DIVISION DE ESTUDIOS DE POSGRADO

FACULTAD DE INGENIERIA

IMPLANTACION DE ALGORITMOS DE PROCESAMIENTO DIGITAL DE SEÑALES CON MICROPROCESADORES

JESUS SAVAGE CARMONA

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)

CIUDAD UNIVERSITARIA

OCTUBRE 1989.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

C A P I T U L O 1

1.- I N T R O D U C C I O N

El objetivo de esta tesis es presentar una serie de experimentos de procesamiento digital de señales (PDS) usando microprocesadores de PDS. Con estos experimentos se trató de cubrir algunas de las aplicaciones más características del PDS y al mismo tiempo hacer una comparación informal de tres diferentes microprocesadores comerciales de PDS. Estos experimentos fueron realizados por una parte en el Laboratorio de Procesamiento de Voz de la División de Estudios de Posgrado de la Facultad de Ingeniería de la U.N.A.M. y por la otra en el Laboratorio de Procesamiento de Imágenes de la Universidad de California en Santa Barbara.

Actualmente se está viviendo una revolución tecnológica en el campo de la electrónica y la computación, que se refleja en la forma de vida de las sociedades modernas.

La aparición de los microprocesadores de procesamiento digital de señales en los principios de los años ochentas está produciendo el mismo efecto que tuvo la aparición de los microprocesadores de propósito general en las años setenta.

Cada vez es más frecuente la incorporación de esta clase de microprocesadores en el diseño y construcción de equipo especializado abriendo para el usuario un gran número de aplicaciones. Tales aplicaciones, que antes eran ejecutadas en máquinas grandes, estaban restringidas a un cierto número de usuarios, debido principalmente a lo costoso que resultaba contar con tal equipo de procesamiento. Pero con la aparición de estos microprocesadores, prácticamente se pueden realizar las operaciones básicas de procesamiento digital de señales con sistemas simples y baratos.

Este trabajo cubre desde la fundamentación teórica de cada uno de los problemas, así como la implantación de las soluciones seleccionadas a nivel de microprocesador.

En el capítulo número 2 se presentan las principales alternativas que existen para solucionar un problema de procesamiento digital de señales que requiera uso de Hardware especializado. En el capítulo 3 se muestran tres problemas que requiren del uso tanto de algoritmos eficientes de procesamiento, así como arquitecturas especialmente diseñadas para su solución.

En el capítulo 4 se analiza la forma en que fue solucionado cada uno de los problemas planteados en el capítulo 3 y por último en el capítulo 5 se da una conclusión de los resultados obtenidos.

Los experimentos que se realizaron fueron: El reconocimiento de comandos hablados en tiempo real; el cálculo de la transformada discreta de Fourier, usando tres diferentes técnicas, en imágenes; y filtrado adaptable, aplicado a la identificación de un sistema.

C A P I T U L O 2

2.- SISTEMAS PARA EL PROCESAMIENTO DIGITAL DE SEÑALES

A diferencia de los microprocesadores de propósito general, los elementos de procesamiento de un sistema de PDS tienen sus propias caracteristicas para cumplir con los requerimientos naturales del tipo de tares que se utilizan en este tipo de procesamiento. Las caracteristicas que debe reunir un microprocesador PDS para una aplicación determinada son: un tamaño adecuado de palabra, multiplicación y acumulación rápida, memorias RAM de alta velocidad, un direccionamiento de tablas de coeficientes rápido, paralelismo de operaciones dentro del microprocesador y un mecanismo eficiente para traer la siguiente instrucción. La clave para que un Hardware de PDS sea eficiente es la de tener los elementos de procesamiento organizados de tal forma que concuerden con el algoritmo de procesamiento del problema a solucionar. Así, el reto a solucionar en una arquitectura general de un "chip" de PDS ,es el de diseñar una arquitectura de PDS, la cual pueda caber en un solo "chip" y que ademas tenga una relativa efectividad para una variedad de aplicaciones de PDS. Las operaciones más importantes en un sistema de PDS son: suma, multiplicación y multiplicación-acumulación, manejo adecuado de estructuras de datos, como son las estructuras tipo mariposa; de esta forma, es importante diseñar una unidad aritmética rápida para este tipo de operaciones. Los bloques básicos de un "chip" de PDS estan mostrados en la Fig 2.1.El diseño arquitectónico de un "chip" de PDS involucra la especificación de estos bloques básicos y la determinación de la forma en la cual ellos están interconectados. Los componentes básicos se describen a continuación :

1.- Unidad de Memoria (UM). Esta unidad consiste de una memoria de datos (MD) y memoria de coeficientes (MC). MD guarda

los datos y algunos resultados intermedios, mientras que la MC guarda coeficientes fijos, tales como los coeficientes de los filtros y las tablas de seno y coseno de la FFT y las aplicaciones de generación de señales senoidales.

2.- Unidad de Control (UC). Esta unidad consiste de un contador de programa (PC), una memoria de programa (MP), un decodificador de la instrucción (DI) y una unidad de cálculo de la dirección (UCD). El PC genera la dirección de la instrucción ha ser ejecutada. El MP guarda la instrucción de programa y el DI decodifica la instrucción para generar las señales de control para llevar a cabo la ejecución de la instrucción. La UCD calcula la dirección de los operandos bajo la dirección de las señales de control generadas por el DI.

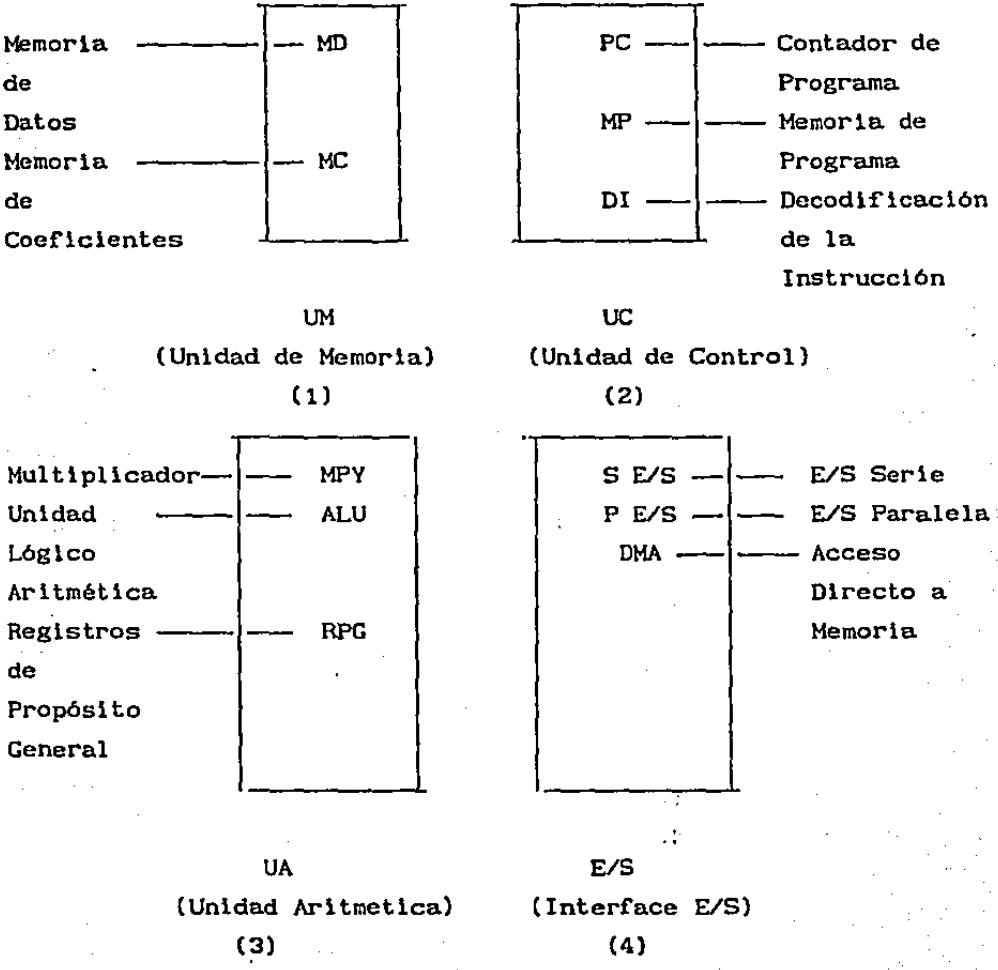


Figura 2.1.- Componentes Funcionales Básicos dc un Sistema de Procesamiento Digital de Señales.

3.- Unidad Aritmética.- La unidad aritmética consiste de un Multiplicador (MPY), una Unidad Lógico Aritmética (ALU) y Registros de Propósito General (RPGs). El MPY es generalmente un

multiplicador paralelo. El ALU realiza la suma, resta, corrimiento y algunas funciones lógicas. Los RPGs retienen datos temporales y usualmente estan directamente conectados al bus de datos y a los otros bloques de construcción básicos. En el caso más sencillo, hay sólamente un registro, el cual es llamado acumulador.

4.- La interface E/S. Esta interface incluye la unidad de E/S serie (S-E/S), la unidad paralela de E/S (P-E/S), y el controlador de acceso directo a memoria (DMA).

Las principales medidas de funcionamiento de un sistema basado en un PDS son precisión y velocidad.

Precision.- La precisión lograda es dictada por el tamaño de la palabra de las UMs y la UA. Los algoritmos de PDS usualmente son iteractivos, y los errores en cálculos intermedios son acumulativos. Para evitar errores excesivos, gran precisión debe ser mantenida durante la ejecución. En aplicaciones de tiempo real, puede ser imposible detectar y corregir errores una vez que ellos ocurren. Asi que la precisión resulta ser crucial. La mayoria de los PDS comerciales de la primera generación cuentan con tamaños de palabras de 16 bits y los de la segunda generación cuentan con tamaños de palabras de hasta 24 bits (Motorola 56000).

Velocidad.- La velocidad de un "chip" PDS depende de varios factores, tales como el tiempo del ciclo de instrucción y el poderio del conjunto de instrucciones soportado por la arquitectura. Si toma N instrucciones para completar la ejecución de un algoritmo, entonces el tiempo total necesario para que éste algoritmo sea ejecutado es Nxt, donde t es el tiempo de ciclo de instrucción. Hay que notar que al final la velocidad de operación es el producto de N por t, asi que un tiempo de ejecucion corto t

no es suficiente para tener una buena operación. Frecuentemente, para la mayoría de los "chips" PDS , un ciclo de instrucción es dictado por la velocidad del multiplicador. Así que se debe tratar de completar todas las demás operaciones, como la traída de otra instrucción (Fetch), decodificación de la instrucción, traída de los operandos, guardar los resultados en la memoria de datos, en el tiempo que se tarda en hacer una multiplicación.

