DIVISION DE ESTUDIOS DE POSGRADO
FACULTAD DE INGENIERIA

" DISEÑO DE UN COPROCESADOR PARA REALIZAR LA SUMA DE PRODUCTOS EN ARITMETICA DE PUNTO FLOTANTE Y PALABRA DE 32 BITS PARA UNA PC - XT "

ING. JORGE ANTONIO POLANIA PUENTES

T E S I S

PRESENTADA A LA DIVISION DE ESTUDIOS DE POSGRADO DE LA FACULTAD DE INGENIERIA DE LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

COMO REQUISITO PARA OBTENER EL GRADO DE :

MAESTRO EN INGENIERIA ELECTRICA
(OPCION ELECTRONICA)
1 9 9 2

DIRECTOR DE TESIS : M. I. JORGE CHRISTEN GRACIA

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

RESUMEN.

El proyecto consiste en el diseño y construcción de una tarjeta de coprocesamiento a insertar en una de las ranuras de la computadora PC-XT, que realice en forma de procesamiento paralelo pipeline la suma de productos, con datos de 32 bits en aritmética de punto-flotante.

La arquitectura del coprocesador tiene dos módulos de memoria RAM para el almacenamiento de los vectores y posterior lectura en forma pipeline, de 1K x 32 BITS cada una. Una unidad aritmética que realiza la suma de productos en arquitectura pipeline, utilizando procesadores matemáticos de punto-flotante: el ADSP-3201KG como multiplicador y el ADSP-3202KG como acumulador, de Analog Devices. La unidad ejecuta una suma de productos en 200 nsg y su rendimiento y productividad dependerá de la longitud de los vectores alimentados al pipeline. La unidad de control no usa un microprocesador especifico, fué diseñada de tal forma que cumpliera con los requisitos de tiempo y produjera las señales de control apropiadas para el funcionamiento del coprocesador.

La comunicación (handshaking) entre el procesador de la PC-XT y el coprocesador, y la lectura del resultado final de una sumatoria de productos, se hace por software, habilitando y leyendo puertos de entrada / salida de la computadora.

La tarjeta fué probada simulando algoritmos para la implantación de un filtro FIR de longitud-80 y de un filtro IIR de longitud-5 en lenguaje de programación Turbo Pascal V.5.0.

1. INTRODUCCION.

El presente proyecto consiste en diseñar un coprocesador para la PC - XT que realice la suma de productos con operandos de 32 bits en aritmética de punto flotante con formato IEEE-754 y en estructura pipeline.

El proyecto se originó basado en que una computadora personal realiza este tipo de tarea aún usando el coprocesador matemático en una forma muy lenta. Por ejemplo, para ejecutar una sola suma de productos con instrucciones de lenguaje ensamblador del 8087 en punto flotante [1] se deben ejecutar las siguientes instrucciones:

FLD : cargar dato1 de memoria al stack ..... 7.6 μseg
FLD : cargar dato2 de memoria al stack..... 7.6 μseg
FMUL : multiplicar dato1 x dato2 ..... 18.0 μseg
FADD : sumar este resultado con el anterior ..... 14.0 μseg

Esto quiere decir que la ejecución completa se llevaría a cabo en 47.2 μseg, lo cual es un tiempo considerable para una aplicación que requiera varias sumas de productos como es el caso de la implementación de un filtro, una correlación, multiplicación de matrices y otras aplicaciones similares. Peor aún si la computadora personal no tiene coprocesador matemático.

Se ha comprobado que tomando los principios del procesamiento paralelo, adicionando tarjetas de coprocesamiento matemático en punto flotante a una computadora personal, se puede llegar a tener a bajo costo una supercomputadora. Es así como, en el Laboratorio

de Alta Velocidad Digital del Departamento de Ingeniería Eléctrica de la Universidad de Florida, fué implantada una arquitectura [2] de coprocesamiento paralelo para una PC-AT consistente de cuatro tarjetas : una tarjeta controladora y tres tarjetas procesadoras idénticas, cada una de las cuales contiene una fila (1x3 procesadores) de un arreglo de 3x3 procesadores en punto flotante.

Con base en la anterior experiencia, se propone en este proyecto, diseñar y construir un coprocesador en estructura pipeline y probar su eficiencia en lo referente al incremento de velocidad de procesamiento, desarrollando algunas aplicaciones y comparando los tiempos de ejecución con la de la computadora PC-XT trabajando sin el coprocesador.

2. PROCESAMIENTO PARALELO.

Las arquitecturas avanzadas de compudoras se centran alrededor del procesamiento paralelo. Pueden clasificarse en : Procesadores Pipeline, Procesadores Matriciales y multiprocesamiento [3].

El procesamiento paralelo puede aplicarse a nivel hardware/software o a nivel algorítmico y de programación. Los elevados rendimientos se obtienen por la realización concurrente de actividades en el cumplimiento de una tarea o trabajo a realizar por la computadora.

Existen tres tipos de arquitecturas en las computadoras que realizan procesamiento paralelo :

1. Procesadores Pipeline
2. Procesadores Matriciales
3. Multiprocesadores

El procesador pipeline efectúa cálculos traslapados para explotar el paralelismo temporal, el Matricial emplea múltiples unidades aritméticas lógicas (ALU) sincronizadas para lograr paralelismo espacial y el Multiprocesamiento emplea un paralelismo asíncrono usando varios procesadores interactivos que disponen de recursos compartidos (memorias, bases de datos, etc).
Las tres estructuras no son excluyentes, aunque la más encontrada en computadoras paralelas es la Pipeline.

2.1 PROCESADORES PIPELINE.

En un procesador pipeline las instrucciones se ejecutan de modo traslapado de la siguiente forma : búsqueda de instrucciones, decodificación de instrucciones, búsqueda de operandos y ejecución. El diseño de este tipo de procesadores tiene que ver con el secuenciamiento de las tareas, prevención de colisiones, control de congestión y administración de bifurcaciones. En la FIG.1 se muestra el procesamiento pipeline a nivel de unidad aritmética y a nivel de procesadores.

2.2 PROCESADORES MATRICIALES.

Un procesador matricial es un procesador con múltiples unidades aritméticas llamados elementos de proceso que operan en paralelo en forma sincronizada por una unidad de control.

Cada uno de los elementos procesadores (EP) tienen asociada una memoria local y se comunican entre ellos a través de una red de interconexión controlada por una unidad de control como se muestra en la FIG.2. Este tipo de procesadores son especialmente diseñados para realizar cálculos vectoriales sobre matrices.

La unidad de control (U.C) tiene su memoria para el almacenamiento de programas. Ella decodifica las instrucciones y determina donde deben de ejecutarse. Las escalares las ejecuta la U.C y las vectoriales las transmite a los elementos procesadores para una ejecución en paralela. Los operandos vectoriales son almacenados en las memorias locales (MEP).

La ejecución de una instrucción vectorial en un elemento

procesador se controla mediante un vector de enmascaramiento. Sólo los E.P activos ejecutan la instrucción vectorial. Los intercambios de datos entre los elementos procesadores se efectúan a través de una red de interconexión controlada por la unidad central.

Normalmente un procesador matrical está conectado a una computadora principal (host) a través de la unidad central. La computadora principal administra los recursos, supervisa periféricos y controla la entrada/salida con el mundo exterior.

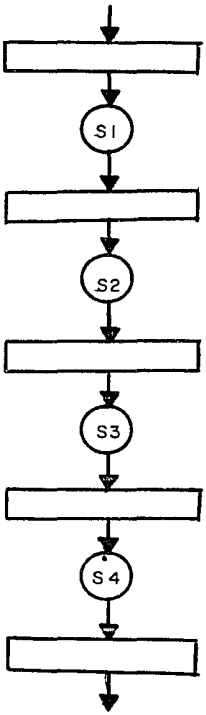
2.3 MULTIPROCESAMIENTO.

Lo forman múltiples procesadores que comparten la memoria, canales de entrada/salida (E/S), periféricos y son cotroladas por un único sistema operativo integrado que facilita las interacciones entre los procesadores y sus programas, a nivel proceso, conjunto de dato, y elementos de datos.

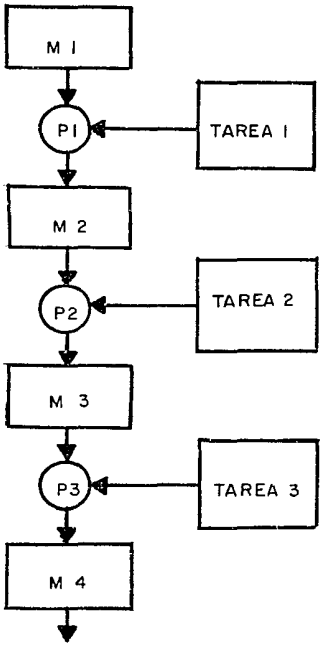
Cada procesador tiene su propia memoria local y la comunicación entre ellos se realiza a través de una memoria compartida o mediante una red de interrupción. La FIG.3 muestra una arquitectura general para multiprocesamiento.

La comunicación entre tareas asignadas a un mismo procesador se realiza a través de la memoria local (ML) y entre tareas asignadas a diferentes procesadores a través de un puerto de comunicación que reside en la memoria de comunicación. A cada procesador se le asocia un puerto de comunicación como puerto de entrada.

PROCESAMIENTO PIPELINE



PIPELINE ARITMETICO



PIPELINE DE PROCESADORES

FIG , 1

PROCESAMIENTO MATRICIAL

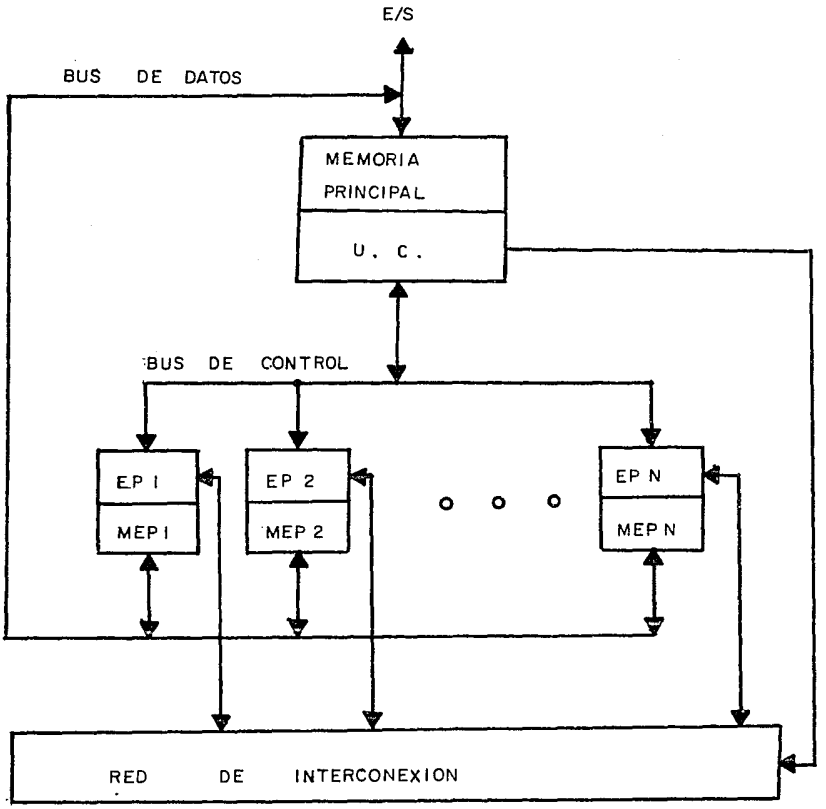


FIG . 2

MULTIPROCESAMIENTO

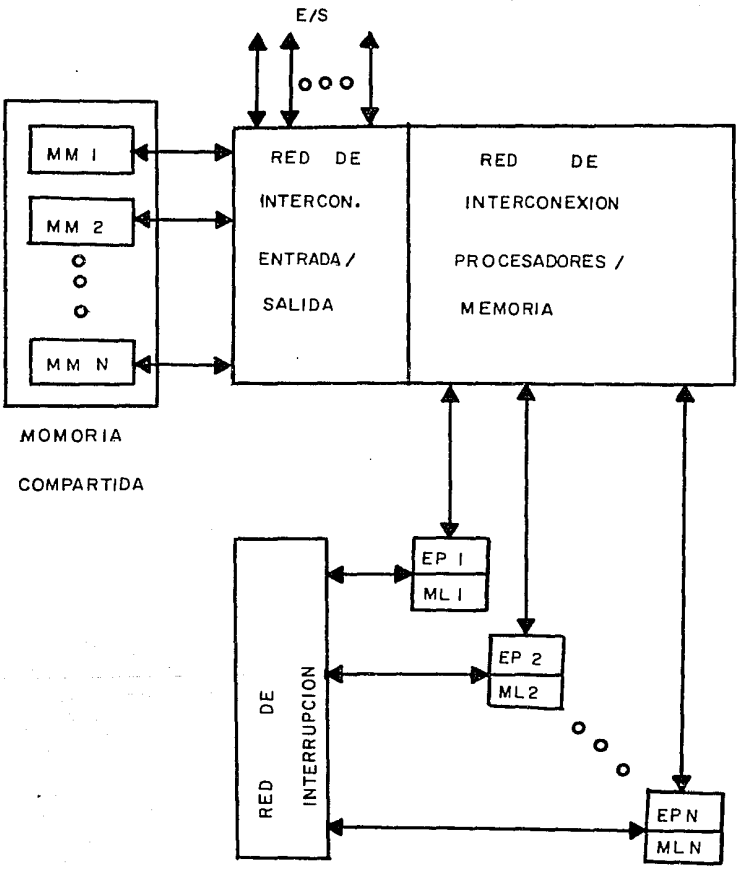


FIG. 3

3. PROCESAMIENTO PIPELINE.

En un procesamiento pipeline, la tarea de entrada se subdivide en una secuencia de subtareas cada una de las cuales es ejecutada por una etapa del pipeline. Las tareas sucesivas pasan por el pipeline y van ejecutándose en modo traslapado a nivel de subtarea como se indica en la FIG.4. El rendimiento dependerá de la segmentación de las tareas de entrada en una secuencia adecuada de subtareas. Por cada pulso de reloj, una subtarea pasa a otra etapa del pipeline, de tal forma, que cuando el pipeline esté lleno se producirá un resultado por cada pulso de reloj [3].

Si se tienen k etapas de pipeline y n número de tareas, el número de ciclos necesarios para ejecutarlas será igual al número de ciclos para llenar el pipeline - ejecución de la primera tarea- que es igual al número de etapas del pipeline, más el número de ciclos para ejecutar las restantes n-1 tareas - un ciclo por tarea-. Por tanto, si

T_k = Número de ciclos para el procesador pipeline , y
T_1 = Números de ciclos para el uniprocesador
se tiene :

$$T_k = k + (n - 1)$$

$$T_1 = n \cdot k$$

3.1 GANANCIA, PRODUCTIVIDAD Y RENDIMIENTO.

La ganancia (speedup) se define como el número de ciclos utilizados por un uniprocesador para ejecutar una determinada tarea, entre el número de ciclos necesarios para ejecutar esa misma tarea por un procesador pipeline.

$$\text{GANANCIA (SPEEDUP)} : S_k = \frac{T_1}{T_k} = \frac{n \cdot k}{k+(n-1)}$$

Como la ganancia máxima es k (cuando el número de tareas es grande), quiere decir que, el tiempo de ejecución para el procesador pipeline es k veces menor que el utilizado por un uniprocesador (no-pipeline).

La productividad (throughput) del pipeline es igual al número de resultados obtenidos entre el total de tiempo de ejecución esto es,

$$\text{PRODUCTIVIDAD (throughput)} : Th = \frac{n}{(k + n - 1)\tau}$$

donde τ es el tiempo de reloj de pipeline. La productividad se mide generalmente en MFLOPS (millones de operaciones en punto flotante por segundo).

El rendimiento es la relación entre el número de espacio - tiempo utilizado y el número total de espacio-tiempo del pipeline, osea,

$$\text{RENDIMIENTO} : \eta = \frac{n \cdot \tau}{(k + n - 1)\tau} = \frac{n}{k + n - 1}$$

Como se puede deducir de la anterior fórmula, al aumentar el número de tareas aumenta el rendimiento del pipeline.

3.2 TABLAS DE RESERVA Y COLISIONES.

Las tablas de reserva y colisiones [3], permiten mostrar cómo se utilizan (reservan) las sucesivas etapas de cauce de un pipeline para la evaluación de una función especifica en sucesivos ciclos de operación. En la FIG.5 se muestra un pipeline no-lineal multifuncional que realiza las funciones A y B. El flujo de operación de cada una de estas funciones dentro del pipeline se manifiesta en las tablas de reserva dadas en ésta misma figura.

Cuando dos o más iniciaciones intentan utilizar la misma etapa del pipeline al mismo tiempo, se produce una colisión. Un problema es entonces, planificar adecuadamente las tareas que esperan en cola su iniciación, de tal forma que en la realización de una tarea se eviten las colisiones y se logre una productividad más alta.

En un pipeline estático (patrón de flujo fijo), todas las iniciaciones se caracterizan por la misma tabla de reservas. En uno dinámico, hay un conjunto de tablas de reservas, una por cada función, que se deben evaluar en su conjunto, para evitar las posibles colisiones.

El diagrama de estados que se hace a partir de una tabla o tablas de reserva, caracteriza las iniciaciones sucesivas de las tareas en un cauce y permite hallando los vectores de colisión, evitar las colisiones en el proceso. Esto se consigue, encontrando la latencia (número de unidades de tiempo entre dos iniciaciones)

mínima permisible, para que sin colisiones haya una alta productividad.

3.3 TIPOS DE MEMORIA PARA PIPELINE.

Los procesadores pipeline requieren de una memoria entrelazada que eviten los conflictos en el acceso. Existen memorias con acceso secuencial, con acceso concurrente o una combinación de las dos.

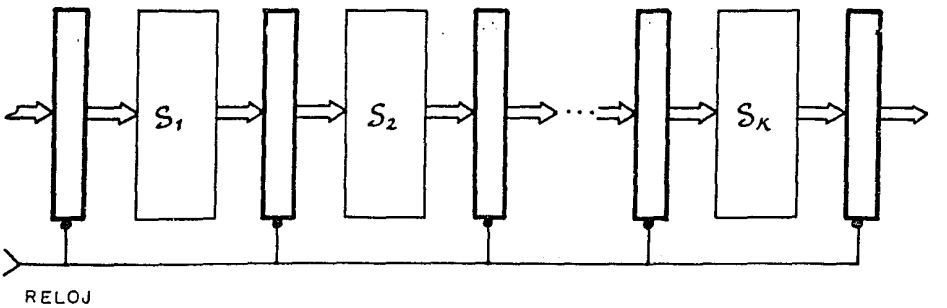
3.3.1 MEMORIA CON ACCESO SECUENCIAL.

En la FIG.6 se muestra la estructura de este tipo de memoria. Los n-m bits correspondientes a la dirección alta se aplican simultáneamente a todos módulos en un acceso y los restantes m bits de la dirección seleccionan la información de un módulo en particular, de tal forma que a la salida del MUX se presentan las palabras en forma consecutiva, ya sea en la lectura o en la escritura.

3.3.2 MEMORIA CON ACCESO CONCURRENTE.

Esta configuración también mostrada en la FIG.6, permite accesar a los módulos en forma concurrente. Los m bits de dirección seleccionan los módulos y los restantes n-m bits seleccionan la palabra deseada dentro del módulo.

PIPELINE LINEAL



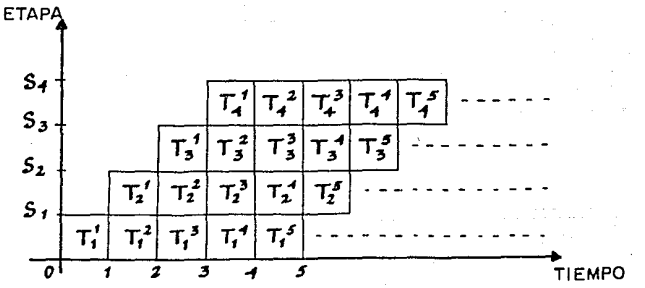
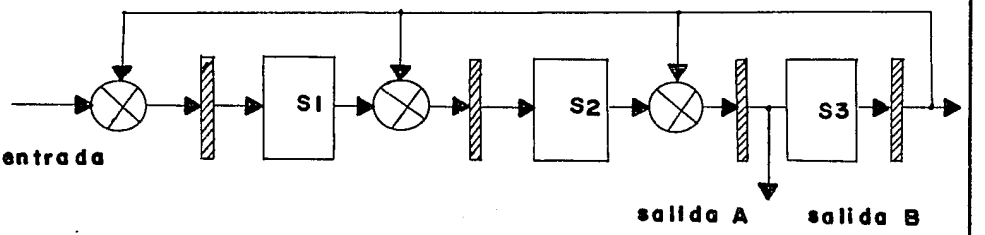


FIG - 4

PIPELINE MULTIFUNCIONAL.



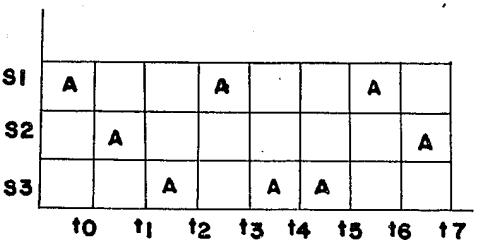


TABLA DE RESERVAS PARA FUNCION A

TABLA DE RESERVAS PARA FUNCION B

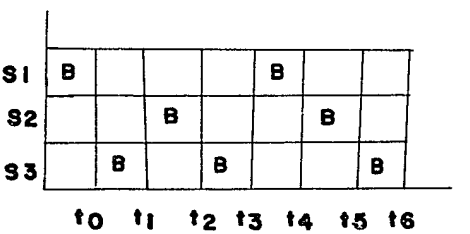
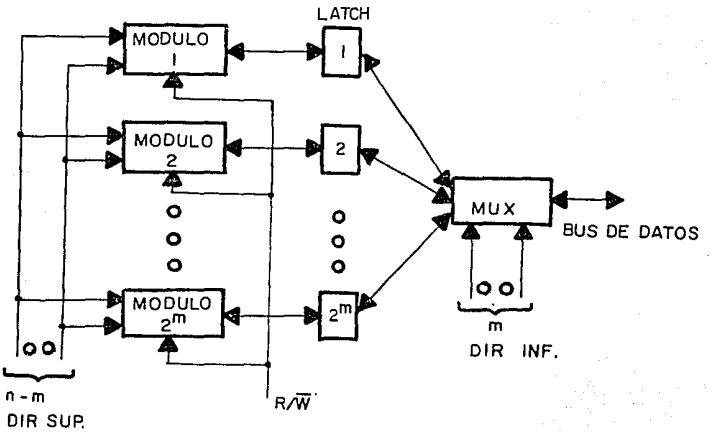


FIG. 5

TIPOS DE MEMORIA PARA PIPELINE

MEMORIA CON ACCESO SECUENCIAL



MEMORIA CON ACCESO CONCURRENTE

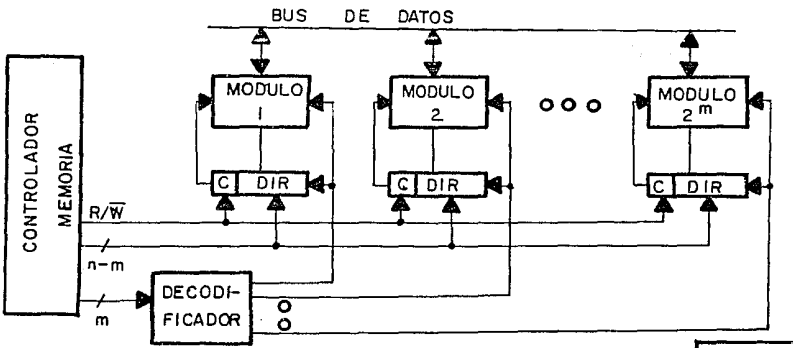


FIG . 6

No Has Hojs
13, 14.

4. ARITMETICA DIGITAL.

Las instrucciones aritméticas en los computadores digitales que se basan en la instrucciones primarias - suma, resta, multiplicación y división -, son llevadas a cabo en dos tipos de aritmética: Punto-fijo y Punto-flotante.

4.1 ARITMETICA DE PUNTO FIJO.

Se considera un número representado en punto-fijo, cuando el punto binario que separa la parte entera de la parte fraccionaria permanece fijo. La parte derecha corresponde a la parte fraccional y la izquierda a la parte entera. El bit de signo es el bit más a la izquierda y es cero (0) si el número es positivo y uno (1) si es negativo.

Para números negativos [4] existen tres tipos de representaciones: Signo y Magnitud, Complemento a Uno y Complemento a Dos. El rango de enteros asociados a cada una de estas representaciones, está determinado por la longitud de la palabra, esto es, en el número de bits (n). Si el entero está fuera de este rango, ocurrirá un *overflow* si el entero positivo excede su límite superior y ocurrirá *underflow* si el entero negativo excede su límite inferior. Según la representación, estos rangos son,

Representación	límite inferior	Límite superior
Signo y Magnitud	$-(2^{n-1}-1)$ (111...1)	$2^{n-1}-1$ (011...1)
Complemento a Uno	$-(2^{n-1}-1)$ (100...0)	$2^{n-1}-1$ (011...1)
Complemento a Dos	-2^{n-1} (100...0)	$2^{n-1}-1$ (011...1)

Este tipo de aritmética, son los que manejan los procesadores generales como los de una computadora personal. Por ejemplo, el 8086/8088 tiene las siguientes instrucciones aritméticas [5],[6]:

1. Suma de números signados y no-signados de 16 bits,
   AX : Primer número (número máximo = 9999)
   BX : Segundo número
   DX,CX : Resultado, en DX el más signif. y en CX el menos sign.

```
SUMA : ADD AX,BX
       MOV CX,AX
       MOV DX,0H
```

2. Resta de números signados y no-signados de 16 bits,
   AX : Minuendo
   BX : Sustraendo
   DX,CX : Resultado

```
RESTA : SUB AX,BX
        CWD
        MOV CX,AX
```

3. Multiplicación signada, usa instrucción : IMUL
   Multiplicación no-signada, usa instrucción : MUL

   AX : Multiplicando
   BX : Miltiplicador
   DX,CX : Producto

```
MULT : IMUL BX
       MOV  CX,AX
```

4. División signada, usa instrucción : IDIV
   División no-signada, usa instrucción : DIV

   DX,AX : Dividendo. En DX el más significativo
   BX : Divisor
   AX : Residuo
   CX : Cociente

```
DIVIS : MOV DX,0
        IDIV BX
        MOV CX,AX
        MOV AX,DX
        MOV DX,0
        MOV BX,0
```

Los tiempos de ejecución (en número de ciclos - un ciclo=210nsg) de estas instrucciones, se dán a continuación [7]:

1. ADD / ADC / SUB / SBB	ciclos
registro-registro	3
registro-memoria	9 a 13
memoria-registro	16 a 24
registro-inmediato	4
memoria-inmediato	17 a 25
acumulador-inmediato	4

2. MUL : Multiplición no-signada	ciclos
registro de 8 bits	70 a 77
registro de 16 bits	118 a 133
memoria de 8 bits	76 a 83
memoria de 16 bits	128 a 143

3. IMUL : Multiplicación signada	ciclos
registro de 8 bits	80 a 98
registro de 16 bits	128 a 154
memoria de 8 bits	86 a 104
memoria de 16 bits	138 a 164

4. DIV : División no-signada	ciclos
registro de 8 bits	80 a 90
registro de 16 bits	144 a 162
memoria de 8 bits	86 a 96
memoria de 16 bits	144 a 172

5. IDIV: División signada	ciclos
registro de 8 bits	101 a 112
registro de 16 bits	165 a 185
memoria de 8 bits	107 a 118
memoria de 16 bits	175 a 194

Como se observa, la ejecución de una instrucción aritmética por un procesador de propósito general es muy lenta y es así cómo para el cálculo numérico se han venido desarrollando procesadores dedicados como los procesadores matemáticos.

