1)$$

Cada etapa de división se denomina decimación y el número total de decimaciones que se realiza para dividir toda la secuencia corresponde a la potencia (v) a la que se eleva la base (2) para obtener el número de datos (N).

Gráficamente, este proceso se representa definiendo una operación básica de dos números, conocida como "mariposa", de la siguiente manera:



donde A y B son datos complejos y W es un coeficiente.
La figura 1 muestra la representación total del cálculo de una transformada discreta de Fourier de 8 puntos, siguiendo el algoritmo descrito. Es importante notar el uso de la propiedad de periodicidad de la exponencial:

$$e^{-j(2\pi/N)(k+N/2)} = -e^{-j(2\pi/N)k}$$

que nos permite ahorrar varias multiplicaciones complejas en cada decimación. En la gráfica podemos observar que se requiere un número total de N sumas complejas y N multiplicaciones complejas por cada decimación, lo que significa una cantidad final de 2Nlog N de operaciones para el cálculo de todos los puntos de la transformada.

Nótese que en cada proceso de decimación, la secuencia de entrada se arregla en sus elementos pares e impares. Esto ocasiona que al finalizar las decimaciones los valores de x(n) aparezcan desordenados en la entrada, mientras que los coeficientes de la transformada discreta se obtienen en el orden adecuado. El orden de la secuencia de entrada se puede determinar mediante el algoritmo de inversión de bits (Rabiner,1975). Supongamos que tenemos una secuencia de entrada x(n), que debe ser ordenada para iniciar el algoritmo de decimación en tiempo. Representando el indice de cada dato en codigo binario, para el ejemplo N=8 necesitamos tres bits. Si se invierte el orden de los bits, tendremos la secuencia "ordenada" para el algoritmo de decimación en tiempo:

Secuencia inicial		Secuencia ordenada	
dec	bin	bin	dec
x(0)	x(000)	x(000)	x(0)
x(1)	x(001)	x(100)	x(4)
x(2)	x(010)	x(010)	x(2)
x(3)	x(011)	x(110)	x(6)
x(4)	x(100)	x(001)	x(1)
x(5)	x(101)	x(101)	x(5)
x(6)	x(110)	x(011)	x(3)
x(7)	x(111)	x(111)	x(7)

El método de decimación en frecuencia es muy similar al descrito y presenta las mismas ventajas en cuanto a la reducción del número de operaciones, por lo que resulta prácticamente lo mismo el empleo de cualquiera de los dos algoritmos.

.

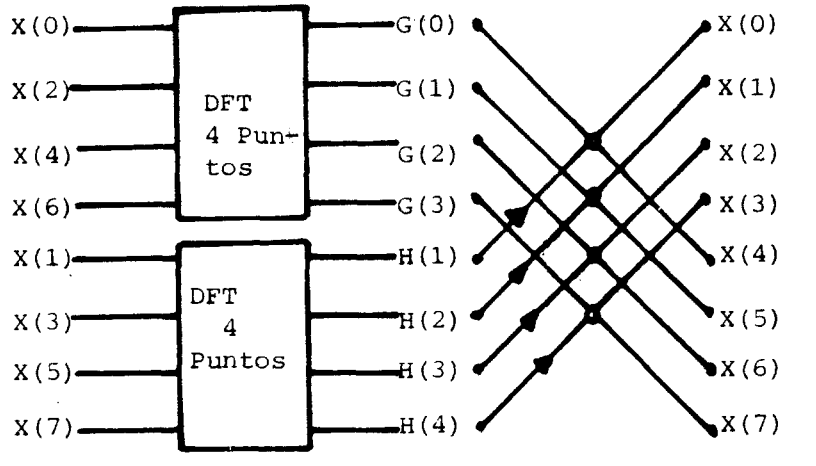


Figura 1. Cálculo de la FFT para 8 puntos por el método de decimación en el tiempo.

DISEÑO DE SISTEMAS DE PROCESAMIENTO DE SEÑALES POR CIRCUITERIA

DISEÑO DE SISTEMAS DE PROCESAMIENTO DE SEÑALES POR CIRCUITERIA

Procedimientos

Para poder llevar a cabo el diseño y la construcción de un sistema para el procesamiento digital de señales por medio de circuitos, se deben seguir 4 pasos (Rabiner, 1975):

1. Hacer la selección de la tecnología de los circuitos lógicos que se utilizarán.
2. Escoger una metodología de construcción.
3. Desarrollar una estructura del sistema.
4. Simular todo o parte del sistema para poder determinar parámetros.

La selección de una familia lógica rara vez proporciona una sola opción, ya que el diseño se puede hacer de distintas maneras, con distintas ventajas para cada uno. Cada familia lógica tiene sus características y ventajas específicas, junto con propiedades que a veces hacen que sean incompatibles unas con otras, aunque a veces esto se puede solucionar con circuitos especiales de interfase o con un diesño cuidadoso. Muchas veces es útil la mezcla de estas distintas tecnologías; por ejemplo, se podrían tener requerimientos moderados de velocidad para un subsistema, mientras que otra parte necesita circuitos de alta velocidad.

Después de que se ha seleccionado el tipo de circuitos que se utilizará, se puede comenzar una metodología de construcción, ya que se conocen las necesidades de velocidad y consumo de potencia. Dentro de esta metodología se toman en cuenta las consideraciones de tamaño físico del sistema, el tipo de construcción, la distribución de alimentación y tierras, así como el tipo de alambrado.

Una vez que se ha hecho la selección del tipo de tecnología que se utilizará y se ha propuesto una metodología de construcción, se puede proceder al desarrollo específico de la arquitectura en cuestión. En muchos sistemas, las consideraciones de los sistemas aritméticos que se usarán, la longitud de los registros las configuraciones de memoria afectarán de manera importante a la arquitectura, al funcionamiento y al costo del sistema. A veces es necesario simular algunos de estos parámetros para predecir como se comportarán cuando estén construídos.

Circuitos Multiplicadores

Desde el punto de vista del diseño lógico, los multiplicadores integrados, se pueden dividir en 2 tipos, los multiplicadores secuenciales y los multiplicadores paralelos. En ambos diseños se obtiene el resultado a partir de una serie de sumas, pero en el multiplicador paralelo, se puede obtener el resultado final sin tener que pasar por resultados intermedios. Los circuitos más rápidos son los "array multipliers" o multiplicadores en arreglo, que consisten en un arreglo bidimensional de sumadores de un bit. Son redes de lógica sin memoria que producen el resultado después de que ha pasado el tiempo de asentamiento de las compuertas. Algunos ejemplos de estos son los array multipliers descritos por Noll, que tienen una velocidad de 330 MHz (Noll, 1986) y las macroceldas que se pueden utilizar dentro de circuitos integrados a base de arreglos de compuertas (gate array), que funcionan a 25 MHz (Henlin, 1985).

El efecto de la cuantización en la FFT

Al construir de un sistema de procesamiento de señales por circuitería, la cuantización de las señales se convierte en una consideración de diseño importante. Es necesario tomar en cuenta los efectos de representar a los datos y a los coeficientes con una precisión finita. Algunos de estos son

el ruido en el redondeo al truncar el resultado de las multiplicaciones, el escalamiento de los datos para evitar el sobreflujo en una multiplicación con precisión fija y las transfornaciones inexactas al representar a los coeficientes de W^k con una precisión finita. Se han hecho varios estudios sobre el ruido de redondeo y el escalamiento de los datos para algoritmos de transformación en radix 2, tanto para decimación en tiempo, como para decimación en frecuencia (Welch, 1969; Oppenheim, 1972).

Si tenemos que calcular la etapa m de una FFT, tenemos que $f_m(i)$ y $f_m(j)$ son las entradas a la mariposa junto con un multiplicador W^p, mientras que las salidas de la mariposa son $f_{m+1}(i)$ y $f_{m+1}(j)$, entonces

$$f_{m+1}(i) = f_m(i) + W^p f_m(j)$$
$$f_{m+1}(j) = f_m(i) - W^p f_m(j)$$

y se puede ver que la magnitud de los números por lo general aumenta de una etapa de decimación a la siguiente, de tal manera que es necesario hacer escalamientos de los datos por medio de corrimientos a la derecha. Welch ha demostrado que el nivel de la señal aumenta con una razón menor a un bit por etapa de decimación. Más especíicamente, la relación del ruido RMS de salida al valor RMS de la señal también de salida es:

$$\frac{\text{RMS(error)}}{\text{RMS(señal)}} \approx \frac{\sqrt{N}\, 2^{-B} (0.3)\sqrt{8}}{\text{RMS(entrada)}}$$

donde B es el número de bits de resolución del sistema y N es el número de datos (Nussbaumer, 1982).

Para resolver el problema del truncamiento, se pueden seguir 3 posibilidades:

1. Hacer un corrimiento a la derecha después de cada iteración para que así no existan sobreflujos.

2. Controlar la secuencia, de tal manera que el valor sea menor al límite ABS ($f_0(i)$) <1/2 para toda i y si cualquier valor es mayor a 1/2, hacer un corrimiento a la derecha sobre todo el arreglo.

3. Probar si hay sobreflujo. En este caso se escalan los datos para que sean menores a uno los valores absolutos de las partes real e imaginaria de las secuencias y si se detecta un sobreflujo, se hace un corrimiento a la derecha sobre todo el arreglo, incluyendo las etapas donde ya se hicieron las mariposas. En este caso puede ocurrir más de un sobreflujo por etapa, pero no más de 2.

La técnica 1 es sencilla pero conduce a un escalamiento excesivo y a una pérdida de la exactitud. La técnica 2 es lenta, ya que es necesario estar calculando las magnitudes constantemente, así que se sugiere usar la tercera opción aunque en este caso, se tiene que recalcular la mariposa donde ocurrió un sobreflujo.

ARQUITECTURA DEL SISTEMA

ARQUITECTURA DEL SISTEMA

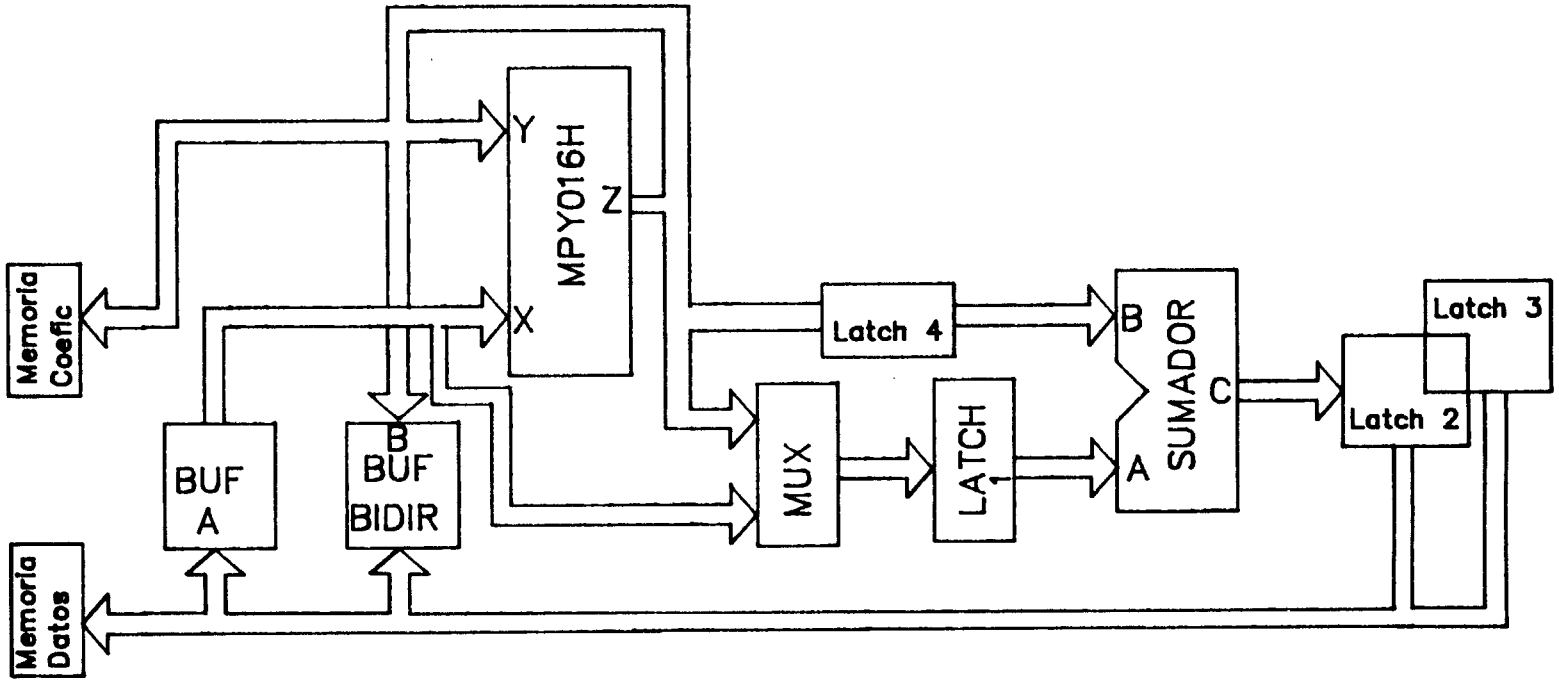
El coprocesador está diseñado para ejecutar la operación de la mariposa de manera eficiente. La distribución y la selección de los componentes reflejan fundamentalmente este criterio. Adicionalmente, puede llevar a cabo las funciones de multiplicación compleja, magnitud cuadrada y convolución sin necesidad de adicionar otros circuitos (Figura 2).

El elemento principal de este sistema es el multiplicador MPY016HJ, que efectúa una multiplicación de 16 x 16 bits en 100 ns. Alrededor de este componente se han agregado una unidad aritmética y lógica y algunos registros de almacenamiento temporal (Figura 3). Todos los elelmentos del coprocesador funcionan con ciclos de 100 ns (reloj de 10MHz).

Los datos que alimentan al sistema provienen de 2 arreglos en memoria RAM estática con tiempo de acceso de 100 ns y que se encuentran dentro de la tarjeta del coprocesador. Estos arreglos se han denominado memoria de datos y memoria de coeficientes,de acuerdo a su función original, aunque pueden utilizarse para otras operaciones.

El sistema cuenta con un canal (BUS) interno de datos, que conecta directamente a la memoria de datos con un buffer unidireccional, el cual se conecta a la entrada X del multiplicador y al multiplexor de datos de entrada al sumador. Este bus tambien conecta a la memoria de datos con los latches o registros de salida del sistema.

La memoria de datos también se conecta al registro Y y a la salida del multiplicador, a la memoria de coeficientes, al multiplexor y a la unidad aritmética y lógica. Ésta recibe su otra entrada del latch1 (conectado al multiplexor) y su salida está unida a el latch2.



COPROCESADOR. Diagrama a Bloques

Figura 2. Diagrama a Bloques del Sistema.