Para que se pueda realizar procesamiento encauzado (Pipeline), cada bloque básico opera en paralelo; así que ellos deben estar interconectados cuidadosamente para así evitar interferencias en el bus. Una palabra de instrucción grande es necesaria para decodificar las señales de control rápidamente para cada unidad que opera en paralelo. Es deseable tener las memorias de datos y de coeficientes separadas de tal forma que se pueda accesar dos operandos al mismo tiempo.

Los factores que influyen en decisiones tecnológicas y costos son consumo de energía, área, velocidad y funcionalidad. Es bien conocido que para una tecnología determinada, entre más rápido sea un circuito más energía consume, simplemente porque los transistores conducen más frecuentemente. Entre más funciones efectúe el dispositivo mayor es el área que se necesita debido al gran número de subcircuitos y sus interconecciones. El procesador de señales puede ser más rápido si la información consumida (input) y la salida (output) es transmitida en paralelo. Esto significa que se requiere un gran número de conexiones con el mundo exterior Para circuitos integrados esto significa un gran número de *pines*. Con la tecnología actual, el empaquetado puede ser más caro que el propio "chip", y el costo del empaquetado depende del número de *pines*. Un número grande de *pines* requiere de un número grande de circuitos de E/S en el "chip"; estos circuitos consumen más energía y hacen más lenta la operación. A continuación se presentan algunas consideraciones importantesen el

diseño de una arquitectura:

1.- Una UA capaz de ejecutar eficientemente la operación de multiplicar y acumular, ya que este tipo de operación se utiliza frecuentemente en la mayoría de los algoritmos de PDS.

2.- Una arquitectura capáz de soportar un estructura de procesamiento encauzado,ya que los algoritmos PDS pueden estar altamente encauzados.

3.- La transferencia de datos entre la memoria y el UA debe estar cuidadosamente balanceado, tal que la efectividad del sistema no se vea degradada por este cuello de botella.

4.- Es necesario un buen lenguaje de programación para soportar procesamiento de arreglos.

5.- Son necesarios puertos de entrada y salida para proporcionar el medio de comunicación entre el mundo exterior.

6.- Se deben considerar diseños de transmisión de datos en serie o en paralelo.

2.1.- DISEÑO DE SISTEMAS UTILIZANDO HARDWARE ESPECIALIZADO

La mayoría de los microprocesadores tienen una estructura y un comportamiento perfectamente definido durante el proceso de fabricación. Si se tiene la necesidad de realizar operaciones que no estan definidas en el microprocesador por medio de Hardware, entonces se tienen que realizar por medio de software. Por supuesto, esto es mucho más lento que la realización directa por Hardware. La velocidad de la mayoría de los microprocesadores también esta limitada por su utilización de tecnología de circuitos MOS, que posibilita una alta densidad de componentes a expensas de rebajar la velocidad de operación con respecto ,por ejemplo, a la máxima velocidad de reloj del CPU permisible. Así, las velocidades realmente altas que se pueden conseguir con varias tecnologías TTL y bipolar ECL no estan disponibles con los microprocesadores estándar. Ademas en el procesamiento digital de señales se requiere tener arquitecturas especializadas que respondan de acuerdo a una aplicación determinada. La modularidad resuelve estos problemas proporcionando bloques constructivos flexibles para desarrollar sistemas de procesamiento digital de señales que pueden aprovechar las ventajas de las tecnologías de circuitos integrados actuales más veloces.

El procesamiento en paralelo y arquitecturas con procesamiento encauzado, usando las tecnologías más rápidas, es el requisito para obtener una alta efectividad en aplicaciones de procesamiento digital de señales.

Diversos fabricantes ofrecen diferentes tipos de módulos que estan encaminados a solucionar las aplicaciones del procesamiento digital de señales. Estos módulos se pueden organizar de acuerdo a la aplicación determinada, dejando en libertad al diseñador de construir su propia arquitectura. La persona o grupo de personas

encargados del diseño de este tipo de sistemas, deberán tener una idea clara de la forma en la que se interconectarán los diferentes módulos, las líneas de conexión que se utilizarán, los diferentes micropasos que se tendrían que realizar para completar una instrucción de lenguaje ensamblador, además serían los encargados de definir el conjunto de instrucciones de lenguaje ensamblador que el sistema se encargará de ejecutar.

Por ejemplo la familia de componentes Am29500 de la compañía Advanced Micro Devices incluyen filtros de respuesta a impulso finita (FIR) e infinita (IIR), procesadores de la transformada rápida de Fourier (FFT), procesadores gráficos, multiplicadores de 16x16 bits de alta velocidad, generadores de funciones seno y coseno, multiplicadores/acumuladores, procesadores de punto flotante. La figura 2.2 muestra un procesador de señales de alta velocidad. Este procesador es construido enteramente con elementos de la familia de procesamiento digital de señales Am29500 y de la familia Am2900.

Dependiendo de la instrucción del ensamblador que se trate, el controlador de microprograma ejecutará los pasos necesarios para llevar a cabo ésta, la memoria de microcódigo contiene todas las microinstrucciones necesarias para los diferentes módulos del sistema. El tamaño de esta memoria es muy grande (entre 60 a 120 bits). Cuando se diseña utilizando esta clase de componentes el tiempo de diseño de un sistema de procesamiento digital de señales es más lento que cuando se utiliza microprocesadores del procesamiento digital de señales, debido principalmente a que prácticamente se debe tener una idea muy clara del funcionamiento del Hardware tanto como del Software. La ventaja de este tipo de sistemas con respecto a los microprocesadores del procesamiento digital de señales, es que pueden ser más rápidos e incluso más económicos que estos ultimos.

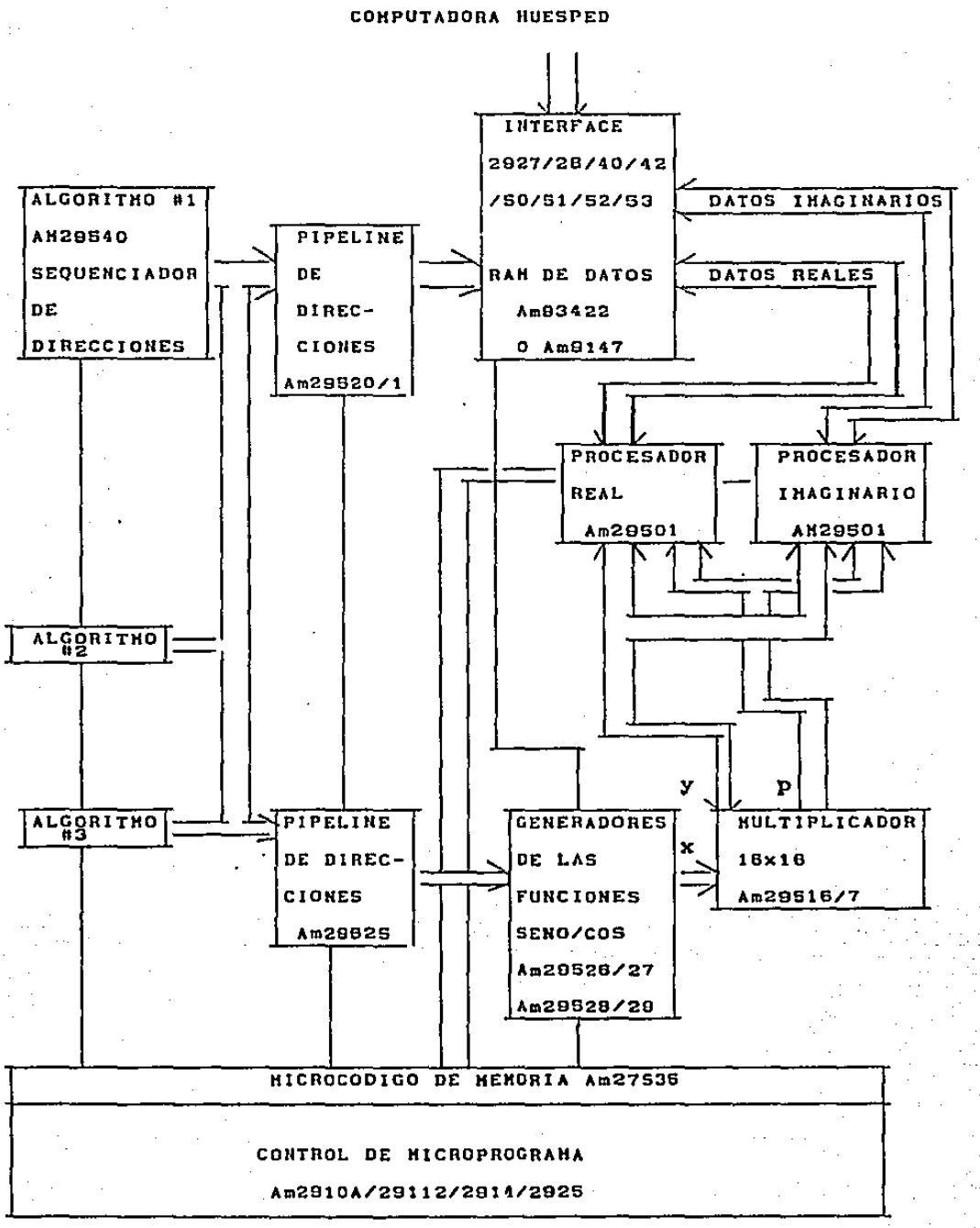


Figura 2.2.- Procesador de Procesamiento Digital de Señales

2.2.- MICROPROCESADORES PARA EL PROCESAMIENTO DIGITAL DE SENALES.

Al principio de los años 1980s, la tecnología 3μm MOS hizo posible la fabricación de dispositivos PDS de un solo chip programables que contenían todos los ingredientes básicos del Hardware de un PDS: el multiplicador, RAM de datos, ALU rápido, memoria de programa, etc. Tipicamente estos dispositivos contenían cerca de 50000 transistores los cuales eran interconectados para ejecutar hasta 5 MIPS (millones de instrucciones por segundo). Era posible utilizar el mismo dispositivo en múltiples aplicaciones sólamente con cambiar el programa. La amplia aceptación que tuvieron estos dispositivos tuvo como resultado que se introdujera la segunda generación basados en tecnología 2μm MOS, integrando hasta 150000 transistores. Estos dispositivos son más veloces, contienen más memoria y sus características son más parecidas a los microprocesadores convencionales que los de la primera generación. Las ventajas en costo y flexibilidad que ofrecen estos dispositivos hacen posible que los dispositivos PDS emigren al campo comercial. Produciendo la misma clase de revolución en el diseño de sistemas de PDS como los microprocesadores lo hicieron en el campo de la computación de propósito general. En los siguientes capítulos se analizan tres microprocesadores de PDS comerciales con los cuales he estado trabajando en diversas aplicaciones en el campo del procesamiento digital de señales.