4.2 ARITMETICA DE PUNTO FLOTANTE.

Un computador con palabra de 32 bits que trabaje en aritmética

de punto-fijo, puede manejar un rango de enteros restringido a $\pm(2^{31} - 1) = \pm 10^{11}$. Este rango es inadecuado para aplicaciones científicas y de ingeniería; para ello se usa la aritmética de punto-flotante que tiene un rango de enteros mucho más amplio.

La representación en punto-flotante [4] tiene dos componentes: la mantisa (m) y el exponente (e) los cuales a su vez están representados en punto-fijo. Su representación, es

$$f = (m, e) = m \times 2^{e}$$

La mantisa dá la precisión de los operandos y el exponente el rango de los números a representar.

Un número en punto-flotante está *normalizado*, cuando el bit más significativo de la mantisa es diferente de cero. Normalizar un número requiere correr la mantisa a la izquierda e ir decrementando el exponente hasta que el bit más significativo sea diferente de cero. En aritmética normalizada, todos los números en punto flotante deben ser prenormalizados antes de ser manipulados. De ahí que cada cálculo intermedio debe ser pos-normalizado para corresponder a la forma normalizada. La mantisa entonces, debe estar en el rango,

$$0.5 \leq | M | \leq 1.0$$

El exponente puede estar polarizado (biased) o no-polarizado (unbiased). Se polariza sumándole una constante (q) de tal forma que el exponente siempre sea positivo. Esta cantidad a sumar es la más grande negativa que se pueda representar. El exponente estará entre los siguientes rangos :

$$- 2^{q-1} \leq e_{polarizd} \leq 2^{q-1} - 1$$

$$0 \leq e_{no-pol} \leq 2^{q} - 1$$

Las operaciones en punto flotante tienen la siguiente aritmética [4]:

4.2.1 SUMA Y RESTA.

$(m1, e1) = x1 = m1 \cdot 2^{e1}$ (primer operando)
$(m2, e2) = x2 = m2 \cdot 2^{e2}$ (segundo operando)
$(m1,e1) \pm (m2,e2)$ es igual a :

1.) $((m1 \pm m2.2^{-(e1-e2)}), e1)$, si $e1 > e2$

2.) $((m1.2^{-(e2-e1)} \pm m2), e2)$, si $e2 \geq e1$

El punto debe ser alineado antes de realizar la operación. Esto se hace comparando la magnitud relativa de los dos exponentes y corriendo la mantisa con exponente más pequeño $| e1-e2 |$ lugares a la derecha. Se pueden presentar dos tipos de problemas. Uno, cuando la suma de las mantisas es mayor de 1 presentándose *overflow* y el otro, que el resultado de la mantisa sea cero y la del exponente diferente de cero. En el primer caso, el problema se resuelve corriendo la mantisa un lugar a la derecha y simultáneamente incrementando el exponente en 1. Para el segundo caso, no hay solución, la posnormalización no será posible y una señal especial debe generarse para indicar OMZ(order of magnitud zero). Un diagrama en bloques para la realización de un sumador de 32 bits en punto flotante se muestra en la FIG.7.

4.2.2 MULTIPLICACION Y DIVISION.

$(m1,e1) \times (m2,e2) = (m1 \times m2, e1 + e2)$

$(m1,m2) / (m2,e2) = (m1 / m2, e1 - e2)$

La multiplicación y división de las mantisas y la suma y resta de los exponentes pueden ejecutarse simultáneamente. La mantisa resultante estará entre :

$$1/4 \leq |m1 \times m2| < 1 \quad y \quad 1/2 < |m1 / m2| < 2$$

Cuando $1/2 \leq |m1 \times m2| < 1$ no se necesita corrección y el producto ya está *normalizado*, pero si $1/4 < |m1 \times m2| < 1/2$ el producto no está *normalizado* y por tanto se debe normalizar corriendo 1 bit a la izquierda la mantisa y decrementando el exponente en 1.

En la división no se necesita *posnormalización*, pero, para $m1 \geq m2$ hay *overflow* de cociente, esto es, $1 \leq |m1 / m2| < 2$. Se corrige corriendo la mantisa 1 bit a la derecha y aumentando el exponente en 1. Un diagrama en bloques para un multiplicador en punto flotante se muestra en la FIG. 8.

En las computadoras personales, las operaciones en punto-flotante son ejecutadas por el coprocesador matemático. Las copmputadoras que tienen como procesador general el 8088, tienen como coprocesador matemático el 8087. Algunas de las instrucciones aritméticas básicas de este coprocesador y su tiempo de ejecución en número de ciclos, se presenta a continuación :

		ciclos
1. FADD / FSUB :	Suma y resta	
	reales de 32 bits	90 a 120
	enteros de 32 bits	108 a 143
	reales de 64 bits	95 a 125
	enteros de 16 bits	102 a 137

2. FMUL :	Multiplicación	ciclos
	reales de 32 bits	110 a 125
	enteros de 32 bits	130 a 144
	reales de 64 bits	112 a 168
	enteros de 16 bits	124 a 138
3. FDIV :	División	ciclos
	reales de 32 bits	215 a 225
	enteros de 32 bits	230 a 243
	reales de 64 bits	220 a 230
	enteros de 16 bits	224 a 238

Como se puede notar una multiplicación de dos números reales de 32 bits se ejecuta en aproximadamente 120 ciclos, esto es, en un tiempo aproximado de 120 x 210 nseg = 25.2 μseg, tiempo aún muy alto para una aplicación rápida como el desarrollo de un algoritmo para el procesamiento digital de señales.

4.3 CIRCUITOS ARITMETICOS.

Los circuitos aritméticos constituyen los elementos básicos del diseño de una unidad aritmética de un coprocesador. El entendimiento de ellos conducirá a la comprensión de la arquitectura de un microprocesador matemático como un sumador, un multiplicador en aritmética de punto-fijo o punto-flotante.

4.3.1 EL SUMADOR.

El elemento básico de un sumador de n-bits, lo constituye el sumador completo, el cual tienen tres entradas (dos bits como

operandos y un bit de acarreo) y dos salidas (un bit de resultado y un bit de acarreo de salida). El 7480 es un circuito de este tipo. Existen además en el mercado sumadores de 2 bits como el 7482 (adder ripple carry) y de 4 bits como el 74283 (adder carry lookahead) [8].

Mediante estos circuitos un sumador o restador de más bits puede diseñarse como el mostrado en la FIG.9 que implementa un sumador/restador de 8 bits en Complemento a Dos [9]. En este circuito para cada sumador $C = A + \overline{B} + C_I$

si la línea de control RESTA = 0, entonces, $C_I = 0$ y

$$C = A + \overline{B} = A + B \text{(operación suma)}$$

si la línea de control RESTA = 1, entonces, $C_I = 1$ y

$$C = A + \overline{B} + 1 = A - B \quad \text{(operación resta)}$$

Arquitecturas de más bits pueden implementarse con esta estructura en forma bit-slice. Circuitos más complejos como por ejemplo un sumador de punto-flotante, requiere cumplir con el algoritmo expuesto en el numeral (4.2.1). Este tipo de circuito se presenta en el esquema de la FIG.10 con 3 etapas de pipeline. Para este sumador,

$$A = a \times 2^p \qquad B = b \times 2^q$$
$$C = A + B = c \times 2^r = d \times 2^s$$
$$\text{donde} \quad r = \text{máx}\,(p,q)$$

Las operaciones efectuadas en las tres etapas son :

1. En la etapa S1, se comparan los exponentes p y q para precisar el exponente mayor y determinar la diferencia $t = |p - q|$.
2. En la etapa S2 ocurre el desplazamiento t bits a la derecha de la fracción asociada al exponente más pequeño, para igualar los exponentes antes de sumar las fracciones. Además, se suma la

fracción predesplazada con la otra fracción para obtener la suma intermedia c.

3. Recuento del número de ceros (u) de la fracción c y desplazamiento de ésta u bits a la izquierda para producir la fracción suma normalizada d con el primer bit significativo igual a uno. Actualización del exponente mayor s efectuando la resta s = r - u para obtener el exponente resultante.

4.3.2 EL MULTIPLICADOR.

Existen diferentes métodos o algoritmos para implementar un multiplicador. A continuación se describen algunos de los algoritmos más comunes :

1. Multiplicación con sumas y corrimientos [10]. Comenzando por la derecha y dependiendo del bit del multiplicador se genera el producto de la siguiente forma: Si el bit del multiplicador es cero, se suma el multiplicando al producto parcial el cual en su comienzo arranca con cero; si el siguiente bit es uno, se corre el producto parcial un bit a la izquierda y luego se suma el multiplicando a este producto parcial corrido. Este proceso continúa hasta llegar al bit más significativo del multiplicador. Si los operandos son de n-bits, el producto generado será de 2n bits.

2. Multiplicación en Complemento a Dos con Algoritmo de Booth [11]. El multiplicando y multiplicador es una fracción con signo. En este algoritmo se producen corrimientos, sumas y restas al producto parcial, dependiendo de la comparación consecutiva de los bits del multiplicando comenzando por la derecha. El algoritmo se indica a continuación :

bits del Mdr		operación
0	0	corra producto parcial 1 bit a la derecha
0	1	sume multiplicando y corra el producto 1 bit a la derecha
1	0	reste multiplicando y corra el producto 1 bit a la derecha
1	1	corra producto parcial 1 bit a la derecha.

El bit de referencia inicial del multiplicador es cero. Después de la última posición no hay más corrimientos.

Una implementación de este tipo de algoritmos se encuentra en la referencia [12].

Estos algoritmos de suma y corrimientos, tiene el inconveniente de implementar multiplicadores muy lentos y por tanto, no son apropiados cuando se requiere ejecutar una aplicación que exige una gran velocidad. Para ello, se han desarrollado otros algoritmos que ejecutan una multiplicación en forma paralela aumentando considerablemente la velocidad de procesamiento. Algunos se describen a continuación :

3. Un método de multiplicación dado por Hayes [9], es el de almacenar inicialmente los productos parciales en memorias ROM y luego mediante sumadores completos (full-adders), obtener el producto final. Este método se describe a continuación para una multiplicación de 8 bits :
   multiplicando : $A = a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0$

   multiplicador : $B = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

   producto : $C = c_{15} c_{14} c_{13} c_{12} c_{11} c_{10} c_9 c_8 c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0$

   Los operandos se pueden representar de la siguiente forma :

$$A = 2^4(a_7a_6a_5a_4) + (a_3a_2a_1a_0)$$

$$B = 2^4(b_7b_6b_5b_4) + (b_3b_2b_1b_0)$$

el producto C = A x B, es

$$C=2^8(a_7a_6a_5a_4)(b_7b_6b_5b_4)+2^4(a_7a_6a_5a_4)(b_3b_2b_1b_0)+$$
$$2^4(a_3a_2a_1a_0)(b_7b_6b_5b_4)+ (a_3a_2a_1a_0)(b_3b_2b_1b_0)$$

Esto quiere decir que el producto final está formado por la suma de cuatro productos parciales corridos, que son

$$P = (a_7a_6a_5a_4)(b_7b_6b_5b_4) \qquad Q = (a_7a_6a_5a_4)(b_3b_2b_1b_0)$$

$$R = (a_3a_2a_1a_0)(b_7b_6b_5b_4) \qquad S = (a_3a_2a_1a_0)(b_3b_2b_1b_0)$$

Estos productos parciales se almacenan en ROM de 256x8 bits en donde las direcciones son operandos de 4 bits. Un esquema de este tipo de multiplicador se muestra en la FIG.11.

4. El multiplicador combinacional es otro tipo de multiplicador paralelo llamado también, multiplicador con salvaguarda de acarreo - SSA- (carry save adder) y en el cual el acarreo (carry) se propaga en el proceso de suma de productos parciales [13]. En esta estructura el acarreo de salida de los sumadores no se alimenta a los sumadores del mismo nivel, sinó que son preservados y propagados diagonalmente al próximo nivel. El diagrama del circuito de un arreglo de 4x4 se muestra en la FIG. 12.

5. Multiplicador en Arbol de Wallace. Este tipo de multiplicador se forma interconectando SSA. El principio del Arbol de Wallace es hacer procesamiento paralelo a los productos parciales. De esta forma se reduce el número de pasos de sumas, el cual es proporcional al logaritmo del número de bits del multiplicador

y por supuesto tiene un mínimo retardo de tiempo. El algoritmo para realizar la multiplicación de dos números de cuatro bits es [3],

$$\begin{array}{cccccccc} & & & & a_3a_2a_1a_0 & & & =A \\ & & & & b_3b_2b_1b_0 & & & =B \\ \hline & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & =W0 \\ & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & =W1 \\ & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & =W2 \\ a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & & =W3 \\ \hline P7 & P6 & P5 & P4 & P3 & P2 & P1 & P0 = P=AxB \end{array}$$

Los productos parciales Wi se generan mediante una matriz de puertas AND (ai.bj). Estos productos parciales pasan a través de SSA hasta llegar a la última etapa donde se encuentra un SPA (sumador con propagación de acarreo) para generar el producto final. En el esquema de la FIG.13 se muestra un multiplicador en Arbol de Wallace de dos operandos de cuatro bits y cuatro etapas de pipeline.

6. Multiplicador Iterativo de Arbol [14]. Es un multiplicador de Arbol de Wallace modificado que reduce la latencia y el hardware de una estructura convencional de arreglos de SSA. En el método iterativo los SSA son más utilizados y se requiere de un número menor de SSA para generar el producto. El elemento básico es un sumador 4:2 (cuatro entradas y dos salidas) en lugar de uno de 3:2 (tres entradas y dos salidas). Este sumador 4:2, se implanta con dos SSA como se indica en la FIG.14. Por cada nivel de pipeline existen dos SSA en lugar de, uno reduciendo de esta forma la frecuencia del reloj. En esta misma

figura se muestra un multiplicador de cuatro entradas iterativo que acumula productos parciales en un acumulador con salvaguarda de acarreo (ASA). Este acumulador es un sumador 4:2 con dos entradas usadas para acumular salidas previas. El ASA es mucho más rápido que un SPA y requiere solamente de una etapa adicional de encauzamiento.

SUMADOR DE PUNTO FLOTANTE DE 32 BITS

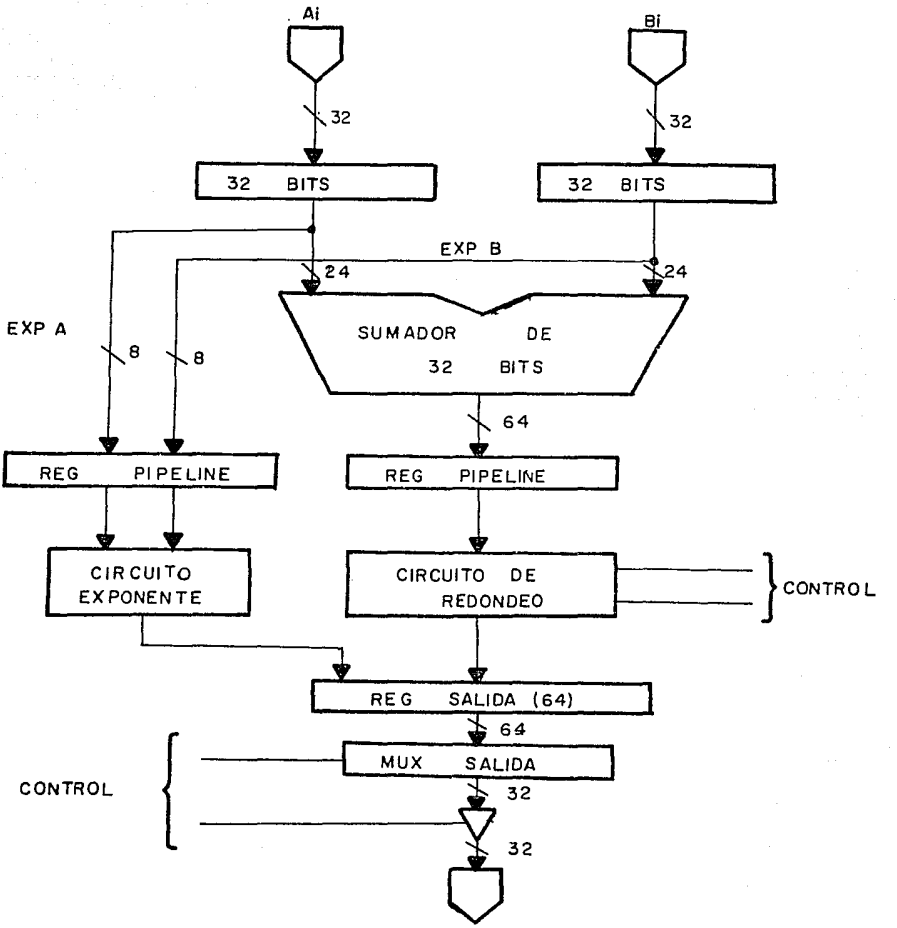


FIG. 7

MULTIPLICADOR DE PUNTO FLOTANTE DE 32 BITS

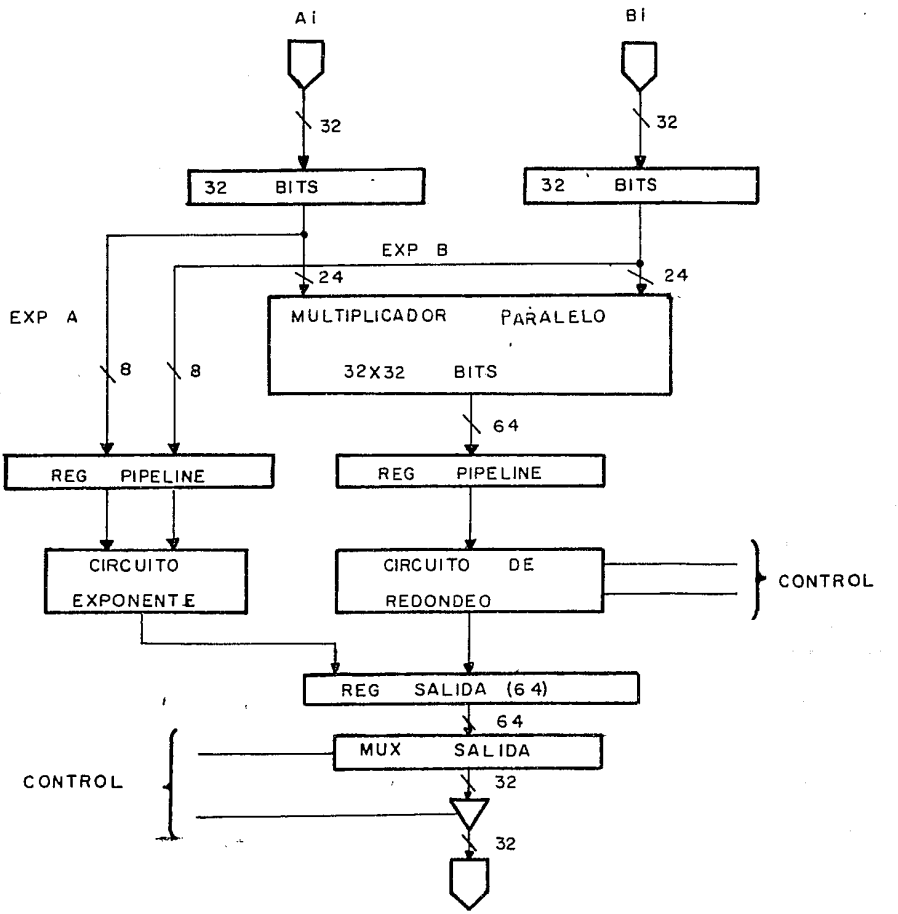


FIG. 8

SUMADOR / RESTADOR EN COMPLEMENTO A DOS

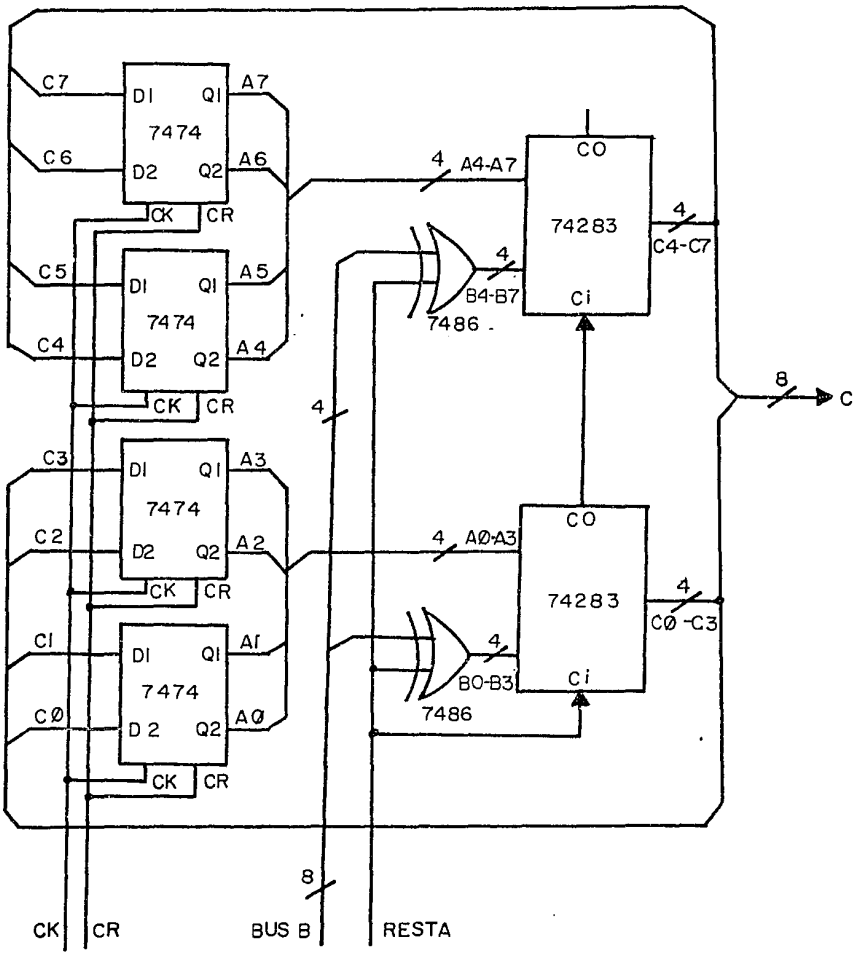


FIG. 9

SUMADOR DE PUNTO FLOTANTE

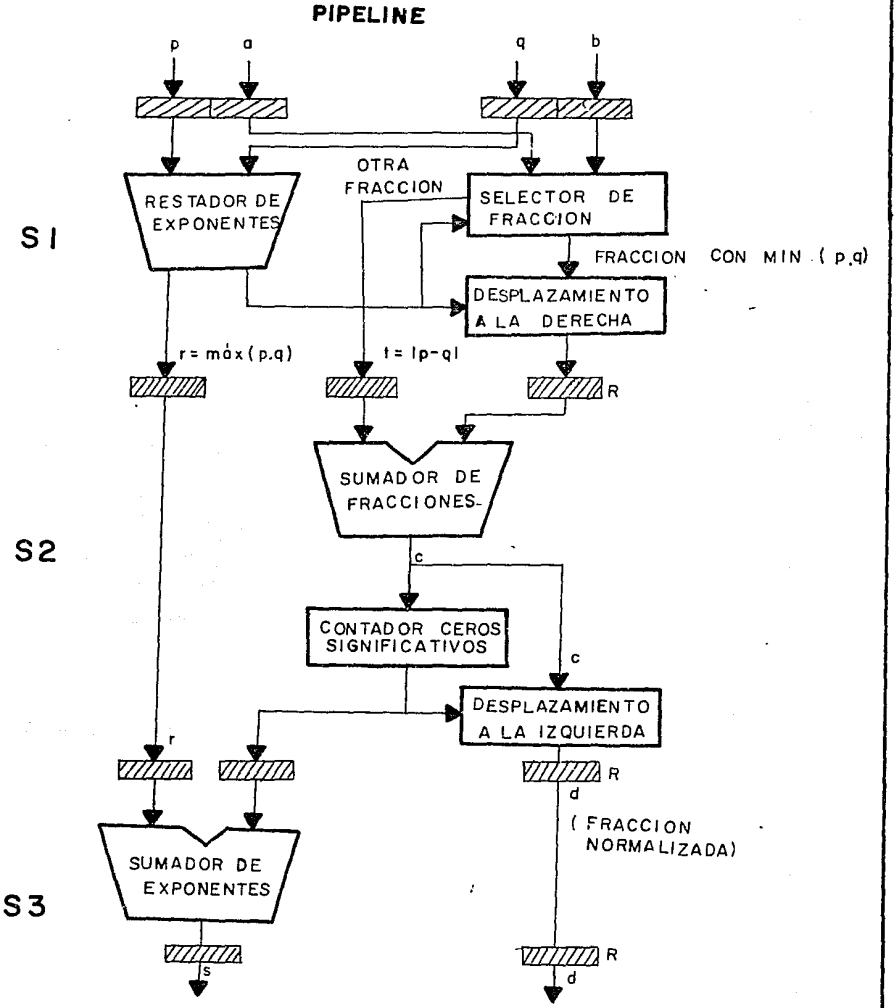


FIG. 10

MULTIPLICADOR PARALELO CON ROM

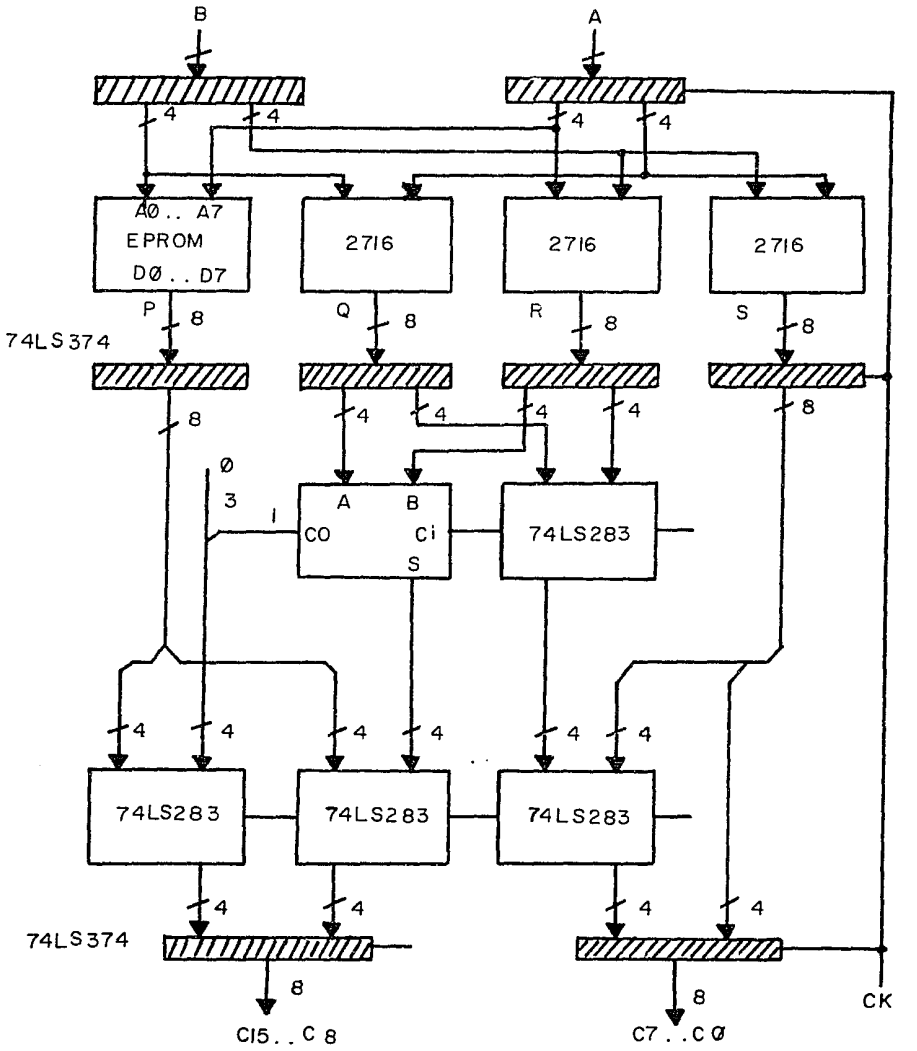


FIG . 11

MULTIPLICADOR EN ARREGLOS S.S.A.