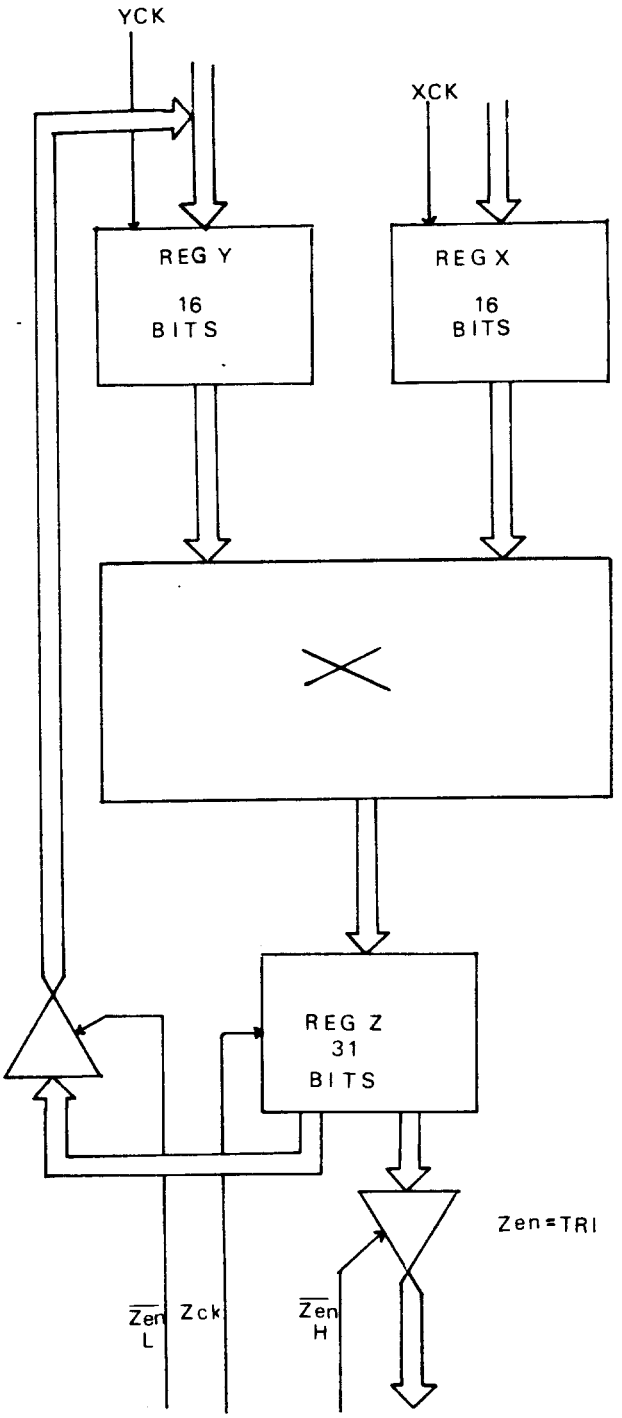


Figura 3. Arquitectura interna del multiplicador MPY016.

La distribución de componentes que se acaba de describir, permite que se lleven a cabo todas las funciones deseadas del procesamiento de señales con facilidad. La secuencia de microinstreucciones resulta compacta y sin problemas de temporización.

Circuitos de interfase de entrada y salida.

La figura 4 muestra los circuitos de interconexión entre el sistema del coprocesador y la microcomputadora huésped. Como en este caso la conexión se puede hacer entre el coprocesador y una máquina tipo IBM-PC o una IBM-PC/AT, es necesario incluir un sistema de "buffers" seleccionables para acoplar el sistema al bus adecuado, y presentar solo una carga tipo TTL al bus.

Buffers unidireccionales.

Los circuitos U2, U3 y U7 son buffers unidireccionales que se encuentran siempre habilitados (74LS244). El circuito U2 proporciona las líneas de direcciones altas (A8-A15) al sistema. El circuito U3 proporciona las líneas de direcciones A0-A7. El circuito U7 proporciona las líneas más bajas de direcciones (A0 y A1), junto con las líneas -IOR e -IOW (lectura y escritura a puertos), las líneas -SMEMR y -SMEMW (lectura y escritura a memoria baja); la línea AEN, la cual indica cuando se está utilizando el bus por el sistema de acceso directo a memoria y las líneas de 0 WAIT STATE y I/OCS16, las cuales indican que no se requieren estados de espera adicionales para la transferencia de datos y que la transferencia de datos a través de puertos será de 16 bits.

Circuitos de decodificación.

El circuito U6 es un comparador de magnitud (74ALS520) y se utiliza para hacer la decodificacion de las direcciones mas altas de la memoria de expansión dentro de la tarjeta. Dentro del sistema PC de IBM, existen algunas localidades

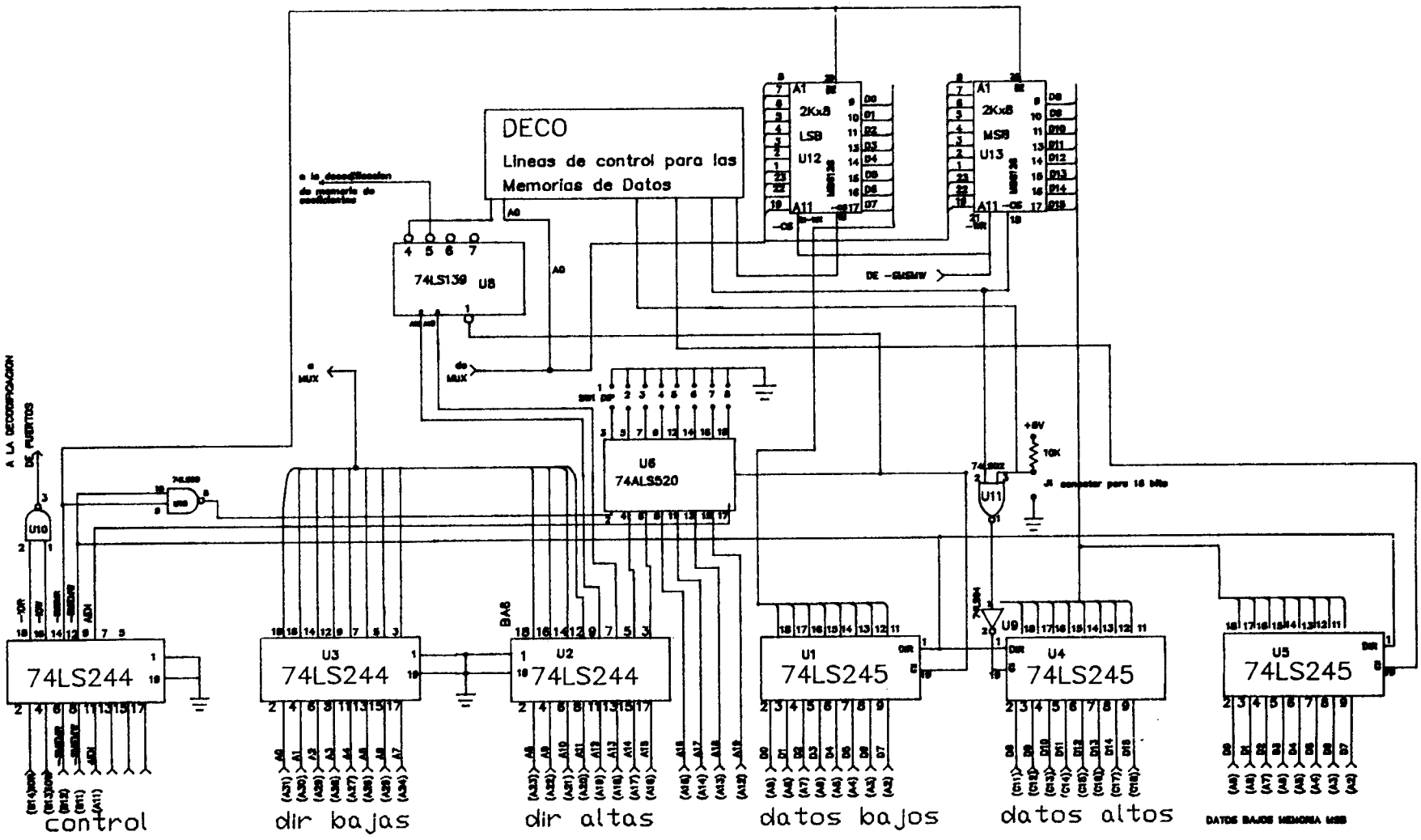


Figura 4. Interfase entre el coprocesador y la computadora.

accesibles al usuario, como lo son las localidades a partir de la dirección C0000h, que inicialmente estaba destinada a la ROM de expansión. En nuestro caso, utilizamos estas localidades para la RAM de 100ns de tiempo de acceso. El circuito U6 se conecta a una serie de interruptores tipo "DIP" que permiten cambiar la direccion de esta memoria si es necesario, a las líneas de dirección A14 y A15 de la salida del circuito U2 y a las líneas de direcciones A16 a A19 que toma directamente del bus. Este circuito tiene una salida que al activarse indica que se ha seleccionado el banco de direcciones correcto. Se utiliza junto con un decodificador adicional para accesar a la memoria de expansión y para activar a los buffers de datos.

El circuito U8 completa el subsistema de decodificación. Es un decodificador 74LS139. Se habilita a partir de la línea de salida del 74ALS520 y selecciona varios bancos de memoria tales como la memoria de datos y la memoria de coeficientes a través de la decodificación de las líneas A12 y A13. En este sistema se cuenta con 2 bancos de memoria principales: los de memoria de datos, que comienzan en la dirección C0000h, y los de coeficientes, que comienzan en la dirección D0000h. Ambos bancos de memoria son de 2K x 16 bits, que internanente se organizarán como 1K x 16 datos complejos con partes real e imaginaria.

Buffers bidireccionales.

La última parte de los circuitos de interfase de entrada y salida está compuesta por los buffers bidireccionales. En esta unidad se tienen 3 circuitos tipo 74LS245. Dos de ellos (U1 y U4) están conectados a las líneas de datos D0-D7 y D8-D15 respectivamente y se activan de manera simultánea en el caso de que la transferencia sea de 16 bits (bus de la IBM-PC/AT). El otro circuito (U5) se activa cuando la transferencia es de 16 bits pero a través del bus de la IBM-PC convencional de 8 bits y cuando se está transfiriendo el

byte más significativo. En todos los casos la dirección de la transferencia se determina por la activación de la línea -IOR o de la línea -SMEMR.

Circuitos de memoria.

La unidad de memoria está construída a partir de circuitos MB8128, que son memorias estáticas de 2Kbytes cada una, con un tiempo de acceso de 100ns. Dos de estas forman el banco de datos, mientras que otras 2 forman la memoria de coeficientes. Desde el punto de vista de operación de la computadora, su funcionamiento es idéntico, y solo varían sus direcciones. Los circuitos U12 y U13 forman la memoria de datos, donde U12 es el byte menos significativo; los circuitos U19 y U20 forman la memoria de coeficientes. U20 es el byte menos significativo.

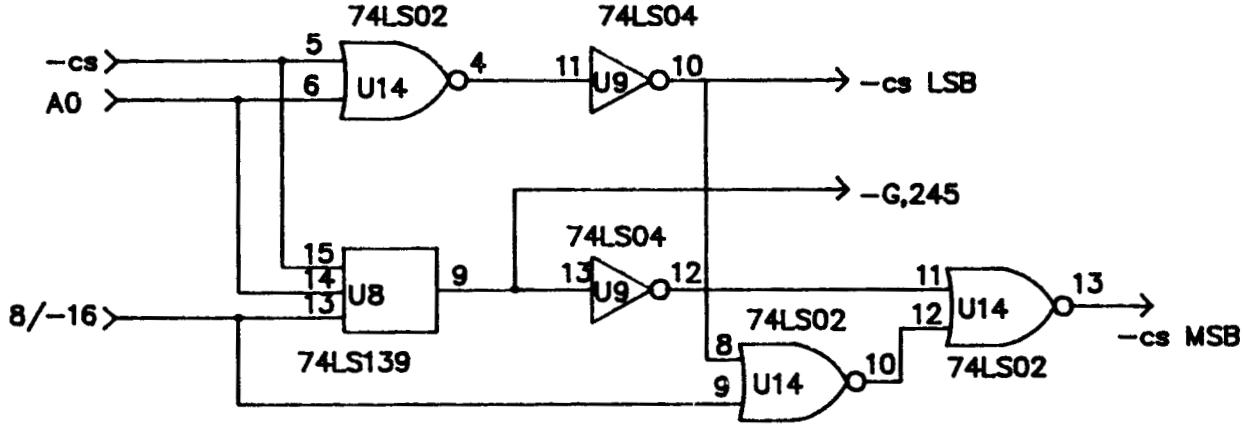
Como la configuración de las memorias es de 2K x 8 bits, es necesario utilizar varios circuitos lógicos para direccionar adecuadamente a las memorias como a una unidad. Los circuitos U9, U10, U11 y U14 se utilizan para proporcionar la decodificación adecuada a las memorias de datos. Toman como entradas a las líneas de selección de dirección provenientes del decodificador 74LS139 (U8) y a la línea de dirección A0, junto con la línea de selección de 8/16 bits de transferencia al bus. A la salida de estos circuitos, se tienen las líneas de habilitación de los buffers bidireccionales (línea -g de los circuitos 74LS245) y las líneas de -CS (selección) de la parte alta y baja de la memoria (bytes menos y más significativos).

De manera similar, se utilizan compuertas de U9 y U14 para proporcionar las líneas de decodificación a la memoria de coeficientes. En este caso se aprovecha una sección de decodificación no utilizada en U8 para minimizar la cantidad de componentes utilizados.

La figura 6 muestra estos circuitos de decodificación. Adicionalmente a estos, están los circuitos de selección de memoria provenientes de la unidad de control. Sin embargo, estos últimos son mucho más sencillos, ya que internamente, la memoria se comporta como de 16 bits y se manejan los 2 circuitos integrados de la memoria de manera simultánea. Las líneas de CS y las líneas de lectura y escritura se conectan a través de una compuerta lógica "O" para que la memoria sea accesada tanto por la computadora, como por la unidad de control.

De manera similar, el direccionamiento de la memoria, puede estar controlado por la computadora o por la unidad de control. Para permitir esto, se utilizan unos circuitos conmutadores de las líneas de dirección, o "multiplexores". La figura 7 muestra de manera general como se obtiene el direccionamiento tanto para la memoria de datos como para la memoria de coeficientes. Se puede observar que las líneas de direcciones provienen de dos fuentes distintas, pero que los datos están conectados al canal común interno del coprocesador.