2.2.1.- NEC μPD7720

Fabricado en NMOS de alta velocidad, el μPD7720 es una microcomputadora de 16 bits en un chip. Memoria ROM es proporcionada para guardar el programa y coeficientes/datos, mientras que cuenta con una memoria RAM para almacenar datos temporales coeficientes y resultados. El poder computacional es proporcionado por un ALU de 16 bits y un multiplicador paralelo de 16 x 16 bits. Esta combinación permite la construcción de sumas de productos en un solo ciclo de instrucción de 250ns. Dos puertos en serie de entrada/salida para poder hacer la interface a codecs y algún otro dispositivo orientado serialmente, además cuenta con un puerto en paralelo. Señales de protocolo incluyendo controles de DMA, permite actuar al μPD7720 como un periférico programable sofisticado, tanto como una microcomputadora en forma aislada.
Características :

* Tiempo de ejecución 250 ns
* Palabra de 16 bits
* Instrucciones de multioperación para la optimización de la ejecución del programa
* Memoria ROM de programa 512 X 23 bits
* Memoria ROM datos/coeficientes 510 X 13 bits
* Memoria RAM interna 128 X 16 bits
* Multiplicador de 16 x 16 con 31 bits de resultado
* Dos acumuladores
* Pila de subrutinas de 4 niveles
* Capacidades múltiples de entrada/salida

  Entrada/salida serie (8 o 16 bits)

E/S paralela

DMA

* Tecnología NMOS

En la figura 2.3 se muestra el diagrama de bloques del microprocesador. Este microprocesador cuenta con una memoria ROM de intrucciones de 512x23 bits, direccionada por un contador de programa de 9 bits. Tiene una pila de 4 localidades. Tiene una memoria RAM de 128 palabras de 16 bits que se direccionan a través de un apuntador de 7 bits (DP). Las características de direccionamiento del DP hacen que este opere simultáneamente con instrucciones aritméticas, así que no es necesario perder tiempo para modificaciones en las direcciones. La memoria de datos puede ser mejor comprendida como dos bloques de memoria, cada uno de ellos con cuatro renglones de 16 palabras.
La memoria de Datos/Coeficientes ROM es organizada en 512 palabras por 13 bits y es direccionada por un apuntador de 9 bits (RP). Esta memoria es ideal para guardar los coeficientes necesarios, tablas de conversión y constantes para el procesamiento. El multiplicador puede efectuar operaciones entre datos de 16 bits para producir soluciones de 31 bits expresadas en complemento a dos con un bit de signo + 30 bits de datos. Una multiplicación puede ser efectuada cada ciclo de instrucción.
Una característica importante en un microprocesador digital de señales es que cuente con diferentes y flexibles dispositivos de entrada/salida. El sistema de entrada/salida del μ7720, concebidos como esclavos, son configurados por software con un tamaño de 8 o 16 bits. Se cuenta con banderas para una revisión interna del

status de E/S, los cuales permiten una sincronización eficiente con el flujo interno del programa. Canales tipo serie de entrada y de salida con su correspondientes lineas de protocolo, cubren las especificaciones estándares de CCITT y AT&T para 2Mbps usando Full-Duplex PCM. La posibilidad de enviar los datos en orden al revés durante la transferencia, incrementa el número de dispositivos serie que pueden ser conectados al µ7720 (ADC, DAC, Codec, FIFO, etc). El puerto paralelo de E/S de 8 bits puede ser usado para transferir datos o leer estatus del microprocesador.

El ALU es una unidad que trabaja en complemento a dos que puede efectuar 16 operaciones distintas. Cuenta con dos acumuladores de 16 bits (AccA, AccB) que guardan los resultados de las operaciones del ALU. Cada acumulador tiene su propio set de banderas que es actualizado después de cada instrucción aritmética. Este microprocesador tiene tres tipos de instrucciones: Una instrucción que es utilizada para realizar operaciones aritméticas, transferencia de datos y además es posible regresar de llamados a subrutinas, este tipo de instrucción es la que más se utiliza y si se hace eficientemente permite realizar hasta siete tareas en un ciclo de instrucción (ver el ejemplo más adelante), el segundo tipo de instrucción es la carga inmediata (LDI) de datos a un destino haciendo uso del bus interno y por último una instrucción en donde es posible hacer saltos condicionales, incodicionales y llamados a subrutinas. Todas las instrucciones necesitan solamente un ciclo de 250 ns usando un reloj de 8 MHz. Durante la ejecución de la presente instrucción, se trae y se decodifica la siguiente instrucción en paralelo. Las funciones de entrada y salida son ejecutadas independientemente del flujo interno del programa. A continuación se presenta un ejemplo tipico de una instrucción que muestra el paralelismo de ejecución de este microprocesador. Este

ejemplo muestra la transferencia de datos a través del bus de datos interno, suma el contenido de la salida del multiplicador (registro M) al acumulador, además modifica los apuntadores de memoria de datos RAM y ROM, y finalmente regresa del llamado de subrutina.

Ejemplo:

```
OP      MOV   @KLM,SI    *RAM(DP+64) ->K,SI-> L
        ADD   ACCA,M     *M+ACCA -> ACCA
        DPINC            *INCREMENTA EL APUNTADOR DPL
        RPDEC            *DECREMENTA EL APUNTADOR RP
        M1               *MODIFICA EL APUNTADOR DPH
        RET             ;*REGRESO DE LA SUBRUTINA
```

Estas 7 instrucciones son ejecutadas en un ciclo de instrucción. El μPD7720 es un microcomputador de un solo chip, sin embargo, es también diseñado para operar como un periférico complejo en una microcomputadora. Tres configuraciones son mostradas en las figuras 2.4-5-6. La primera es con el microprocesador operando como un periférico complejo, introduciendo y sacando datos serialmente así como comunicandose con el bus del sistema de una microcomputadora. El segundo ejemplo muestra dos μPD7720 en una configuración de cascada y el último ejemplo es una aplicación con un μPD7720 trabajando solo.