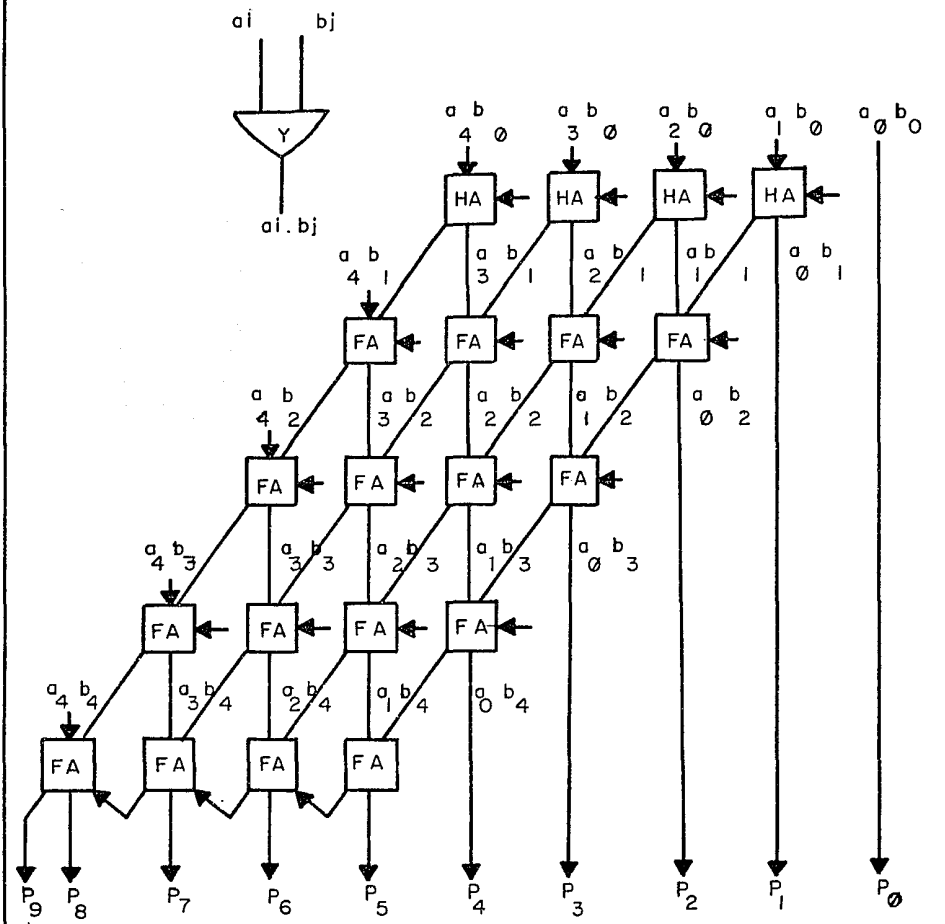
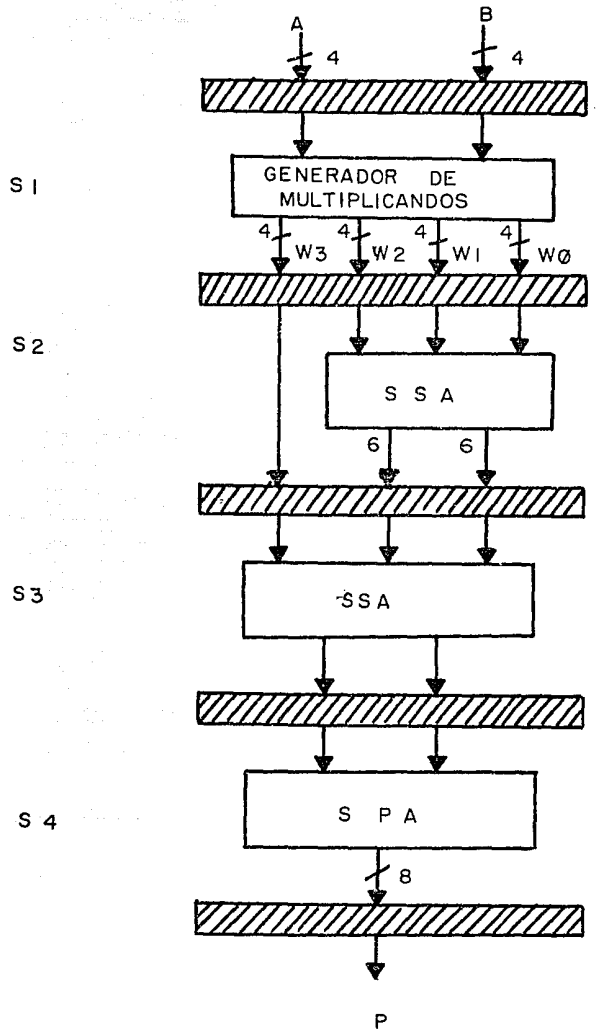
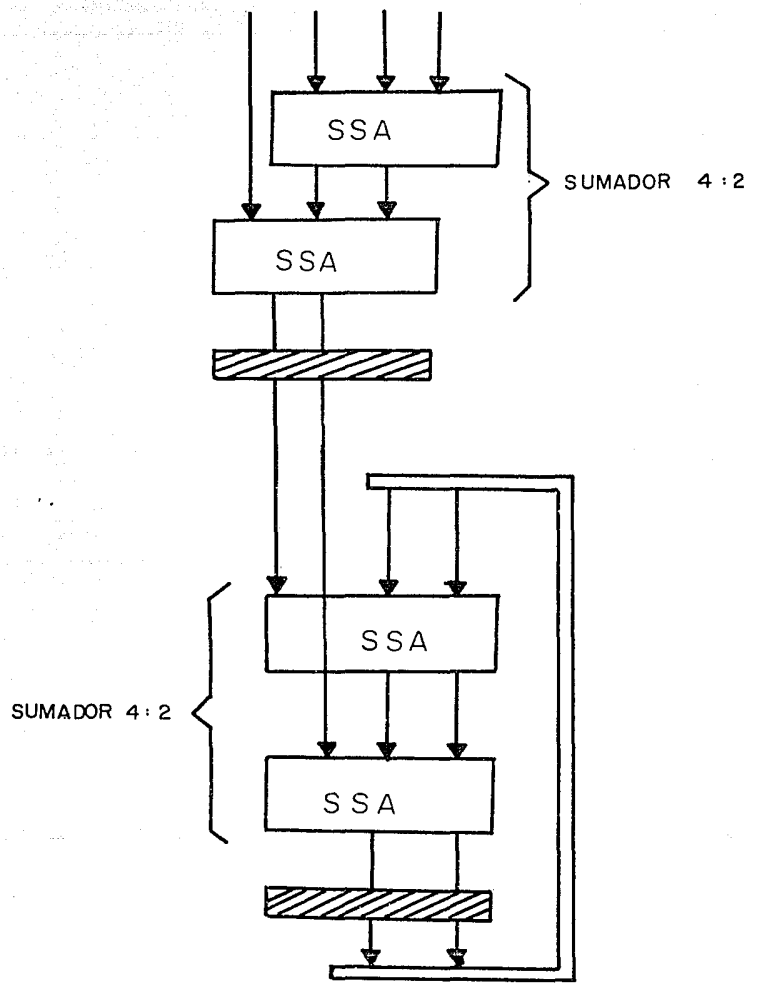


FIG. 12



MULTIPLICADOR EN ARBOL DE WALLACE

FIG. 13



WALLACE CON SUMADORES 4:2

FIG. 14

5. COPROCESADOR MATEMATICO.

En algunos casos un coprocesador matemático está diseñado como un aparato de entrada/salida de propósito especial que interactúa con un procesador central comunicándose mediante métodos estandar vía E/S programado o interrupciones [9]. En otros casos, se acopla al procesador central como una extensión de la unidad de ejecución en donde los operandos e instrucciones aritméticas especiales son ejecutadas por el coprocesador.

En general un coprocesador matemático tiene las siguientes características:

1. Tiene un conjunto de instrucciones más pequeño y especializado.
2. No es capaz de buscar (fetch) sus propias instrucciones y operandos.
3. Transfiere resultados vía bus bidireccional de datos.
4. Está acoplado al procesador central mediante los tres buses: datos, direcciones y control.
5. Además de su unidad de control tiene un *stack* para el almacenamiento de los operandos, una ALU (unidad aritmética-lógica) de ejecución de instrucciones, una interface, un registro de estado (status), un registro de comando y registros de trabajo.

Para entender el proceso de comunicación entre un procesador central (host) y un coprocesador, se analizará este *handshake* entre el 8085 (host)y el AMD9511 (coprocesador) y entre el 8088 (host) y el 8087 (coprocesador).

5.1 EL AMD9511.

Es un coprocesador matemático de 16 bits que se comunica con el procesador central de la siguiente forma:

1. El procesador central (8085) envía los operandos al 9511 para la ejecución de una instrucción aritmética en particular.
2. El 8085 envía el código de la instrucción al 9511, éste la decodifica y luego la ejecuta.
3. Cuando el 9511 ha ejecutado la instrucción aviza al 8085 de su finalización.
4. El 8085 toma el resultado del 9511 ejecutando una o más instrucciones de entrada.

Hay tres métodos para que el 8085 se informe de la ejecución de una instrucción:

1. A través del servicio de interrupción SVERQ.
2. Leyendo el 8085 por *polling* el registro de estado del 9511 para saber si está ocupado o nó.
3. Utilizando la línea $\overline{\text{PAUSE}}$ conectándola a la línea READY del 8085. Si ésta línea está activa, el procesador debe esperar hasta que el 9511 ejecute la instrucción.

Como características importantes de este coprocesador se tienen:

1. Instrucciones aritméticas de punto-fijo y de punto-flotante de 16 y 32 bits.
2. Instrucciones de conversión de punto-fijo a punto-flotante y viceversa.
3. Formato de punto-flotante no-estandar: fracción de 24 bits, exponente de 7 bits y un bit de signo.

4. Registro de estado que informa sobre overflow, underflow, cero, carry y otras banderas del resultado de una operación.
5. ALU y stack de 16 bits y 8 niveles.

En la FIG. 15, se muestra la interface del 8085 con el 9511. Como se observa, las líneas de bus de datos, reloj y señales de lectura y escritura, son conectadas directamente a las mismas líneas del 8085. Dos direcciones del mapa de direcciones de E/S (o de memoria) del 8085 son asignadas al 9511. Una para accesar al registro de comandos (escritura) y al registro de estado (lectura) y la otra para el stack de datos. La línea de control $C/\overline{D}$ (comando/dato) es la encargada de distinguir estas dos direcciones. Estando el coprocesador activado ($\overline{CS}$ = 0), las señales de interface son:

$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	operación
0	1	0	el operando del bus de datos pasa al stack
0	0	1	el operando del stack pasa al bus de datos
1	1	0	se transfiere comando del bus de datos al registro de comando
1	0	1	se transfiere reg. de estado al bus de datos

Estas señales de interface deben ser sincronizadas con las de dirección y control generadas por el 8085 en la transferencia de datos durante las instrucciones de entrada/salida.

5.2 EL INTEL 8087.

Difiere al 9511 en la forma como es acoplado a su procesador

central (8088). Esta interconexión 8088 - 8087 se muestra en la FIG.15. En ella, el bus de datos, de dirección, las líneas de status (S_0, S_1, S_2) y las líneas de cola de status (QS_0, QS_1) van conectada directamente del 8088 al 8087.

La señal BUSY del 8087 es conectada a la entrada $\overline{TEST}$ del 8088. Si el 8087 está realizando un cálculo, le informará al 8088 mediante una instrucción WAIT que espere hasta que la línea $\overline{TEST}$ se ponga en activo bajo. Un nivel bajo en BUSY del 8087 indica que el 8087 ha terminado el cálculo.

Los códigos de operación del 8087 son puestos en memoria al igual que los códigos del 8088. Como el 8088 toma las instrucciones de memoria y las pone en su cola, el 8087 también toma estas instrucciones y las pone en su cola interna. El 8087 decodifica cada instrucción de la cola; si encuentra que es una instrucción del 8088 la trata como un NOP. De otro lado, cuando el 8088 decodifica una instrucción de su cola y encuentra que es del 8087 la trata como un NOP. Osea, ambos procesadores decodifican todas las instrucciones, pero solamente ejecutan sus propias instrucciones. Las instrucciones del 8087 se distinguen de las del 8088, porque todas tienen 11011 como bits más significativos de su primer byte de código.

El compartimiento del bus en la transferencia de datos entre memoria y procesador o coprocesador, se lleva a cabo activando la señal $\overline{RQ}$/GT0 (8088) y $\overline{RQ}$/GT1 (8087) que se conectan directamente y son líneas bidireccionales. La transferencia de datos es un tipo de DMA entre procesadores y memoria.

La sincronización entre el procesador host y el coprocesador para que el 8088 no ejecute una instrucción antes de terminarse la ejecución de una instrucción del 8087, se lleva a cabo mediante la

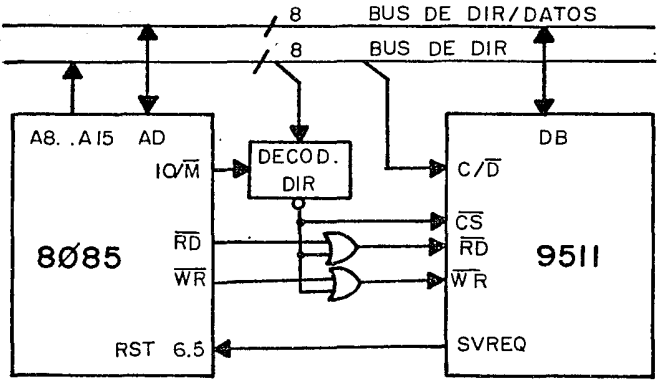
conexión de las líneas BUSY y $\overline{TEST}$ y la adición de la instrucción WAIT en el programa. Siempre que el 8087 esté ejecutando una instrucción, su línea BUSY se pone baja. Puesto que la línea BUSY está conectada directamente a la línea $\overline{TEST}$ del 8088, el host puede checar este pin para ver si el 8087 está ejecutando una instrucción. El uso de la instrucción WAIT hace que el 8088 entre en un loop interno donde repetidamente checa el nivel lógico de la entrada $\overline{TEST}$. Cuando se pone en activo bajo, indica que el 8087 ha completado la ejecución de la instrucción y el 8088 sale del loop interno y ejecuta su próxima instrucción.

El 8087 trabaja en aritmética de punto-fijo y en punto-flotante. En punto-fijo opera con enteros de 16, 32 y 64 bits. El bit más significativo es el de signo y los números negativos operan en complemento a dos. En punto-flotante se tienen representaciones de 32 bits (simple-precisión), 64 bits (doble-precisión) y 80 bits. Usa formato estandar IEEE-754.

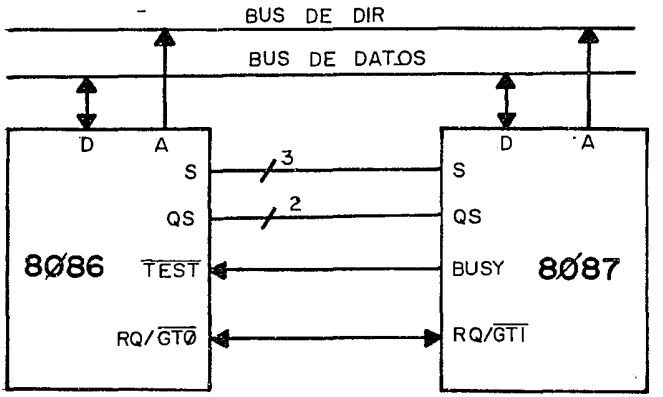
Con base en la forma como operan y se comunican dos procesadores vistos en los dos casos anteriores (8085-9511 y 8088-8087) y la forma de enlace entre dos chips en multiprocesamiento dado en [16], se perfila cómo debe ser la comunicación entre el CPU de una PC-XT y el coprocesador pipeline que ejecutará la suma de productos a diseñar en este proyecto. La comunicación entre estos dos procesadores se esquematiza en la FIG. 16 en forma de multiprocesamiento débilmente acoplado.

El coprocesador tiene una memoria local (M.L) que es un espejo de parte del espacio de memoria de la PC-XT. El procesador central envía un código de comando al coprocesador vía puerto de escritura para que comience a ejecutar la tarea asignada. Una vez terminada la tarea, mediante la lectura del registro de estado del coprocesador, el procesador central se entera de la finalización,

lee el resultado y toma nuevamente el control ejecutando sus propias instrucciones para terminar con la ejecución de una aplicación en particular.

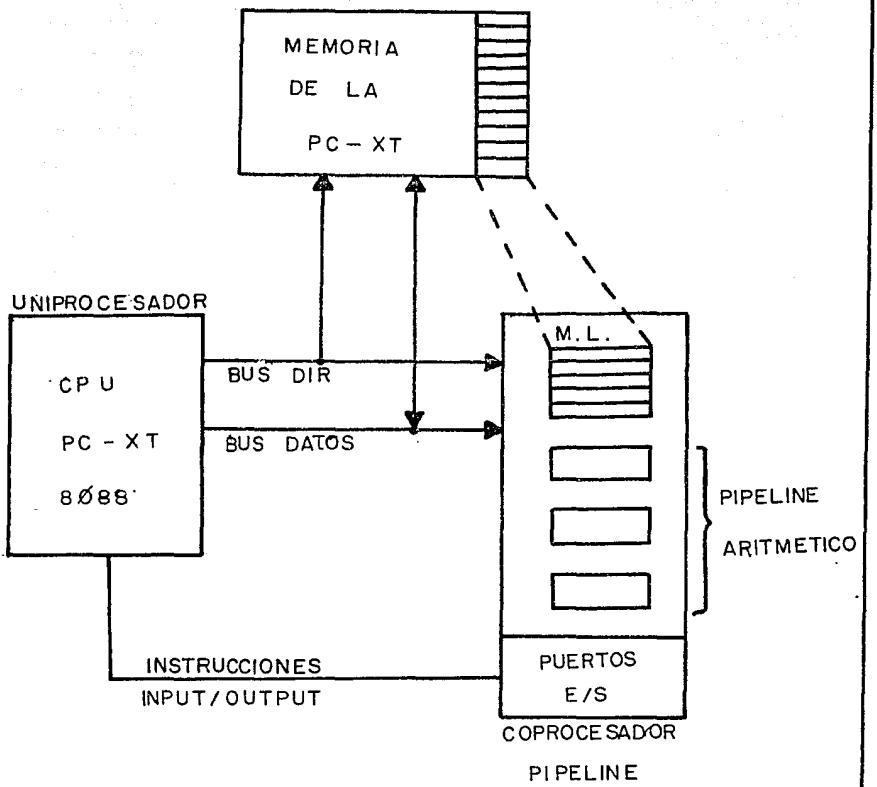


COMUNICACION ENTRE EL 8085 Y EL AMD 9511



COMUNICACION ENTRE EL 8088 Y EL 8087

FIG. 15



COMUNICACION ENTRE CPU DE PC-XT Y COPROCESADOR PIPELINE DE SUMA DE PRODUCTOS

FIG. 16

6. ARQUITECTURA GENERAL.

6.1. ARQUITECTURAS DE PROCESADORES DIGITAL DE SEÑALES.

Teniendo en cuenta que el procesador tendrá aplicaciones para el desarrollo de algoritmos en procesamiento digital de señales, la arquitectura a implementar considerará algunas sugerencias dadas por Stanley [15] en el diseño de arquitecturas para DSP (Digital Signal Processing). Estas recomendaciones son:

1. Configuración de multibus. Esto es, independencia entre los buses de datos, de dirección y de control,
2. Memoria local para el almacenamiento de datos y valores de los coeficientes,
3. Memoria ROM para almacenar el microprograma a ejecutar,
4. Capacidad de realizar una suma de productos en un solo ciclo, que incluya el multiplicador y el acumulador.

Basado en estas cuatro recomendaciones se han diseñado los DSP más comerciales como los de la familia TMS32010 [16] de la Texas Instruments y el 56000/1 de Motorola [17].

El TMS32010 tiene una arquitectura (FIG. 17) tipo Harvard modificado en el que las memorias de programa y de datos se encuentran en espacios distintos, con lo cual los procesos de búsqueda y ejecución de una instrucción suceden simultáneamente, incrementando así su velocidad de procesamiento. Como partes relievantes de su arquitectura se pueden describir :

- Acumulador (ACC) de 32 bits que almacena los resultados de una operación,
- Unidad aritmética lógica (ALU) con entradas de dos puertos de 32 bits,
- Bus de datos de 16 bits,
- Bus de programa de 16 bits para instrucciones,
- Memoria de datos RAM de 144 x 16 bits,
- Memoria de programa ROM de 1536 x 16 bits,
- Multiplicador paralelo de 16 x 16 bits con resultado de 32 bits,
- Contador de programa (PC) de 12 bits que contiene la dirección de la siguiente instrucción a ejecutar,
- Registro de corrimiento de 32 bits para hacer corrimientos a la izquierda de 0 a 15 bits,
- Registro de estado (ST) de 5 bits que indica el estado del microprocesador,
- Un pila para el anidamiento de subrutinas de 4 niveles,
- Dos MUX para comunicación externa de datos y direcciones con periféricos.

El ciclo de instrucción de éste microprocesador es de 200 nsg.

La multiplicación se realiza en tres ciclos de instrucción, esto es, 600 nsg, pero puede ser reducido a 400 nsg cuando se efectúan multiplicaciones y acumulaciones en forma sucesiva.

La implementación de un filtro FIR [20] de longitud-5

$$y[n] = \sum_{K=0}^{4} x[n-k].h[k]$$

en la Forma Directa requiere el siguiente segmento de código del TMS32010 suponiendo que los coeficientes del filtro h[k] y los valores previos x[n-k] están almacenados en la memoria del microprocesador.

```
NXTP    IN   XN,PA2      * Tome el valor de entrada x[n] *
        ZAC              * Borre acumulador *
        LT   XNM4        * Cargue x[n-4] *
        MPY  H4          * x[n-4].h[4] *
        LTD  XNM3        * Cargue x[n-3] y acumule *
        MPY  H3          * x[n-4].h[4] + x[n-3].h[3] *
        LTD  XNM2        * Cargue x[n-2] y acumule *
        MPY  H2          * x[n-4].h[4] + ...+ x[n-2].h[2] *
        LTD  XNM1        * Cargue x[n-1] y acumule *
        MPY  H1          * x[n-4].h[4] + ...+ x[n-1].h[1] *
        LTD  XN          * Cargue x[n] y acumule *
        MPY  H0          * x[n-4].h[4] + ... + x[n].h[0] *
        APAC             * Deje resultado en el acumul. *
        SACH YN,1        * Almacene el resultado en YN *
        OUT  YN,PA2      * Saque resultado a puerto *
        *
        B    NXPT        * Salte a próximo punto *
```

Se observa en el anterior segmento de programa que para realizar las cinco sumas de productos de 16 bits se requieren 10 instrucciones - de la LT XNM4 a MPY H0 - ,osea, un tiempo de 200nsg x 10 = 2 μsg, sin tener en cuenta el overhead que implica llamar cada instrucción.

El DSP 56000/1 (FIG. 18) tiene como unidades importantes en su arquitectura las siguientes.

- ROM para memoria de programa de 512 x 24 bits,
- Dos espacios para memoria de datos X, Y de 256 x 24 bits cada una. Además la X tiene grabada la tabla de compresión - Ley μ - y la memoria Y la tabla del seno,
- Tiene una ROM adicional de 32 x 24 bits para cargar el programa de un periférico o memoria externa, rutina que se carga a través del puerto de huésped,

- Tiene una ALU de direcciones para encontrar la dirección final cuando se usan los módulos
- Un multiplicador-acumulador (MAC) de 24 x 24 bits con resultado de 56 bits,
- Dos acumuladores de 56 bits,
- Cuatro buses internos de 24 bits para datos de X, de Y, de programa y un bus global,
- Puertos I/O de 24 bits,
- ALU de datos,
- Tres buses de direcciones XA, YA, PA de 16 bits. El controlador de bus envía uno de estos buses hacia afuera.

La velocidad de procesamiento es de 10.25 MIPS (millones de instrucciones por segundo), esto es, un tiempo de ciclo aproximado de 100 nsg. Comparado éste microprocesador con el TMS32010 se nota que es doblemente veloz, además de tener la ventaja de ejecutar instrucciones en paralelo como las siguientes :

- Traer la instrucción siguiente,
- Multiplicar datos de 24 x 24 bits,
- Sumar datos de 56 bits,
- Hacer movimientos de datos entre la X y la Y,
- Actualizar dos de los apuntadores de direcciones, pudiéndose usar tres tipos de aritmética (lineal, módulo e invertir el orden de los bits).

La implementación de un filtro IIR (de respuesta impulsional infinita) tipo Butterworth de segundo orden con el DSP56000 se presenta a continuación [21], en donde básicamente se implementa la ecuación,

$$y[n] = \sum_{K=0}^{4} b_k . \; x[n-k] + \sum_{K=0}^{4} a_k . \; y[n-k]$$

El programa se almacena en la memoria P, los valores previos de x[n-k] y de y[n-k] en la memoria X y los coeficientes b_k, a_k en la memoria Y :

```
DATIN     EQU   $100          ; Direc. datos de entr.enX
DATOUT    EQU   $100          ; Direc. datos de sal. en Y
NPTS      EQU   1024          ; Número de puntos
PREV_X    EQU   5             ; Valores previos de x[n]
PREV_Y    EQU   4             ; Valores previos de y[n]
TOTAL     EQU   9             ; Número de coeficientes
SCALE     EQU   3             ; Factor de escala
          ORG   X:$0          ; Direc. entradas previas
STATE_X   DSM   PREV_X        ; Define módulo para x[n-k]
          ORG   X:$10         ; Direc. salidas previas
STATE_Y   DSM   PREV_Y        ;Define módulo para y[n-k]
          ORG   Y:$0          ; Direc. para coeficientes
COEF      DC    b0,b1,b2,b3,b4,a1,a2,a3,a4
          ORG   Y:$0          ; Direc. memoria P
START
          MOVE  #0,X1         ; Borra registro X1
          MOVE  #DATIN,R2; Apuntador de entrada
          MOVE  #DATOUT,R5    ; Apuntador de salida
          MOVE  #STATE_X,R0   ; Apuntador de x[n-k]
          MOVE  #PREV_X-1,M0  ; Módulo para x[n-k]
          MOVE  #COEF,R4      ; Apuntador de coefic.
          MOVE  #TOTAL-1,M4   ; Módulo para coef.
          MOVE  #STATE_Y,R1   ; Apuntador de y[n-k]
          MOVE  #PREV_Y-1,M1  ; Módulo para y[n-k]
          REP   #PREV_X       ; Borra entradas previas
          MOVE  X1,X:(R0)+
          REP   #PREV_Y       ; Borra salidas previas
          MOVE  X1,X:(R1)+
          DO    #NPTS,_ENDP   ; Una pasada por muestra
          MOVE  X:(R2)+,X0
```

```
        CLR     A                   X0,X:(R0)+      Y:(R4)+,Y0
        REP   #PREV_X-1           ;Términos para entr. prev.
        MAC     X0,Y0,A             X:(R0)+,X0      Y:(R4)+,Y0
        MAC     X0,Y0,A             X:(R1)+,X0      Y:(R4)+,Y0
        REP     #PREV_Y-1           ; términos para sal. prev.
        MAC     X0,Y0,A             X:(R1)+,X0      Y:(R4)+,Y0
        MACR    X0,Y0,A
        CLR     B           ; Borra acumulador B
        REP     #SCALE; Suma A a B  SCALE veces
        ADD     A,B
        MOVE    B,Y:(R5)+           ; Valor del dato de salida
        MOVE    B,X:(R1)            ; Almacena valor del dato
        MOVE    (R0)-
_ENDP
        STOP
        END
```

Se observa claramente que se tienen que repetir la suma de productos, tanto para los términos de x[n-k], como para y[n-k], aunque ya para éste microprocesador se ejecutan otras instrucciones en paralelo con la multiplicación. De todas formas, la operación más importante suma de productos repetitivas se hace por software y además necesita de otras instrucciones para cargar los datos y los coeficientes en las memorias del microprocesador.