La figura 8 es un diagrama más específico de las conexiones a través de los multiplexores. Las direcciones provenientes de la computadora entran a las líneas I0a-d de los multiplexores 74LS257, provenientes de los buffers unidireccionales 74LS244 que se conectan directamente al bus de expansión de la computadora y que están siempre hablitados. Las direcciones de la unidad de control entran a las líneas I1a-d de los multiplexores. La selección de las direciones que entrarán a las memorias se hace a través de la línea de control S (pata 1) de los 74LS257. Como se utilizan 11 líneas de direcciones (desde A1 hasta A11, ya que A0 se usa solo para el caso del acceso desde la computadora), se utilizan 3 circuitos multiplexores



DECO1 lineas de control para memoria de coeficientes

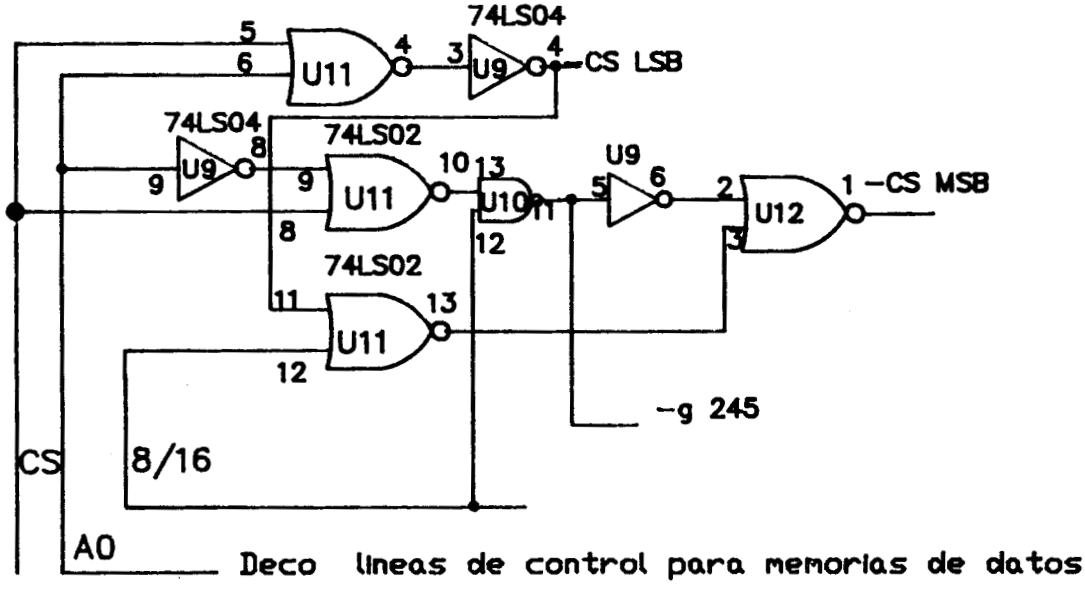


Figura 6. Circuitos de decodificación de las memorias de datos y de coeficientes.

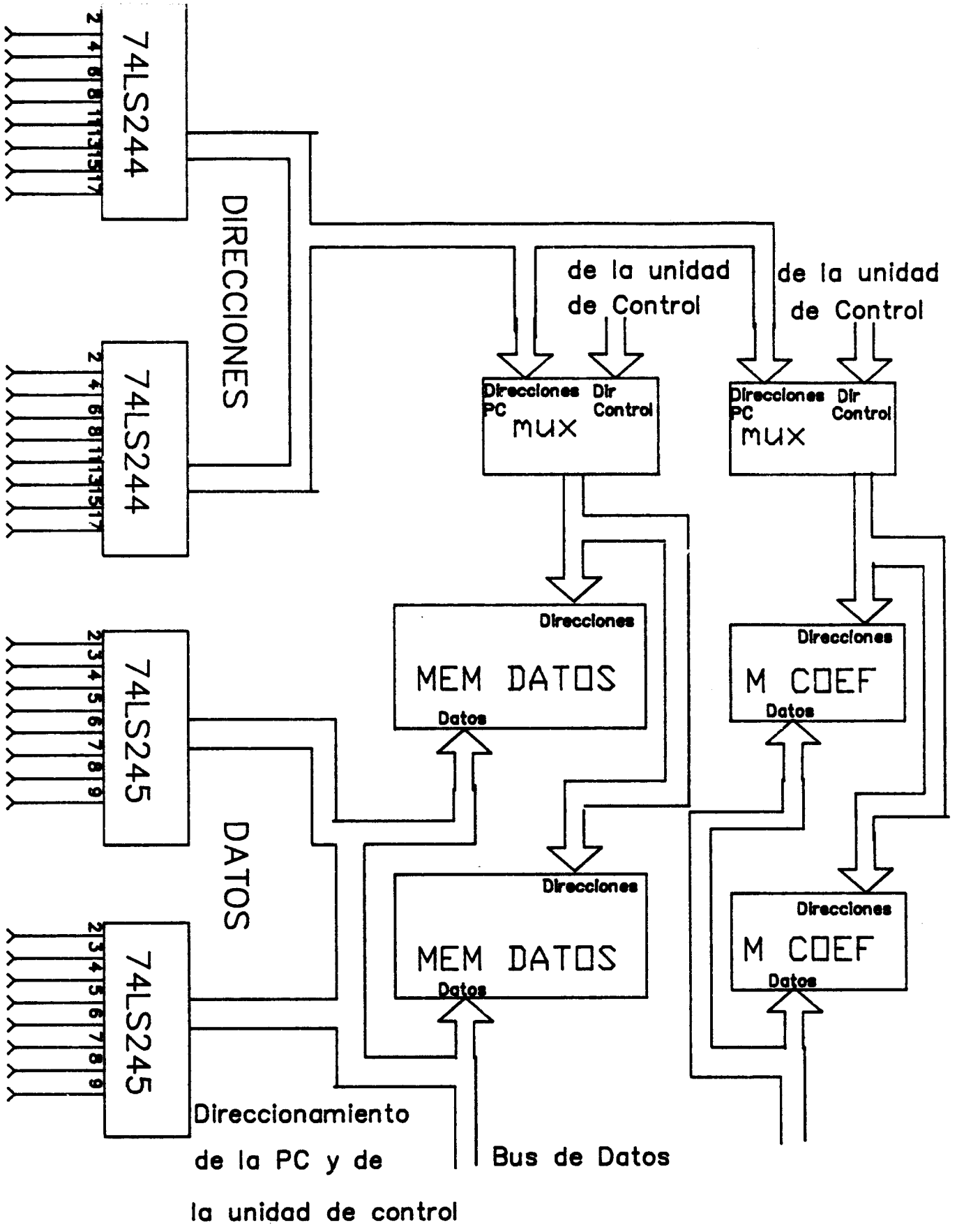


Figura 7. Manejo de la memoria de datos y la memoria de coeficientes a través de los multiplexores.

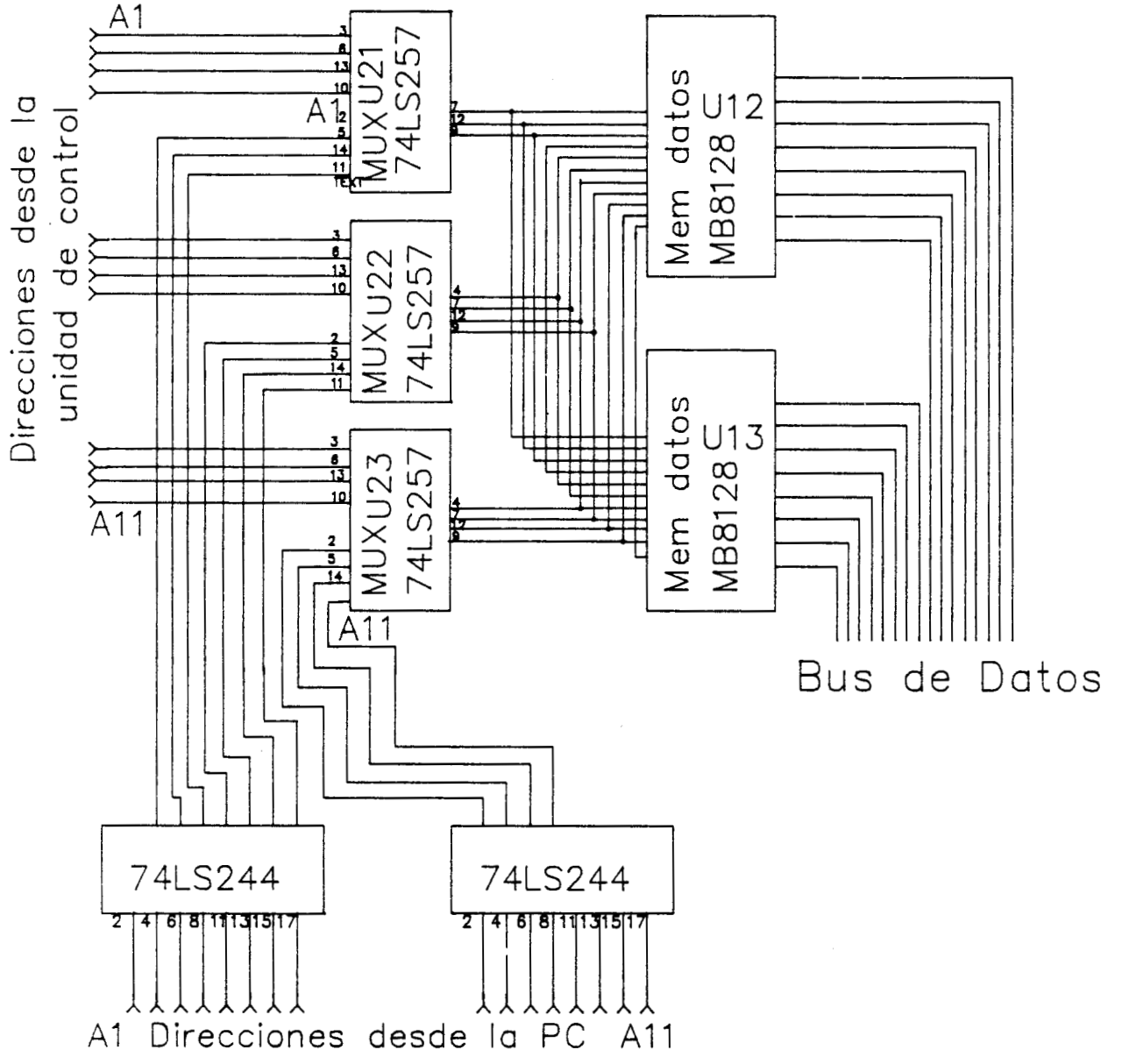


Figura 8. Selección del direccionamiento: Unidad de control o memoria de la computadora.

(U21,U22,U23). El circuito U21 maneja las líneas A1-A3 y deja libre una línea de ambas entradas. Los circuitos U22 y U23 manejan las líneas A4 - A7 y A8 - A11 respectivamente.

Multiplicador y circuitos asociados.

La figura 9 muestra la configuración interna del multiplicador.

Las líneas de control del MPY016HJ son las siguientes:

CLK X - Reloj del registro X_{in}
CLK Y - Reloj del registro Y_{in}
CLK L - Reloj del registro LSP
CLK M - Reloj del registro MSP
TRI L - Control de 3er estado LSP
TRI M - Control de 3er estado MSP
RS - Corrimiento a la derecha de MSP 1 bit; quita el bit de signo a LSP
FT - "Feedthrough", hace transparentes a los registros de salida
TCX - Indica el formato de las palabras de entrada
TCY - Formato de palabras de entrada: complemento a 2 (1 lógico) o magnitud (0 lógico)
RND - Suma 1 al bit más significativo de LSP independientemente de la posición de corrimiento

Se tienen 2 registros o latches de entada (registros X y Y), los cuales se controlan por las líneas XCk y YCk. A la salida se tiene un doble registro Z de 32 bits. Cada sección de este registro se controla separadamente. Adicionalmente, la salida menos signficativa del producto está conectada internamente a la entrada Y. Las líneas TCX y TCY se utilizan para señalar el tipo de aritmética que se utiliza: complemento a 2 (TCX o TCY en alto) o magnitud. Como la multiplicación se puede hacer en un formato mixto, TCX puede estar en un estado mientras TCY está en otro. Los valores lógicos de estas señales se leen junto con los datos

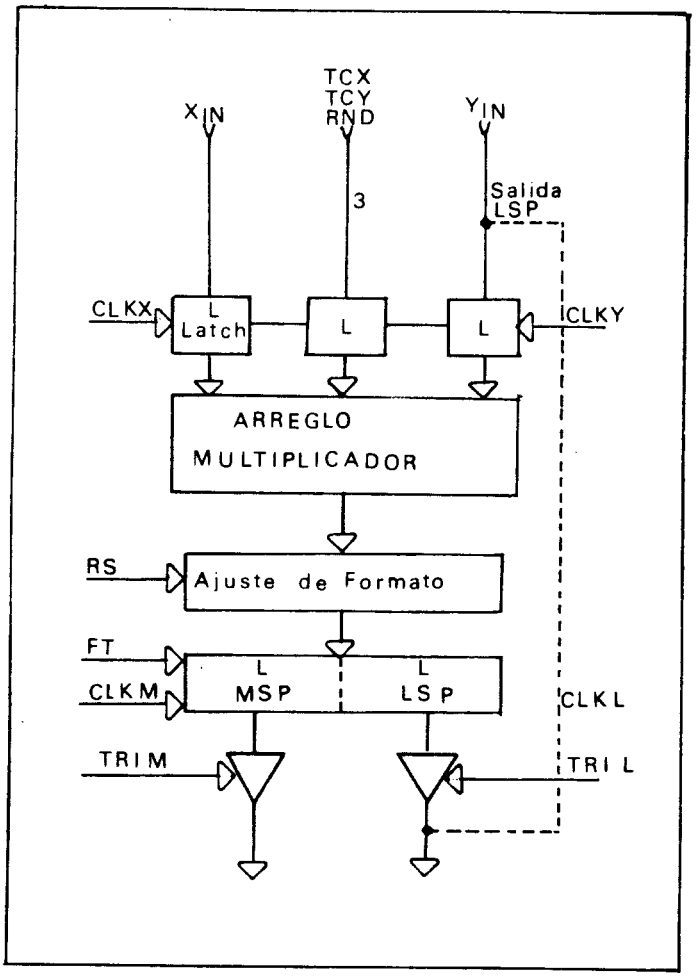


Figura 9. Configuración interna del multiplicador.

al activarse las líneas de XCk y YCk. La línea RND también se lee en el momento de la activación de las líneas XCk o YCk y se utiliza para llevar a cabo la función de redondeo de los datos.

Los datos que entran a los latches de entrada, se comunican a un multiplicador asíncrono, de donde sale el producto. La línea RS permite que se haga un corrimiento a la derecha, de tal manera que el bit de signo de la parte menos significativa del producto se elimina (está duplicado en la prte más significativa) y se hace un corrimiento del bit menos significativo de la parte alta del producto a esta localidad.
La salida de esta sección de ajuste de formato se conecta a los latches de salida. Se cuenta con 2 latches independientes que manejan a la parte alta y la parte baja del producto. Estos latches pueden manejarse tanto por flanco (al utilizar las líneas ClkL y ClkM), como por nivel (por medio de la línea FT) .
Las líneas TRIM y TRIL manejan las salidas de 3er estado de las partes altas y bajas del producto respectivamente.