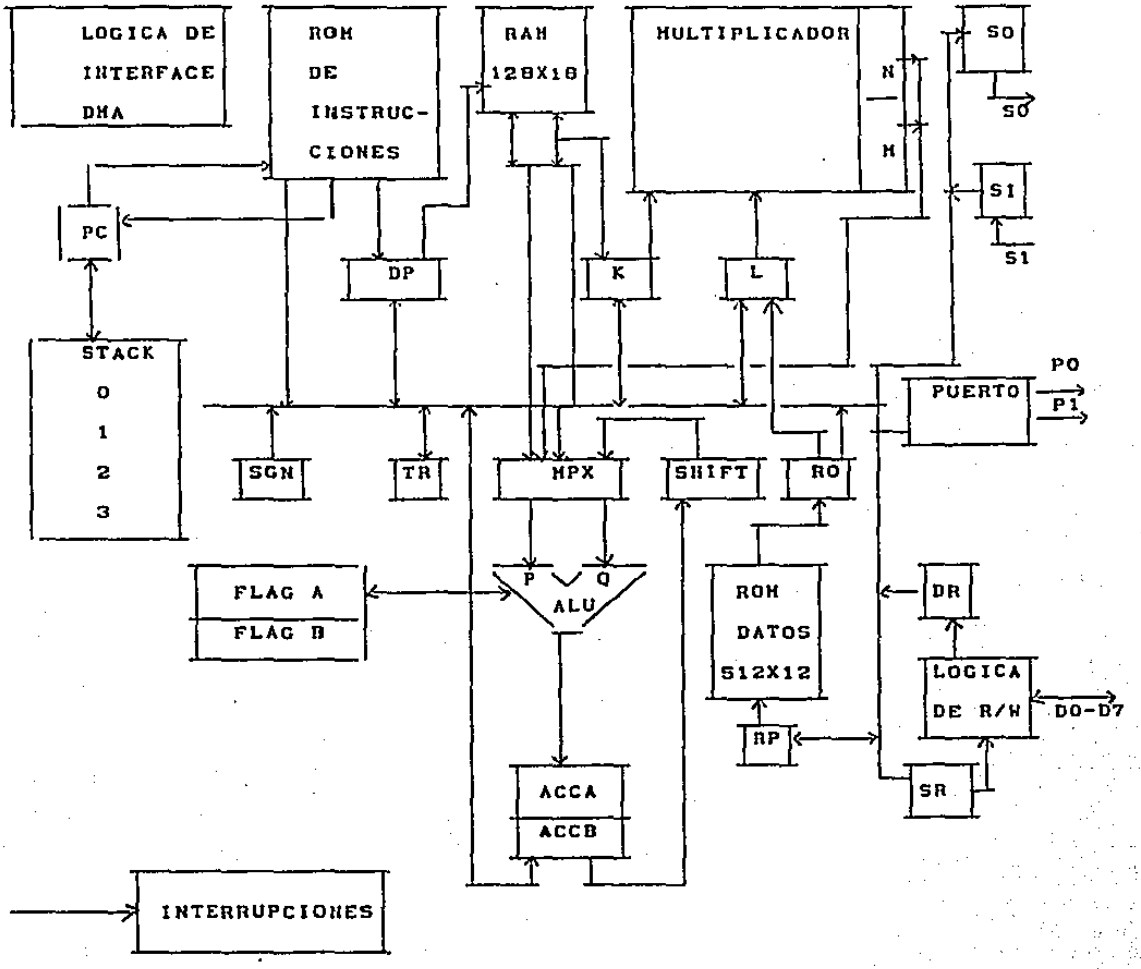


Figura 2.3.- Diagrama de bloques de μPD7720

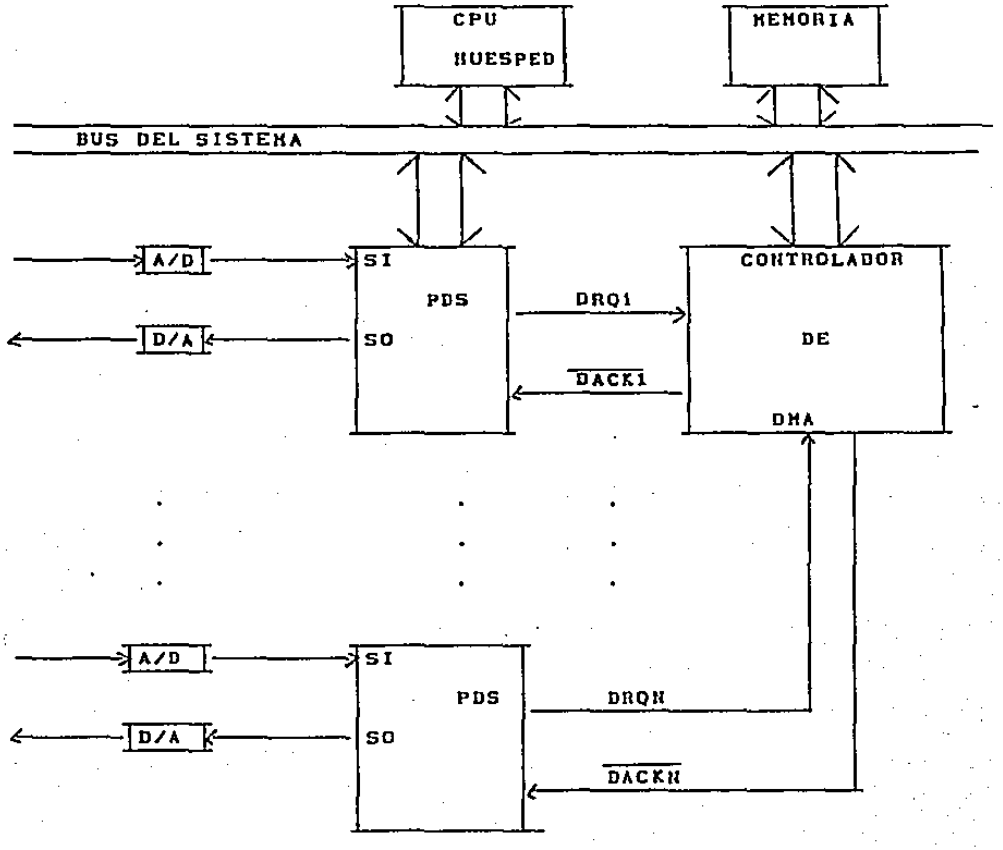


Figura 2.4.- Uso del μPD7720 como un periférico complejo de computadora.

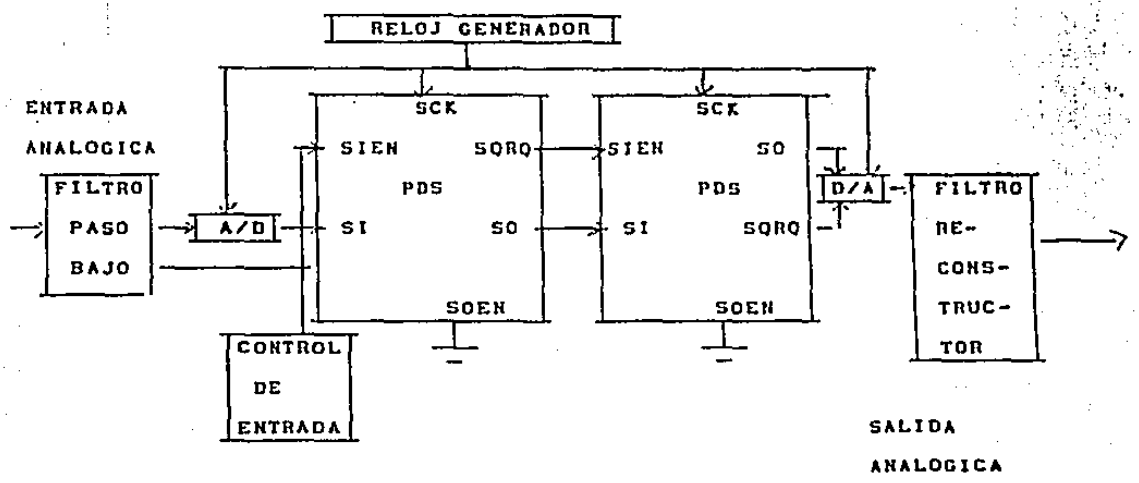


FIGURA 2.5.- Sistema de procesamiento digital de señales usando dos μPD7720 en cascada, comunicación a traves de los puerto serie

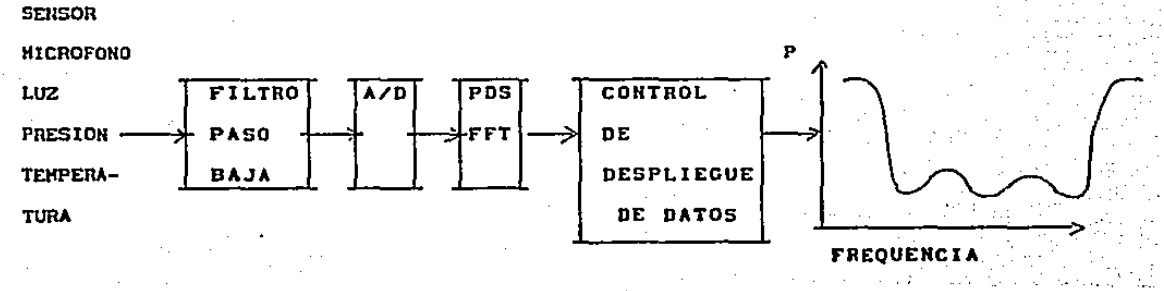


Figura 2.6.- Sistema de análisis espectral usando el μPD7720.

2.2.2.- FAMILIA TMS320

2.2.2.1.- LA PRIMERA GENERACION DE LA FAMILIA TMS320

La familia de procesadores para el procesamiento digital de señales TMS320 de 16/32 bits combina la flexibilidad de un controlador de alta velocidad con la capacidad numérica de un procesador de arreglos, ofrece una alternativa barata para la persona que utiliza componentes VLSI y procesadores BIT-SLICE. EL TMS32010 es el primer procesador de procesamiento digital de señales de la familia TMS320 que fué introducido en 1983. Su conjunto de instrucciones es bastante poderoso, su flexibilidad inherente, su capacidad para realizar operaciones a altas velocidades, su arquitectura ha hecho que se pueda tener todas estas capacidades, es una solución ideal para aplicaciones en telecomunicaciones, computación, comerciales, industrales y muchas otras. La familia TMS320 ha sido ahora expandida en tres generaciones de procesadores: TMS320C1x, TMS320C2x y TMS320C3x. Muchas características existen en común entre estas generaciones. La compatibilidad en el software es mantenida através de la familia como protección del usuario. Cada procesador tiene herramientas de software y hardware que facilitan un diseño rápido.

DESCRIPCION GENERAL

La combinación de una arquitectura tipo Harvard (separación de los buses de datos y de programa) y su conjunto de instrucciones especial para el procesamiento digital de señales proporciona rapidéz y flexibilidad para producir un microprocesador capaz de ejecutar 6.25 MIPS (Millón de instrucciones por segundo). La familia de TMS320 optimiza velocidad por medio de la creación de funciones en hardware que otros microprocesadores las realizan en software o en microcódigo. La siguiente tabla presenta una

revisión del grupo de procesadores TMS320C1x con comparaciones de tecnología, memoria, Entrada/Salida, ciclos de reloj.

Dispositivo	Tecn.	Memoria				E/S		Ciclos
		Dentro del chip			Fuera del chip	SER	PAR	(ns)
		RAM	ROM	EPROM	PROG			
TMS32010-14	NMOS	144	1.5K	-	4K	-	8x16	280
TMS32010	NMOS	144	1.5K	-	4K	-	8x16	200
TMS32010-25	NMOS	144	1.5K	-	4K	-	8x16	160
TMS32011	NMOS	144	1.5K	-	-	2	6x16	200
TMS320C10	CMOS	144	1.5K	-	4K	-	8x16	200
TMS320C10-25	CMOS	144	1.5K	-	4K	-	8x16	160
TMS320C15	CMOS	256	4K	-	4K	-	8x16	200
TMS320C15-25	CMOS	256	4K	-	4K	-	8x16	160
TMS320E15	CMOS	256	-	4K	4K	-	8x16	200
TMS320C17	CMOS	256	4K	-	-	2	6x16	200
TMS320C17-25	CMOS	256	4K	-	-	2	6x16	160
TMS320E17	CMOS	256	-	4K	-	2	6x16	200

Algunas de las características de los dispositivos TMS320C1x son listadas a continuación. Se encierran en paréntesis los dispositivos que cumplen con la característica especificada.

Ciclo de reloj para una instrucción
160ns (TMS32010-25/C10-25/C15-25/C17-25)

200ns (TMS32010/C10/11/C15/E15/C17/E17)

280NS (TMS32010-14)

144/256 palabras de memoria de datos dentro del chip
1.5/4k palabras de memoria de pograma ROM dentro del chip
4k palabras de memoria de programa EPROM dentro del chip
Código de protección para la memoria EPROM
4K de memoria externa total a máxima velocidad

Bus de datos de 16 bits con una razón de transferencia de 50-Mbps
32 bits ALU/acumulador
Multiplicador de 16 x 16 bits con un producto de 32 bits
Corrimientos de 0 a 16 bits
Generador de reloj dentro del chip
Canales de entrada y de salida de ocho bits
Puerto en serie de canal doble con reloj (TMS32011 /C17/E17)
Interface directa a los codecs(TMS32011/C17/E17)
Ley de compancion Mu/A dentro del chip armada en Hardware (TMS32011/C17/E17)

Interface de coprocesador de 16 bits (TMS320C17/E17)
Fuente de 5-V

Tecnología :
NMOS (TMS32010/11)
CMOS (TMS320C10/C15/E15/C17/E17)

Versiones comerciales y militares

Aplicaciones Tipicas

Debido a su versatilidad y a que puede ser utilizado en aplicaciones de tiempo real la familia TMS320 puede ser aplicado en una variedad de aplicaciones. Se enlistan a continuación aplicaciones tipicas.