6.2 ARQUITECTURA DEL COPROCESADOR.

Con base en el análisis de las arquitecturas de estos dos microprocesadores digitales de señales, se diseñó la arquitectura del coprocesador que se muestra en la FIG. 19.

Este coprocesador trabajará en un multiprocesamiento débilmente acoplado con el procesador central de la computadora -8088- y realizará la sumatoria de productos en forma repetitiva en aritmética de punto flotante y palabra de 32 bits.

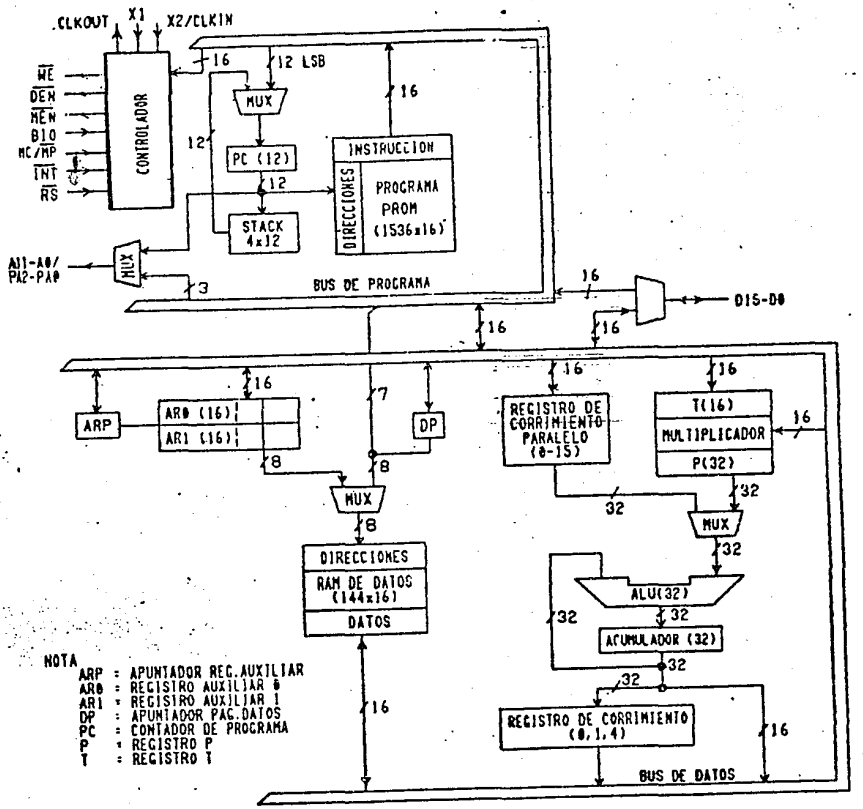
Tiene las siguientes características :

. Dos memorias RAM de 1024 x 32 bits cada una que almacenarán los operandos o vectores a procesar. Las memorias son leídas en forma simultánea por la unidad de control del coprocesador.
. Una unidad aritmética compuesta por un multiplicador - sumador en pipeline que realiza la suma de productos en forma hardware, en punto flotante y 32 bits a una velocidad de coprocesamiento de 100 nsg y con una latencia del pipeline de 480 nsg.
. Una unidad de control compuesta por el secuenciador y el controlador, encargada de leer los datos a ser procesados y controlar la ejecución de la suma de productos. También es la encargada del control de la comunicación entre la CPU de la computadora (HOST) y el coprocesador.
. La interface I/O por intermedio de la cual se establecerá la comunicación entre la computadora y el coprocesador leyendo o escribiendo en los puertos del coprocesador asignados según el espacio disponible de puertos de la PC.
. Un acumulador de 32 bits que será leído como cuatro puertos de 8 bits por parte de la PC par obtener el resultado final de la sumatoria de productos.
. Un registro de comando configurado como puerto de escritura mediante el cual se envía un código al coprocesador para que inicie la ejecución de la suma de productos.
. Un registro de status configurado como puerto de lectura le indica a la CPU de la computadora, el estado del coprocesador necesario para la lectura del resultado final.

El coprocesador se implementará en dos tarjetas. Una de ellas , contendrá la unidad de memoria y la adquisición de datos, y la otra, la unidad aritmética, unidad de control, interface I/O, registro de comando y registro de status.

El coprocesador operará de la siguiente forma :

1. Los operandos o vectores a procesar son cargados en la memoria del coprocesador, siendo esta memoria un espejo del espacio de memoria de la computadora. Esto es, el almacenaje de los operandos es simultáneo tanto en la memoria de la computadora como en la memoria del coprocesador.
2. Cargados los datos en la memoria del coprocesador, la unidad de control hará que los datos sean leídos en forma paralela y secuencialmente llevados a la unidad aritmética para su ejecución.
3. Terminada la ejecución, esto es, realizada la suma de productos, el resultado de 32 bits será leído por la computadora como datos de puerto de 8 bits.
4. Este resultado lo puede seguir procesando la computadora para concluir con una aplicación en particular.



NOTA
ARP : APUNTADOR REG.AUXILIAR
AR0 : REGISTRO AUXILIAR 0
AR1 : REGISTRO AUXILIAR 1
DP : APUNTADOR PAG.DATOS
PC : CONTADOR DE PROGRAMA
P : REGISTRO P
T : REGISTRO T

ARQUITECTURA DEL TMS32ØIØ

FIG. 17

DSP56000/1 BLOCK DIAGRAM

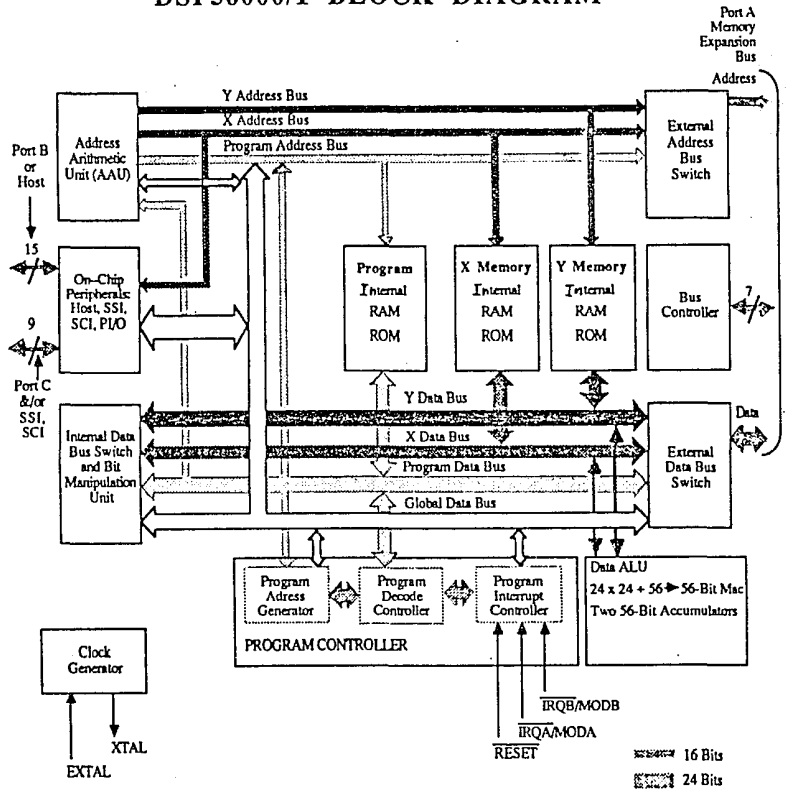
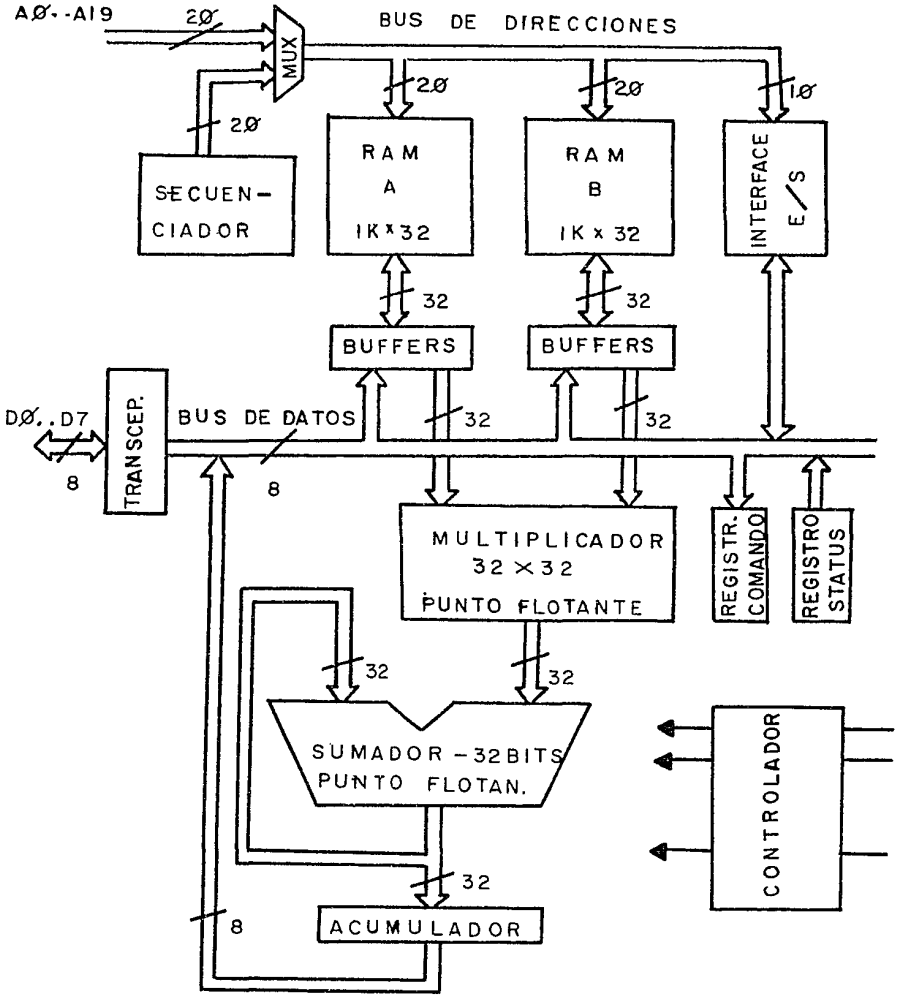


FIG. 18



ARQUITECTURA GENERAL

FIG. 19

7. UNIDAD DE MEMORIA.

7.1 ADQUISICION DE DATOS.

Interrupciones y accesos directos a memorias son dos maneras de mover datos entre periféricos y memoria de una computadora [20]Otra forma es hacer el polling que requiere que la CPU periódicamente pregunte a la tarjeta de adquisición de datos y a todos los otros periféricos sobre la necesidad del servicio del bus. Aunque el polling puede ser rápido exige una complejidad en el sistema.

Con interrupciones, sin embargo, los periféricos necesitan servicio de la CPU. La CPU detiene el programa que está ejecutando y salva sus registros internos. Luego mediante un programa especial atiende la interrupción.

El acceso directo a memoria (DMA) es la manera más rápida de transferir datos de un periférico a una computadora. El periférico envía una solicitud al controlador de DMA, éste avisa al CPU de la petición, cede el bus al controlador y transfiere el dato directamente a memoria. Cuando la transferencia es concluida, el controlador retorna el bus a la CPU. Para la mayoría de aplicaciones de adquisición de datos, tasas de transferencia de 2 MBytes/sg son adecuadas.

El bus de la PC tiene cuatro canales de DMA, de los cuales uno de ellos está reservado al refrescamiento de la memoria dinámica. De los otros, dos son para el floppy y el disco duro, quedando uno disponible para periféricos. Debido a que un canal es usado para

refrescar memoria cada 16 μsg, transferencias de bloques de datos por DMA que tomen largo tiempo, no es posible. Además, el sistema operativo inicializa todos los canales a modo de sólo transferencia, en donde la CPU realiza al menos una instrucción de máquina entre cada transferencia de DMA.

Toda transferencia de DMA, excepto el refresco de memoria, toma 5 pulsos de reloj, osea, 1.05 μsg (5 x 210 nsg). Esto dá teóricamente una tasa de transferencia de 833 KBytes/sg. Ahora, si se tiene en cuenta que se adiciona un ciclo de CPU entre ciclos de DMA, esta tasa de transferencia se reduciría a cerca de 500 KBytes/sg (2 μsg/byte), que es todavía adecuada para un gran rango de aplicaciones.

7.2 ALTERNATIVAS PARA LA TRANSFERENCIA DE DATOS.

Para cumplir con el objetivo de diseño del coprocesador para realizar la suma de productos, se analizaron las siguientes cuatro alternativas de transferencias de datos entre la memoria de la computadora y el coprocesador :

1. Transferir los datos por DMA a la tarjeta y realizar el coprocesamiento en forma secuencial en la medida que lleguen los datos. Como la arquitectura del coprocesador es en pipeline con un tiempo de reloj de 100 nsg, el pipeline del coprocesador no obraría puesto que cada byte llega por DMA a una tasa de 2 μsg como se mencionó anteriormente. Como cada dato es de 4 bytes (32 bits), el par de operandos llegaría cada 16 μsg y el pipeline actuaría como un uniprocesador llevándose a cabo el procesamiento en forma lenta y no cumpliría con el objetivo central del proyecto.

2. Transferir los datos u operandos almacenados en la RAM de la computadora al coprocesador por DMA y almacenarlos en una memoria local del coprocesador para posteriormente procesar la suma de productos de los vectores almacenados. Aunque en este caso, la arquitectura del coprocesador trabaja en pipeline, durante la ejecución existe el problema de que hay una pérdida en tiempo por la demora en el almacenamiento o carga de los datos en la memoria local del coprocesador.

3. Almacenar los datos directamente en la memoria local del coprocesador desde la CPU como un espacio de memoria de la computadora y luego procesarlos. Es una de las mejores opciones, pero tiene el inconveniente desde el punto de vista de las aplicaciones, que no pueden manipularse esta memoria desde la PC para el reordenamiento de los vectores, como es el caso cuando se simula por ejemplo un filtro. Como se mencionó en la descripción de la arquitectura general del coprocesador, la memoria local sólo puede ser leída por la unidad de control del coprocesador.

4. Almacenar los datos directamente en la memoria local del coprocesador, pero a diferencia de la opción anterior, el espacio de memoria utilizado será un espejo de un espacio de la memoria física de la computadora. Esto permitirá manipular los datos en modo de lectura en la memoria de la computadora y escribirlos como un espejo en la memoria local, haciendo el coprocesamiento mucho más rápido y eficiente. Esta alternativa es la escogida para el diseño de la unidad de memoria del coprocesador.

7.3 INTERFASE DE MEMORIA.

La memoria usada por el coprocesador será un espejo de la memoria de la computadora. Esto quiere decir, que la carga de los vectores para realizar la suma de productos se hace simultáneamente en la memoria de la PC y en los bancos de memoria del coprocesador. La carga de estos parámetros se hace por bytes pero deben quedar almacenados de tal forma que para el coprocesamiento de la suma de productos queden formando dos bancos de 32 bits para hacer su posterior lectura en forma paralela. En un banco quedarán los valores de un vector y en el otro los del otro vector.

El diseño de esta interface se muestra en la FIG.20 y 21. En este circuito, el 74LS688 selecciona el espacio de memoria a utilizar en correspondencia con el mapeo de memoria [21] de la computadora (FIG.22) según las posiciones del DIP SWITCH. La salida de este comparador activa al decodificador 74LS138 para seleccionar el módulo de memoria. Cada módulo es decodificado por las líneas de dirección A0, A1, A12. Con A12 = 0 se escoge el banco 1 para almacenar los valores de un vector (A_1) y con A12 = 1 se almacenan los valores del otro vector (B_1). Las líneas A0 Y A1 seleccionan el módulo en particular para cada banco, y las líneas de A2 a A11 son utilizadas para direccionar la locación de memoria dentro de cada módulo.

Para que los datos que vienen de memoria central puedan ser almacenados en la memoria local del coprocesador, es necesario que los buffers bidireccionales estén en modo de entrada. Esto lo hace el controlador activando las líneas de control C3 como se explicará más adelante en la Unidad de Control.

Las puertas AND permiten en la etapa de ejecución de la suma de

productos hacer la carga paralela de los operandos o vectores a la unidad aritmética. Esto se realiza colocando la línea de control C1 = 0 y activando la línea de lectura MR dada por el controlador. La línea de control C2 (para el MUX) permitirá el direccionamiento de la memoria, tanto por la CPU central en la escritura, como por el coprocesador en la lectura de datos. En la FIG.23 y FIG.24 se muestran los tiempos requeridos para la lectura y escritura de la memoria.

La capacidad de la memoria local del coprocesador utilizada para el almacenamiento de los operandos es de 8 KBytes. Debido a que los datos son de 32 bits, cada banco puede almacenar hasta 1000 operandos, que son suficientes para realizar una determinada aplicación de las planteadas en el proyecto.

7.4 SELECCION DEL TIPO DE MEMORIA.

Dos parámetros básicos fueron considerados para la selección del tipo de memoria RAM a utilizar en la conformación de la memoria local del coprocesador : el tiempo de acceso y el consumo de potencia.

En lo referente al tiempo de acceso, se debe tener en cuenta que sea compatible con las etapas pipeline de la unidad aritmética, esto es, un tiempo aproximado de acceso de 100 nsg. Respecto al consumo, se debe tener en cuenta que la memoria no vaya a consumir lo recomendado por slots [21] para una tarjeta que se inserte, osea, aproximadamente una corriente de unos 800 mA. Además se debe tener en cuenta que en la tarjeta en donde se alambrará la memoria, se encontrarán los circuitos correspondientes a los buffers bidireccionales, decodificadores, multiplexers, puertas y

otros circuitos que limitan el consumo para las 8 memorias a aproximadamente 400 mA, esto es, a unos, 50 mA por chip de RAM.

Para la escogencia del tipo de chip para conformar la memoria local, se compararon las RAM tradicionales de la serie 2064 y las 6116 que no son tan tradicionales. A continuación se describen sus caracteríticas más importantes :

RAM estática 2064 :

Tiene las siguientes características [22] :

. Una sola fuente de +5V
. Buffers de salida controlado por línea activa baja
. Salidas en tres estados
. Entradas y salidas compatibles con TTL
. Entradas protegidas contra cargas electrostáticas
. Tiempos de acceso de : 100 nsg (2064-10), 120 nsg (2064-12)
150 nsg (2064-15)
. Consumo de corriente : 80 mA (en operación)
10 mA (en stanby)
. Un solo ciclo de lectura controlado por $\overline{OE}$ y donde $\overline{WE}$ = 1.

Tres ciclos de escritura con $\overline{OE}$ = 1

(1) Controlado por $\overline{WE}$

(2) Controlado por $\overline{CS1}$

(3) Controlado por $\overline{CS2}$

RAM estática 6116 :

Las diferencias más significativas con la memoria anterior [23] son :

. Tiempo de acceso : 120 nsg (6116-2)
150 nsg (6116-3)
200 nsg (6116-4)

. Consumo de corriente : 36 mA (en operación)
20 μA (en stanby)

. Tres ciclos de lectura :

(1) Controlado por $\overline{OE}$ y $\overline{WE}$ = 1 (igual a la 2064)

(2) La selección es continua con $\overline{CS}$ = 0, $\overline{WE}$ = 1, $\overline{OE}$ = 0

(3) Selección con $\overline{CS}$ y $\overline{WE}$ = 1, $\overline{OE}$ = 0

. Dos ciclos de escritura :

(1) Controlado por $\overline{OE}$ y $\overline{WE}$

(2) Controlado por $\overline{WE}$ y $\overline{OE}$ = 0

Como los tiempos de acceso de las dos memorias son aproximadamente iguales, éste factor no es decisivo en la escogencia de uno de los dos tipos de memoria. Tomando el otro parámetro, el consumo, la RAM 6116 consume en operación menos de la mitad de corriente que consume la RAM 2064. La memoria consumiría un total de :

RAM 2064 : 80 mA x 8 = 640 mA
RAM 6116 : 36 mA x 8 = 288 mA

Otra ventaja que ofrece la RAM 6116 además del consumo, es que tiene un ciclo de lectura (2) más simple que la RAM 2064 lo cual permitirá que el diseño de la unidad de control del coprocesador sea más sencillo.

Como conclusión, para el diseño de la unidad de memoria se

seleccionarán las RAM 6116-2 utilizando el ciclo de lectura (2) y el ciclo de escritura (2). Para el ciclo de escritura se utilizarán las señales dadas por el sistema de bus de la computadora - FIG.24 - y para el ciclo de lectura las señales necesarias las enviará el controlador del coprocesador.

En la FIG.25, se dán los diagramas de tiempo de los ciclos de lectura y escritura de la RAM 6116 los cuales tienen la siguiente temporización :

Lectura :

t_{RC}	: tiempo de ciclo de lectura, min	= 120 nsg
t_{AA}	: tiempo de acceso de dirección, max	= 120 nsg
t_{OE}	: salida válida al $\overline{OE}$, max	= 80 nsg
t_{OH}	: t_{HOLD} al cambio de dirección, min	= 10 nsg
t_{ACS}	: tiempo de acceso de $\overline{CS}$, max	= 120 nsg

Escritura :

t_{WC}	: tiempo de ciclo de escritura, min	= 120 nsg
t_{CW}	: selec. del chip fin de escrit.,min	= 70 nsg
t_{AW}	: direc. válida fin de escrit., min	= 105 nsg
t_{AS}	: t_{SET-UP} de direc., min	= 20 nsg
t_{WP}	: ancho del pulso de escritura, min	= 70 nsg
t_{WR}	: tiempo de recuperación escr., min	= 35 nsg
t_{DS}	: t_{SET-UP} del dato, min	= 35 nsg
t_{DH}	: t_{HOLD} del dato, min	= 5 nsg

La sincronización de los ciclos [24] con la temporización dada en el sistema de bus de la computadora es muy importante a tener

en cuenta para que el proceso de escritura de datos en la memoria local del coprocesador se lleve a cabo correctamente. Comparando los tiempos de ciclo de bus de la PC - FIG.24 - , los requeridos por la RAM 6116 en la escritura y los retardos de los circuitos integrados que hacen parte de la unidad de memoria (FIG.20 y 21), se debe cumplir que :

1) $t_{12}-t_D(74LS244)+t_D(74LS157)+t_{13}-2t_D(74LS244)$
$+t_{11}+t_D(74LS244)+t_D(74LS157) > t_{WP}$

reemplazando tiempos en nsg :

$112-12-14+297-2x12+10+12+14 = 395$ nsg > 70 nsg

2) $t_7+ t_D(74LS157) > t_{AS}$

reemplazando tiempos en nsg :

$91.5+14 = 105.5$ nsg > 20 nsg

3) $t_{11}+t_D(74LS244)+t_{13}-t_D(74LS25)-t_D(74LS244) > t_{DS}$

reemplazando tiempos en nsg :

$10+12+297-8-12 = 399$ nsg > 35 nsg

Con lo anterior se demuestra que la temporización para el proceso de escritura en la RAM 6116 se cumple con el diseño propuesto.

Para el ciclo de lectura(2), el proceso es más sencillo. El controlador del coprocesador debe poner la líneas,

$\overline{WE} = 1$, $\overline{CS} = 0$ y $\overline{OE} = 0$

y los latches de entrada de la unidad aritmética deben atrapar este dato para el coprocesamiento correspondiente.

INTERFACE DE LA MEMORIA.