Para el sistema que se presenta, solo se necesitan utilizar las líneas de adquisición de los datos de entrada, CkX y CkY, así como la línea de salida del producto menos significativo CkL. Las demás líneas de control se mantendrán en estados constantes, por ejemplo, se utiliza el formato de aritmética en complemento a 2, de tal manera que TCX y TCY están en un nivel alto.

La figura 10 muestra las líneas de conexiones específicas entre el multiplicador y los buffers. La entrada X del multiplicador se conecta a los buffers unidireccionales U15 y U17 (74LS244), los cuales permiten el paso de los datos provenientes de la memoria. La entrada Y se conecta a los buffers bidireccionales U16 y U18 (74LS245), de tal manera

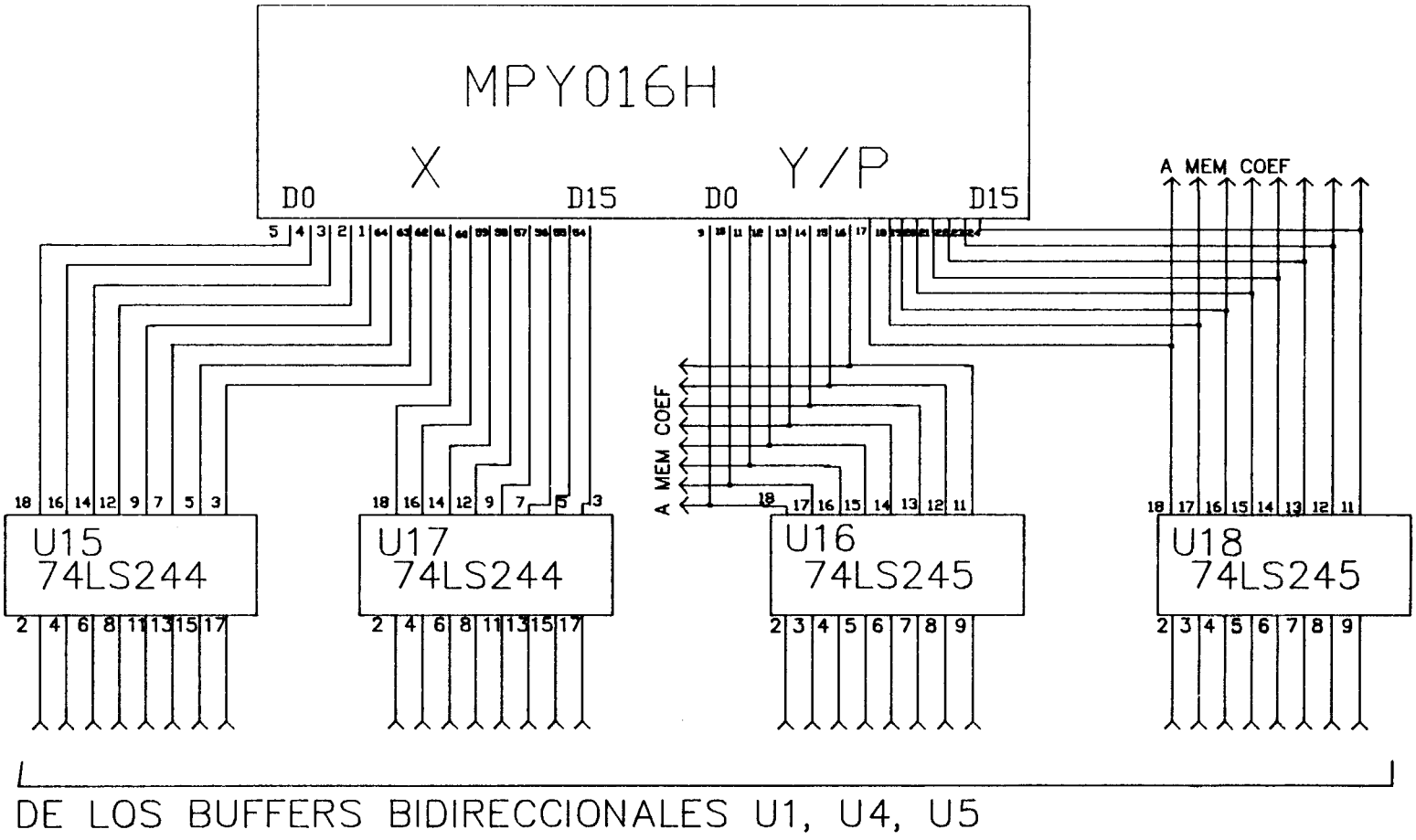


Figura 10. Conexiones entre los buffers y el multiplicador.

que se puede cargar un dato de la memoria si se desea. Estos circuitos permiten el paso de los datos provenientes de la computadora, a la memoria de coeficientes. Esta misma entrada Y del multiplicador, es la salida de la parte menos significativa del producto, de tal manera que estas líneas van también a la entrada de los multiplexores de datos, 74LS257 (U26a,U28,U29 y U30).

Circuitos multiplexores, latches y sumador.

Adicionalmente a los datos de salida del multipicador, los datos provenientes de los buffers unidireccionales U17 y U17 entran a los circuitos multiplexores 74LS257. La figura 11 muestra esta conexión, así como la salida de estos circuitos y su entrada tanto al LATCH1 (U31 y U32), como a la entrada B del sumador compuesto por los circuitos 74LS181 (U33,U34,U39 y U40). En este diagrama se muestra como se tienen los datos provenientes de la salida del multiplicador (MPYZ) conectados a las entradas A de los circuitos sumadores. La salida C de estos circuitos se conecta a los registros denominados LATCH2 y LATCH3 de manera paralela, mientras que la salida de estos regresa al canal interno de datos.

El sumador, compuesto por 4 elementos 74LS181 (unidades aritméticas y lógicas) tiene como funciones únicas la suma y la resta de los datos que entran. Esto facilita el manejo de las líneas de control, ya que con una sola línea se puede manejar la operación de este elemento (sumar/restar). Los registros de salida, LATCH2 y LATCH3 se utilizan principalmente para almacenar temporalmente los resultados intermedios que se producen en en cálculo de la multiplicación compleja.

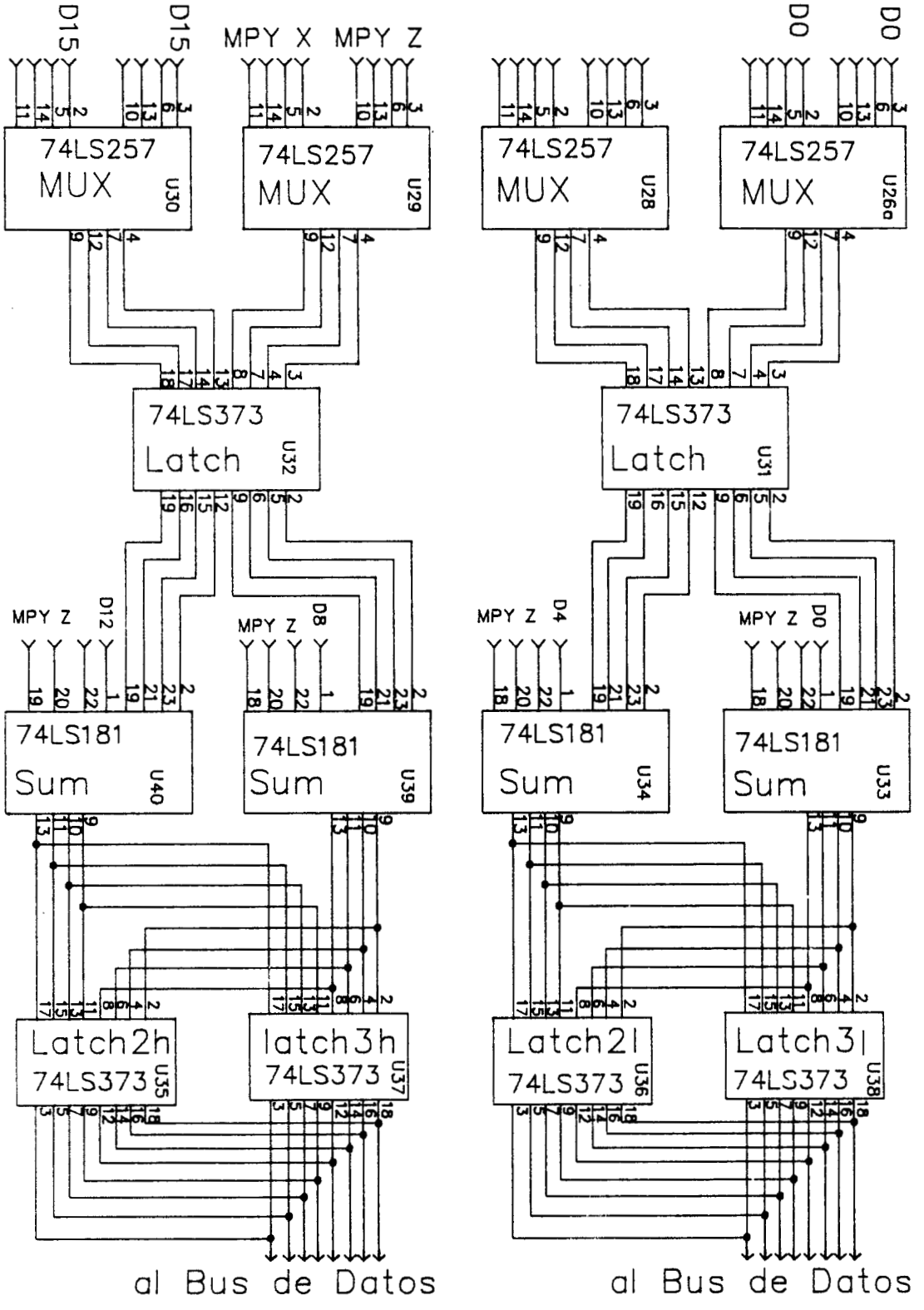


Figura 11. Conexiones entre las salidas del multiplicador, sumadores y los registros temporales.

Unidad de control.

Las operaciones del coprocesador se ejecutan por medio de la activación y desactivación coordinada de las líneas de control individuales de cada dispositivo que forma parte del sistema. Cada operación está compuesta de varias micro-instrucciones, que son los ciclos fundamentales del funcionamiento de la unidad y que consisten en el manejo de los estados de las líneas de control para que se lleve a cabo un solo evento dentro de la secuencia que conduce a la ejecución de la operación.

Las líneas de control se manejan a través de las salidas de un conjunto de memorias programables (PROMS). El contenido de cada localidad de las memorias es el estado de las líneas de control para cada microinstrucción. Para ejecutar una operación, la unidad de control lleva a cabo varios accesos a la memoria PROM, de tal manera que cada acceso a la memoria produce una salida en ésta y como resultado, se ejecuta la serie de microinstrucciones que forma una operación. La figura 12 muestra la estructura general de la unidad de control. Aquí se puede observar que el control maestro está dado por la microcomputadora a través de la lectura y escritura de datos en los puertos de control. Un circuito decodificador permite que la computadora tenga acceso a los registros de la unidad de control, a través de la activación de un buffer de datos. Se tienen 2 latches que almacenan los valores de las direcciones para las memorias de datos y de coeficientes y un circuito latch-contador que hace el direccionamiento de la PROM de microinstrucciones.

El direccionamiento de la PROM está dado de la siguiente manera: La parte más significativa del Byte contiene el código de la instrucción e indica el bloque de memoria que se utilizará para la ejecución de ésta. La parte menos significativa contiene el número de ciclos de microinstrucción

227469

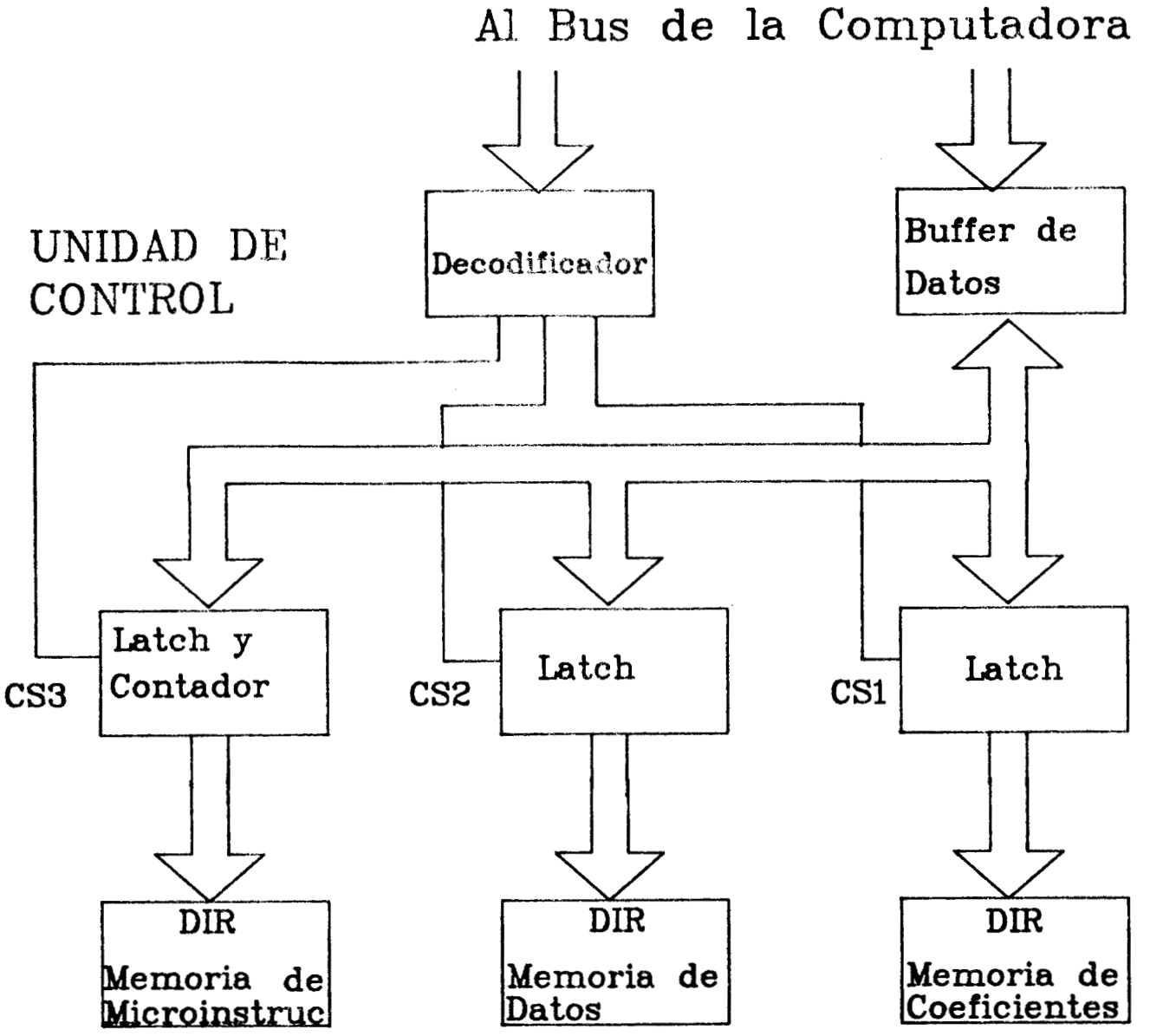


Figura 12. Estructura general de la unidad de control.

para cada operación. La parte que lleva a cabo el direccionamiento de la PROM consiste en un registro de instrucciones IR que se carga desde la computadora con la parte alta de la dirección en PROM de la operación que se quiere llevar a cabo, y un secuenciador de instrucciones que permite que se obtenga el contenido de las localidades de la PROM paso a paso, para así manejar las líneas de control para cada microinstrucción. Este secuenciador de instrucciones o contador en anillo decrementa su valor interno cada vez que se ejecuta una microinstrucción y cuando toda la operación se ha llevado a cabo, se autoinicializa en ceros.