Propósito General DSP

1.- Filtrado Digital
2.- Convolución
3.- Correlaciones
4.- Transformada de Hilbert
5.- Transformada rápida dc Fourier
6.- Filtrado adaptivo
7.- Ventaneo
8.- Generación de formas de onda

Gráficas/Imágenes

1.- Rotación en 3 dimensiones
2.- Visión de Robots

3.- Transmisión de Imágenes/Compresión
4.- Reconocimiento de Patrones
5.- Acrecentamiento de Imágenes
6.- Procesamiento Homomorfónico
7.- Estaciones de Trabajo
8.- Transformación de Animación/Digital

Instrumentación

1.- Análisis Espectral
2.- Generación de Funciones
3.- Igualación de Patrones
4.- Procesamiento Sísmico
5.- Análisis de transitorios
6.- Filtrado Digital

VOZ

1.- Correo de Voz
2.- Vocoders
3.- Reconocimiento de la Voz
4.- Verificación del Parlante
5.- Acrecentamiento de la Voz
6.- Voz a texto

Control

1.- Control de discos
2.- Control de Servomecanismos
3.- Control de Robots
4.- Control de Impresoras Laser
5.- Control de Ingenios
6.- Control de Motores

Telecomunicaciones

1.- Cancelación de Eco
2.- Transcoderes ADPCM
3.- Digital PBXs
4.- Repetidores de Lineas
5.- Multiplexores de Canal
6.- Modems de 1200 a 19200 bps
7.- Ecualizadores Adaptivos
8.- Encodificación/Decodificación DTMF
9.- Encriptografía
10.- FAX
11.- Teléfonos Celulares

12.- Teléfonos Parlantes
13.- Voz Digital
14.- Interpolación (DSI)
15.- X.25 Packet Switching
16.- Video Conferencias
17.- Comunicaciones

Automotores

1.- Control del Motor
2.- Análisis Vibratorio
3.- Control adaptivo del manejo
4.- Posicionamiento Global
5.- Navegación
6.- Comandos con Voz
7.- Radio Digital
8.- Teléfonos Celulares

Consumidor

1.- Detector de Radares
2.- Herramientas de Poder
3.- Audio/TV digitales
4.- Sintetizador de Música
5.- Juegos Educacionales

Industria

1.- Robots
2.- Control Numérico
3.- Acceso de Seguridad
4.- Monitores de Potencia de Línea

Médicos

1.- Ayuda para sordos
2.- Monitoreo de pacientes
3.- Equipo de Ultasonido
4.- Herramientas de Diagnóstico
5.- Prostetics
6.- Monitoreo de Fetos

2.2.2.2.- LA SEGUNDA GENERACION DE LA FAMILIA TMS320

La segunda generación de procesadores de procesamiento digital de señales incluye dos miembros, el TMS32020 y el TMS32025. La arquitectura de estos dispositivos ha evolucionado del TMS32010, el primer miembro de la familia TMS320 . Las características de la segunda generación de TMS320 son las siguientes:

- Tiempo del ciclo de instrucción
  - 100 ns (TMS320C25)
  - 200 ns (TMS320C20)
- 4K palabras de memoria ROM dentro del chip (TMS320C25)
- 544 palabra de memoria RAM dentro del chip.
- 128k palabras de memoria de programa RAM
- Ocho registros auxiliares con una unidad aritmética
- Pila de ocho niveles
- Puerto en serie doble
- Estados de espera para comunicación con memorias lentas
- Puerto en serie para multiprocesamiento o para conectarlo a Codecs
- Modos de direccionamiento para poder efectuar la transformada rápida de Fourier(TMS320C25)
- DMA concurrente utilizando una operación de detenido extendida (TMS320C25)
- Aritmética de precisión extendida y soporte para filtrado adaptable (TMS320C25)
- Operación a velocidad completa de instrucciones MAC/MACD para memorias externas (TMS320C25)
- Bit de acarreo del acumulador e instrucciones relacionadas (TMS320C25)

El diagrama de bloques funcional mostrado en la Figura 2.7 describe los bloques principales y las rutas de los datos dentro de los procesadores TMS320C2x. La arquitectura del TMS320C2x es construida alrededor de dos buses principales: el bus de programa y el bus de datos. El bus de programa acarrea el código de instrucción y operandos inmediatos de la memoria de programa. El bus de datos interconecta varios elementos, tales como la Unidad Central Lógico Aritmética (CALU, siglas en ingles) y el archivo

de registros auxiliares, a la RAM de datos. Los buses de datos y programas pueden acarrear datos de la memoria interna del chip o de la memoria externa de programa al multiplicador en un ciclo sencillo para operaciones de multiplicación/acumulación.

El TMS320C2x tiene un alto grado de paralelismo; mientras los datos son operados en el CALU, operaciones aritméticas pueden también ser realizadas en la Unidad Aritmética de los Registros Auxiliares (ARAU). Tal paralelismo resulta en un conjunto poderoso de operaciones aritmético-lógicas y de manipulacion de bits en que todas ellas pueden ser manipuladas en un ciclo sencillo de máquina. El Hardware interno realiza funciones que otros microprocesadores tipicamente realizan por medio de software o microcódigo. El TMS320C2x proporciona un total de 544 16 bits palabras de memoria de datos RAM interna, de las cuales 288 palabras siempre son memorias de datos y las restantes 256 palabras pueden ser configurados ya sea como memoria de datos o o bienmemoria de programa. El TMS320C25 también proporciona 4k palabras de memoria 4K palabras de memoria ROM. El TMS320C2x puede direccionar un total de 64K palabras de memoria de datos. La memoria de datos interna y las localidades reservadas son asignadas al espacio de memoria de datos de las localidades bajas de 1K. La memoria de datos es directamente expandible hasta 64k palabras manteniendo la misma velocidad de operación. Una linea de READY es proporcionada para hacer la interface con memoria lentas y menos costosas. El TMS320C2x proporciona tres espacios separados para memoria de programa, memoria de datos y para operaciones de entrada/salida. Además proporciona instrucciones para movimiento de bloques de programa o de datos que eficientemente utiliza la memoria configurable dentro del chip.

La Unidad Central Lógico Aritmética (CALU) contiene un registro de corrimiento de escalamiento, un multiplicador paralelo de 16x16 bits, una unidad lógico aritmética (ALU) de 32 bits, un acumulador de 32 bits (ACC), y registros de corrimiento

adicionales a las salidas del acumulador y del multiplicador. Utiliza un multiplicador de 16x16 bits, el cual es capáz de calcular productos de datos con signo o sin signo de 32 bits como resultado en un ciclo sencillo de máquina . La salida de producto puede ser escalado a la izquierda por 1 o 4 bits, esto es útil para las operaciones con aritmética fraccional o para justificar fracciones de productos. Hay dos instrucciones para realizar multiplicacciones y al mismo tiempo acumular el resultado en el acumulador. Los datos de estas operaciones pueden residir en cualquier lugar de la memoria externa o interna, o puede ser transferido al multiplicador cada ciclo vía los buses de programa y de datos.

Contiene un contador de programa (PC) de 16 bits y un pila de cuatro (TMS32020) o ocho (TMS320C25) localidades. El PC direcciona la memoria interna y externa para traer las instrucciones que seran ejecutadas. Cuenta además con una caracteristica que permite la ejecución de una instrucción por N+1 veces.

El procesamiento encauzado de las instrucciones (PIPELINE) consiste en una secuencia de operaciones con el bus externo que ocurre durante la ejecución de las instrucciones. El procesamiento encauzado de la traida de la instrucción decodificación y ejecución es escencialmente invisible para el usuario, exepto en algunos casos cuando el encauzamiento debe ser roto(cuando por ejemplo ocurre una instrucción de salto). En la operación del encauzamiento, la traida de la instrucción, decodificación y ejecución son independientes, lo cual permite ejecución de las instrucciones se traslapen.Así, durante un determinado ciclo, dos o tres instrucciones pueden estar activas, cada una de ellas en un estado diferente de completación. En la figura 2.8 se presenta la operación del encauzamiento para el TMS320C25.