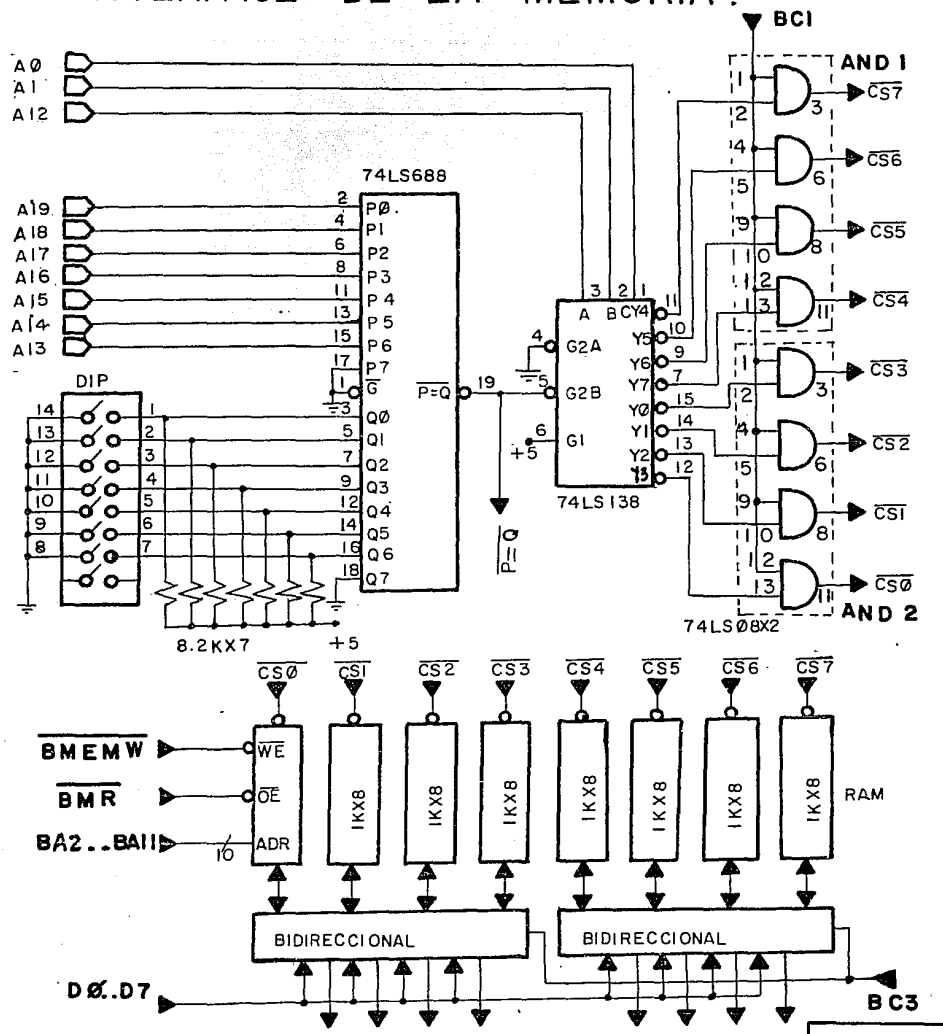


FIG. 20

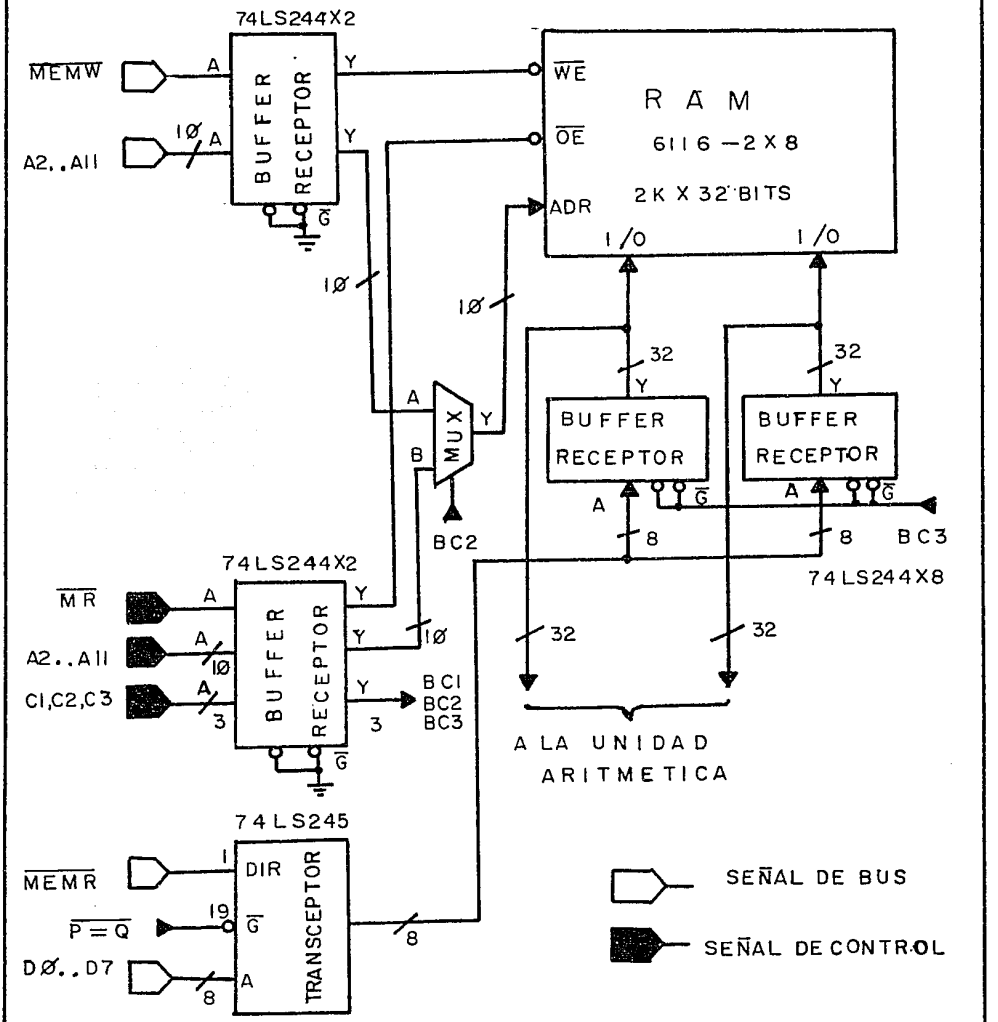
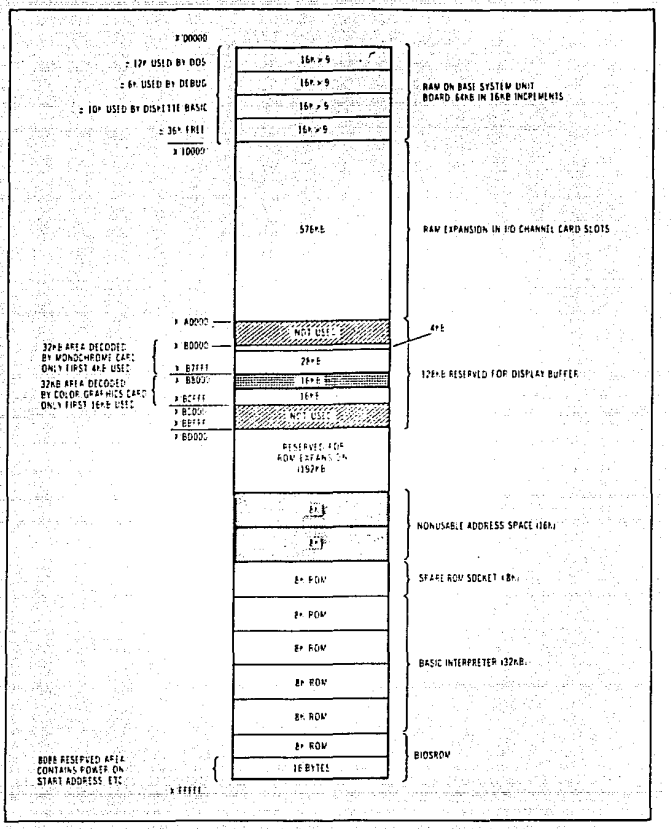
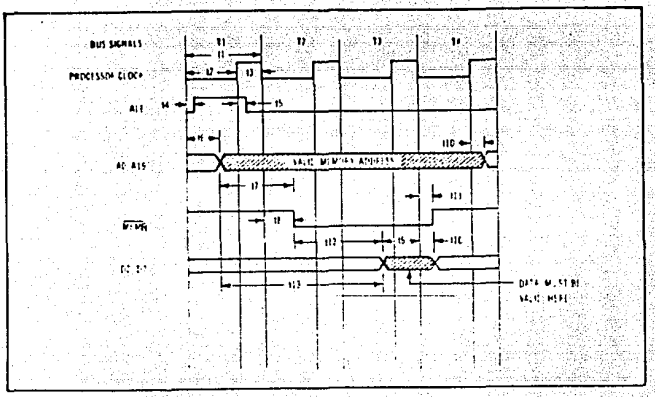

74LS244X2
MEMW
A2..A11
BUFFER RECEPTOR
WE
OE
ADR
RAM
6116 — 2X8
2K X 32 BITS
I/O
I/O
MUX
BUFFER RECEPTOR
BUFFER RECEPTOR
BC2
BC3
74LS244X8
74LS244X2
MR
A2..A11
C1,C2,C3
BC1
BC2
BC3
74LS245
MEMR
P=Q
D0..D7
DIR
TRANSCEPTOR
A LA UNIDAD ARITMETICA
SEÑAL DE BUS
SEÑAL DE CONTROL

FIG. 21



MAPEO DEL ESPACIO DE MEMORIA DE LA PC-XT

FIG. 22

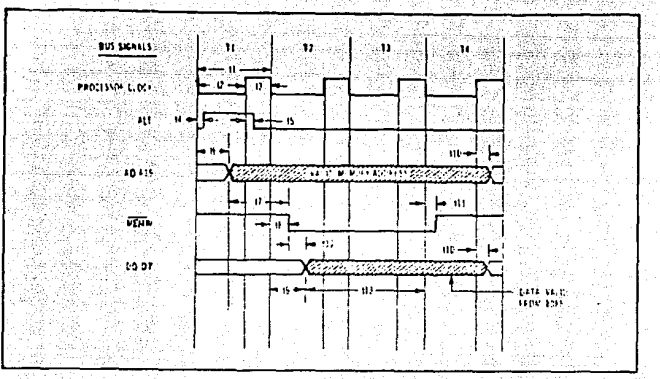


Symbol	Max	Min
t1	—	209.5
t2	—	124.5
t3	—	71.8
t4	15	—
t5	15	—
t6	126	16
t7	—	91.5
t8	35	10
t9	—	42
t10	—	10
t11	35	10
t12	—	342
t13	—	458.5

*All times are in nanoseconds.

TIEMPOS DE LECTURA DE MEMORIA

FIG. 23



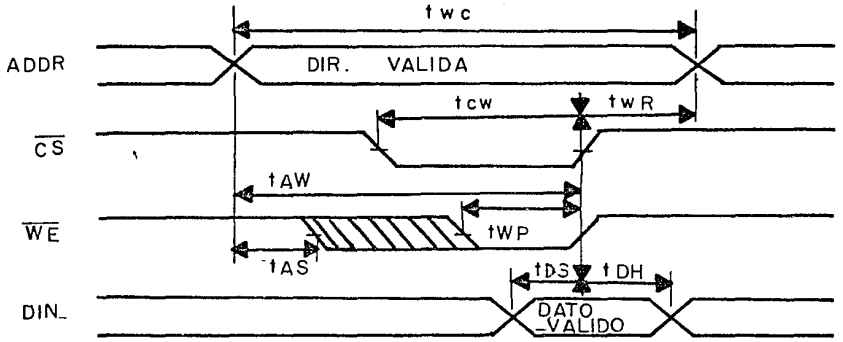
Symbol	Max	Min
t1	—	209.5
t2	—	124.5
t3	—	71.8
t4	15	—
t5	15	—
t6	128	16
t7	—	91.5
t8	35	10
t9	122	14
t10	—	10
t11	35	10
t12	112	—
t13	—	297

*All timings are in nanoseconds

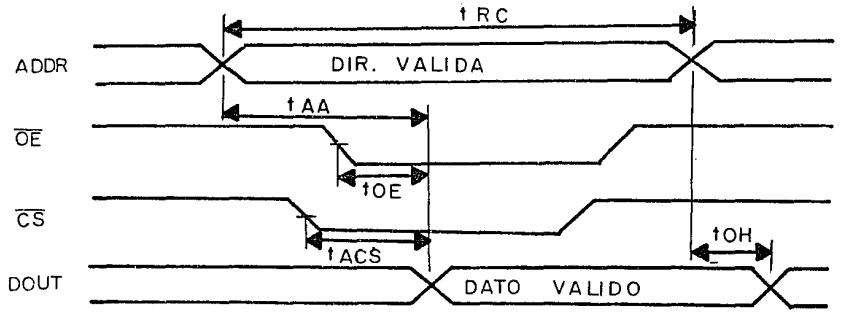
TIEMPOS DE ESCRITURA DE MEMORIA

FIG .24

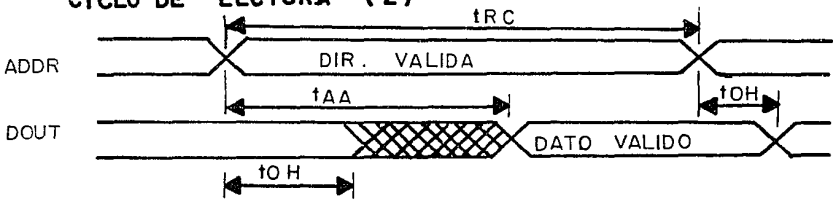
CICLO DE ESCRITURA (2)



CICLO DE LECTURA (1)



CICLO DE LECTURA (2)



CICLOS DE LECTURA / ESCRITUA DE RAM 6116-2

FIG. 25

8. INTERFASE ENTRADA / SALIDA.

El procesador de la PC (8088) es el que maneja la transferencia de datos en la lectura/escritura de puertos I/O utilizados en la Tarjeta de Control del coprocesador.

Un ciclo [21] de lectura/escritura tiene como mínimo cinco periodos de reloj de la PC, esto es, aproximadamente un ciclo de 1.05 μseg. En este proceso se generan los estados T1, T2, T3, TW y T4 necesarios en la transferencia de datos.

La señal ALE (habilitación de direcciones) es la que informa a la tarjeta que hay una dirección válida para puerto indicada por las líneas de dirección A0 a A15. La lectura de puerto se realiza ejecutando la instrucción IN. En esta instrucción se genera la línea IOR (activa baja) que le indica al puerto que debe colocar el dato en el bus. (ver FIG..26).

Ejecutando una instrucción OUT, se hace una escritura a puerto generándose la línea IOW (activa baja). Esta línea le indica al puerto direccionado que debe tomar el dato que se le envía. (ver FIG.27).

Para la decodificación de puertos solamente se utilizan los 10 bits más bajos de dirección, de la A0 a la A9. La línea A9 selecciona los puertos para el sistema (tarjeta maestra) cuando A9 = 0 y selecciona los puertos para periféricos en ranuras (slots) con A9 = 1.

En la FIG.28 se muestra un decodificador de puertos programable, que tiene la particularidad de seleccionar un determinado grupo de

puertos según la posición de los DIP SWITCH. En el circuito el 74LS688 selecciona el grupo de puertos de lectura/escritura activando su salida $\overline{P=Q}$ cuando la dirección del puerto que llega de la PC coincide con la posición de los DIP SWITCH. Variando la posición de estos suiches se varía el espacio de puertos a utilizar. Los decodificadores 74LS138 seleccionan el puerto en particular de lectura o escritura según se active $\overline{IOR}$ o $\overline{IOW}$. Las líneas $\overline{READ}$ Y $\overline{WRITE}$ activarán los registros I/O correspondientes (FIG.29) que almacenarán el dato a leer o a escribir. El 74LS245 es el transceptor cuya función será la de dejar pasar los datos de entrada o salida controlado por la línea $\overline{IOR}$. Cuando $\overline{IOR}$=0 permitirá la salida de datos (lectura de puertos) y con $\overline{IOR}$=1 permitirá la entrada de datos (escritura a puertos).

La utilización de los puertos I/O en el diseño de éste coprocesador, se debe a que las órdenes de comando de activación se enviarán con una escritura a un determinado puerto y la lectura del estado del coprocesador (registro de status) y del resultado mismo de la suma de productos se hará con lecturas a puertos I/O.

Para la realización de ésta comunicación por puertos se ha dejado disponible con los decodificadores 74LS138 (FIG.28) la posibilidad de manejar hasta 8 puertos de entrada (escritura) y 8 puertos de salida (lectura).

La determinación del espacio de puertos a utilizar se realiza conociendo el mapeo de ellos en la computadora. El mapeo del espacio de puertos en la computadora es el siguiente :

Espacio de puertos : 0000 a FFFF

Tarjeta Maestra : 0000 - 01FF (512 puertos)

0000 - 000F : DMA (8237)
0020 - 0021 : PIC (8259)
0022 - 003F : LIBRE
0040 - 0043 : TIMER (8253)
0044 - 005F : LIBRE
0060 - 0063 : PPI (8255)
0064 - 007F : LIBRE
0080 - 0083 : REG. PAG. DMA (74670)
0084 - 009F : LIBRE
00A0 : NMI MASK BIT
00A1 - 00BF : LIBRE
00C0 - 01FF : LIBRE

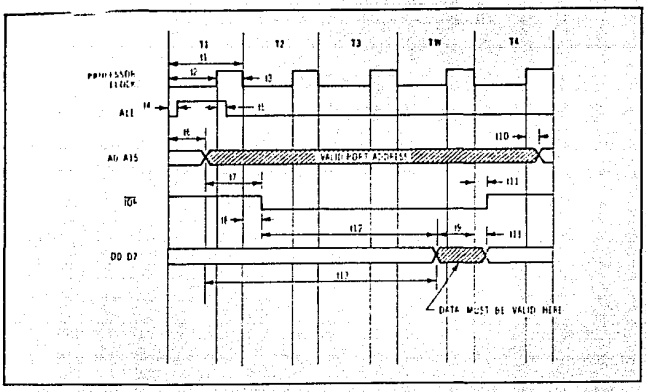
SLOTS (RANURAS) : 0200 - 03FF

0200 : LIBRE
0201 : GAME CONTROL
0202 - 0277 : LIBRE
0278 - 027F : PUERTO2 IMPRESORA
0280 - 02F7 : LIBRE
02F8 - 02FF : PUERTO SERIE - COM 1
0300 - 0377 : LIBRE
0378 - 037F : PUERTO1 IMPRESORA
0380 - 03AF : LIBRE
03B0 - 03BF : MONOCROMATICO
03C0 - 03CF : LIBRE
03D0 - 03DF : COLOR GRAPHICS
03E0 - 03EF : LIBRE
03F0 - 03F7 : DISKETTE
03F8 - 03FF : PUERTO SERIE - COM 0
0400 - FFFF : NO USADO POR LA PC

Se seleccionó el espacio de puertos para la Tarjeta de Control de 02F0 a 02F7. Esto es,

A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	0	1	1	1	1	0	0	0	0	(02F0)
1	0	1	1	1	1	0	0	0	1	(02F1)
.	.	.	.	.	.	.	.	.	.	
1	0	1	1	1	1	0	1	1	1	(02F7)

Las líneas A_9 a A_3 seleccionan el espacio de puertos del coprocesador (selector deposiciones- DIP SWITCH) y las líne A_2 a A_0 seleccionan uno de los 8 puertos I/O dentro de la tarjeta.

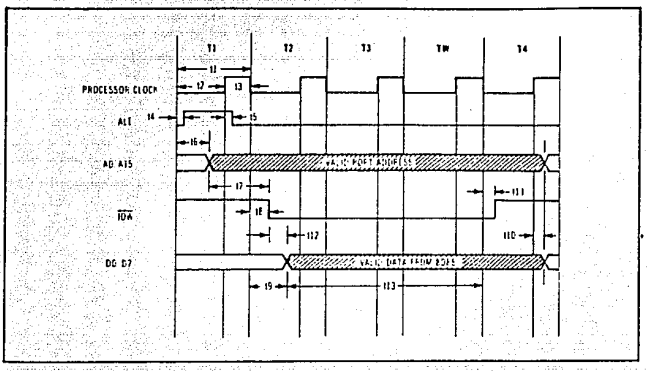


Symbol	Max	Min
t1	—	209.5
t2	—	124.5
t3	—	71.8
t4	15	—
t5	15	—
t6	128	16
t7	—	91.5
t8	35	10
t9	—	42
t10	—	10
t11	35	10
t12	—	551.5
t13	—	666

*All times are in nanoseconds.

TIEMPOS DE LECTURA DE PUERTOS

FIG.26



Symbol	Max	Min
t1	—	209.5
t2	—	124.5
t3	—	71.8
t4	15	—
t5	15	—
t6	128	16
t7	—	91.5
t8	35	10
t9	122	14
t10	—	10
t11	35	10
t12	112	—
t13	—	506.5

*All times are in nanoseconds.

TIEMPOS DE ESCRITURA DE PUERTOS

FIG.27

INTERFACE DE PUERTOS

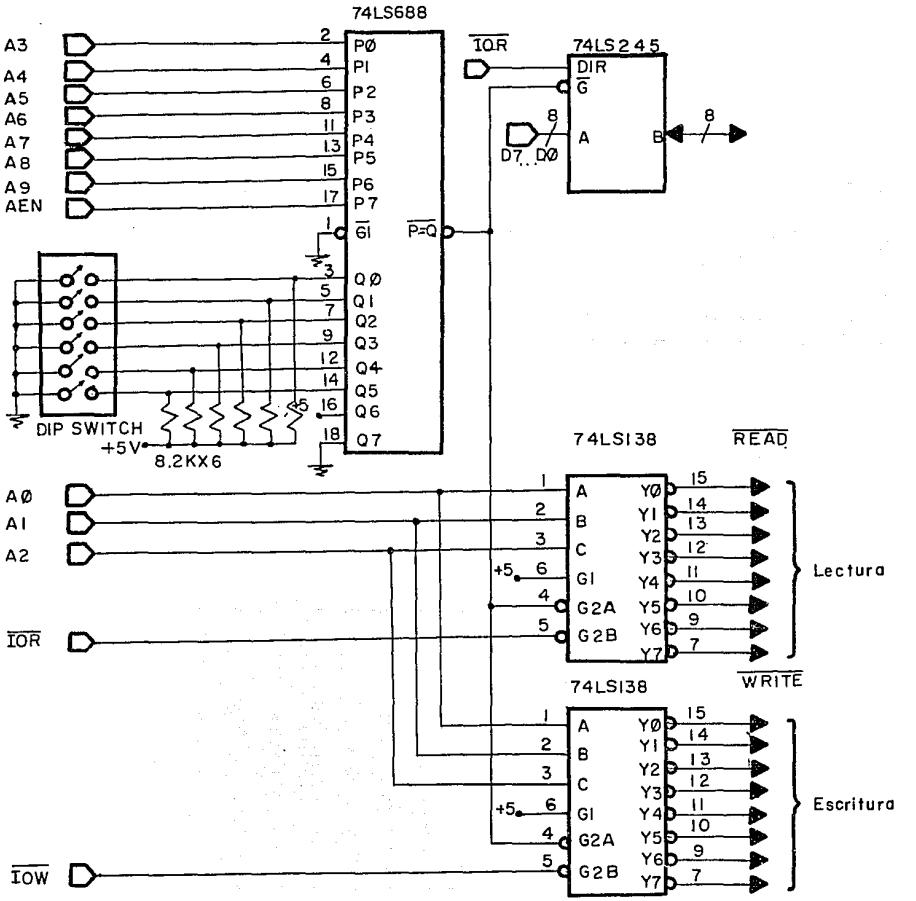


FIG.28

REGISTROS I/O

DE INTERFACE

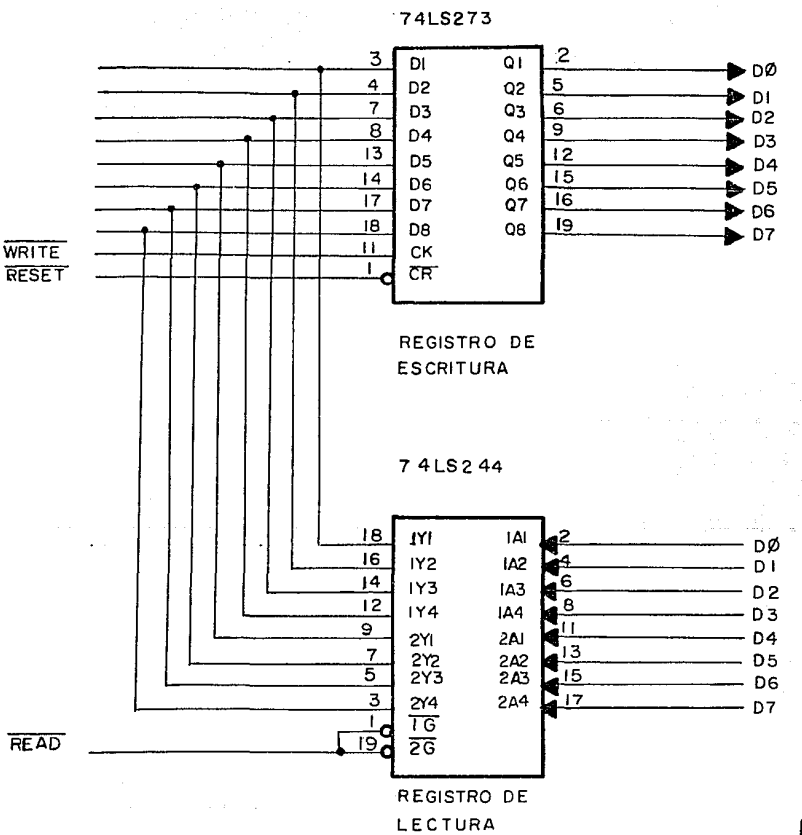


FIG.29

9. UNIDAD ARITMETICA.

El corazón de un coprocesador lo constituye su unidad aritmética. El diseño de esta unidad depende de las operaciones que tiene que realizar y del tipo de aritmética que utilice en el procesamiento de los datos.

La aritmética de punto-fijo se usa en aplicaciones comerciales o de negocios en forma exacta. En cálculos de ingeniería o científicos este tipo de aritmética presenta algunas dificultades puesto que su rango es limitado y su precisión es rígida. En este tipo de cálculos se deben redondear los números constantemente para reducir el número de dígitos a una cantidad manejable. Es así como en aplicaciones de procesamiento digital de señales, la acumulación de errores en el redondeo produce ruido al sistema, altera la posición de los polos y ceros en un filtro cambiando la función de transferencia y causando posiblemente inestabilidad. Con operaciones en punto-flotante se evitan generalmente estos problemas.

La aritmética de punto-flotante tiene la ventaja de tener un gran rango dinámico y alta precisión. Aunque requiere más hardware, la aritmética de punto-flotante ha sido universalmente aceptada para cálculos científicos de alta velocidad.

El formato estandar IEEE para punto-flotante, estandariza la aritmética de punto-flotante para microcomputadoras y minicomputadoras. El propósito de esta estandarización, es la asegurar un ambiente de software uniforme para los programadores. Puede ser implantado por software, hardware o por una combinación de ambos. Esta estandarización describe en forma precisa el formato de los datos y los resultados de una operación aritmética.

En términos de precisión, algunas aplicaciones requieren de punto-flotante en simple-precisión (32 bits) y en otras, se requiere de doble-precisión (64 bits) o de precisión-extendida (80 bits). En formato estandar IEEE-754 para simple-precisión se tienen 23 bits para la mantisa (los menos significativos), 8 bits para el exponente y 1 bit de signo; el exponente está polarizado con el decimal 127 (7FH) y tiene un rango decimal de $1.2x10^{-38} \leq |x| \leq 3.4x10^{38}$. Para doble-precisión se tienen 52 bits para la mantisa, 11 bits para el exponente y 1 bit de signo; el exponente está polarizado con 1023 (3FFH) y tiene un rango de $2.3x10^{-308} \leq |x| \leq 1.7x10^{308}$. En precisión-extendida se tienen 64 bits para la mantisa, 15 bits para el exponente y 1 bit de signo; el exponente está polarizado con 16383 (3FFFH) y tiene un rango dinámico de $3.4x10^{-4932} \leq |x| \leq 1.1X10^{4932}$.

9.1 ARQUITECTURA DE LA UNIDAD ARITMETICA.

El estudio realizado en el capítulo correspondiente al procesamiento pipeline, nos llevó a la conclusión que la velocidad de un procesador pipeline es mayor a la velocidad de un uniprocesador en una proporción que dependía del número de etapas del pipeline.

Como el interés de este proyecto se centra principalmente en el diseño de un coprocesador que realice la sumatoria de productos, la arquitectura de la unidad aritmética de éste coprocesador se hará en estructura pipeline de tres etapas tal como se indica en la FIG.30. Como se observa en este esquema, el multiplicador lo constituye las dos primeras etapas y el sumador (acumulador) la última etapa. Esta decisión se tomó con base en el análisis de dos estructuras pipeline; una de cuatro etapas con un tiempo de ciclo de 100 nsg y la otra de tres etapas con un tiempo de ciclo de 200

nsg, ambas estructuras con una latencia de 200 nsg. En la FIG.31 se muestran estos dos pipeline con sus correspondientes tablas de reserva. El análisis es el siguiente :

PIPELINE 1 : Para n = 10, k = 4, τ = 100 nsg,

$$Th = n / [k + 2(n - 1)]\tau = 4.5 \text{ MFLOPS}$$

$$S = n.k / [k + 2(n - 1)] = 1.9$$

$$\eta = n.k.\tau / k.[k + 2(n - 1)]\tau = 45\%$$

Para n = inf, se tiene, Th = 5.0 MFLOPS, S = 2, η = 50%

PIPELINE 2 : Para n = 10, k = 3, τ = 200 nsg,

$$Th = n / [k + (n - 1)]\tau = 4.2 \text{ MFLOPS}$$

$$S = n.k / [k + (n - 1)] = 2.5$$

$$\eta = n.k / k [k + (n - 1)] = 83\%$$

Para n = inf, se tiene, Th = 5.0 MFLOPS, S = 3, η = 100%

Como análisis de estas dos estructuras, se puede concluir que la configuración PIPELINE 2 ofrece mejores características que el PIPELINE 1. Estas características se mejoran aún más, cuando el número de entradas n (elementos de los vectores es grande (n = inf). Por tanto, para el diseño de la unidad aritmética del coprocesador, se escogerá la estructura PIPELINE 2.

9.2 CIRCUITOS VLSI PARA PUNTO-FLOTANTE.

Circuitos VLSI de punto flotante son fabricados corrientemente por Advanced Microdevices [25], texas Instruments [26], Weitek y

Analog Devices [27]. Weitek tiene el WTL1032 y el WTL1033 que son multiplicador y ALU de 32 bits en punto-flotante con velocidad de 10 MFLOPS. Advanced Microdevices ofrece el AMD29325 que es un procesador de punto-flotante de 32 bits con tres buses (dos de entrada y uno de salida), soporta formato IEEE-754 y tiene una velocidad de 10 MFLOPS. Tiene la gran desventaja, que consume cerca de 2 amperios. Texas Instruments ha desarrollado el SN74ACT8837/47 que son procesadores de 64 bits de punto-flotante con multiplicador y ALU integrados dentro del mismo chip para que puedan operar en paralelo, puede ser configurado para pipeline y es compatible con el formato IEEE-754. Tiene un solo bus de entrada de 64 bits, pero puede ser configurado como dos buses de 32 bits. Este tipo de procesador fue el usado en la Universidad de Florida para la realización del procesador paralelo adicionado a la computadora PC-AT como lo publica la referencia [2].

Analog Devices es la que más ha desarrollado circuitos aritméticos de punto-flotante con buses de 32 y 64 bits. Dentro de ellos se tienen :

ADSP-3210/3211 : Multiplicadores de doble precisión
ADSP-3220/3221 : ALUs de doble precisión.