El IR es un latch 74LS175 y guarda los valores altos cuando se escribe en el puerto de control. El secuenciador es un par de contadores 74LS193 que se cargan con una palabra que indica cuantos ciclos de microinstrucciones se deben de ejecutar para cada operación. El valor específico M que se manda a estos contadores es: M=N/4, donde N es el número de microinstrucciones. Este valor se seleccionó para poder utilizar hasta 64 ciclos de microinstrucciones a partir de los 6 bits de direccionamiento. La figura 13 muestra el diagrama completo de la unidad de control.

Como se necesitan 6 líneas de direcciones, se utilizan solo los bits menos significativos del contador alto. Este se conecta a las líneas D2 y D3 del canal de datos de la microcomputadora. Sus salidas se conectan a las líneas internas de direcciones A4 y A5 de la PROM de microinstrucciones. El contador bajo tiene como entradas a las líneas D0 y D1 del bus de la computadora y sus salidas son A0-A3. Esto significa que la entrada en D0 y D1 será un submúltiplo de 4 del número de ciclos de cuenta que llevará a cabo (D0 y D1 se conectan a las líneas de cuenta más significativas del circuito para lograr esto).

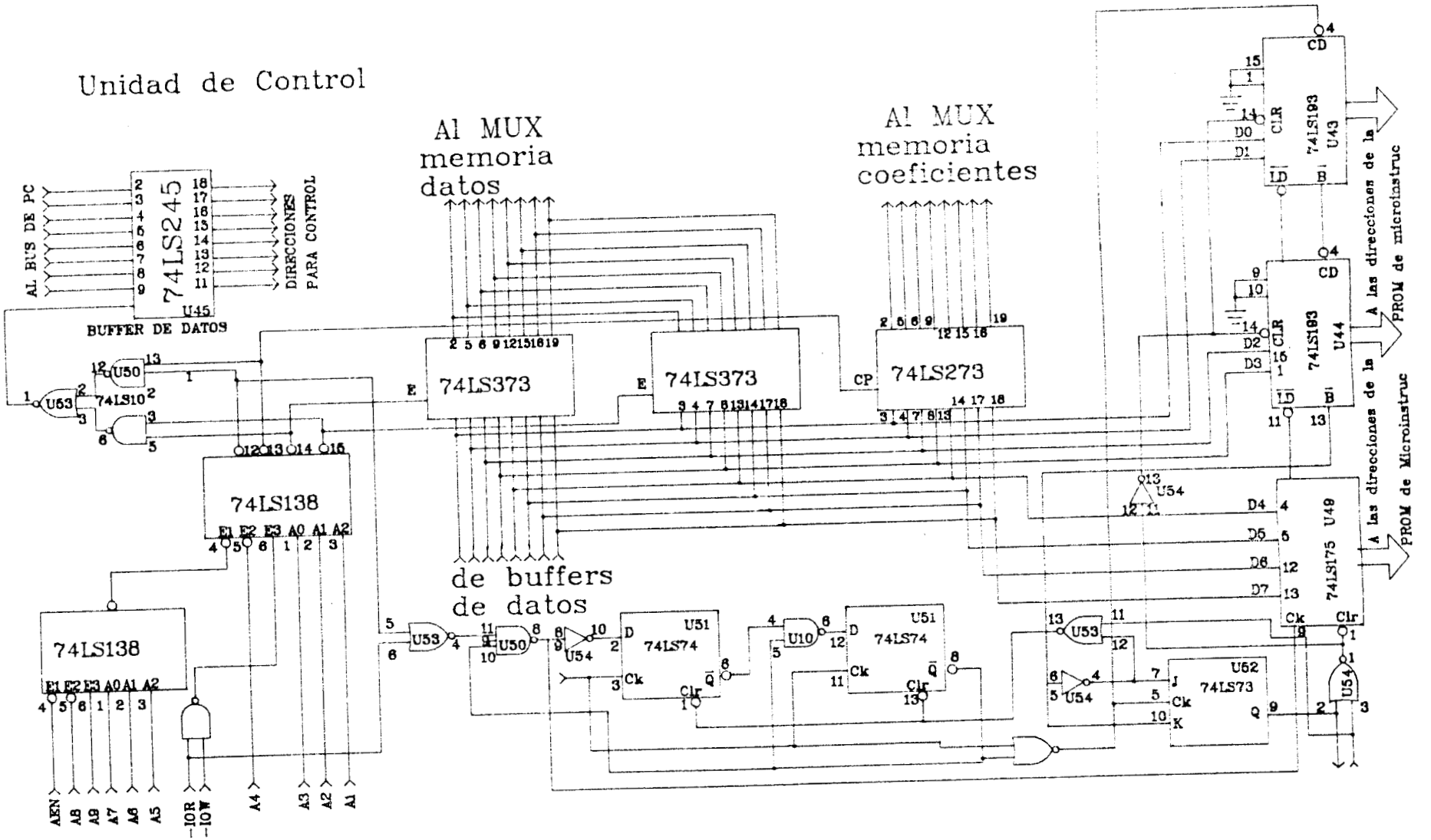


Figura 13. Unidad de control: Diagrama completo.

Para evitar que el contador cicle varias veces, la línea de "borrow" más significativa produce una señal de "clear" o inicialización el contador, a través de un flip-flop que se encarga de sincronizar a esta señal con el reloj del sistema. Esta señal de inicialización se activa también con la señal de reset o inicializacion maestra, proveniente de la microcomputadora.

El último bloque de la unidad de control es un circuito de inicialización de cuenta, que se utiliza para impedir que la señal del reloj interno del coprocesador llegue a los contadores antes de que estos hayan sido cargados con los datos correspondientes al número de cuenta. Para lograr esto, se controla un retraso por medio de las líneas DIR e IOW junto con unos flip-flops 74LS74.

El inicio de la operación se efectúa cuando se ha cargado el valor requerido en los registros y han pasado los 2 ciclos de reloj. El fin de la operación se puede leer en la línea de salida Q del flip-flop 74LS73. Cuando esta línea se encuentra en estado bajo, el coproc lor se encuentra ocupado.

Este diseño permite que se adicionen más instrucciones fácilmente al ocupar más localidades de la PROM de microinstrucciones con los valores de las líneas de control para otras operaciones.

PROGRAMACION

Programa para el cálculo de la FFT en ensamblador

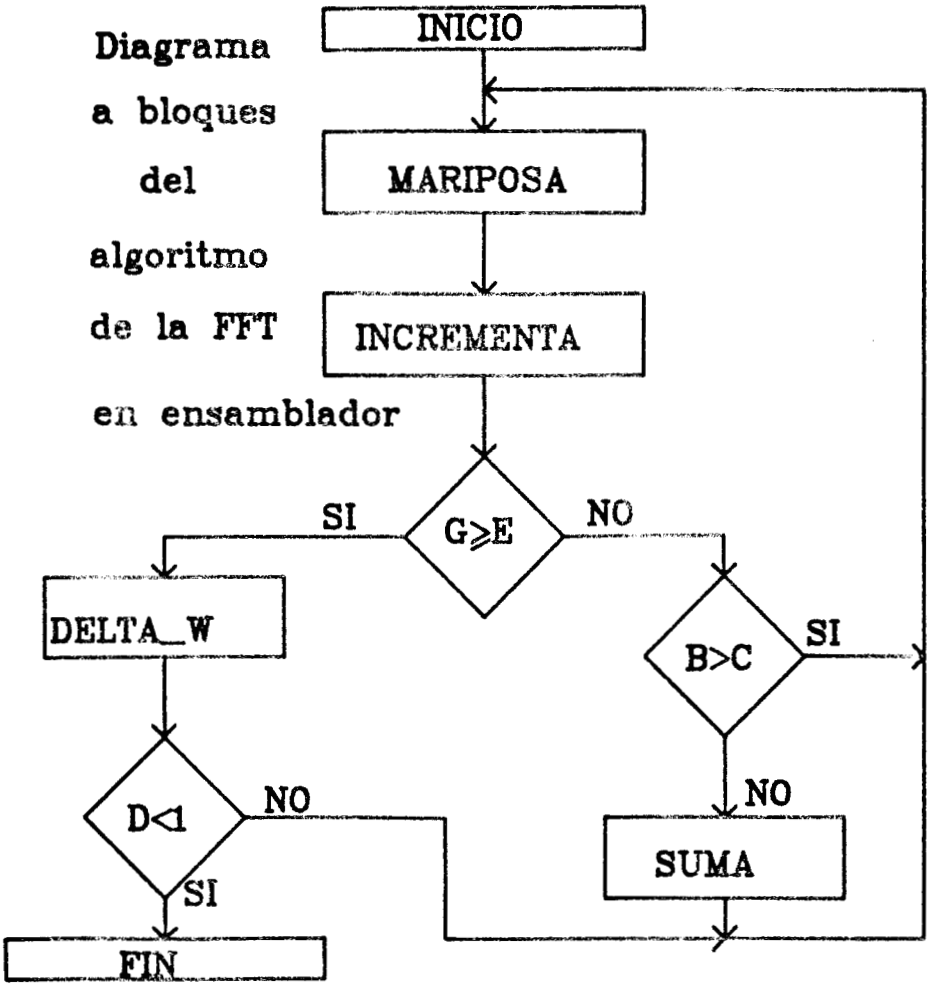
Una de las maneras directas para aumentar la velocidad de ejecución de un programa, es el escribir las rutinas más utilizadas en ensamblador. Para los propósitos del trabajo que se presenta, este programa cumple una función doble:

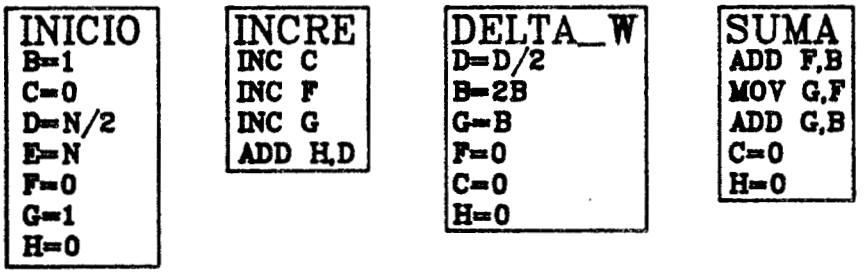
1. Puede incluirse dentro de sistemas y paquetes de programas para procesamiento de señales y aumentar así la velocidad de ejecución de éstos. Así, en algunos casos, puede ser ésta la solución al largo tiempo de procesamiento que se requiere para el análisis de señales.
2. Puede utilizarse como marco básico, dentro del cual se pueden insertar las rutinas que puede efectuar el coprocesador. Por ejemplo, puede calcular todos los índices y hacer los corrimientos para evitar el sobreflujo y dejar los procedimientos del cálculo de la mariposa al coprocesador. De esta manera se puede tener un verdadero sistema de ejecución en paralelo.

El algoritmo que se utilizó fué de decimación en tiempo y radix2. El procedimiento principal se dedica a calcular los índices de los datos y de los coeficientes, mientras que otros se ocupan de tareas tales como el cálculo de la mariposa, el incremento de índices y el manejo de los coeficientes previamente calculados.

La figura 14 muestra el diagrama de flujo del algoritmo. Las variables que utiliza son las siguientes:

B	Indice de decimación. Valor inicial:1
C	Contador de mariposas. Valor inicial:0
D	Valor del incremento del indice de los coeficientes. Valor inicial:N/2
E	Longitud del arreglo. Valor inicial:N
F	Indice de datos. Valor inicial:0
G	Indice de datos. Valor inicial:1
H	Indice de coeficientes. Valor inicial:0





Procedimientos del algoritmo de la FFT

Figura 14. Algoritmo para la FFT

El algoritmo está diseñado alrededor de la operación de la mariposa. Los distintos índices se modifican de acuerdo al número de datos, el índice de la mariposa y la etapa de decimación. Así, siguiendo al diagrama de flujo y para un ejemplo donde se tienen 8 datos, tenemos al inicio los valores siguientes:

B=1, C=0, D=4, E=8, F=0, G=1, H=0

La primera mariposa se ejecuta entre los datos 0 y 1, con el coeficiente 0. Se incrementan los coeficientes C, F, G y H toma el valor H =H+D, de tal manera que B=1, C=1, D=4, E=8, F=1, G=2, H=4. Se hace la comparación entre E y G, y como E es mayor, se hace la comparación entre B y C. Como B no es mayor a C, entonces se pasa a un bloque de suma donde F = F + B (F=2) y luego G = F+B, de manera que G=3, se limpian los valores de C y H, y se ejecuta la siguiente mariposa entre los datos 2 y 3, con el coeficiente 0.

Este procedimiento se continúa hasta que se han agotado los datos de esta primera decimación (cuando E=G). Posteriormente se calculan los nuevos valores de D (el tamaño del incremento de un indice de coeficientes a otro) y de B (el nuevo valor de la decimación) y se procede al cálculo de los índices de esta etapa.

La tabla 2 muestra los valores de los índices para el ejemplo de 8 datos. Se puede observar que las primeras 4 mariposas corresponden a la primera decimación, ya que el valor del indice de decimación B se mantiene en B=0. En esta primera etapa se utiliza el coeficiente 0 exclusivamente. Los índices de los datos se incrementan de uno a uno, de tal manera que los datos que entran a la mariposa son las parejas 0,1, 2,3, 4,5, 6,7. La segunda etapa de decimación utiliza los valores 0 y 2 para los

coeficientes y los índices de los datos se agrupan en las parejas 0,2, 1,3, 4,6, 5,7 . En la última etapa de decimación se utilizan los índices 0,1,2,3 para los coeficientes y los valores 0,4, 1,5, 2,6, 3,7 para los índices de datos. El valor del índice D se decrementa con cada decimación y cuando llega a 0 termina el procedimiento de la FFT. Es importante recordar que los índices de los datos de entrada deben estar previamente recalculados, ya que este algoritmo produce una serie de datos a la salida con los índices desordenados. Este nuevo cálculo de índices puede hacerse a la entrada o la salida de los datos y consiste, para el caso de los algoritmos radix 2 de la FFT, en utilizar el valor binario de los índices e invertir el valor de los bits. Este procedimiento no ocupa mucho tiempo de procesamiento y puede hacerse en cualquier lenguaje.