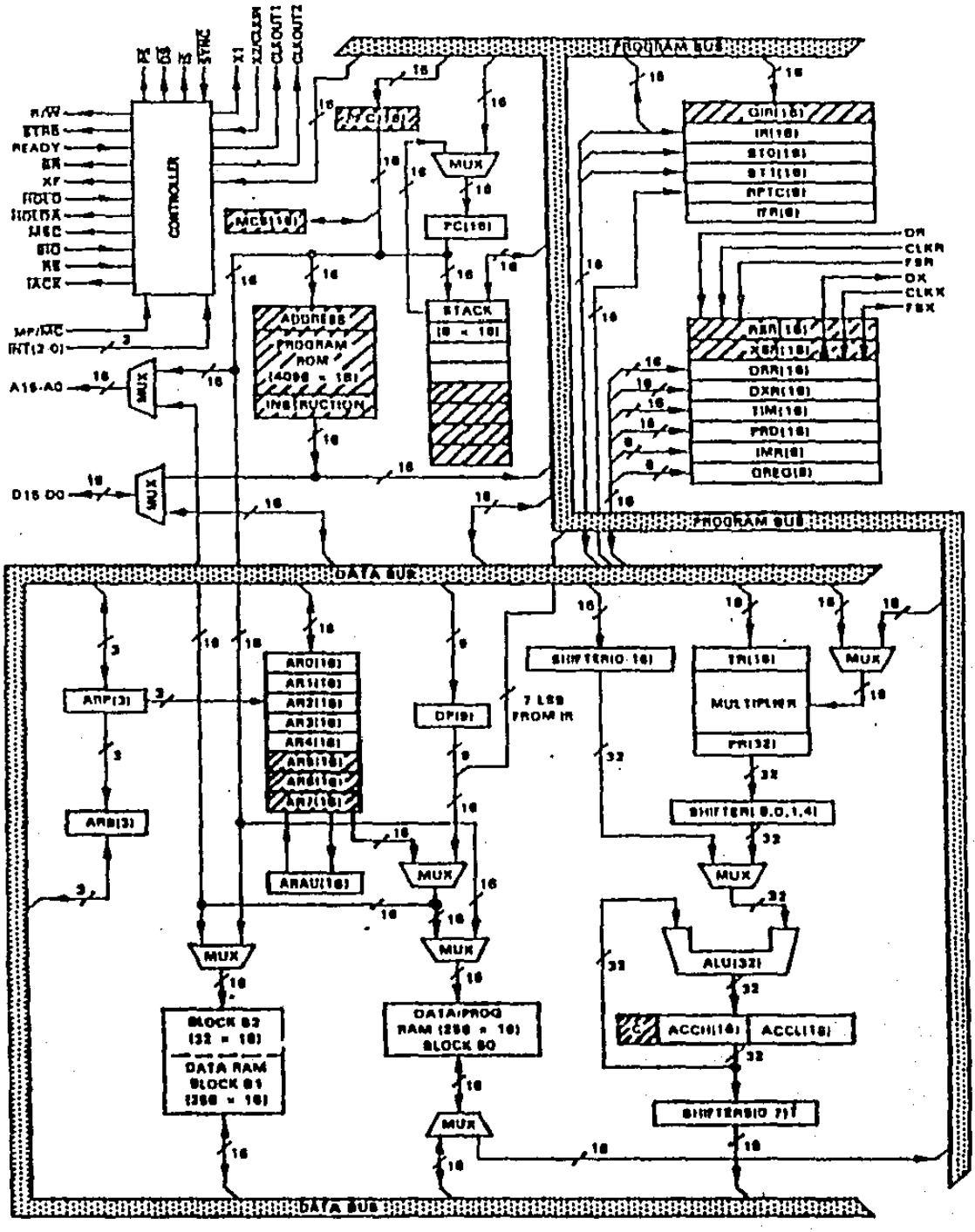


Figura 2.7.- Diagrama de bloques del TMS320C2x

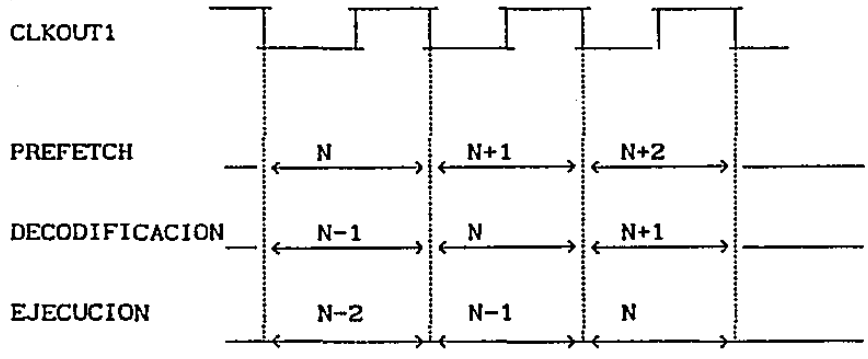


Figura 2.8.- Operación de tres niveles en el encauzamiento.

2.2.2.3.- LA TERCERA GENERACION DE LA FAMILIA TMS32020

El TMS320C30 es el miembro de la tercera generación de procesadores de preocesamiento digital de señales de la familia TMS320. Con una capacidad de 33 MFLOPS (Million Floating-Point Operation per Second), el TMS320C30 excede ampliamente la operación de cualquier procesador de procesamiento digital de señales en la actualidad (1988). El tiempo de procesamiento es dramáticamente reducido con la disponibilidad de operaciones de punto flotante, instrucciones de propósito general y herramientas de desarrollo. Características principales del TMS320C30 :

* Tiempo de ejecución de 60ns

* Dos bloques de memoria RAM de 1K x 32 bits

* Un bloque de memoria RAM de 4K x 32 bits

* 64 x 32 bits de memoria cache

* Palabras de datos e instrucciones de 32 bits, y de direcciones de 24 bits

* Multiplicador de punto flotante y entero de 32/40 bits

* ALU de punto flotante, enteros y lógicos de 32/40 bits

* Corrimientos de 32 bits

* Ocho registros de precisión extendida

* Dos generadores de direcciones con ocho registros auxiliares

* Controlador de acceso directo a memoria (DMA) interno para operaciones concurrentes de I/O y CPU

* Buses de periféricos y módulos para un uso sencillo

* Soportado por lenguajes de alto nivel

* Instrucciones interconectadas para poder realizar multiproceso

2.2.3.- MOTOROLA 56000

El DSP56000 de Motorola es un microprocesador para el procesamiento digital de señales construido con tecnología HCMOS, de bajo consumo de energía. La base de este microprocesador consiste de tres unidades de ejecución: El ALU de los datos, el ALU de Direcciones, y el Controlador de Programa, los cuales operan en paralelo. Las características de este microprocesador se presentan a continuación:

* 10.25 Millones de instrucciones por segundo
* ALU de un solo ciclo de operación, sin encauzamiento
  - Multiplicación/Acumulación paralela de 24x24 -> 56bits
  - 10 registros de datos
  - 2 buses de datos con limitadores y corrimientos
* ALU de direcciones orientado al PDS
  - 24 registros de direcciones
  - Módulo doble de la unidad aritmética
  - Generación de direcciones tipo lineal, módulo, al revés
* Controlador de programa Avanzado
  - Una pila en Hardware de 15 niveles
  - Iteraciónes DO realizadas por Hardware
* Conjunto de instrucciónes altamente ortogonal
  - 62 tipos de instrucción
  - El procesamiento encauzado es invisible al usuario
  - Diseñado para tener compiladores de alto nivel
* Buses Múltiples
  - 4 Buses de Datos
  - 3 Buses de Direcciones
  - Tiempos de acceso fuera del chip programable
* Periféricos dentro del chip
  - Puerto de 24 pins de E/S programable

Interface de ocho bits paralela con un huesped MPU/DMA
Puerto de comunicación serial con generador de baudaje
Puerto serie sincrono(codec) con generador de reloj

* Memoria dentro del chip
  Dos RAMs de datos independientes de 256x24 bits
  Dos ROMs de datos independientes de 256x24 bits
  2Kx24 bits de ROM de programa.
* Memoria de expansión fuera del chip
  128Kx24 Memoria de Datos
  64Kx24 Memoria de programa

La arquitectura del DSP56000 ha sido diseñada para maximizar su operación cuando se hace un uso intensivo de datos. Este objetivo ha resultado en una arquitectura dual expandible con periféricos E/S internos sofisticados y de propósito general. Es dual porque permite tener dos espacios de memoria de datos expandibles, dos unidades aritméticas de direcciones, y en el ALU de datos se cuenta con dos acumuladores. La dualidad de la arquitectura facilita escribir Software para aplicaciones de PDS. Por ejemplo los datos son particionados en espacios X y Y para aplicaciones de gráficas y procesamiento de imágenes, en convoluciones teniendo los coeficientes y los datos en cada uno de estos espacios, y datos reales e imaginarios para realizar aritmética compleja.
El programdor puede ver la arquitectura del DSP56000 como tres unidades de ejecución el ALU de datos, el ALU de direcciones, y el controlador de programa. Los principales componentes del DSP56000 son los siguientes:

Buses de Datos
Buses de Direcciones
ALU de Datos
ALU de Direcciones
Memoria de Datos X

Memoria de Datos Y

Controlador de Programa

Memoria de Programa

Entrada/Salida

- Entrada/Salida de propósito general
- Periféricos de Interface y puerto de DMA-Interface con un Huesped
- Interface de comunicación serie tipo RS232
- Interface serial sincrona para Codecs
- Interrupciones externas mascarables por programa
- Bus de Memoria Externa
- Circuito oscilador de cristal dentro del chip.

En la figura 2.8 se muestra la arquitectura de este microprocesador.

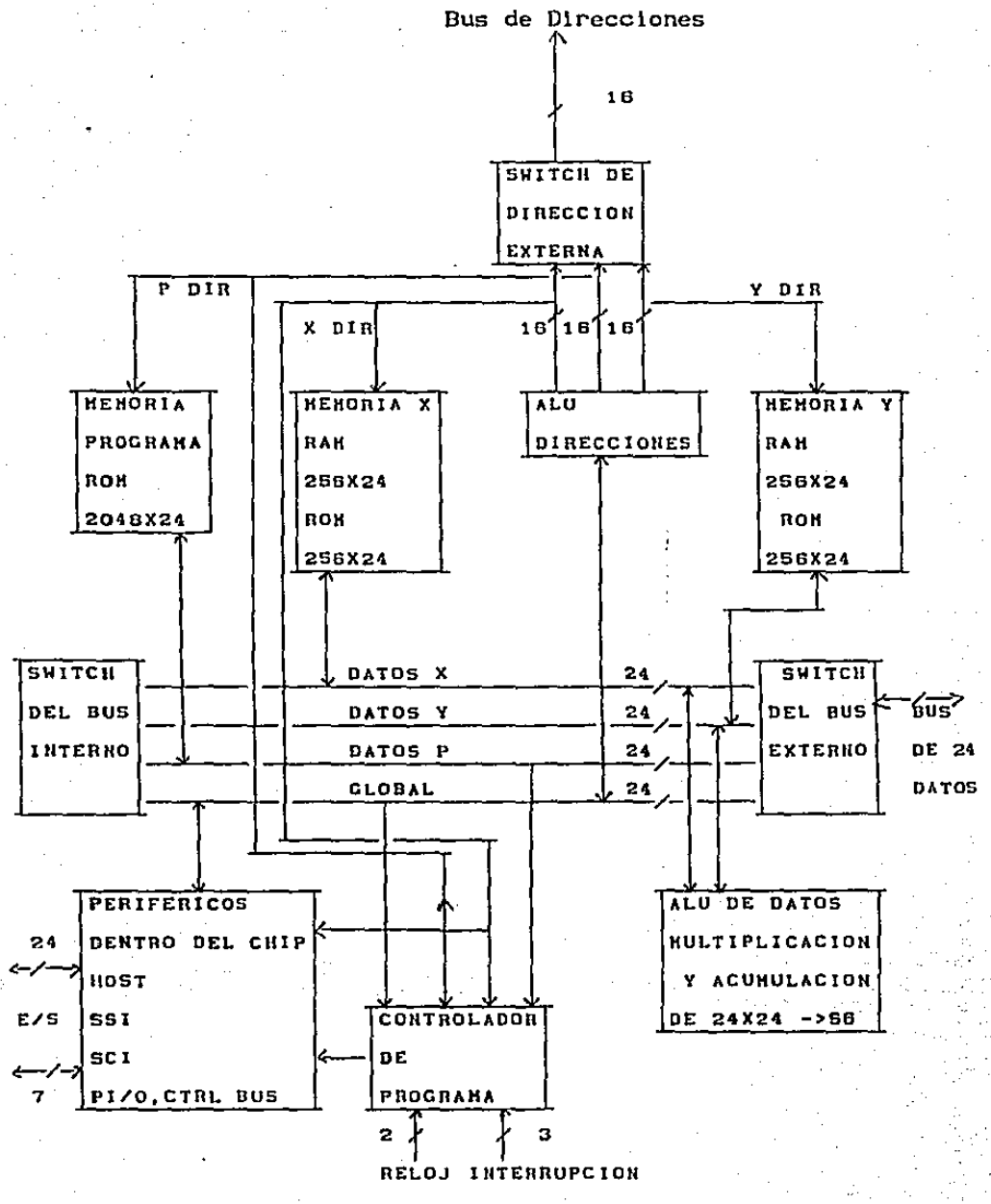


Figura 2.8 Arquitectura 56000

El movimiento de datos en el DSP56000 ocurre sobre tres buses bidireccionables de 24 bits: Bus de datos X (XD), Bus de datos Y (YD), y el bus de datos global (GD). Los buses X y Y pueden ser tratados también, en ciertas instrucciones como un bus de 48 bits de datos con la concatenación de los buses XD y YD. Tranferencia de datos entre los espacios de memoria X y Y y el ALU ocurre a través de los buses XD y YD. Todas las demás transferencias ocurren sobre el bus de datos global.