Estos chips procesan las operaciones en tres tipos de formatos : formato IEEE para simple-precisión y 32 bits, punto-fijo en 32 bits y doble-precisión en 64 bits. El 3211 tiene un throughput de 20 MFLOPS en simple-precisión, 5 MFLOPS en doble-precisión y 20 MIPS en punto-fijo, tiene tres puertos. El 3210 sólo tiene dos puertos y un throughput de 16.6 MFLOPS en simple-precisión, 4 MFLOPS en doble-precisión y 16.6 MIPS en punto-fijo. Tanto el 3220 como el 3221 tienen tres puertos y un throughput de 10 MFLOPS para simple y doble-precisión y 10 MIPS para punto-fijo.

ADSP-3212 multiplicador y ADSP-3222 ALU : Es la siguiente generación del 3211 y el 3221. Son totalmente compatibles con el formato IEEE-754 en todas las operaciones aritméticas, solamente tienen un registro interno de pipeline y son de baja potencia.

ADSP-3201 multiplicador y ADSP-3202 ALU : Ambos procesan números de 32 bits en punto-flotante y punto-fijo en formato IEEE-754, tienen estructura de tres puertos y operan a una frecuencia de 10 Mhz. El consumo de potencia es bajo, aproximadamente 750 mW.

Con base en esta información sobre las características de los diferentes chips de procesamiento aritmético en punto-flotante y a la configuración de la estructura de pipeline definida para la Unidad Aritmética, se decide escoger como multiplicador el ADSP-3201 KG y como sumador (una instrucción de la ALU) el ADSP-3202 KG que tienen un productividad para 32 bits de 100 nsg (10MFLOPS) y una latencia de 240 nsg.

Seguidamente se estudiarán las caracteríticas con más detalle de estos dos procesadores aritméticos.

9.3 El ADSP-3201 y el ADSP-3202.

Tienen las siguientes características :

- Operan con formato estandar IEEE-754
- Throughput de pipeline : 100 nsg (32 bits), 240 nsg (64 bits) para el 3201 KG/3202 KG
  125 nsg (32 bits), 300 nsg (64 bits) para el 3201 JG/3202 JG
- Dos etapas de pipeline con baja latencia de 240 nsg

- Formatos de 32 bits en punto-flotante, 32 bits en punto-fijo en Complemento a Dos y 32 bits en punto-fijo en Signo-Magnitud.
- Tres buses de 32 bits : dos de entrada y uno de salida
- Disipación máxima de potencia de 750 mW (150 mA)
- Pines de status para : overflow, underflow, operación inválida, resultado inexacto y división por cero
- Chips de 144 pines.

Los datos en formato IEEE-754 tienen la siguiente representación:

- Tienen una fracción signada (la menos significativa) de 23 bits, un exponente no-signado de 8 bits y un bit de signo.
- Los números normalizados (NORM) tienen cualquier fracción y un exponente entre 1 y 254 (polarizado con +127).
- Los números denormalizados (DNRM) tienen un exponente igual a cero y una fracción o mantisa diferente de cero. El verdadero valor del exponente se obtiene restándole -126.
- El cero (ZERO) es representado por exponente y mantisa igual a cero.
- Un número no-definido (not a number-NAN) es representado con mantisa diferente de cero y exponente igual a 255.
- El infinito (INF) se representa con mantisa igual a cero y exponente igual a 255.
- El máximo número normalizado (NORM.MAX) es $2^{+127}. (2 - 2^{-23})$
- El mínimo número normalizado (NORM.MIN) es 2^{-126}

Las líneas de control mediante las cuales operan estos dos procesadores aritméticos (FIG.32, FIG.33) son :

FAST/IEEE : Hace trabajar a los chips en una operación simple, trabajando sólo con datos normalizados.

RESET : Es activada para la inicialización. Borra todas las funciones de control. En reset las líneas de selección de los

registros de entrada deben estar en activo-bajo. Esta línea debe ser activada al poner la potencia.

IPORT1/0 : Son utilizadas para la configuración de los puertos.

IPORT1	IPORT0	CONFIGURACION
0	0	Ain → Reg B, Bin → Reg A (bipuerto)
0	1	Bin → Reg A y Reg B (unipuerto)
1	0	Ain → Reg A y Reg B (unipuerto)
1	1	Ain → Reg A, Bin → Reg B (bipuerto)

SELA3..A0/SELB3..B0 : Seleccionan los registros de entrada. En la configuración bipuerto, los registros pares se cargan con el frente de subida del pulso y los impares con el frente de bajada.

SP : Línea de control que selecciona el formato de los datos. Esta línea es solamente usada por el multiplicador ADSP3201. con SP = 0 se trabaja en punto-fijo y con SP = 1 se trabaja en punto-flotante.

I8..0 : Son líneas de control que seleccionan el tipo de instrucción a ser ejecutada por la ALU ADSP3202.

RDA1/0,RDB1/0 : Línea de control del MUX para seleccionar el registro de datos a leer.

DA1	RDA0	Reg
0	0	A_2
0	1	A_3
1	0	A_0
1	1	A_1

RDB1	RDB0	Reg
0	0	B_2
0	1	B_3
1	0	B_0
1	1	B_1

ABSA/B : Convierte un operando seleccionado por RDA/B a su valor absoluto antes de ser procesado.

WRAPA/B : Es un control de entrada para datos denormalizados.

TCA/B : Línea de control para operandos en Complemento a Dos o no-signados en punto-fijo. En punto -flotante estas líneas son ignoradas.

RND1/0 : Seleccionan el modo de redondeo para números de punto-flotante en formato IEE-754. Existen cuatro modos de redondeo,

Tipo de redondeo	RND1	RND0
RN (round to nearest) : redondeo al más cercano	0	0
RZ (round toward zero) : redondeo a cero	0	1
RP (round toward plus infinity) : redondeo a + infinito	1	0
RM (round toward minus infinity) : redondeo a - infinito	1	1

REDONDEO A +INF : Si el resultado antes del redondeo es mayor que NORM.MAX pero diferente a +inf,el resultado se redondea a +inf. Si es menor que -NORM.MAX pero diferente a -inf, el resultado se redondea a -NORM.MAX.

REDONDEO A -INF : Si el resultado antes del redondeo es mayor que NORM.MAX pero diferente a +inf, el resultado se redondea a

NORM.MAX. Si es menor que -NORM.MAX pero diferente a -inf, el resultado se redondea a -inf.

REDONDEO A CERO : Si el resultado antes del redondeo tiene una magnitud mayor NORM.MAX pero diferente de inf, el resultado se redondea a NORM.MAX del mismo signo (truncación).

REDONDEO AL MAS CERCANO : Si el resultado antes del redondeo está exactamente a la mitad de dos números que difieren del LSB, el resultado se redondea al que tiene el LSB = 0. Si hay sobreflujo antes del redondeo, es decir, mayor a NORM.MAX + 1/2 LSB, el resultado se redondea a inf del mismo signo.

FLAGS de STATUS : Tiene cinco flags o banderas que indican el estado del procesador aritmético. El overflow (OVRFLO), underflow (UNDFLO), resultado inexacto (INEXO) y operación inválida (INVALOP). La condición de división por cero es indicada por la simultaneidad de overflow y operación inválida.

CONTROLES DE SALIDA : SHLP, HOLD, MSWEL y OEN.

En las FIG.34 y FIG.35 se muestran los diagramas de tiempo de estos dos procesadores aritméticos, con la siguiente temporización :

t_{CY} : Ciclo de reloj máx = 100 nsg
t_{DS} : Tiempo de setup min = 15 nsg
t_{DH} : Tiempo hold min = 3 nsg
t_{OPD} : Tiempo de operación máx = 100 nsg
t_{LAD} : Tiempo de latencia máx = 240 nsg

9.4. CONSIDERACIONES DE DISEÑO.

Con base en los criterios de selección citados anteriormente, se escogió el ADSP 3201-KG como multiplicador y el ADSP 3202-KG como sumador (ALU), que son procesadores de punto flotante de 32 bits.

Para su funcionamiento normal con los requerimientos de operación del proyecto necesarios en la implantación de la Unidad Aritmética, se diseña la siguiente configuración :

- La línea RESET de los procesadores se genera de la Unidad de Control cada vez que termina un proceso de cálculo (suma de productos) con el fin de dejar sus registros internos borrados.
- Los puertos de entrada se han configurado en forma de bipuerto, esto es, $A_{in} \longrightarrow$ Reg A y $B_{in} \longrightarrow$ Reg B, seleccionando,

  IPORT0.IPORT1 = 1 1
- Se han tomado como líneas de selección de los registros de entrada SELA0 y SELB0
- El formato de datos para el multiplicador (3201) se selecciona el punto-flotante activando la línea SP = 1 y para la ALU (3202) suma de datos en punto-flotante 32 bits activando las líneas

$$I_8 I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0 = 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1$$

- Como registro de datos a leer se han seleccionado los registros A0, B0 activando

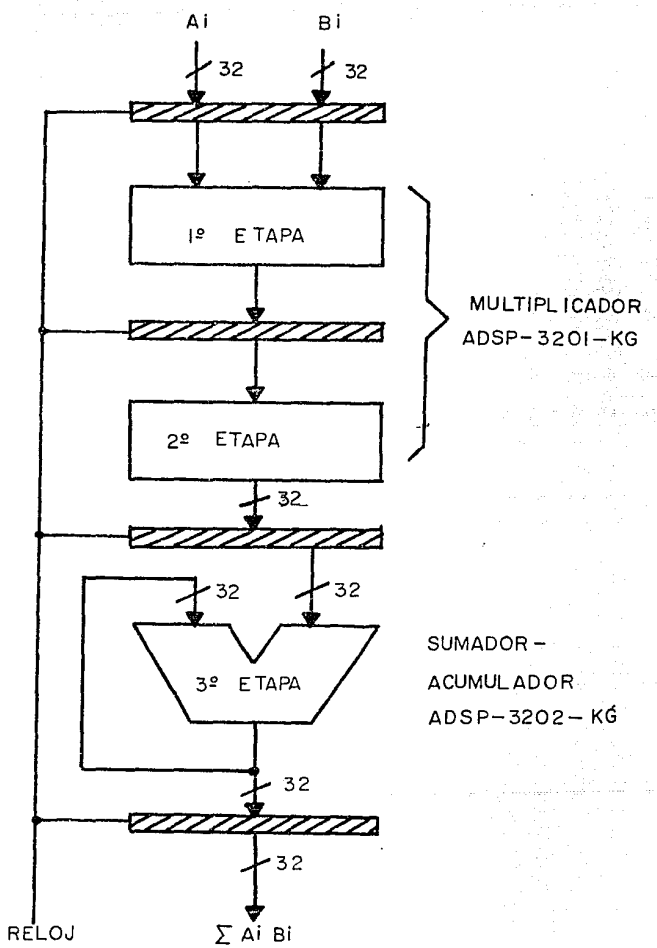
  RDA1.RDA0 = 1 0 RDB1.RDB0 = 1 0
- De los cuatro modos de redondeo posibles se ha tomado el redondeo al más cercano (RN) activando las líneas

  RND1.RND0 = 0 0

- Para el manejo de datos a la salida se activan la líneas

  HOLD = 1 MSWL = 1 OEN = 1

- Como señal de reloj para la ALU se usará la señal de reloj de la computadora (100 nsg) y para el multiplicador una señal de 200 nsg generad de la unidad de control. Esto se debe a que los procesadores internamente están conformados por una estructura pipeline de dos niveles y como la ALU está realizando las sumas acumuladas ésta debe trabajar al doble de velocidad que el multiplicador.



ARQUITECTURA DE LA UNIDAD ARITMETICA

FIG.30

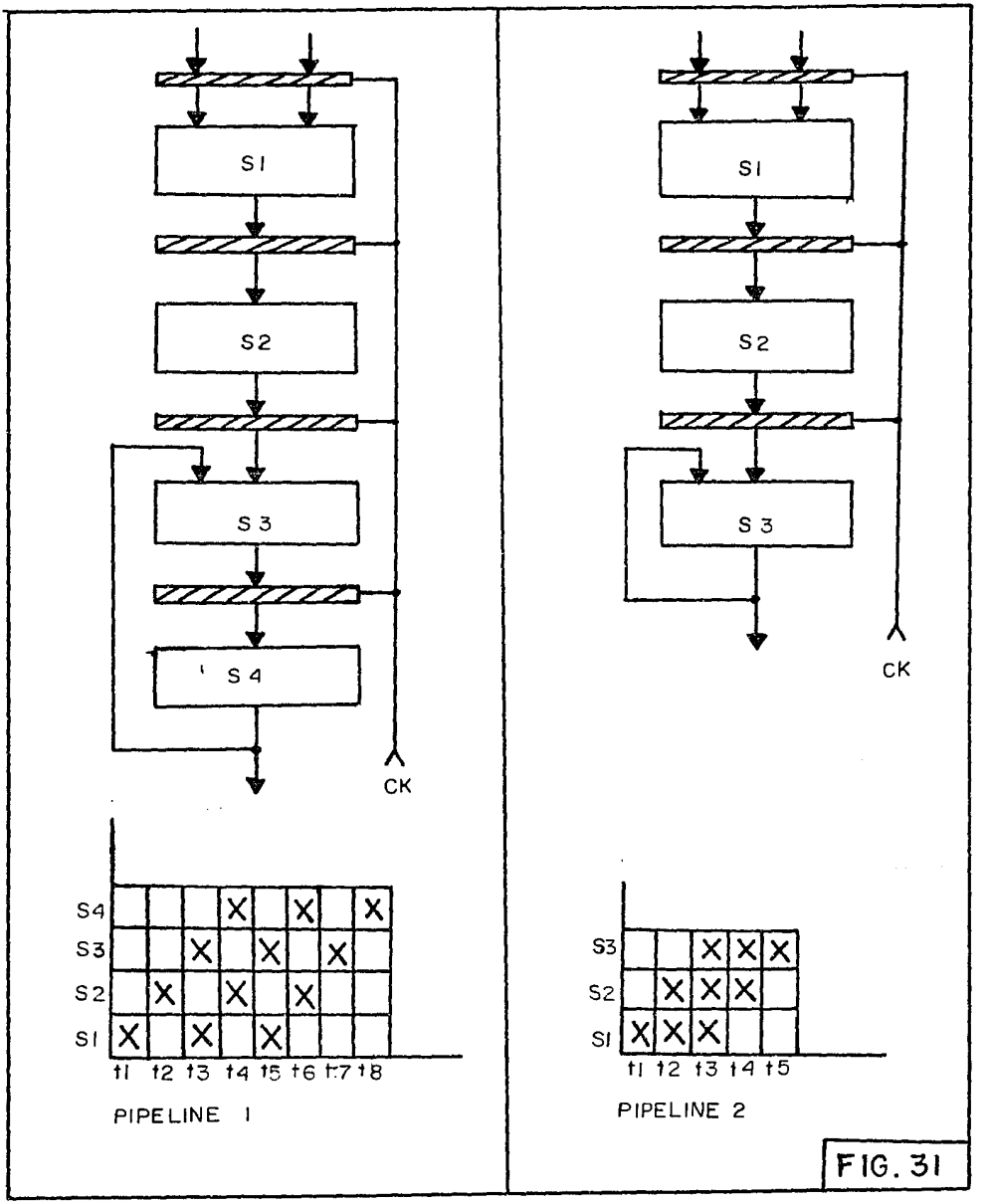

S1
S2
S3
S4
CK
S4
S3
S2
S1
t1 t2 t3 t4 t5 t6 t7 t8
PIPELINE 1
S1
S2
S3
CK
S3
S2
S1
t1 t2 t3 t4 t5
PIPELINE 2

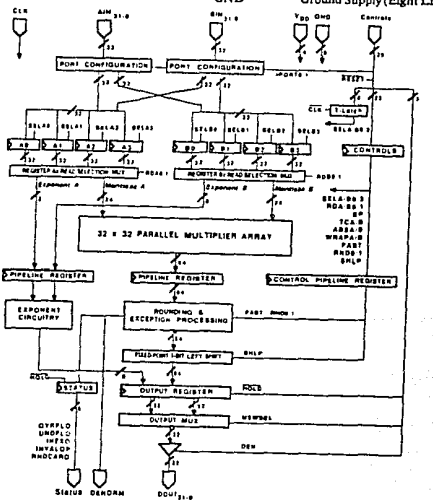
FIG. 31

PIN DEFINITIONS AND FUNCTIONAL BLOCK DIAGRAMS

All control pins are active HI (positive true logic naming convention), except $\overline{\text{RESET}}$ and $\overline{\text{HOLD}}$. Some controls are registered at the clock's rising edge (REG); other controls are latched in clock HI and transparent in clock LO (LAT); and others are asynchronous (ASYN).

ADSP-3201 Floating-Point Multiplier Pin List

PIN NAME	DESCRIPTION	TYPE
Data Pins		
AIN_{31-0}	32-Bit Data Input	
BIN_{31-0}	32-Bit Data Input	
$DOUT_{31-0}$	32-Bit Data Output	
Control Pins		
$\overline{\text{RESET}}$	Reset	ASYN
$\overline{\text{HOLD}}$	Hold Control	ASYN
IPORT0	Input Port Configuration Control 0	ASYN
IPORT1	Input Port Configuration Control 1	ASYN
SELA0	Load Selection for A0	LAT
SELA1	Load Selection for A1	LAT
SELA2	Load Selection for A2	LAT
SELA3	Load Selection for A3	LAT
SELB0	Load Selection for B0	LAT
SELB1	Load Selection for B1	LAT
SELB2	Load Selection for B2	LAT
SELB3	Load Selection for B3	LAT
RDA0	Register Ax Read Selection Control 0	REG
RDA1	Register Ax Read Selection Control 1	REG
RDB0	Register Bx Read Selection Control 0	REG
RDB1	Register Bx Read Selection Control 1	REG
WRAPA	Wrapped Contents in Register Ax	REG
WRAPB	Wrapped Contents in Register Bx	REG
TCA	Twos-Complement Integer in Register Ax	REG
TCB	Twos-Complement Integer in Register Bx	REG
ABSA	Read Absolute Value of Ax	REG
ABSB	Read Absolute Value of Bx	REG
SP	Single-Precision Floating-Point Mode	REG
DP	Double-Precision Mode	REG
RND0	Rounding Mode Control 0	REG
RND1	Rounding Mode Control 1	REG
FAST	Fast Mode	REG
SHLP	Shift Left Fixed-Point Product	REG
MSWSEL	Select MSW of Output Register	ASYN
OEN	Output Data Enable	ASYN
Status Out		
INEXO	Inexact Result	
OVRFLO	Overflowed Result	
UNDFLO	Underflowed Result	
INVALOP	Invalid Operation	
DENORM	Denormal Output	
RNDCARO	Round Carry Propagation Out	
Miscellaneous		
CLK	Clock Input	
V_{DD}	+5V Power Supply (Four Lines)	
GND	Ground Supply (Eight Lines)	



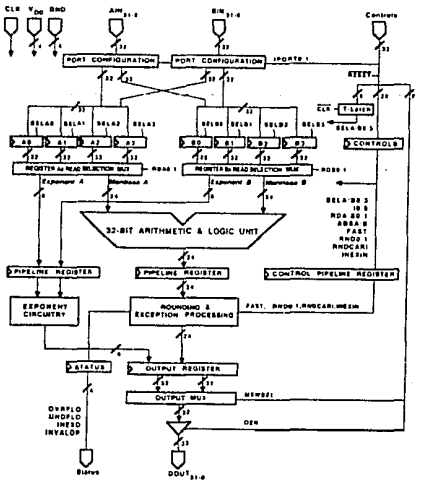
ADSP-3201 Functional Block Diagram

ADSP-3201

FIG. 32

PIN NAME	DESCRIPTION	TYPE
Data Pins		
AIN_{31-0}	32-Bit Data Input	
BIN_{31-0}	32-Bit Data Input	
$DOUT_{31-0}$	32-Bit Data Output	
Control Pins		
$\overline{RESET}$	Reset	ASYN
IPORT0	Input Port Configuration Control 0	ASYN
IPORT1	Input Port Configuration Control 1	ASYN
SELA0	Load Selection for A0	LAT
SELA1	Load Selection for A1	LAT
SELA2	Load Selection for A2	LAT
SELA3	Load Selection for A3	LAT
SELB0	Load Selection for B0	LAT
SELB1	Load Selection for B1	LAT
SELB2	Load Selection for B2	LAT
SELB3	Load Selection for B3	LAT
RDA0	Register Ax Read Selection Control 0	REG
RDA1	Register Ax Read Selection Control 1	REG
RDB0	Register Bx Read Selection Control 0	REG
RDB1	Register Bx Read Selection Control 1	REG

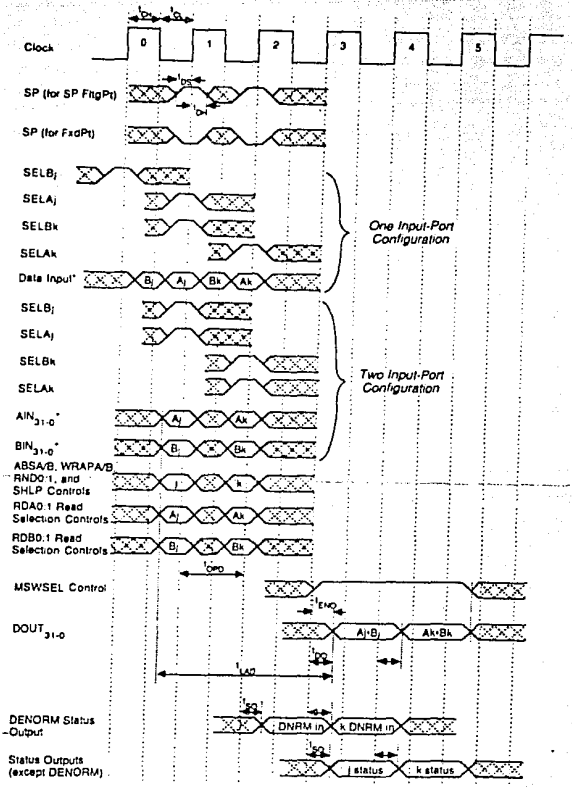
PIN NAME	DESCRIPTION	TYPE
I_{6-0}	ALU Instruction	REG
RND0	Rounding Mode Control 0	REG
RND1	Rounding Mode Control 1	REG
FAST	Fast Mode	REG
MSWSEL	Select MSW of Output Register	ASYN
OEN	Output Data Enable	ASYN
Status In		
INEXIN	Inexact Data In	REG
RNDCAR1	Round Carry Propagation In	REG
Status Out		
INEXO	Inexact Result	
OVRFLO	Overflowed Result	
UNDFLO	Underflowed Result	
INVALOP	Invalid Operation	
Miscellaneous		
CLK	Clock Input	
V_{DD}	+5V Power Supply (Four Lines)	
GND	Ground Supply (Four Lines)	



ADSP-3202 Functional Block Diagram

ADSP-3202

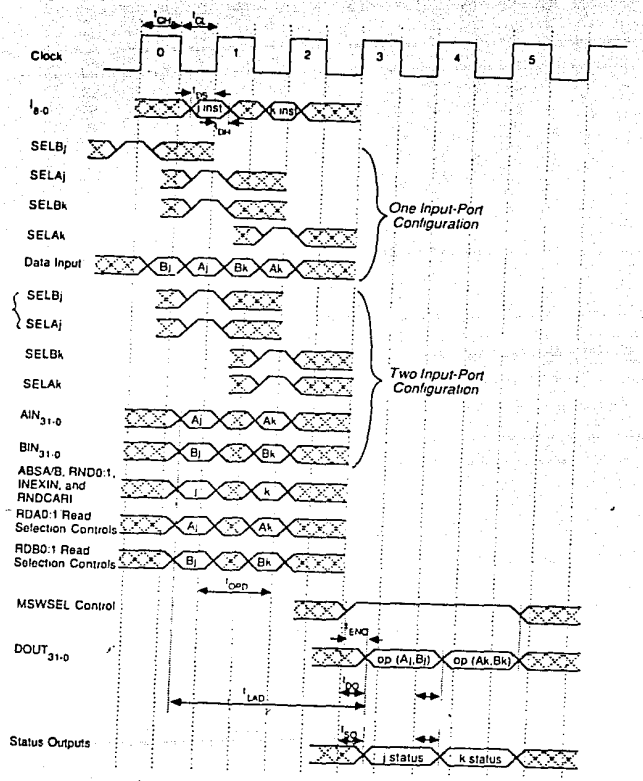
FIG.33



ADSP-3201 32-Bit Single-Precision Floating-Point and Fixed-Point Multiplications

TEMPORIZACION DEL ADSP-3201

FIG. 34



ADSP-3202 32-Bit Single-Precision Floating-Point Logical, and Fixed-Point ALU Operations

TEMPORIZACION DEL ADSP-3202

FIG. 35

10. UNIDAD DE CONTROL.

La unidad de control está conformada por el secuenciador que genera las direcciones para la memoria del coprocesador y el controlador encargado de generar las señales de control necesarias para sincronizar el funcionamiento del coprocesador a implementar.

10.1. EL SECUENCIADOR

Está constituido por un contador ascendente/descendente (up/down) que genera las 10 líneas de dirección internas de cada uno de los chips de la memoria (A_{11} .. A_2). Ver FIG.20.

En el proceso de escritura de memoria, se almacenan los dos vectores (datos de 32 bits) a partir de la posición cero (offset) tanto en la memoria de la computadora como en la del coprocesador. En esta escritura actúa como reloj la señal generada CS0 (ver FIG.20) con el fin de que cada vez que se almacenen los dos operandos de 32 bits en memoria el contador aumente su cuenta en uno (forma ascendente). Al terminar la escritura de los operandos, el contador contendrá la dirección de los últimos dos datos almacenados. La línea HAB selecciona el reloj en la lectura y escritura. Con HAB = 0 el reloj que actúa es CS0 y con HAB = 1 actúa la línea de control CK1 generada por el controlador para el proceso de lectura.

Durante la lectura de memoria, el secuenciador pone la dirección y la memoria envía los datos a la Unidad Aritmética comenzando por los últimos almacenados. La memoria entrega los dos operandos simultáneamente. Como el contador está ahora en forma descendente,

él informa que todos los datos han sido leídos cuando llegue a cero activando la línea FIN. Esta línea activa otro contador que produce un retardo necesario para realizar la purga del pipeline, esto es, permitir que los últimos datos leídos de memoria pasen por todos los niveles del pipeline. Este último contador produce la línea RET que se genera después de contar tres pulsos de reloj necesarios para la purga.

Las líneas RESET que viene de la computadora y la línea LD1 del controlador, se usan para inicializar el secuenciador en cada proceso de cálculo de la suma de productos. En la FIG 36 se presenta el diagrama en bloques de esta unidad y en la FIG 37 el diseño correspondiente.

10.2. EL CONTROLADOR

Es la etapa de diseño más crítica y de difícil implementación porque es la encargada de coordinar todas las acciones que se deben realizar para sincronizar todas las unidades del coprocesador y permitirla comunicación entre el procesador de la computadora y el coprocesador a implementar.

10.2.1. EL AUTOMATA.

El autómata define las acciones que debe tomar el controlador según se activen las entradas al mismo. Como se observa en la FIG. 38, el autómata tiene siete estados y en cada uno de ellos se deben generar las salidas que se indican en la FIG 39. Cada línea realiza la siguiente acción :

C1 : Es la línea que controla la selección de los chips de la RAM (6116-2) durante la lectura y escritura. Con C1 = 1, los chip selects de los (CS) de los integrados de memoria se

activan independientemente para almacenar datos de 8 bits; con C1 = 0 se activan los chips selects simultáneamente para leer los dos operandos de 32 bits. Ver FIG 20 y FIG 21.

C2 : Controla el MUX de direcciones de memoria (FIG 21). Con C2=0 se deja pasar la direeción que viene de la computadora para el proceso de escritura en la memoria RAM y con C2=1 se habilita el MUX para que pase la dirección del secuenciador del coprocesador en la lectura de datos.