Para escribir este algoritmo en lenguaje ensamblador, se redistribuyeron los procedimientos y se estructuró un poco el flujo del programa.

El primer procedimiento se dedica a inicializar los índices. Se ha definido un grupo para ensamblar en las mismas localidades los segmentos de código y de datos, y el procedimiento INICIO se declara como público.

Al escribir el programa en ensamblador, se dividió en procedimientos pequeños para lograr que el código fuera más compacto y para probar los distintos elementos del programa fácilmente por separado. Los procedimientos se ensamblaron por separado y se probaron individualmente. Se utilizó el ensamblador de Microsoft (Microsoft Macroassembler Versión 3.0). Una vez que se depuraron y que se verificó que funcionaban adecuadamente, se ligaron en una sola unidad.

INDICES								INDICES UTILIZADOS	
B	C	D	E	F	G	H		MARI	W
1	0	4	8	0	1	0		0,1	0
1	1	4	8	1	2	4			
1	0	4	8	2	3	0		2,3	0
1	1	4	8	3	4	4			
1	0	4	8	4	5	0		4,5	0
1	1	4	8	5	6	4			
1	0	4	8	6	7	0		6,7	0
2	1	2	8	7	8	4			
2	1	2	8	0	2	4		0,2	0
2	0	2	8	0	2	0			
2	1	2	8	1	3	2		1,3	2
2	2	2	8	2	4	4			
2	0	2	8	4	6	0		4,6	0
2	1	2	8	5	7	2		5,7	2
2	2	2	8	6	8	4			
4	0	1	8	0	4	0		0,4	0
4	1	1	8	1	5	1		1,5	1
4	2	1	8	2	6	2		2,6	2
4	3	1	8	3	7	3		3,7	3
4	4	0	8	4	8	4			
		FIN							

Tabla 2. Valores de los indices para una FFT de 8 datos.

Ejecución de la mariposa por el coprocesador.

Los datos que se manejan para el cálculo de la FFT son complejos de 16 bits. Por esto, ocupan 4 bytes cada uno: la palabra baja contiene la parte real y la palabra alta contiene la parte imaginaria.

Para tener acceso al par de datos sobre los cuales se va a operar, se utilizará el bit 1 de direcciones para determinar se se trata de la parte real (bit 1 en estado bajo) o la parte imaginaria. El bit 0 no se utiliza, ya que internamente, el sistema opera como memoria de 16 bits.

El resto de las direcciones específicas se mandad del programa de cálculo de direcciones, escrito en ensamblador, a través de los puertos de entrada y salida.

Operaciones.

Las operaciones básicas del coprocesador son la multiplicación compleja y la magnitud cuadrada. La mariposa y el resto del procesamiento de los datos para terminar la transformada de Fourier son modificaciones de estas.

Multiplicación Compleja.

Si tenemos un par de datos complejos A y B, donde A = Ar + j(Aj) y B = Br + j(Bj); Ar y Br son las partes reales de los datos y Aj y Bj son las partes imaginarias, entonces tenemos que A x B=C donde C también es complejo. La operación completa de multiplicación es:

$$C = Ar\ Br + AjBj + j(ArBj + AjBr)$$

$$C = Cr + j\ (Cj)$$

Para las operaciones específicas del coprocesador, la secuencia será:

227469

1. X = Ar; Y = Br
2. Z = ArBr; Latch1 = ArBr
3. X = Aj; Y = Bj
4. Z = AjBj; C = ArBr + AjBj
5. Mem = ArBr + AjBj

6. X = Ar; Y = Bj
7. Z = ArBj; Latch1 = ArBj
8. X = Aj; Y = Br
9. Z = AjBr; C = AjBr + ArBj
10. Z = AjBr + ArBj

La operación completa toma 10 ciclos de 100 ns cada uno, de talmanera que el tiempo de ejecución es de 1 microsegundo.

Magnitud cuadrada.
La ejecución de esta operación es muy sencilla en este sistema. Si tenemos un dato complejo A = Ar + j (Aj), la magnitud cuadrada se define como la parte reall al cuadrado mas la parte imaginaria al cuadrado:

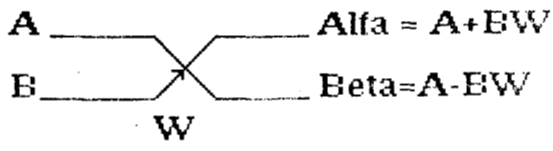
$$A^2 = Ar^2 + Aj^2$$

La secuencia de microinstrucciones es la siguiente:

1. X = Ar ; Y = Ar ;
2. Z = Ar^2 ; Latch1 = Ar^2
3. X = Aj ; Y = Aj ;
4. Z = Aj^2 ; C = $Ar^2 + Aj^2$
5. MEM = $Ar^2 + Aj^2$

La operación toma 500 nanosegundos

Mariposa:



ALFA= Ar + R(BW) + Aj + I(BW) = A + BW

BETA = Ar - R(BW) + Aj - I(BW) = A - BW

R(BW) = BrWr - BjWj

I(BW) = BrWj + BjWr

Secuencia de operaciones:

1. X = Br ; Y = Wr
2. Z = BrWr ; L1 = BrWr
3. X = Bj ; Y = Wj
4. Z = BjWj ; C=BrWr-BjWj ; L2 = BrWr-BjWj = R(BW)

5. X = Br ; Y =Wj
6. Z = BrWj ; L1 = BrWj
7. X = Bj ; Y = Wr
8. Z = BjWr ; C = BrWj + BjWr ; L3 = BrWj + BjWr = I(BW)

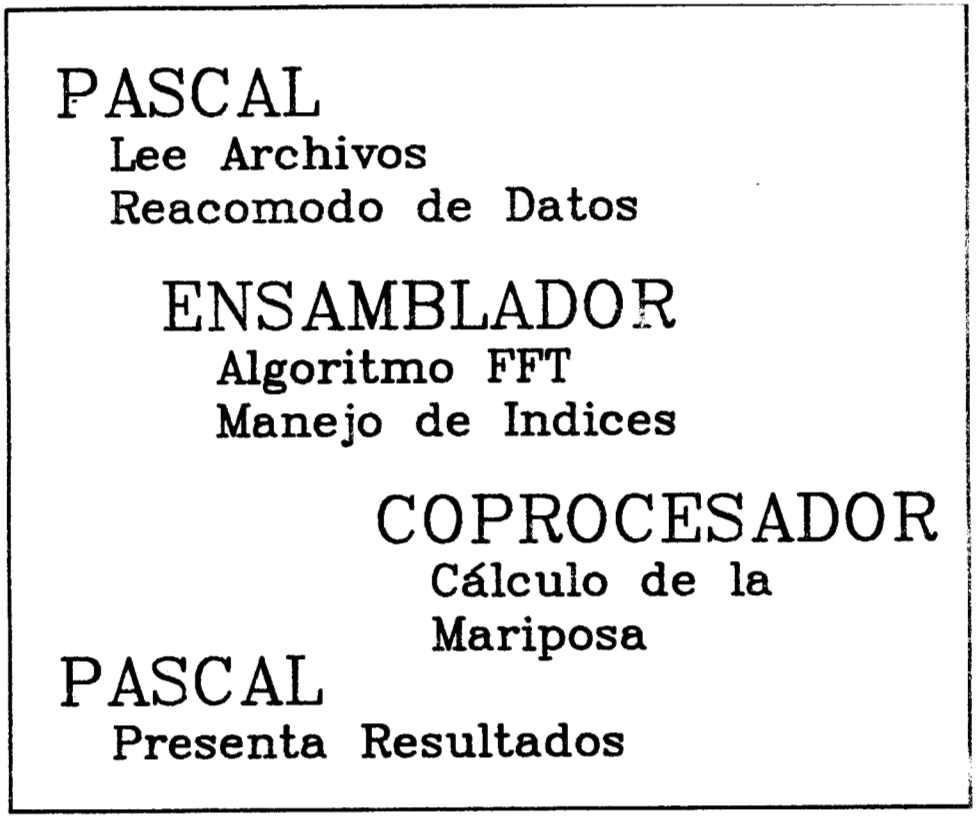
9. L1 = Ar
10. C = Ar + Br ; MEM =Ar + R(BW)
11. L1 = Aj
12. C = Aj + I(BW) ; MEM = Aj + I(BW)

13. L1 = Ar; L4 = R(BW)
14. L2 = Ar + R(BW)
15. MEM = Ar + R(BW)
16. L2 = Ar - R(BW)

17. MEM = Ar - R(BW)
18. L1 = Aj; L4 = I(BW)
19. L2 = Aj + I(BW)
20. MEM = Aj + I(BW)

21. L2 = Aj - I(BW)
22. MEM = Aj - I(BW)

Este sistema de programación está diseñado para poder utilizarse tanto en la modalidad de procesamiento en paralelo entre la microcomputadora y el coprocesador, y como un sistema para el procesamiento rápido a nivel ensamblador. La figura 15 muestra el tipo de programación que se empleó. Se tienen distintos módulos escritos en Pascal, ensamblador y microinstruccines para el coprocesador. El apéndice 1 contiene los listados que se utilizaron para calcular los coeficientes que se cargan en la memoria del coprocesador y los programas para el cálculo de la FFT en ensamblador y los procedimientos del cálculo de la mariposa para el coprocesador.



PROGRAMACION MODULAR

Indice1 [Indice2 Mariposa1] [Indice3 Mariposa2] [Indice4 Mariposa3 ... Indice n−1 ... Mariposa n]

EJECUCION DE PROCEDIMIENTOS PARALELOS

Figura 15. Jerarquía y ejecución de los procedimientos para el cálculo de la FFT.

RESULTADOS Y CONCLUSIONES

RESULTADOS

El sistema desarrollado permite el procesamiento de señales en tiempos breves. Este sistema consiste en una tarjeta con componentes discretos que forman una unidad de procesamiento en paralelo que permite que se realicen cálculos para la transformada rápida de Fourier en tiempos cortos, y una serie de programas para la ejecución del algoritmo de la transformada rápida de Fourier. Estos programas, escritos en ensamblador, están diseñados de tal manera que pueden trabajar conjuntamente con la unidad de procesamiento en paralelo, o pueden utilizarse para incrementar la velocidad de los algoritmos convencionales escritos en lenguajes de alto nivel. Solo difieren en la ejecución del algoritmo para el cálculo de la mariposa. En el caso de la utilización del coprocesador, este último ejecuta la mariposa, mientras que la microcomputadora calcula los índices para los datos y coeficientes que se requieren. Este tipo de operación es un verdadero procesamiento en paralelo, ya que mientras se está ejecutando una mariposa, se calculan los índices de los datos siguientes, de tal manera que cuando el coprocesador se dedica a ejecutar la mariposa, la microcomputadora se dedica a calcular los índices para la mariposa siguiente.

El programa en ensamblador para la mariposa toma 1024 ciclos de reloj aproximadamente (varía en menos del 2% dependiendo de las condiciones del algoritmo), donde casi la mitad del tiempo se ocupa en hacer multiplicaciones. Cada multiplicación toma 123 ciclos de reloj (INTEL, 1981). Las 4 operaciones de multiplicación para una mariposa toman 123 x 4 = 492 ciclos de reloj. Para una frecuencia de reloj de 4.77 MHz, los 1024 ciclos toman 215 microsegundos.

El circuito coprocesador toma 30 ciclos para el cálculo de la mariposa. Cuando el coprocesador opera a su velocidad

máxima de 10 MHz, el cálculo de la mariposa toma 3 microsegundos. Esto es 70 veces más rápido que el programa en ensamblador.

Para calcular una transformada rápida de Fourier de 512 datos, se necesita calcular 256 mariposas por etapa de decimación y se tienen 8 etapas. El total de mariposas necesarias es de 1248. En el programa escrito en ensamblador, esto se convierte en 267.6 milisegundos, mientras que para el coprocesador esto se convierte en 3.49 milisegundos. Es importante hacer notar que a estos tiempos es necesario agregar el tiempo que toma el cálculo de las direcciones de los datos y los coeficientes dentro del algoritmo de la transformada rápida de Fourier.

Para ambos casos, el programa en ensamblador toma aproximadamente 400 ciclos de reloj para calcular los índices siguientes. En tiempo, esto corresponde a 84 microsegundos cada vez que se calculan los índices. En total, el tiempo de ejecución para el programa escrito en ensamblador es de (84 +215) x 1248 = 498 milisegundos. Cuando se utiliza el coprocesador, el tiempo necesario para calcular la mariposa se elimina y solo es necesario tomar el cuenta el tiempo del cálculo de los índices. En este caso este tiempo es de 84 x 1248 = 105 milisegundos.
La velocidad se incrementa en este caso por un factor de cinco.

CONCLUSIONES

La arquitectura del coprocesador se ha modificado varias veces, siempre con el fin de mejorar la velocidad de ejecución de la FFT y para eliminar dificultades y retrasos en la programación. Algunos de los cambios que han sido fundamentales para incrementar la velocidad del coprocesador son la adición de varios registros de almacenamiento temporal, los cuales eliminan la necesidad de almacenar los

resultados intermedios en la memoria y así eliminan decenas de ciclos de reloj por cada cálculo de la mariposa.
Es evidente, a partir de los tiempos de ejecución indicados anteriormente, que existe un desbalance muy grande entre los tiempos del cálculo de los índices y el tiempo de ejecución de la mariposa. Este desbalance hace que el sistema en sí sea ineficiente: por un lado se tiene una unidad de procesamiento que ejecuta su tarea en 3.5 milisegundos, y por otro, un programa que toma 105 milisegundos. Cuando estos dos subsistemas tienen que convivir en un ambiente de procesamiento paralelo, el coprocesador tiene que esperar un tiempo demasiado largo antes de recibir las nuevas direcciones.

Para resolver este problema de ineficiencia, se pueden tener dos soluciones: el utilizar una versión más poderosa de las computadoras tipo PC, como lo son las versiones PC/AT con reloj interno de 8 MHz, o modificar la arquitectura del coprocesador para que calcule independientemente la FFT.

La opción de utilizar una computadora tipo PC/AT es la que en este caso resulta ser más atractiva porque dentro del objetivo inicial de obtener un sistema de procesamiento paralelo, permite que se ejecuten los algoritmos para el cálculo de la FFT con una relación entre los tiempos de ejecución mas aceptable y porque actualmente, este tipo de computadoras están reemplazando a las computadoras tipo PC-XT, ya que los sistemas de programación y los paquetes comerciales requieren de mayor potencia computacional.