Las direciones internas son especificadas por medio de los buses de direcciones de X (XA) y de Y (YA). Espacios de memoria externos son direccionados por medio de un bus de direcciones unidireccional manejado por un multiplexor de tres entradas que puede seleccionar entre el bus de direcciones X (XA),el bus de direcciones Y (YA), o el bus de dirección de programa. Cuando se utiliza memoria externa, un ciclo de instrucción es necesario para cada acceso de memoria externo.

El ALU de datos realiza todas las operaciones aritméticas y lógicas. Consiste de 4 registros de 24 bits,dos acumuladores de 48 bits,dos extensiones de los acumuladores de 8 bits, un registro de corrrimiento, dos buses de datos con corrimiento/limitador, y una unidad de Multiplicación/Acumulación en paralelo. El ALU es capaz de realizar las operaciones de multiplicación, multiplicación-acumulación con acumulación positiva y negativa, suma, resta, corrimiento y operaciones lógicas en un ciclo de instrucción. Los operandos fuentes del ALU pueden ser de 24, 48, o 56 bits y puden provenir de los registros del ALU. Los resultados del ALU siempre son guardados en los registros del ALU. Las operaciones del ALU siempre tienen como resultado 56 bits.

El ALU de direcciones realiza todo el almacenamiento de direcciones y calculaciones efectivas de direcciones necesario para direccionar los operandos en memoria. El ALU de direcciones cuenta con 8 registros de direcciones (R0-R7), 8 registros de

offset (N0-N7), y 8 registros de modificación (M0-M7). Cada registro de dirección es de 16 bits. Los registros de modificación y de offset son utilizados para controlar la actualización de los registros de dirección.

La lógica de control de programa realiza la operación de traída de las instrucciones a ser ejecutadas (Prefetch), decodificación de la instrucción, y procesamiento de las excepciones. El contador de programa de 16 bits puede direccionar hasta 65536 localidades.

La ejecución de las instrucciones se hace usando procesamiento encauzado para permitir que estas se ejecuten cada ciclo de instrucción. Sin embargo, algunas instrucciones requieren un tiempo adicional para ser ejecutadas. El procesamiento encauzado permite que las operaciones de traer una instrucción, decodificación, y ejecución ocurra durante la traída de otra instrucción.

Mientras que una instrucción es ejecutada, la siguiente instrucción ha ser ejecutada es decodificada, y la siguiente instrucción de la instrucción que esta siendo decodificada es traída de la memoria de programa.

2.3.- PROCESAMIENTO EN PARALELO

2.3.1.- TRANSPUTERS

El INMOS transputer chip es un microprocesador que cuenta con el Hardware necesario para realizar procesos y comunicación concurrentes, la base de computación de arreglos. En la figura 2.10 se muestra el transputer INMOS T424. Este chip es claramente una computadora completa. Adopta el diseño popular de una arquitectura de computadora con un conjunto de instrucciones reducido (RISC,siglas en inglés). Este tipo de computadoras tienen un conjunto de instrucciones más simple que las mayoría de computadoras y evita tener caracteristicas que requieran complejas estructuras de control. Las caracteristicas más importantes de una arquitectura tipo RISC son:(1) Un conjunto de instrucciones regular y simple, que permite el uso de las mismas unidades de Hardware simples, para casi todas las instrucciones. (2) La mayoría de las instruciones se ejecutan en un ciclo de instrucción (Pipeline). Estas arquitecturas estan orientadas a registros: Todas las operaciones en los datos objetos son realizadas en los registros. Solamente las instrucciones de carga y guardado accesan la memoria.(3) Instrucciones de tamaño fijo con una variedad de formatos pequeña es usada. Las ventajas de una arquitectura tipo RISC proviene de la cercana iteración entre el hardware y la puesta en marcha de las aplicaciones. Las ventajas de RISC incluye: decodificación rapida de la instrucción, ejecución de casi todas las instruciones en un solo ciclo y con un periodo corto de reloj . Las desventajas potenciales de un sistema RISC proviene del tiempo de acceso a la memoria y el software adicional que se requiere. En aplicaciones para el procesamiento digital de señales una de las desventajas de la arquitectura RISC es cuando, para obtener instrucciones rápidas de multiplicación y

multiplicación/acumulación, éstas requerirán probablemente más de un ciclo de instrucción y una cantidad sustancial de microcódigo. El transputer INMOS T424 tiene un procesador de 32 bits capáz de 10 MIPS, 4 kbytes de 50 ns RAM estática y, de gran significación, cuenta con una variedad de interfaces de comunicación. Estas hacen posible la construcción de varias redes de procesos. Utiliza un mecanismo de transferencia de bloques de DMA que transmite mensajes entre la memoria y otros transputers vía los canales de comunicación. Las ligas de comunicación y el procesador operan concurrentemente, permitiendo que el procesamiento continúe mientras que los datos son transferidos sobre todas las ligas. Los datos son transmitidos por medio de estas interfaces a una velocidad de hasta 1.5 Mbytes/s. Los datos son transmitidos como una secuencia de bytes, cada byte siendo reconocido por el receptor antes de que el próximo sea transmitido. Ya que los procesos deben ser independientes excepto cuando estan en comunicación, este protocolo es escencial de acuerdo al modelo computacional. Permite que los procesos corran asincronamente , solamente sincronizandose cuando ellos necesitan comunicarse. El transputer debe ser visto como un poderoso block de construcción del cual nuevos dispositivos concurrentes pueden ser construidos. Por supuesto tiene varias limitaciones para muchas de las aplicaciones del procesamiento digital de señales. Por ejemplo no cuenta con un procesador de punto flotante y además solamente cuenta con cuatro conecciones para comunicarse. Los transputers cuentan con un tamaño de palabra de punto fijo de 32 bits, una precisión adecuada para la mayoría (pero no para todas) de las aplicaciones del procesamiento digital de señales. Arreglos de transputers pueden ser utilizados para que realizen computación concurrente a la velocidad que requieran las aplicaciones de procesamiento digital de señales. Tomando la FFT como ejemplo, para cubrir todo el espectro de audio hasta 100 khz,se

necesitarían 6, 8, y 10 transputers para una FFT de 64, 256, 1024 puntos respectivamente. Un punto débil de los transputers en aplicaciones de procesamiento digital de señales es que en este tipo de aplicaciones se requiere una unidad aritmética rápida y versatil,y un direçcionamiento de alta velocidad de tablas, y en este tipo de operaciones el transputer no es del todo efectivo. Para eliminar estas limitaciónes, especialmente en la multiplicación, se podria usar externamente un dispositivo especializado de alta velocidad que se encargará de la multiplicación. Existe un lenguaje de alto nivel llamado Occam el cual permite escribir programas concurrentes; se pueden definir procesos independientes que utilizan definición de variables locales que se pueden comunicar solamente via los canales de comunicación declarados.

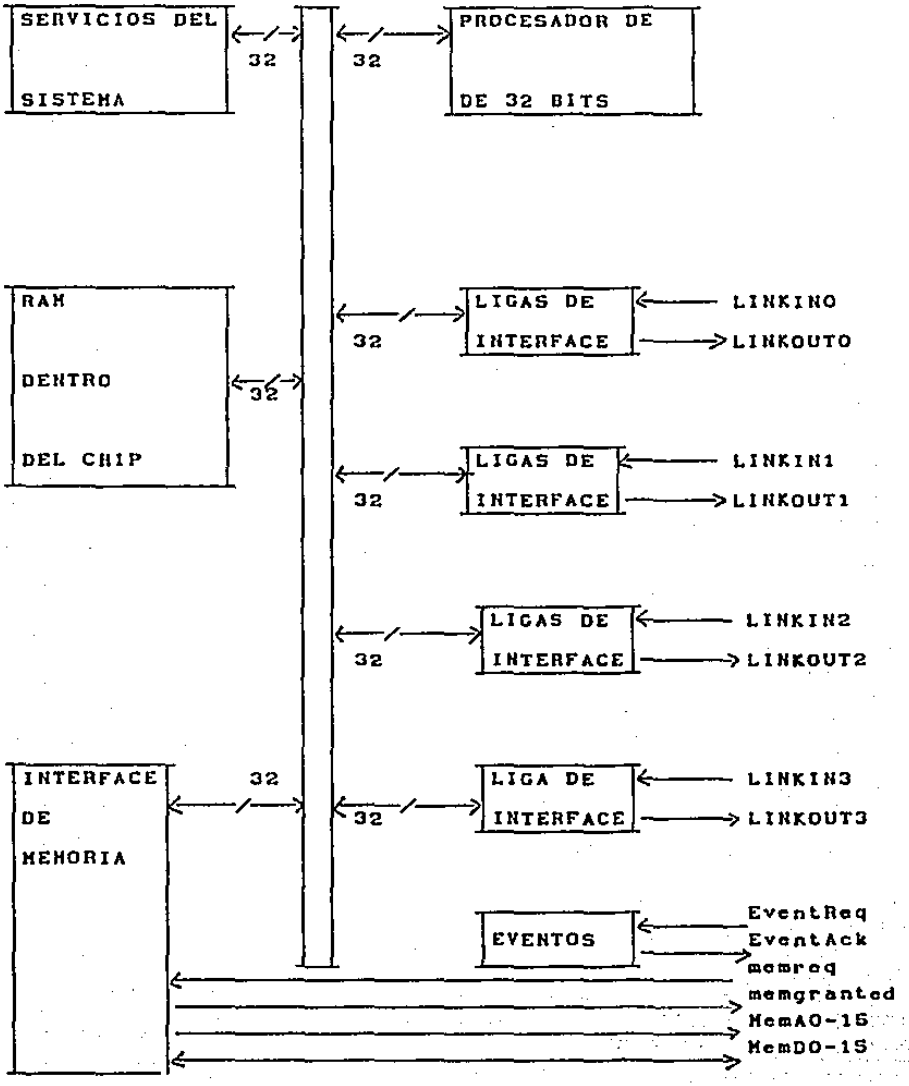


Figura 2.10.- Diagrama funcional del TRANSPUTER T424

C A P I T U L O 3

3.- A P L I C A C I O N E S

3.1.- RECONOCIMIENTO DE COMANDOS AISLADOS USANDO LA VOZ.