C3 : Con C3=0 se habilitan los buffers (FIG 21) para que pasen los datos que vienen de la computadora y con C3=1 se dejan los buffers en estado flotante (tri-state) para que la memoria del coprocesador envie los datos a la Unidad Aritmética.

CLR: Se utiliza para borrar los registros pipeline de la Unidad Aritmética haciendo CLR=0.

UD1: Habilita el conteo del secuenciador. Con UD1=0 cuenta en forma ascendente y con UD1=1 en forma descendente.

LD1: Se utiliza para cargar el secuenciador con ceros cada vez que se inicie un nuevo proceso de cálculo (LD1=0).

CK1: Señal de reloj generada para el secuenciador en el proceso de lectura de datos. Debe estar la línea HAB activada (HAB=1).

CK2: Señal de reloj a utilizar como reloj del multiplicador en la estructura pipeline. CK1 = CK2 = 200 nsg.

MR : Señal que habilita la lectura de la memoria RAM del coprocesador (MR = 0).

W : Línea que indica a la CPU de la computadora que el coprocesador ya ejecutó la suma de productos (W = 1).

10.2.2. IMPLANTACIÓN.

La implementación de este tipo de circuitos secuenciales se puede realizar por diferentes métodos: los microprogramados y los no-microprogramados. Los microprogramados tienen la ventaja de disminuir el hardware de implementación a causa de grabar las microinstrucciones del autómata en un memoria EPROM. Como los tiempos de acceso que se requieren para esta implementación son del orden de los 100 nsg, esto causaría conseguir memorias rápidas que harían más costoso el proyecto; memorias comunes como la 2716 son más baratas pero su tiempo de acceso es del orden de los 300 nsg y no serían compatibles con la velocidad de la RAM usada en el proyecto con tiempos de acceso de 120 nsg y de los circuitos usados en la Unidad Aritmética con tiempos de 100 nsg. Debido a lo anterior se optó por hacer la implementación en forma no-microprogramada utilizando circuitos TTL de tecnología LS (bajo consumo y alta velocidad) con MUX como control de entradas (74LS151) y contador de carga paralela (74LS161) como manejador de los estados, metodología establecida en la referencia [28]. En las FIG.40 y FIG.41, se muestra la implementación del controlador.

10.2.3. COMUNICACION PROCESADOR-COPROCESADOR.

La comunicación entre el procesador de la computadora (8088) y el coprocesador diseñado, se realiza a través de las líneas HAB y WAIT.

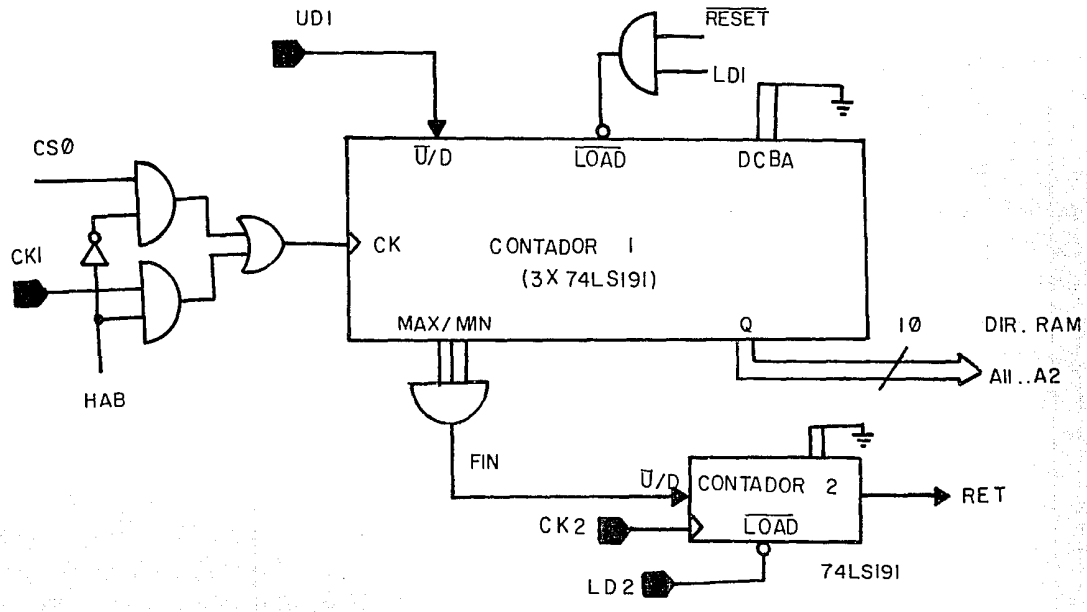
Cuando el procesador necesita de un coprocesamiento de suma de productos en forma pipeline, activa la tarjeta de coprocesamiento enviando por puerto de escritura una señal de habilitación. Esto se hace pasando la entrada HAB del controlador a un nivel alto (HAB=1). Esta línea inicializa al controlador para que lea los

datos de la memoria del coprocesador y los procese en la Unidad Aritmética. Terminada la ejecución, el coprocesador activa la línea de control W (WAIT) indicando que ya finalizó su trabajo. Este estado es leído por el procesador en forma software y lee el resultado de la operación (suma de productos) leyendo los puertos de entrada de la interfase del coprocesador. Leído el resultado, el procesador deshabilita el coprocesador y éste queda listo para una nueva ejecución cuando se le vuelva a necesitar.

10.2.4. TEMPORIZACION DE SEÑALES.

Las señales de comunicación entre procesador y coprocesador (handshaking) se presentan en la FIG.42 con la siguiente temporización :

t_1 = Tiempo de ciclo de 100 nsg
t_2 = Retardo del secuenciador de 44 nsg
t_3 = Tiempo de acceso de la RAM de 120 nsg
t_4 = Tiempo de purga del pipeline de 3x200=600 nsg
t_5 = Retardo de WAIT de aprox. 100nsg
t_6 = Latencia del 3201 y 3202 de 240 nsg
t_7 = Tiempo de procesamiento de N x 100 nsg
N = Número de pares de operandos de 32 bits



UNIDAD DE CONTROL: SECUENCIADOR.

FIG.36

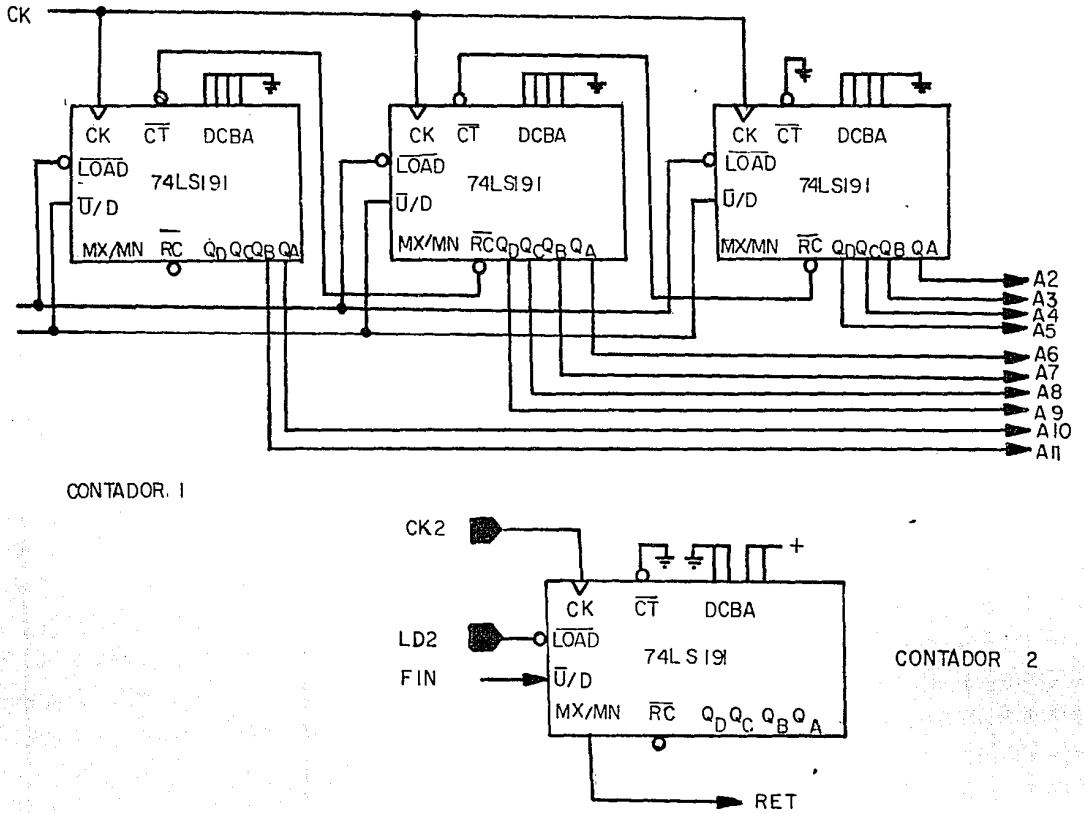
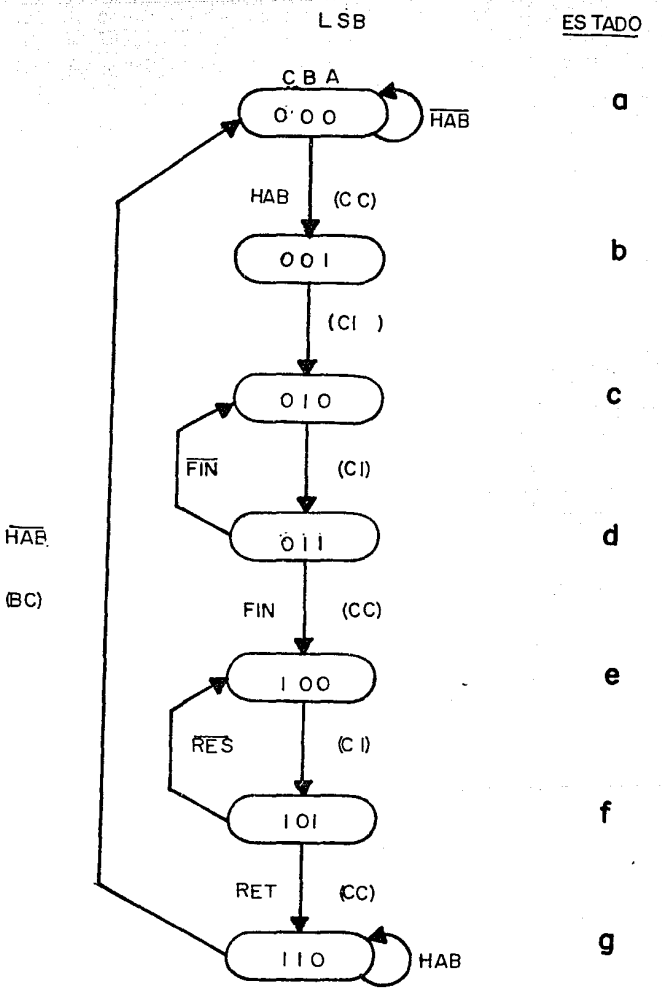

CK
CK CT DCBA
LOAD
U/D
74LS191
MX/MN RC QD QC QB QA
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
CONTADOR 1
CK2
LD2
FIN
+
RET
CONTADOR 2

FIG.37



CONTROLADOR: AUTOMATA

FIG.38

ESTADO	C B A	C1	C2	C3	UD1	LD2	LD1	CK2	CK1	MR	W
a	0 0 0	1	0	0	0	0	1	1	1	1	0
b	0 0 1	0	1	1	1	0	1	1	1	1	0
c	0 1 0	0	1	1	1	0	1	0	0	0	0
d	0 1 1	0	1	1	1	0	1	1	1	0	0
e	1 0 0	0	1	1	1	1	1	0	1	0	0
f	1 0 1	0	1	1	1	1	1	1	1	0	0
g	1 1 0	0	1	1	1	1	0	1	1	1	1

CONTROLADOR: MAPA DE SALIDAS

FIG.39

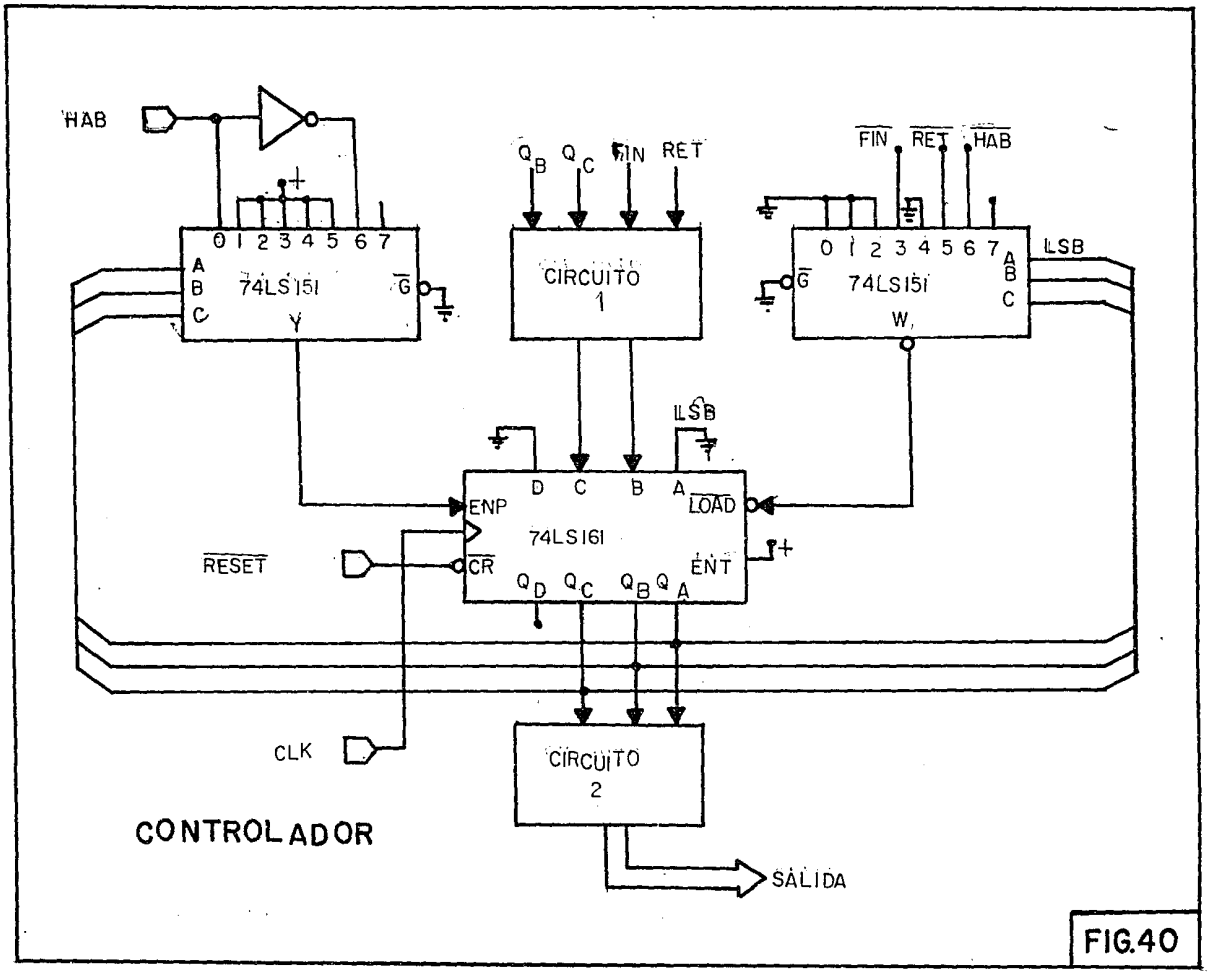
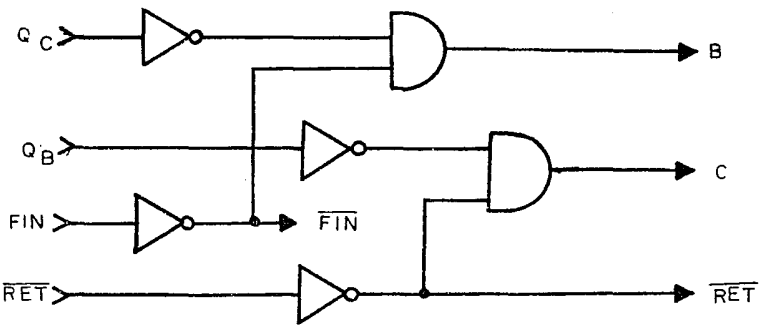

HAB
0 1 2 3 4 5 6 7
A
B
C
74LS151
Y
G
Q_B Q_C FIN RET
CIRCUITO 1
FIN RET HAB
0 1 2 3 4 5 6 7
G
74LS151
W
A
B
C
LSB
LSB
D C B A
ENP
LOAD
74LS161
RESET
CR
ENT
Q_D Q_C Q_B Q_A
CLK
CIRCUITO 2
CONTROLADOR
SALIDA

FIG.40

CIRCUITO 1



CIRCUITO 2

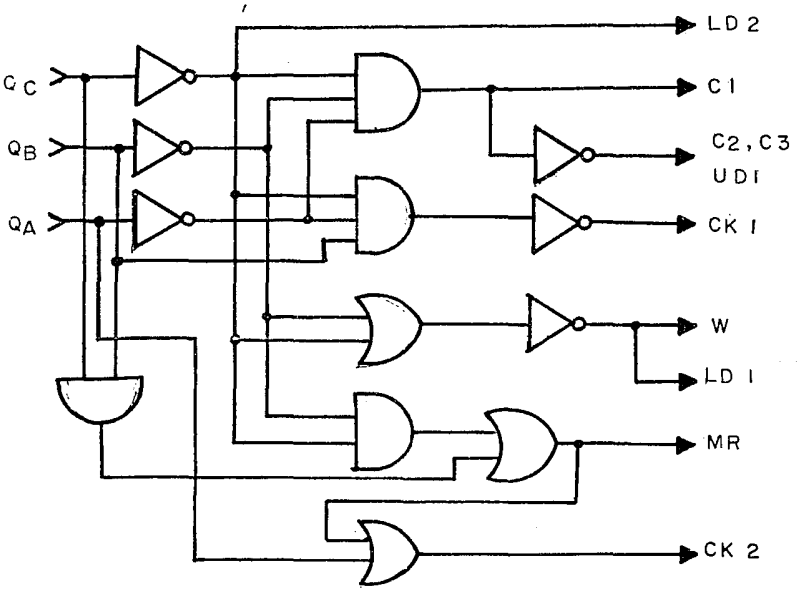
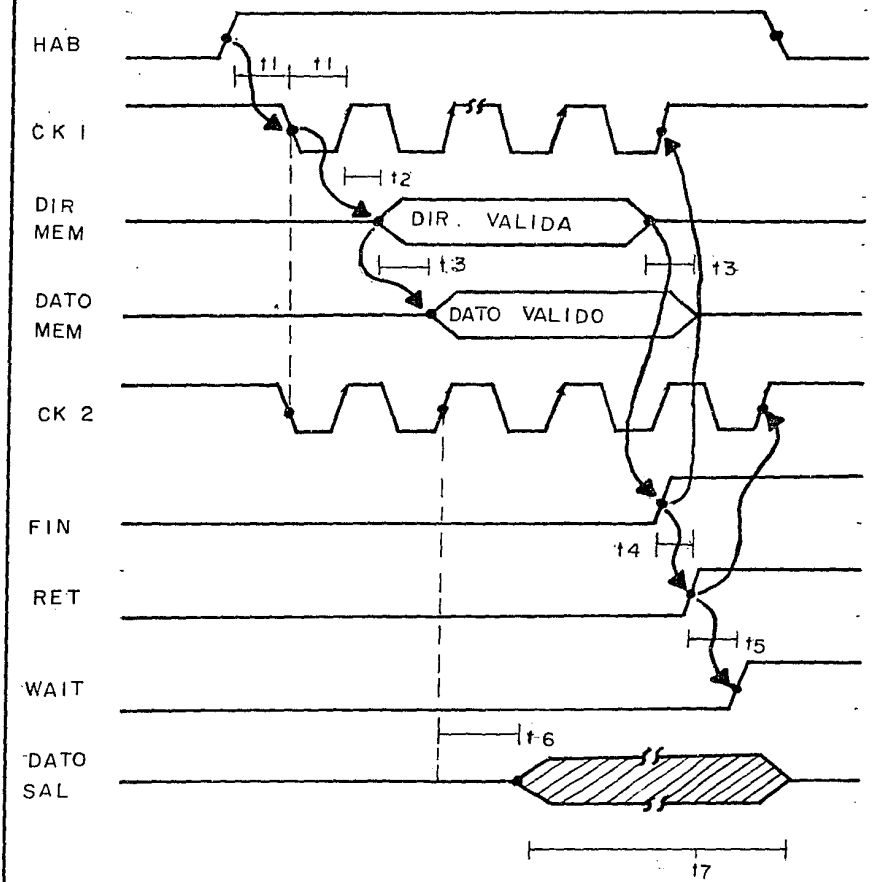


FIG. 41



HANDSHAKING

FIG.42

11. EJEMPLOS DE APLICACION.

Con el fin de validar la operación del coprocesador diseñado, se presentarán dos ejemplos de aplicación: uno sobre la implementación de un filtro de respuesta impulsional finita (FIR) y otro sobre un filtro de respuesta impulsional infinita (IIR). En ambos ejemplos, el almacenamiento de los vectores dados por los coeficientes y los valores de las muestras, se almacenarán en la memoria de la computadora a partir de la posición 6000:0000H y se usarán para la simulación 1000 datos de entrada en lenguaje de programación Turbo Pascal V 5.0.

11.1. FILTRO DE RESPUESTA IMPULSIONAL FINITA - FIR.

Se implantará un filtro pasabanda de longitud-80, en forma directa. La ecuación a implantar, es la siguiente:

$$y[n] = \sum_{k=0}^{79} h[k]^* x[n-k]$$

El filtro tiene las siguientes características:

- Pasabanda de 1375 Hz a 3625 Hz
- Frecuencia de muestreo de 10 KHz
- Región de transición de 1000 a 1375Hz y de 3625 a 4000Hz

En la simulación del filtro en la computadora PC-XT se usaron 1000 datos de entrada en punto flotante en formato de 32 bits (tipo single en Pascal) que se generaron mediante un barrido de frecuencia de 0 a 5000 Hz. El la FIG.43 se muestran estos datos

de entrada graficados. El programa que simula el filtro usando la computadora sin la tarjeta de coprocesamiento se muestra en la FIG.44 , y su respuesta de salida en la FIG.45 .

Usando la tarjeta de coprocesamiento de suma de productos, se tiene el procesamiento siguiente :

τ : Tiempo de ciclo de pipeline = 210 nsg
k : Número de etapas del pipeline = 3
n : Número de elementos del vector = 80
N : Número de tareas (sumas de productos) = 1000

t(procesamiento/tarea) = (k + n -1)τ = 17.2 μsg
T(procesamiento total) = 16.4 μsg * 1000 = 17.2 msg

La productividad (throughput) del coprocesador es la siguiente:

$$Th = \frac{n}{(k + n - 1)^* \tau} = 4.7 \text{ MFLOPS}$$

El rendimiento del coprocesador para la realización de este filtro es de :

$$\eta = \frac{n}{k + n - 1} = 98\%$$

Este mismo filtro ha sido implementado en el procesador de señales TMS 32020 y su programa se encuentra en la referencia [18].

11.2. FILTRO DE RESPUESTA IMPULSIONAL INFINITA - IIR.

Se implementará un filtro IIR Butherworth de segundo orden con frecuencia central de 1000 Hz, pasabanda de 500 Hz a 1500 Hz y frecuencia de muestreo de 10 KHz. En la referencia [19], éste filtro fué implementado en el procesador de señales Motorola 56001.

La implementación del filtro, obedece a la siguiente ecuación :

$$y[n] = \sum_{K=0}^{4} b_K * x[n-k] + \sum_{K=0}^{4} a_K * y[n-k]$$

Los datos de entrada se generaron haciendo un barrido de frecuencias de 0 a 2000 Hz como se indica en la gráfica de la FIG. 46. El programa en Pascal que simula el filtro en la computadora se muestra en la FIG. 47 y su respuesta en la FIG. 48.

Con la tarjeta de coprocesamiento se tiene el siguiente rendimiento de procesamiento :

τ : Tiempo de ciclo de pipeline = 210 nsg
k : Número de etapas del pipeline = 3
n : Número de elementos del vector = 10
N : Número de tareas (sumas de productos) = 1000

t(procesamiento/tarea) = (k + n -1)τ = 2.5 μsg
T(procesamiento total) = 2.4 μsg * 1000 = 2.5 msg

La productividad (throughput) del coprocesador es la siguiente:

$$Th = \frac{n}{(k + n - 1)^* \tau} = 3.9 \text{ MFLOPS}$$

El rendimiento del coprocesador para la realización de este filtro es de :

$$\eta = \frac{n}{k + n - 1} = 83\%$$

Para la ejecución de los programas de la FIG. 44 y FIG. 47 utilizando la tarjeta de coprocesamiento construida, se debe reemplazar la instrucción for de la suma de productos, por :

```
Port [PHAB] := $01;    { Habilita coprocesador }
  s := Port [PSTU];    { Lee puerto de status }
if s = 1  then         { Si s = 1, entonces, el coprocesador
                         concluyó su tarea }
YI[j] := Port [psal]; { Lee puerto de salida, osea, el
                         resultado }
Port [PHAB] := $00;    { Deshabilita coprocesador }
```

11.3. RESULTADOS.

El comportamiento de la unidad aritmética pipeline se muestra en la TABLA 1., en donde se aprecia que para el filtro FIR la unidad es más eficiente en ganancia, productividad y rendimiento, dado que la longitud del filtro es mayor comparado con el filtro IIR.

Los filtros fueron simulados en una computadora personal PC - XT con reloj de 4.77 MHz (210 nsg), procesador 8088 y coprocesador aritmético 8087. Los resultados de procesamiento de las 1000 muestras de entrada se presentan en la TABLA 2.

TABLA 1. COMPORTAMIENTO DE LA UNIDAD ARITMÉTICA.

	FIR - 80	IIR - 5
Sk	2.9	2.5
Th	4.7 MFLOPS	3.9 MFLOPS
η	98 %	83 %

Sk : Ganancia (speedup)

Th : Productividad (Throughput)

η : Rendimiento

TABLA 2. RESULTADOS DE LA SIMULACIÓN.

	FILTRO	Σ A * B (1)
FIR (sin cop)	17.74 sg	12.54 sg
FIR (con cop)	7.19 sg	1.99 sg
Gfir	2.47	6.27
IIR (sin cop)	1.81 sg	1.15 sg
IIR (con cop)	0.87 sg	0.21 sg
Giir	2.08	5.48

(1) El tiempo de ejecución de la unidad aritmética es de 17.2 msg para el FIR y de 2.5 msg para el IIR.