El procesador de la computadora PC/AT (80286, en sustitución del 8088 de la PC) puede ejecutar el mismo conjunto de instrucciones del procesador de la PC. Sin embargo, existen tres diferencias fundamentales que aumentan significativamente la velocidad de ejecución:

--La frecuencia del reloj se aumenta de 4.77 a 8 MHz. Las operaciones se ejecutan en menor tiempo.
--Los ciclos de lectura y escritura son de 16 bits, de tal manera que se reducen los tiempos de acceso a memoria a la mitad.
--Las instrucciones son más eficientes. Se ejecutan en menos ciclos de reloj.

Para ejectuar la operación de la mariposa, el procesador 80286 toma 303 ciclos de operación, en vez de utilizar 1024 ciclos. Este ahorro, junto con el incremento en la velocidad del reloj hace que el tiempo de operación se reduzca de 214.5 a 38 microsegundos. Esto significa que el procesador 80286 es capaz de ejecutar las operaciones del 8088 con un incremento de velocidad de significativo (5.6 veces).

Para el cálculo de los índices de datos, el tiempo se reduce de 215 a 38 microsegundos, y el tiempo total para el cálculo de la FFT se reduce de 105 a 18.75 milisegundos

La opción de modificar la arquitectura del coprocesador tiene la ventaja de que los tiempos de procesamiento se minimizan. En este caso una opción posible es el incluir dentro de la unidad de control un banco de memoria rápida donde las direcciones para cada iteración estén previamente calculadas y almacenadas en forma de una tabla. De esta manera, se tendrían las direcciones disponibles y con solo dos ciclos de reloj para cada acceso. Los tiempos de ejecución se podrían reducir a 4 milisegundos para el cálculo de la FFT, pero entonces la limitante serían los tiempos de acceso y la transferencia de datos desde la computadora.
La opción de utilizar una computadora tipo AT debe evaluarse en la práctica. Es posible que con este sistema, se puedan lograr reducciones importantes en el tiempo de

procesamiento. Por ejemplo, para el cálculo de las densidades espectrales electroencefalográficas de 12 canales, con arreglos de 512 datos para cada canal, se tienen tiempos de procesamiento de 8 minutos para programas escritos en Pascal y con una computadora PC normal. Este tiempo tan largo es una seria limitación al empleo de este sistema en un medio clínico real. La utilización de este sistema y una computadora tipo AT permitirán que programas como este puedan tener tiempos de ejecución de unos cuantos segundos.

BIBLIOGRAFIA

BIBLIOGRAFIA

1 AMI, S28214 FAST FOURIER TRANSFORMER, **AMI Product Reference Manual**, (1983).

2 AT&T, WE DSP16 Digital Signal Processor, **Western Electric DSP Data Sheet**, 10 (1986).

3 AT&T, WE DSP32 Digital Signal Processor, **Western Electric DSP32 Data Sheet**, (1987).

4 Blanken, J.D. and Rustan, P.L., Selection Criteria for Efficient Implementation of FFT algorithms, **IEEE Transactions on Acoustics, Speech and Signal Processing**, ASSP-30 (1982) 107-109.

5 Borland International, **Turbo Pascal 4.0 Owner's Handbook**, Borland, Scotts Valley, CA, 1987, 640 pp.

6 Caspe, R.A., Array Processors, **Mini-Micro Systems**, 7 (1978) 54-55.

7 Cooley, J.W. and Tukey, J.W., An algorithm for the machine calculation of complex Fourier series, **Mathematics of Computation**, 19.20 (1965).

8 DataFlowImaging, Data Flow Processing Board For IBM-PC and Compatible Computers, **DF-1Board Data Sheet**, (1986).

9 Essig, D., A Second Generation Digital Signal Processor, **IEEE Journal of Solid State Circuits**, SC-21 (1986) 86-91.

10 Finn, W.J., LSI HARDWARE IMPLEMENTS SIGNAL PROCESSING ALGORITHMS, **Computer Design**, 3 (1980) 137-142.

11 Good, I.J., The Interaction Algorithm and Practical Fourier Analysis, **J.Royal Statistical Society**, 20 (1968).

12 Hagiwara, Y., A Single Chip DSP and its Application to Real Time Speech Analysis, **IEEE Journal of Solid State Circuits**, SC-18 (1983) 91-99.

13 Hays, W.P., A 32 bit VLSI Digital Signal Processor, **IEEE Journal of Solid State Circuits**, SC-20 (1985) 998-1004.

14 Henlin, D.A., A 16 bit x 16 bit pipelined multiplier macrocell., **IEEE Journal of Solid State Circuits**, SC-20 (1985) 542-547.

15 IntelCorp, **iAPX 86/88 User's Manual**, INTEL, Santa Clara, CA, 1981,

227469

16 Mehalic, M., Rustan, P. and Route, G., Effects of Architecture Implementation on DFT Algorithm Performance, **IEEE Transactions on Acoustics, Speech and Signal Processing**, 33,No3 (1985) 684-693.

17 Morris, L.R., A comparative study of time efficient FFT and WFTA programs for general purpose computers., **IEEE Transactions on Acoustics, Speech and Signal Processing**, ASSP 26 (1978) 141-150.

18 Murray, W.H. and Pappas, C.H., **80386/80286 Assembly Language Programming**, Osborne/McGraw-Hill, Berkeley, CA, 1987, 547 pp.

19 Nishitani, T., A Single Chip DSP for Telecommunications Applications, **IEEE Journal of Solid State Circuits**, SC-16 (1981) 372-376.

20 Noll, T.G., A 330 MHz Array Multiplier, **IEEE Journal of solid State Circuits**,SC-21 (1986) 411-416.

21 Norton, P. and Socha, J., **Peter Norton's Assembly Language Book for the IBM PC**, Brady/Prentice-Hall, New York, N.Y., 1986,

22 Nussbaumer, H.J., **Fast Fourier Transform and Convolution Algorithms**, 2nd Ed., Springer-Verlag, New York, 1982,

23 Oppenheim, A.V. and Shafer, R.W., **Digital Signal Processing**, Prentice-Hall,

24 Pedersen, J.E., Fast dedicated microprocessor for real-time frequency analysis of ultrasonic blood velocity measurements., **Medical & Biological Engineering & Computing**, 20 (1982) 681-686.

25 Pezaris, S.D., A 40 ns 17 bit by 17 bit Array Multiplier, **IEEE Transactions on Computers**, (1971) 442-447.

26 Quint, S.R., Wunk, D., Messenheimer, D. and Johnson, R., Multicomputer based Power Spectra for the Operating Room, **IEEE Frontiers of Engineering and Computing in Health Care**, (1984) 82-84.

27 Rabiner, L.R. and Gold, B., **Theory and Application of Digital Signal Processing**, Prentice-Hall, 1975,

28 Rudnick, P., Notes on the Calculation of Fourier Series, **Mathematics of Computation, 20 (1966).**

29 Steeger, G.H., A Signalprocessor for Real-Time Measurements in Neurophysiology, **Signal Processing**, 3 (1981) 397-406.

30 Swan, T., **Mastering Turbo Pascal 4.0**, Hayden, Indianapolis, IA, 1988

31 TexasInstrumentsInc, **TMS 32010 User's Guide** Digital Signal Processor Products, Texas Instruments Inc., 1985, pp.

32 TexasInstrumentsInc, **TMS32020 User's Guide** Digital Signal Processor Products,Texas Instruments Inc., 1986.

33 Wunk, D.F., Realtime EEG Processing Using a High-Speed Coprocessor, **Proc.IEEE 7th Ann. Conf.EMBS**, (1985) 108-111.

.

APENDICES

```
AGRUP     GROUP     CODE_SEG, DATA_SEG
          ASSUME    CS:AGRUP, DS:AGRUP

CODE_SEG            SEGMENT         PUBLIC
          ORG       100H
          EXTRN     INICIO:NEAR, SUMA:NEAR, DELTA_W:NEAR
          EXTRN     INCRE:NEAR, MARIPOSA:NEAR
DATA_SEG            SEGMENT         PUBLIC
          EXTRN     IXB:BYTE, IXC:BYTE, IXD:WORD, IXE:WORD,
          EXTRN     IXF:WORD, IXG:WORD, IXH:WORD
DATA_SEG            ENDS

FFT2                PROC            NEAR
;-------------------------------------------------------------
;
;  Este es el programa principal de la FFT.  LLama a varios
;  procedimientos;
;  que se utilizan en el algoritmo y hace las comparaciones
;  necesarias para ;
;  tomar decisiones de acuerdo a los valores de los indices
;  IXB...IXH.       ;
;-------------------------------------------------------------

          PUSH      AX
          PUSH      BX
          PUSH      CX
AQUI:     CALL      MARIPOSA
          CALL      INCRE
          MOV       AX,IXE
          MOV       BX,IXG
          CMP       AX,BX           ;PRIMER LAZO DE COMPARACION
          JLE       DELTA           ;SI E>G REPITE
          MOV       AH,IXB
          MOV       BH,IXC
          CMP       AH,BH           ;SEGUNDO LAZO DE COMPARACION
          JLE       SUMAR
          JMP       AQUI            ;REPITE MIENTRAS B<=C
SUMAR:    CALL      SUMA
          JMP       AQUI
DELTA:    CALL      DELTA_W
          MOV       CX,IXD
          CMP       CX,0
          JG        AQUI
          POP       CX
          POP       BX
          POP       AX
          INT       20h
FFT2                ENDP
CODE_SEG            ENDS

          END       FFT2
```

```
AGRUP     GROUP     CODE_SEG, DATA_SEG

          ASSUME    CS:AGRUP,DS:AGRUP
          PUBLIC    INICIO
CODE_SEG            SEGMENT        PUBLIC
INICIO    PROC       NEAR

DATA_SEG            SEGMENT        PUBLIC
          PUBLIC    IXB,IXC,IXD,IXE,IXF,IXG,IXH
;-------------------------------------------------------;
;Los indices IB...IH se utilizan durante el algoritmo de la
;  FFT y tienen ;
;unos valores iniciales.
;
;-------------------------------------------------------------
IXB         DB         1                  ;INDICE DE DECIMACION
IXC         DB         0                  ;CONTADOR DE MARIPOSAS
IXD         DW         100h               ;VALOR INCREMENTAL DE
                                           Wn, AL INICIO=N/2
IXE         DW         200h               ;LONGITUD DEL ARREGLO.
                                           AQUI N=8
IXF         DW         0                  ;INDICE DE DATOS
IXG         DW         1                  ;INDICE DE DATOS
IXH         DW         0                 ;INDICE DE COEFICIENTES

DATA_SEG            ENDS
INICIO              ENDP
CODE_SEG            ENDS
END
```