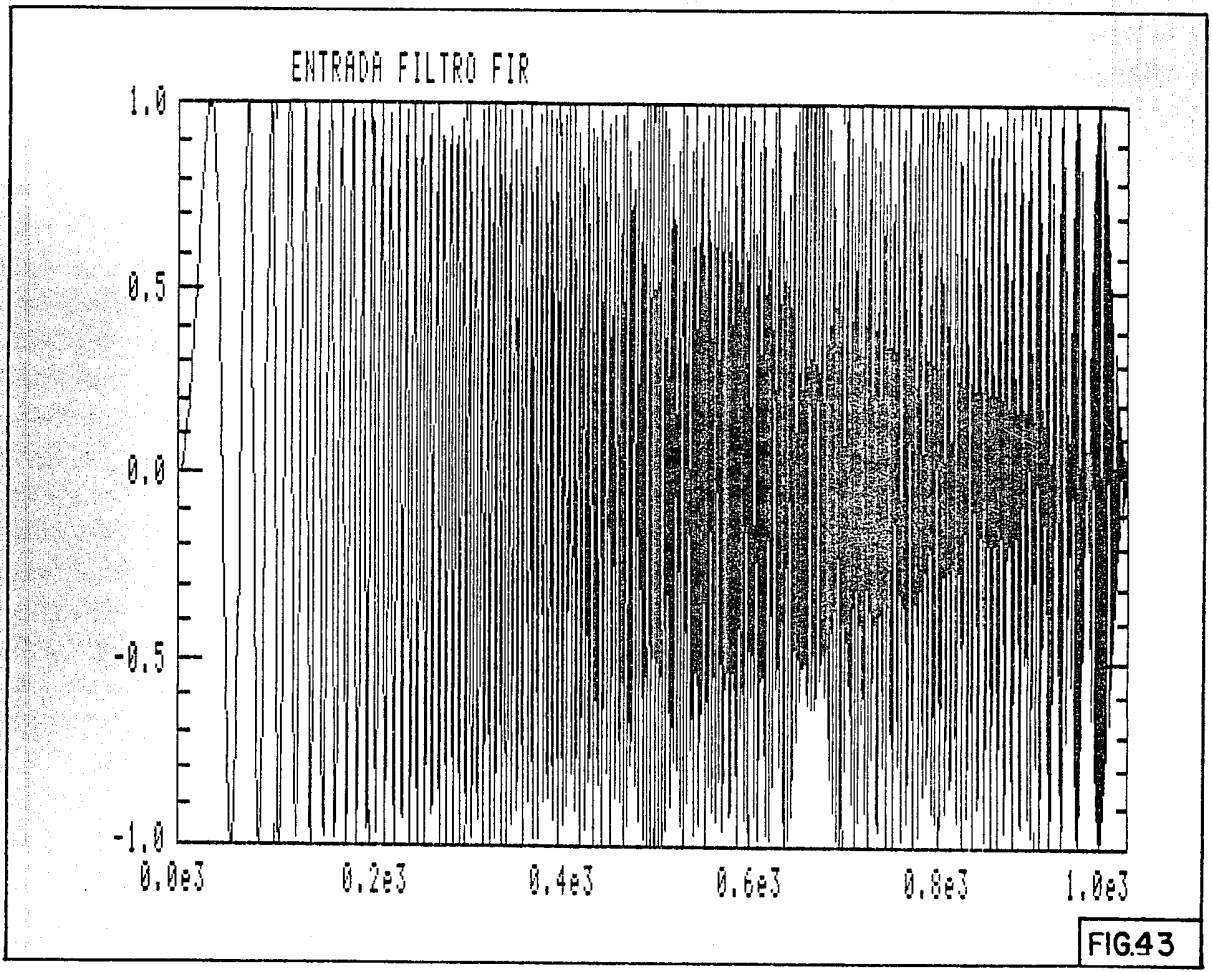

ENTRADA FILTRO FIR
1.0
0.5
0.0
-0.5
-1.0
0.0e3
0.2e3
0.4e3
0.6e3
0.8e3
1.0e3

FIG.43

```
program filtro_fir;

{ COEFICIENTES DEL FILTRO EN ARCHIVO : COEF2.DAT
  DATOS DE ENTRADA EN ARCHIVO : ENTFIR.DAT
  DATOS DE SALIDA EN ARCHIVO : SALFIR.DAT }

uses crt,dos;

const   num_dat = 1000;
        per_mues = 0.0001;
        f0 = 2500;
        df = 2500;

type

 vector1=array[1..81] of single;
 vector2=array[1..1000] of single;

var

 coefic,dato,salida,arch: text;
 t,h,x,y,x1,x2,limite   : single;
 hr,m,s,c,i,j           : word;
 XN    : vector1 absolute $6000:0000;
 HI    : vector1 absolute $6100:0000;
 XI,YI : vector2;

begin

    clrscr;
    writeln;
    writeln;
    writeln;
    writeln(' ':10,'****************************************** ');
    writeln(' ':10,'*                                        * ');
    writeln(' ':10,'*  GENERACION DE LA SENAL DE ENTRADA     * ');
```

```
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'*   BARRIDO DE FRECUENCIAS DEL SENO        * ');
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'*         DE  : 0 HZ  -  5000 HZ           * ');
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'*     NUMERO DE PUNTOS : 1000              * ');
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'*   ARCHIVO DE DATOS : ENTFIR.DAT          * ');
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'* TIPO DE DATO :PUNTO-FLOTANTE,32 BITS     * ');
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'*   POR : JORGE A. POLANIA P.              * ');
    writeln(' ':10,'*                                          * ');
    writeln(' ':10,'********************************************* ');

    assign (arch,'B:entfir.dat');
    rewrite (arch);
    j:=1;
    t := 0;
    limite := per_mues * num_dat ;
    repeat
        x1 := f0 * t + df * ( sqr(t) /limite - t);
        x2 := sin ( 2 * pi* ( x1- trunc(x1) ) );
        writeln (arch,x2);
        XI[j]:=x2;
        t := t + per_mues ;
        j:=j+1;
    until  t >  limite ;
    close (arch);

clrscr;
writeln;
writeln;
writeln;
writeln(' ':10,'********************************************* ');
writeln(' ':10,'*                                           * ');
writeln(' ':10,'*FILTRO DE RESPUESTA IMPULSIONAL FINITA-FIR* ');
```

```
 writeln(' ':10,'*                                                    * ');
 writeln(' ':10,'* PASABANDA : 1375 HZ - 3625 HZ EN  -3DB             * ');
 writeln(' ':10,'*                                                    * ');
 writeln(' ':10,'*  FRECUENCIA DE MUESTREO : 10 KHZ                    * ');
 writeln(' ':10,'*                                                    * ');
 writeln(' ':10,'*      DATOS DE ENTRADA : 1000                       * ');
 writeln(' ':10,'*                                                    * ');
 writeln(' ':10,'*     LONGITUD DEL FILTRO : 80                       * ');
 writeln(' ':10,'*                                                    * ');
 writeln(' ':10,'* TIPO  DE DATOS  : PUNTO-FLOTANTE, 32 BITS          * ');
 writeln(' ':10,'*                                                    * ');
 writeln(' ':10,'****************************************************** ');


 assign(coefic,'B:coef2.dat');
 reset(coefic);
 assign(salida,'B:salfir.dat');
 rewrite(salida);


{ LECTURA DE LOS COEFICIENTES DE DISCO A RAM }

 i:=1;
 while not eof(coefic) do
  begin
   readln(coefic,h);
   HI[i]:=h;
   i:=i+1;
  end;
 close(coefic);

 { CALCULO DEL FILTRO }

 settime(0,0,0,0);
 for i:=1 to 80 do
  XN[i]:=0;
 for j:=1 to 1000 do
```

```
  begin
   XN[1]:=XI[j];
   y:=0;
   for i:=1 to 80 do
    y:=HI[i]*XN[i] + y;
   YI[j]:=y;
   for i:=0 to 78 do
     XN[80-i]:=XN[79-i];
  end;
 gettime(hr,m,s,c);
 writeln;
 writeln;
 writeln('':10,'TIEMPO DE PROCESO:',hr,'hr',m,'min',s,'.',c,'sg');

 { ESCRITURA DE LOS DATOS DE SALIDA DE RAM A DISCO }

 for j:=1 to 1000 do
   writeln(salida,YI[j]);
 close(salida);

end.
```

FIG. 44 . SIMULACIÓN DEL FILTRO FIR EN PASCAL.

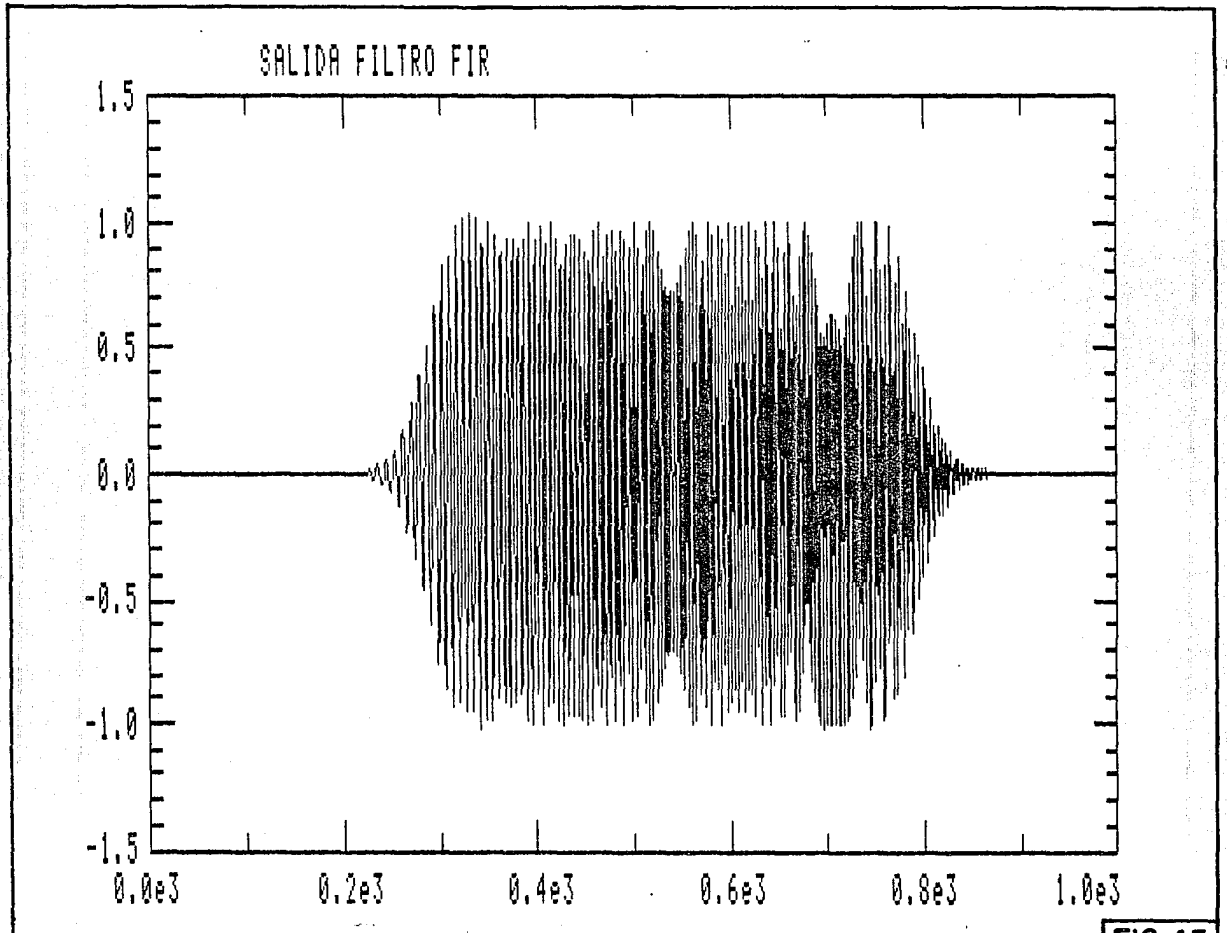

SALIDA FILTRO FIR
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0.0e3
0.2e3
0.4e3
0.6e3
0.8e3
1.0e3

FIG. 45

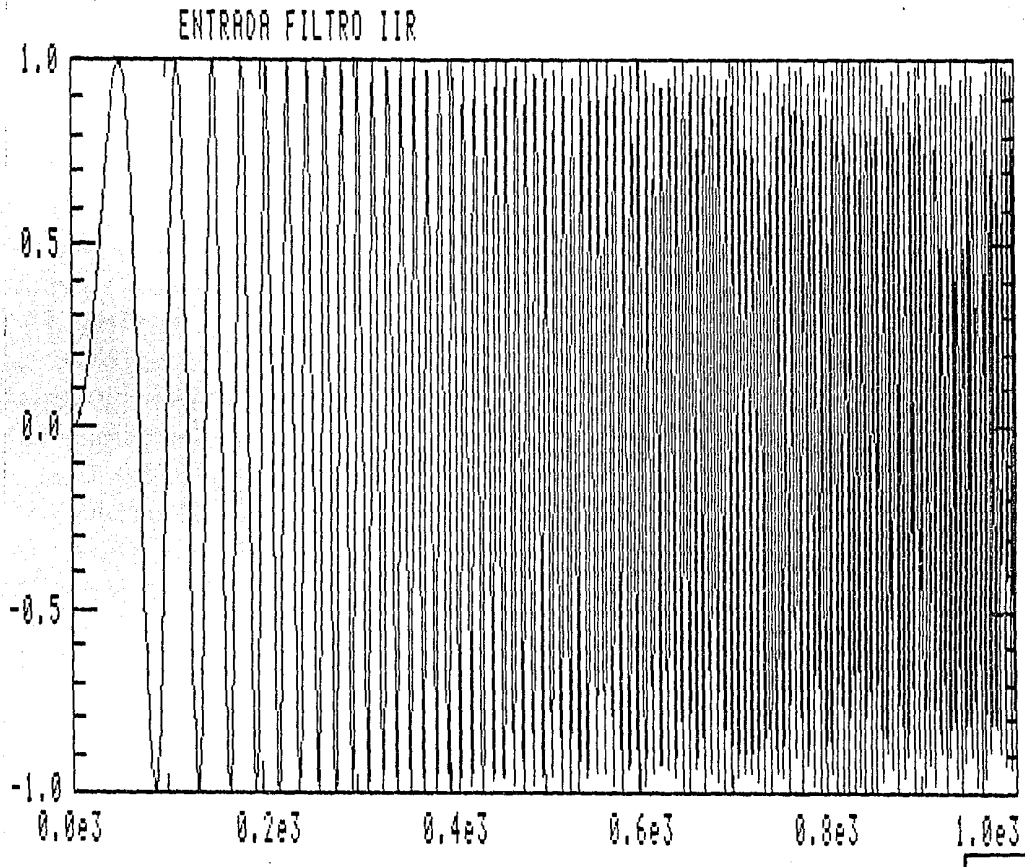

ENTRADA FILTRO IIR
1.0
0.5
0.0
-0.5
-1.0
0.0e3
0.2e3
0.4e3
0.6e3
0.8e3
1.0e3

FIG.46

```
program filtro_iir;

{ COEFICIENTES DEL FILTRO EN ARCHIVO : COEF1.DAT
  DATOS DE ENTRADA EN ARCHIVO : ENTIIR.DAT
  DATOS DE SALIDA EN ARCHIVO : SALIIR.DAT }

uses crt,dos;

const   num_dat = 1000;
        per_mues = 0.0001;
        f0 = 1000;
        df = 1000;

type
 vector1=array[1..5] of single;
 vector2=array[1..1000] of single;

var
 coefic,dato,salida,arch:text;
 t,a,b,x1,x2,x,y,limite : single;
 hr,m,s,c,i,j   : word;
 YN  :    vector1 absolute $6000:0000;
 XN :     vector1 absolute $6000:0020;
 AI :     vector1 absolute $6100:0000;
 BI :     vector1 absolute $6100:0020;
 XI,YI : vector2;

begin

    clrscr;
    writeln;
    writeln;
    writeln;
    writeln (' ':10,'************************************* ');
    writeln (' ':10,'*                                   * ');
    writeln (' ':10,'*  BARRIDO DE FRECUENCIAS DEL SENO  * ');
    writeln (' ':10,'*                                   * ');
```

```
      writeln (' ':10,'*   DE  : 0 HZ  -  2000 HZ          * ');
      writeln (' ':10,'*                                    * ');
      writeln (' ':10,'*   NUMERO DE PUNTOS : 1000          * ');
      writeln (' ':10,'*                                    * ');
      writeln (' ':10,'*   ARCHIVO DE DATOS : ENTIIR.DAT    * ');
      writeln (' ':10,'*                                    * ');
      writeln (' ':10,'* TIPO DE DATOS:PUNTO-FLOTANTE,32 BITS* ');
      writeln (' ':10,'*                                    * ');
      writeln (' ':10,'*  POR : JORGE A. POLANIA P.         * ');
      writeln (' ':10,'*                                    * ');
      writeln (' ':10,'************************************** ');

      assign (arch,'B:entiir.dat');
      rewrite (arch);
      j:=1;
      t := 0;
      limite := per_mues * num_dat ;
      repeat
           x1 := f0 * t + df * ( sqr(t) /limite - t);
           x2 := sin ( 2 * pi* ( x1- trunc(x1) ) );
           writeln (arch,x2);
           XI[j]:=x2;
           t := t + per_mues ;
           j:=j+1;
      until  t >  limite ;
      close (arch);

clrscr;
writeln;
writeln;
writeln;
writeln(' ':10,'******************************************* ');
writeln(' ':10,'*                                         * ');
writeln(' ':10,'*FILT. DE RESPUEST IMPULSIONAL INFINITA-IIR* ');
writeln(' ':10,'*                                         * ');
writeln(' ':10,'* PASABANDA :  500 HZ - 1500 HZ EN  -3DB   * ');
writeln(' ':10,'*                                         * ');
```

```
 writeln(' ':10,'*  FRECUENCIA DE MUESTREO : 10 KHZ          * ');
 writeln(' ':10,'*                                           * ');
 writeln(' ':10,'*     DATOS DE ENTRADA : 1000               * ');
 writeln(' ':10,'*                                           * ');
 writeln(' ':10,'*      NUMERO DE COEFICIENTES : 5           * ');
 writeln(' ':10,'*                                           * ');
 writeln(' ':10,'* TIPOS DE DATOS: PUNTO-FLOTANTE, 32 BITS   * ');
 writeln(' ':10,'*                                           * ');
 writeln(' ':10,'********************************************* ');


 assign(coefic,'B:coef1.dat');
 reset(coefic);
 assign(salida,'B:saliir.dat');
 rewrite(salida);

{ LECTURA DE LOS COEFICIENTES DE DISCO A RAM }

 i:=1;
 while not eof(coefic) do
  begin
   readln(coefic,b,a);
   AI[i]:=a;BI[i]:=b;
   i:=i+1;
  end;
 close(coefic);

{ CALCULO DEL FILTRO }

settime(0,0,0,0);
for i:=1 to 5 do
   YN[i]:=0; XN[i]:=0;
for j:=1 to 1000 do
  begin
   XN[1]:=XI[j]; YN[1]:= y;
   y:=0;
```

```
    for i:=1 to 5 do
     y:=AI[i] * YN[i] + BI[i] * XN[i] + y;
    YI[j]:= y;
    for i:=0 to 3 do
     begin
      YN[5-i]:=YN[4-i];
      XN[5-i]:=XN[4-i];
     end;
   end;
  gettime(hr,m,s,c);
  writeln;
  writeln;
  writeln(''':10,'TIEMPO DE PROCESO:',hr,'hr',m,'min',s,'.',c,'sg');

  { ESCRITURA DE LOS DATOS DE SALIDA DE RAM A DISCO }

  for j:=1 to 1000 do
    writeln(salida,YI[j]);
  close(salida);

end.
```

FIG. 47 . SIMULACIÓN DEL FILTRO IIR EN PASCAL.

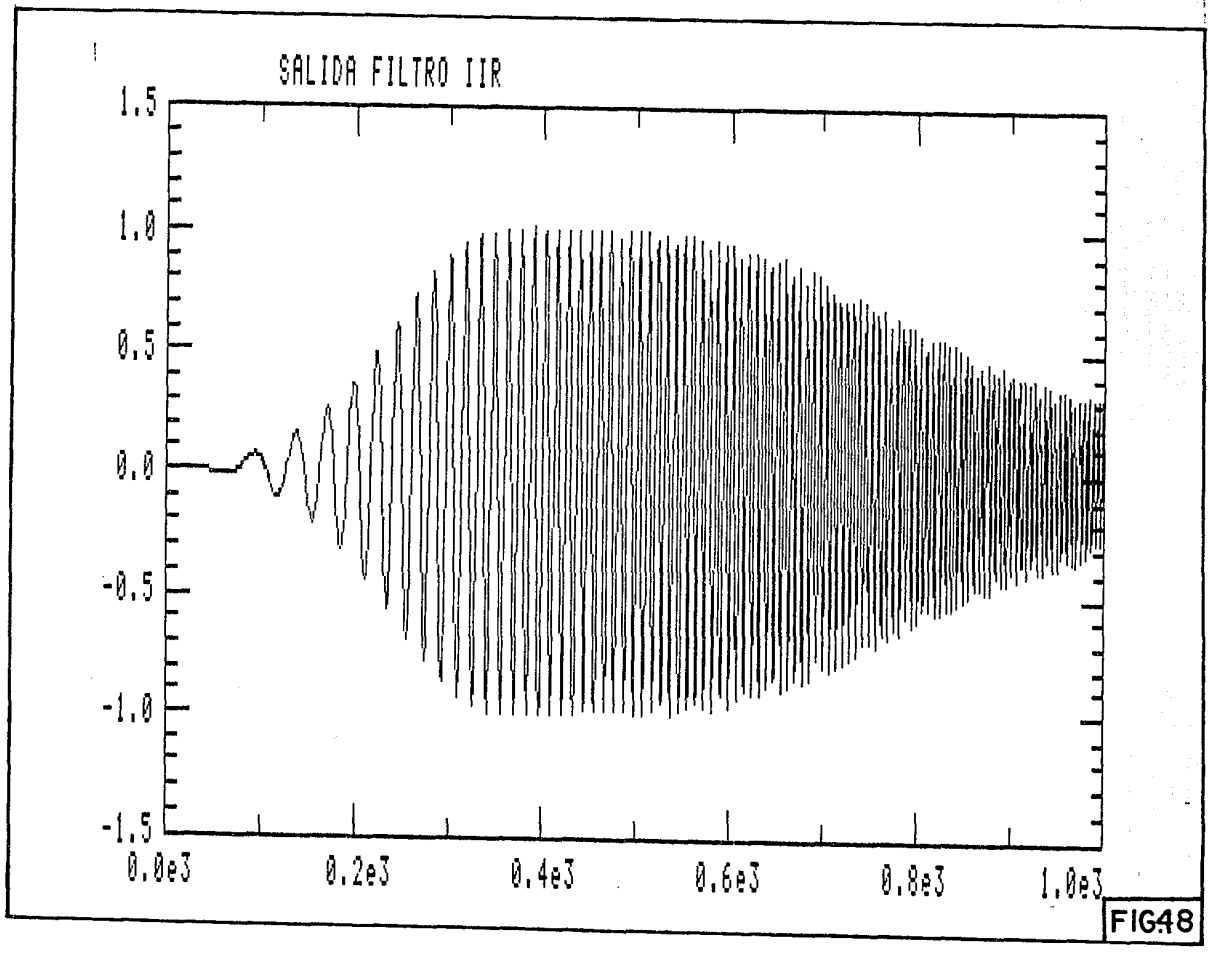

SALIDA FILTRO IIR
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0.0e3
0.2e3
0.4e3
0.6e3
0.8e3
1.0e3

FIG48

12. CONCLUSIONES Y RECOMENDACIONES.

El desarrollo de este proyecto permitió el análisis de las diferentes alternativas en la implantación del coprocesador, tales como: arquitectura más óptima, tipo de memoria a utilizar, forma de transmitir los datos a la tarjeta, tipos de procesadores digital de señales existentes, así como, los procesadores matemáticos de punto-flotante comerciales que se adecuaran al proyecto perfilado. Otro aprovechamiento alcanzado con el desarrollo del presente proyecto, fué el estudio teórico-práctico que se tuvo que realizar para conocer con alguna profundidad la arquitectura de la computadora PC-XT con el fin de hacer una buena comunicación entre el procesador de la computadora y el coprocesador construido.

Como conclusión se puede decir, que la adición de la tarjeta de coprocesamiento de suma de productos a la computadora, hizo que se aprovechara en forma eficiente el software de la computadora personal en la simulación de un algoritmo en particular y el coprocesamiento en forma hardware en la arquitectura pipeline construida. La velocidad de procesamiento se mejora sustancialmente, alcanzándose un rendimiento superior en la simulación del filtro FIR, que en la simulación del filtro IIR. Lo que nos lleva a validar la teoría que se tiene para el procesamiento pipeline, que dice que el rendimiento en una arquitectura pipeline aumenta, a medida que la longitud de los vectores de alimentación crece. Para el caso del filtro IIR los vectores tenían una longitud de 5 elementos y en el filtro FIR, los vectores eran de 80 elementos.

Dado que se cumplieron con los objetivos trazados en el proyecto, se recomienda proyectar uno nuevo, que considere las

diferentes posibilidades en cuanto a longitud de palabra (32 y 64 bits) y tipo de aritmética utilizada (punto-fijo y punto-flotante) y se cuanticen los errores que se puedan presentar en los cálculos para una aplicación especifica, como por ejemplo, un filtro, una correlación, una multiplicación de matrices, etc. Este proyecto podría ser implantado en una PC-AT que usa como procesador el 80286, 80386 o 80486 y se ganaría aún más en velocidad de procesamiento, teniendo en cuenta que la velocidad de operación del procesador es mayor (doble o triple) y el bus de datos se duplica.

Paralelo a este proyecto y con el fin de ir avanzando en el área de Arquitectura de Computadores y en el manejo de Procesadores Digital de Señales, sería importante implantar otro proyecto que integre la arquitectura de una computadora PC-AT y la de una tarjeta a diseñar que procese señales en tiempo real con un procesador como el DSP56000 (punto fijo) o el DSP96000 (punto flotante), u otro similar, como los comercializados por la Texas Instruments, como el TMS32025 o el TMS32030.

B I B L I O G R A F I A.

[1] INTEL CORP. User's Manual. March, 1983

[2] SMITH K., DRAFZ R. Turn a PC into a supercomputer with plug-in boards. Electronics Design. Nov, 1988

[3] HWANG Kai, BIGGS Tai. Computer architecture and parallel processing. McGraw-Hill, 1984

[4] HWANG Kai. Computer arithmetic: principles, architecture and design . John Wiley, 1979

[5] THORNE Michael. Programming the 8086/8088 for the IBM PC and compatibles. Publishing Company, 1987

[6] RUSSEL R. The 8086/8088 Book. McGraw-Hill, 1980

[7] JOURDAIN Robert. Programmer's solver for the IBM PC. Prentice-Hall, 1986

[8] TEXAS INSTRUMENTS. The TTL Data Book. 1984

[9] HAYES John. Digital system design and microprocessor. McGraw-Hill, 1984

[10] MOWLE F. J. A Systematic Approach to Digital Logic Design. Weley, 1976

[11] AHMAD M.O. Design of a Simple Two's Complement Multiplier Circuit. IEEE Transactions on circuits and systems. June, 1985

[12] KUNG S. Y. VLSI Array Processors. Prentice Hall, 1988

[13] SANTORO M. R. SPIM: A Pipeline 64 x 64 - bits Iterative

Multiplier. IEEE Journal of Solid State Circuits. april, 1989

[14] SUNDARARAJAN D., AHMAD M.O. Interface links 8 bits chip to provide multiprocessing. Electronics. Oct, 1983

[15] STANLEY, DOUGHERTY. Digital Signal Processing. Prentice Hall, 1984

[16] TMS32010, User's Manual. Texas Instruments. 1985

[17] DSP56000. User's Manual. Motorola. 1986

[18] KUN SHAN LIN. Digital Signal Processing Applications with the TMS320 Family. Vol 1. Texas Inst. Prentice Hall. 1987

[19] SIMPSON, VARLEY. Digital Filtering using the Motorola DSP 56000/1: A Laboratory Based Case Study. Proceeding of IEEE. 1989

[20] PHILLIPS B. Pick the right bus for PC based data acquisition. Electronics Design. April, 1989

[21] EGGEBRECHT Lewis. Interfacing to the IBM personal computer. Howards w. Sams, 1983

[22] TOSHIBA CORP. MOS Memory Products. Aug, 1985

[23] HITACHI. IC Memory Data Book. Sept, 1985

[24] PEATMAN J. B. Microcomputer Based Design. McGraw-Hill. 1981

[25] ADVANCE MICRO DEVICES. Bipolar Microprocessor Logic and Interface Data Book. 1985

[26] TEXAS INST. The New 32-bit Processor Systems. 1986

[27] ANALOG DEVICES. DSP Products Data Books. 1989

[28] FLETCHER Willian. An engineering approach to digital design. Prentice-Hall ,1980.