.

```
AGRUP    GROUP     CODE_SEG, DATA_SEG
         ASSUME CS:AGRUP, DS:AGRUP

CODE_SEG         SEGMENT       PUBLIC
         ORG     100H
         PUBLIC                 MARIPOSA
;-------------------------------------------------------------
;Este procedimiento hace la operacion de la mariposa.
Utiliza los indices IXF, IXG e IXH para obtener los valores
de  los datos A y B y para obtener el valor del coeficiente
W.  Los datos son complejos, por lo que es necesario hacer
las   operaciones   con   su   parte real   y su parte
imaginaria.    Si existe sobreflujo (overflow) en las
multiplicaciones,  es  necesario  hacer  un  escalamiento
automatico        sobre        todo        el        arreglo.
;
;-------------------------------------------------------------

MARIPOSA PROC       NEAR
         PUSH       AX
         PUSH       BX
         PUSH       CX
         PUSH       DX
MARIP:   XOR        AX,AX              ;LIMPIA LOS REGISTROS
         XOR        BX,BX
         XOR        CX,CX
         XOR        DX,DX
         MOV        BX,IXH
         SHL        BX,1
         MOV        AL,Wr[BX]          ;CARGAR Wr
         MOV        CL,AL              ;GUARDAR Wr
         MOV        BX,IXG
         SHL        BX,1
         MOV        DX,DATOr[BX]       ;CARGAR Br
         CLC
         MUL        DX                 ;HACER BrWr
         JNC        ALLA
         JNO        ALLA
         CALL       OVERFLOW
ALLA:    PUSH       AX                 ;GUARDAR BrWr
         XOR        AX,AX
         MOV        BX,IXH
         SHL        BX,1
         MOV        AL,Wj[BX]          ;CARGAR Wj
         MOV        CH,AL
         CLC
         MUL        DX                 ;HACER BrWj
         JNC        AQUI               ;NO HUBO SOBREFLUJO
         JNO        AQUI
         CALL       OVERFLOW
AQUI:    PUSH       AX                 ;GUARDAR BrWj
         XOR        AX,AX
         MOV        BX,IXG
         SHL        BX,1
```

```
        MOV     DX,DATOj[BX]    ;CARGAR Bj
        MOV     AL,CL           ;CARGAR Wr
        CLC
        MUL     DX              ;HACER BjWr
        JNC     XYXY            ;NO HUBO SOBREFLUJO
        JNO     XYXY
        CALL    OVERFLOW
XYXY:   PUSH    AX              ;GUARDAR BjWr
        XOR     AX,AX
        MOV     AL,CH           ;TOMAR Wj
        CLC
        MUL     DX              ;HACER BjWj
        JNC     XXXX            ;NO HUBO SOBREFLUJO
        JNO     XXXX
        CALL    OVERFLOW
XXXX:   MOV     BX,AX           ;BX TIENE BjWj
        POP     CX              ;CX TIENE BjWr
        POP     AX              ;AX TIENE BrWj
        ADD     CX,AX          ;CX TIENE BrWj+BjWr=Im(BW)
        POP     AX              ;TOMAR BrWr
        PUSH    CX              ;GUARDAR Im(Bw)
        SUB     AX,BX           ;AX TIENE BrWr-BjWj
        MOV     DX,AX           ;GUARDAR Re(Bw)
        MOV     BX,IXF
        MOV     AX,DATOr[BX]    ;CARGAR Ar
        MOV     CX,AX
        ADD     AX,DX           ;HACER Ar+Re(BW)
        MOV     RESr[BX],AX     ;GUARDAR Ar+Re(BW)
        MOV     CX,AX
        SUB     AX,DX           ;HACER Ar-Re(BW)
        MOV     BX,IXG
        SHL     BX,1
        MOV     RESr[BX],AX     ;GUARDAR Ar-(BW)
        MOV     BX,IXF
        MOV     AX,DATOj[BX]    ;SACAR Aj
        MOV     CX,AX
        POP     DX
        ADD     CX,DX           ;HACER Aj+Im(BW)
        MOV     RESj[BX],CX     ;GUARDAR LO MISMO
        SUB     AX,DX           ;HACER Aj-Im(BW)
        MOV     BX,IXG
        SHL     BX,1
        MOV     RESj[BX],AX     ;GUARDARLO
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
MARIPOSA        ENDP
```

```
OVERFLOW            PROC      NEAR
          PUSH      AX
          PUSH      BX
          PUSH      CX
          XOR       BX,BX
          MOV       CX,08H          ;EL LAZO SE HARA PARA 8
DATOS
INICIO:   MOV       AX,DATOr[BX]
          SHR       AX,1            ;HACER EL CORRIMIENTO
          MOV       DATOr[BX],AX
          MOV       AX,DATOj[BX]
          SHR       AX,1
          MOV       DATOj[BX],AX
          INC       BX         ;DOS INCREMENTOS PARA DIRECCION
          INC       BX                ;PAR DE DATOS DE 16 BITS
          LOOP      INICIO
          POP       CX
          POP       BX
          POP       AX
          JMP       MARIP   ;DESPUES DEL CORRIMIENTO REINICIA

OVERFLOW            ENDP
```

```
;-------------------------------------------------;
;Este procedimiento se usara para guardar los     ;
;valores de los coeficientes W en memoria. Como   ;
;son valores complejos se tienen Wr y Wj, ambos   ;
;de 8 bits en complemento a 2                     ;
;-------------------------------------------------;


DATA_SEG          SEGMENT       PUBLIC

          EXTRN     IXF:WORD, IXG:WORD, IXH:WORD  ;IX SON LOS
INDICES PARA LOS DATOS

          PUBLIC Wr

          Wr        DB
127,127,127,127,126,126,126,126,125,125
                    DB
124,123,122,122,121,120,118,117,116,115
                    DB
113,112,111,109,107,106,104,102,100,98
                    DB
96,94,92,90,88,85,83,81,78,76,73,71,68,65
                    DB
63,60,57,54,51,49,46,43,40,37,34,31,28,25,22
```

```
                    DB
19,16,12,9,6,3,0,253,250,247,244,240,237,234
                    DB
231,228,225,222,219,216,213,210,207,205,202
                    DB
199,196,193,191,188,185,183,180,178,175,173,171
                    DB
168,166,164,162,160,158,156,154,152,150,149
                    DB
147,145,144,143,141,140,139,138,136,135,134,134
                    DB
133,132,131,131,132,133,134,134,135,136,138,139
                    DB
140,141,143,144,145,147,149,150,152,154,156,158
                    DB
160,162,164,166,168,171,173,175,178,180,183,185
                    DB
188,191,193,196,199,202,205,207,210,213,216,219
                    DB
222,225,228,231,234,237,240,244,247,250,253,0
                    DB
3,6,9,12,16,19,22,25,28,31,34,37,40,43,46,49,51
                    DB
54,57,60,63,65,68,71,73,76,78,81,83,85,88,90,92
                    DB
94,96,98,100,102,104,106,107,109,111,112,113,115
                    DB
116,117,118,120,121,122,122,123,124,125,125,126
                    DB        126,126,127,127,127

          PUBLIC    WJ

          WJ        DB
0,253,250,247,244,240,237,234,231,228,225,222
                    DB
219,216,213,210,207,205,202,199,196,193,191,188
                    DB
185,183,180,178,175,173,171,168,166,164,162,160
                    DB
158,156,154,152,150,149,147,145,144,143,141,140
                    DB
139,138,136,135,134,134,133,132,131,131,130,130
                    DB
130,129,129,129,129,129,129,129,130,130,130,131
                    DB
131,132,133,134,134,135,136,138,139,140,141,143
                    DB
144,145,147,149,150,152,154,156,158,160,162,164
                    DB
166,168,171,173,175,178,180,183,185,188,191,193
                    DB
196,199,202,205,207,210,213,216,219,222,225,228
                    DB
231,234,237,240,244,247,250,253,0,3,6,9,12,16,19
```

```
                        DB
22,25,28,31,34,37,40,43,46,49,51,54,57,60,63,65
                        DB
68,71,73,76,78,81,83,85,88,90,92,94,96,98,100
                        DB
102,104,106,107,109,111,112,113,115,116,117,118
                        DB
120,121,122,122,123,124,125,125,126,126,126,127
                        DB
127,127,127,127,127,127,126,126,126,125,125,124
                        DB
123,122,122,121,120,118,117,116,115,113,112,111
                        DB
109,107,106,104,102,100,98,96,94,92,90,88,85,83
                        DB
81,78,76,73,71,68,65,63,60,57,54,51,49,46,43,40
                        DB
37,34,31,28,25,22,19,16,12,9,6,3

          PUBLIC     DATOr
          DATOr     DW
1797,1805,1870,1926,1989,2009,1959,1906,1855,1837
                        DW
1853,1925,2025,2073,2061,2004,1929,1841,1849,1881
                        DW
1931,2025,2039,2051,2019,1840,1721,1745,1897,2069
                        DW
2171,2157,2091,1989,1893,1815,1706,1689,1813,2034
                        DW
2145,2145,2078,2019,1914,1789,1694,1661,1787,2011
                        DW
2191,2294,2244,1997,1720,1596,1547,1544,1704,2040
                        DW
2281,2340,2253,2117,1988,1784,1641,1599,1699,1974
                        DW
2246,2381,2310,2115,1947,1800,1661,1592,1541,1608
                        DW
1799,2017,2133,2085,2015,1930,1864,1834,1743,1802
                        DW
2043,2129,2173,2230,2123,1954,1798,1649,1621,1759
                        DW
1922,2081,2155,2145,2084,1960,1804,1672,1708,1831
                        DW
1911,2057,2183,2165,2035,1874,1704,1638,1647,1767
                        DW
1928,2043,2174,2249,2202,2023,1837,1852,1936,1950
                        DW
2027,2126,2151,2112,2007,1930,1867,1824,1842,1968
                        DW
2056,2083,2053,2011,1983,1934,1957,1960,1944,1949
                        DW
2017,2045,2061,2057,2076,2045,1937,1827,1812,1940
                        DW
2052,2092,2061,2044,2037,1979,1880,1801,1812,1871
```

```
                        DW
1989,2085,2072,2012,2000,2013,1951,1874,1866,1823
                        DW
1825,1965,2068,2109,2103,2016,1893,1893,1811,1741
                        DW
1862,1966,2081,2165,2085,2024,2017,1896,1800,1769
                        DW
1794,1921,2059,2111,2113,2067,2033,1990,1907,1829
                        DW
1872,1960,2026,2027,1983,1984,1984,1962,1920,1901
                        DW
1967,2013,1993,1973,1940,1917,1985,1981,1942,1904
                        DW
1855,1925,1989,2006,2022,2024,2037,1995,2001,1981
                        DW
1929,1966,1915,1839,1869,1955,2002,1931,1892,1888
                        DW
1920,1998,2010,2007,2024,2033,1992,1997,2011,1985
PUBLIC                  DATOj
        DATOj     DW          512 DUP (0)
PUBLIC                  RESr
        RESr      DW           512 DUP (0)
PUBLIC                  RESj
        RESj      DW           512 DUP (0)


DATA_SEG        ENDS
CODE_SEG        ENDS
END
```

.

```
AGRUP     GROUP     CODE_SEG, DATA_SEG
          ASSUME    CS:AGRUP, DS:AGRUP

CODE_SEG            SEGMENT          PUBLIC

          PUBLIC    INCRE
DATA_SEG            SEGMENT          PUBLIC
          EXTRN     IXB:BYTE, IXC:BYTE, IXD:BYTE, IXE:BYTE
          EXTRN     IXF:BYTE, IXG:BYTE, IXH:BYTE
          EXTRN     DATO:WORD
DATA_SEG            ENDS
          EXTRN     DATOS:NEAR

INCRE               PROC             NEAR

;-------------------------------------------------------------
;Este procedimiento incrementa unos indices,despues de que
;se hizo la mariposa.
;
;-------------------------------------------------------------
PUSH       AX
          CALL      DATOS
          INC       IXC              ;INCREMENTA LOS REGISTROS
          INC       IXF
          INC       IXG
          MOV       AL,IXD           ;SUMAR IXH+IXD
          ADD       IXH,AL
POP        AX
          RET

INCRE               ENDP

CODE_SEG            ENDS
          END
.
```

```
AGRUP     GROUP     CODE_SEG, DATA_SEG
          ASSUME    CS:AGRUP, DS:AGRUP

CODE_SEG            SEGMENT         PUBLIC

          PUBLIC    DELTA_W

DATA_SEG            SEGMENT         PUBLIC
          EXTRN     IXB:BYTE, IXC:BYTE, IXD:WORD, IXE:WORD
          EXTRN     IXF:WORD, IXG:WORD, IXH:WORD
DATA_SEG            ENDS

DELTA_W             PROC            NEAR

;-------------------------------------------------------------
;
;En este procedimiento cambia el tamaño del brinco para
calcular el nuevo indice del valor incremental de los
coeficientes IXD, incrementa al indice de decimacion IXB, el
indice de datos IXG, e inicializa en ceros a losindices IXF,
IXC e IXH.
;
;-------------------------------------------------------------
;*************************************************************;
;ESTE PROCEDIMIENTO FUNCIONA BIEN.   PROBADO 18 FEB 1988;
;*************************************************************;

          PUSH      AX
          MOV       AX,IXD
          SHR       AX,1
          MOV       IXD,AX                          ;IXD=IXD/2
          MOV       AL,IXB
          SHL       AX,1
          MOV       IXB,AL                          ;IXB=IXB*2
          MOV       IXG,AX                          ;IXG=IXB
          XOR       AX,AX                           ;AL=0
          MOV       IXF,AX
          MOV       IXC,AL
          MOV       IXH,AX      ;INICIALIZA IXF,IXC,IXH CON 0
          POP       AX
          RET

DELTA_W             ENDP
CODE_SEG            ENDS
          END
.
```

227469

```
AGRUP     GROUP     CODE_SEG, DATA_SEG
          ASSUME    CS:AGRUP, DS:AGRUP

CODE_SEG            SEGMENT         PUBLIC

          PUBLIC    SUMA

DATA_SEG            SEGMENT         PUBLIC
          EXTRN     IXB:BYTE, IXC:BYTE, IXD:BYTE, IXE:BYTE
          EXTRN     IXF:BYTE, IXG:BYTE, IXH:BYTE
DATA_SEG            ENDS

;-------------------------------------------------;
;Este procedimiento hace sumas de indices.        ;
;-------------------------------------------------;
SUMA                PROC            NEAR

          PUSH      AX
          PUSH      BX
          MOV       AL,IXB                  ;AL TOMA A IXB
          MOV       AH,AL                   ;AH TOMA A IXB
          MOV       BL,IXF                  ;BL TOMA A IXF
          ADD       AL,BL                   ;SUMAR IXB + IXF
          MOV       IXF,AL                  ;RESULTADO EN IXF
          ADD       AL,AH              ;SUMAR IXB AL RESULTADO
          MOV       IXG,AL                  ;GUARDARLO EN IXG
          XOR       BX,BX                   ;PONER BX EN 0
          MOV       IXC,BL
          MOV       IXH,BL         ;INICIALIZAR IXC E IXH EN 0
          POP       BX
          POP       AX
          RET

SUMA                ENDP

CODE_SEG            ENDS
          END
```

.

```
AGRUP    GROUP     CODE_SEG, DATA_SEG
         ASSUME CS:AGRUP, DS:AGRUP

CODE_SEG         SEGMENT       PUBLIC
         ORG     100H
EXTRN IXF:WORD, IXG:WORD, IXH:WORD
EXTRN PUERTO_1:WORD, PUERTO_2:WORD, PUERTO_3:WORD
EXTRN PUERTO_4:WORD, INSTRU:WORD
         PUBLIC                MARIPOSA
;-------------------------------------------------------
;Este procedimiento hace la operacion de la mariposa por
medio del coprocesador.  Utiliza los indices IXF, IXG e IXH
para obtener los  valores  de  los datos A y B y para
obtener el valor del coeficiente W.  Los datos son
complejos,por lo que es necesario hacer las  operaciones
con  su  parte real  y su parte imaginaria.  Si existe
sobreflujo (overflow) en las multiplicaciones, es necesario
hacer un escalamiento  automatico  sobre todo el arreglo.
Los resultados se guardan automaticamente en la memoria.
;
;-------------------------------------------------------
;
MARIPOSA PROC       NEAR
         PUSH       AX

         PUSH       CX
         MOV        AX,IXF
         MOV        CX,PUERTO_1;DIRECCION DE DATO1
         OUT        CX,AX
         MOV        AX,IXG
         MOV        CX,PUERTO_2;DIRECCION DE DATO2
         OUT        CX,AX
         MOV        AX,IXH
         MOV        CX,PUERTO_3;DIRECCION DE COEFICIENTE
         OUT        CX,AX
         MOV        AX,INSTRU;DIRECCION DE MICROINSTRUCCION
         MOV        CX,PUERTO_4;LECTURA DE OVERFLOW Y FIN DE
                              OPERACION
         POP        CX
         POP        AX

         RET
MARIPOSA             ENDP


DATA_SEG         ENDS
CODE_SEG         ENDS
END
```

```
PROGRAM calcula_omegas;
{ESTE PROGRAMA CALCULA LOS COEFICIENTES W PARA FFT Y LOS
GRAFICA EN LA PANTALLA A DISTINTAS RESOLUCIONES}
USES   crt,graph;
Var
  grDriver,grMode,GrError:integer;
  wrtrun,wjtrun,wr3:integer;
  color,N1,M,i: integer;
  wr1,wr2,wj0,wj1:array[0..511]of integer;
  wr,wj:array[0..511]of real;
Const
  n=511;
BEGIN
  grDriver:=Detect;
  InitGraph( grDriver, grMode,'');
  N1:=getmaxX;
  M:=getmaxY;
  color:=getmaxcolor;
  for i:=0 to N do
   Begin
     wr[i]:=cos((2*pi/N)*i);
     wj[i]:=-sin((2*pi/N)*i);
     wrtrun:=round(256*wr[i]);
     wr1[i]:=round(128*wr[i]);
     wr2[i]:=round(1024*wr[i]);
     wr3:=round(4096*wr[i]);
     wj1[i]:=round(128*wj[i]);
     wj0[i]:=round(256*wj[i]);
   End;
 {    writeln('Este programa imprime los valores de la parte
real de los');
     writeln('coeficientes W redondeados a 12,10,9,8 y 7
bits mas signo');
     writeln(lst,i,'     ',wr,'      ',wr1,'      ',wj,'
',wj1);
     }

     setcolor(red);
     for i:=0 to N do
     lineto(i,round((128-wr1[i])*1.35));
     setcolor(cyan);
     moveto(0,0);
     For i:=0 to N do
     lineto(i+10,round((1024-wr2[i])*0.17));
     setcolor(lightgreen);
     moveto(0,0);
     for i:=0 to 511 do
     lineto(i+20,round((1-wr[i])*M/2));
     setcolor(magenta);
     moveto(0,round((1-wj[0])*m/2));
     for i:=0 to 511 do
     lineto(i,round((1-wj[i])*M/2));
     setcolor(lightred);
     moveto(10,round((256-wj0[1])*0.681));
```

```
     for i:=0 to 511 do
     lineto(i+10,round((256-wj0[i])*0.681));
     setcolor(white);
    moveto(20,round((128-wj1[1])*1.350));
    for i:=0 to 511 do
    lineto(i+20,round((128-wj1[i])*1.350));
     repeat until keypressed;
END..
```