En la actualidad, debido principalmente al avance tecnológico que se a dado en los campos de la electrónica y la computación, es posible realizar proyectos que el hombre había concebido antes solamente como fantasías. Este es el caso de los sistemas de reconocimiento de comandos hablados por medio de computadora, capaces de reconocer palabras aisladas, tomadas de un vocabulario preestablecido, en tiempo real. Antes de analizar en detalle la forma en que la computadora realiza el reconocimiento, es pertinente hacer mención de algunas de la características de la voz humana. Si se tomaran en cuenta todos los mecanismos del cuerpo humano que intervienen en éste proceso nos daríamos cuenta que en realidad se trata de un proceso mecánico muy complejo que involucra la interacción coordinada de pulmones, garganta, cuerdas vocales, traquea, nariz y lengua. Para cada sonido que se emite, el cerebro coordina todos estos órganos y generalmente lo hace en una forma automática. La voz se produce con una corriente de aire que sale de los pulmones. Los pulmones retienen en su interior cerca de tres cuartos de su capacidad con aire, y el diafragma puede enviar cantidades de aire controladas en un tiempo determinado y en una cantidad precisa hacia la garganta.

La cantidad y la presión del aire que se envía a la garganta (traquea) determina algunas características del sonido emitido. Un murmullo utiliza muy poco aire, mientras que un grito requiere de una cantidad muy grande. Aunque este flujo de aire no es propiamente voz, actúa como la fuente de excitación para los sonidos que componen la voz. Al salir dichas corrientes de aire con la

fuerza suficiente hacen vibrar el aire que se encuentra fuera del cuerpo y estas vibraciones son ondas acústicas que son perceptibles por el oído humano. El órgano que genera las vibraciones de las corrientes de aire son las cuerdas vocales. Usando estas se puede bloquear parcial o totalmente el flujo de aire y de esta forma el flujo de aire se convierte en un conjunto de pequeños zumbidos. La rapidez con la cual las cuerdas vocales se abren o se cierren es la frecuencia del zumbido que se produce. Pero la frecuencia del zumbido no es lo único que distingue un sonido de voz de otro. Después de salir de la laringe, el flujo de aire pasa hacia la traquea (faringe), la boca y la naríz. Todos ellos cambian el sonido ligeramente. A éstos organos se les conoce como el tracto vocal. Cuando la corriente de aire pasa hacia el tracto vocal, la resonancia natural de éstas cavidades modifica la vibración para darle otras características adicionales al sonido emitido. Moviendo la lengua en varias posiciones dentro de la boca, cerrando y abriendo los dientes, levantando y bajando el paladar,o de alguna otra manera cambiando la forma o el tamaño del tracto vocal,se modifica la resonancia. De esta forma,el flujo de aire es modificado; a ésta modificación se le conoce como modulación. Los ingenieros deben representar procesos que ocurren en el mundo físico a través de modelos matemáticos, con los cuales se pueda experimentar bajo las condiciones de laboratorio. De esta forma se han creado modelos matemáticos que representan la forma en la que los diferentes órganos vocales generan la voz. A partir de estos modelos se han desarrollado una serie de algoritmos de reconocimiento de patrones.

El sistema de reconocimiento que se desarrolló para esta tesis es capaz de crear los patrones de reconocimiento de cualquier palabra aislada; de éste modo, se puede tener un conjunto de patrones de acuerdo con las necesidades de cada usuario. Para la aplicación que se desarrollo se pueden reconocer los números del 0

al 9 y las palabras si o no. Pero se podrían tener otras palabras en éste sistema dependiendo de la aplicación. La aplicación que se desarrolló era crear un conmutador telefónico que reconozca comandos hablados. Este conmutador sería capaz de hacer la conexión de una línea telefónica externa con cualquiera de las extensiones internas que maneje éste, sin la ayuda de una operadora. La extensión se pide diciendo dígito por dígito el número de la extensión requerida. Este sistema es una de las tantas aplicaciones de un sistema de reconocimiento de comandos, pero éstas ideas pueden ser ampliadas para tener diferentes aplicaciones, como por ejemplo, la comunicación que se podría hacer entre una institucion bancaria y sus cuenta habientes . Para hacer operaciones bancarias por teléfono se necesita en la actualidad un generador de tonos conectado al teléfono para indicar la operación que se quiere realizar. Usando el sistema de reconocimiento de comandos hablados el generador de tonos puede ser eliminado y en su lugar se puede seleccionar la operación deseada por medio de comandos hablados. Otras aplicaciones podrían ser encontradas en la medicina, donde se podría contar con sistemas que ayuden a personas con problemas físicos, tales como sordos, mudos, ciegos, paralíticos,etc. Por ejemplo se podría tener en un cuarto de hospital un sistema de reconocimiento que pudiera efectuar diferentes acciones para un enfermo inmovilizado, tales como tener el control de la iluminación del cuarto, tener el control de la posición de la cama donde se encuentra el paciente,etc. Otro ejemplo sería una silla de ruedas con motor donde el paciente indique con voz, la dirección que debe tomar esta (derecha, izquierda, adelante, atrás). Aunque el número de aplicaciones esta limitado solo por la imaginación, todas ellas serían ligeras variaciones de la misma filosofía básica de diseño presentada en ésta tesis. En los siguientes capítulos se dan todas las bases matemáticas para la solución de éste tipo de problemas.

3.1.1.- MODELO DEL APARATO VOCAL

Un modelo lineal de producción de voz para sonidos no nasales, es decir, sonoros, ha sido propuesto por Fant [1959,1960], y la asumpción de éste modelo ha sido analizada en detalle [Fant,1960; Flanagan, 1972]. El término voz se refiere únicamente a los sonidos vocales en este estudio. Como se muestra en la figura 3-1, este modelo esta compuesto por tres filtros acústicos. La principal justificación para éste modelo está basado en la teoría del tubo acústico, medidas del volumen de velocidad y la presión de la forma de onda del sonido, los datos de los rayos-x, y los resultados obtenidos usando circuitos eléctricos para sintetizar los sonidos vocales. La mayor asumpción de éste modelo es la separabilidad de los segmentos del filtro durante la generación de la voz.

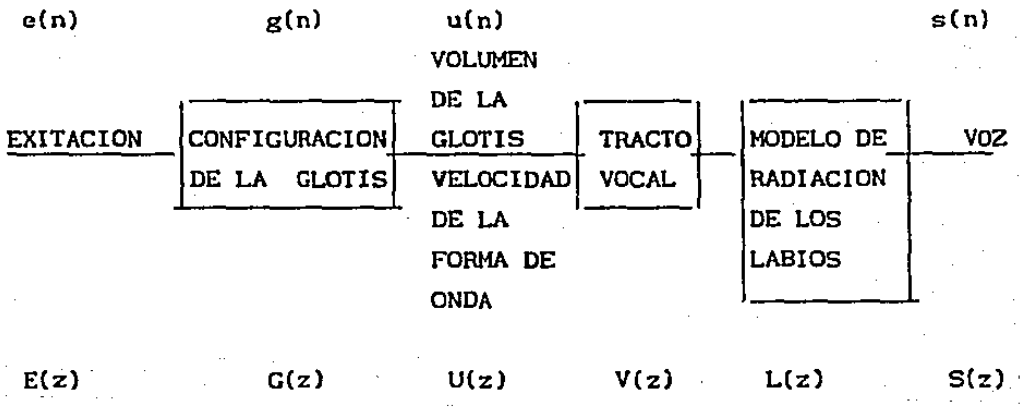


Fig.3.1.- Modelo Lineal de Producción de Voz

La exitación e(n) es modelada como una serie de impulsos unitarios escalados y espaciados con un periodo que corresponde al tono (Pitch) para sonidos llamados vocales, y es expresado como

$$e(n) = E_0 \sum_{j=0}^{\infty} \partial\,(n-JP)\ ,$$

donde E_0 es un factor de escala, $\partial(n)$ es la función delta de Kronecker. La transformada z es igual a

$$E(z) = \frac{E_0}{1 - z^{-P}}$$

donde P es la periodicidad del tono.

El filtro de la forma de la glotis es un filtro cuya respuesta impulso esta dada por

$$g(n) = G_0(n+1)\, e^{-cnT}$$

y cuya transformada z es

$$G(z) = \frac{G_0}{(1 - e^{-cT} z^{-1})^2}\ .$$

El filtro del tracto vocal es usado para modelar las caracteristicas acusticas de resonancia de los espacios de aire contenidos entre la glotis y los labios. Esta aproximación del tubo acústico esta compuesto de un número especifico de secciones, cada sección tiene una área constante.

La transformada z de V(z) del filtro del tracto vocal puede ser modelada por un número pequeño finito k (generalmente 3 o 4) de polos complejos de banda angosta,

$$V(z) = \prod_{i=1}^{k} \frac{1}{(1-z^{-1} z_i)(1-z^{-1} z_i^{*})}$$

o

$$V(z) = \prod_{i=1}^{k} \frac{1}{1-e^{-\pi b_i T} \cos(2\pi f_i T) z^{-1} + e^{-2\pi b_i T} z^{-2}}$$

Una resonancia espectral con frecuencia central f_i y ancho de banda de dos lados b_i es definida por cada par de polos complejos (z_i, z_i^*) , donde $z_i = \exp[-\pi b_i + j2\pi f_i T]$, z_i^* es el Xcomplejo conjugado de z_i, y T es el periodo de muestreo. Estas resonancias son también llamados formantes. Los formantes tienen sus centros de frecuencia generalmente menos de 5 kHz y anchos de banda generalmente menores a 100 Hz.

El modelo de radiación de los labios representa la transformación de una forma de onda con una volúmen y velocidad a una forma de onda de sonido. La transformada z de esta función es igual a

$$L(z) = L_0 (1 - z^{-1}) ,$$

donde L_0 es un factor de escala.

El modelo total lineal de la producción de voz para S(z) es el producto de la transformada z de cada uno de los filtros mencionados anteriormente

$$S(z) = L(z)V(z)G(z)E(z)$$

o

$$= \frac{L_0 G_0 E_0 (1 - z^{-1})(1 - e^{-cT} z^{-1})^{-2} (1 - z^{-P})^{-1}}{\prod_{i=1}^{k} 1 - e^{-\pi b_i T} \cos(2\pi f_i T) z^{-1} + e^{-2\pi b_i T} z^{-2}} .$$

Este modelo es simplificado haciendo un análisis de la siguiente forma: el factor en el numerador $[1-z]^{-1}$ se asume que se cancela aproximadamente por uno de los factores del numerador $[1 - e^{-cT} z^{-1}]^{-1}$ ya que cT es generalmente menor que la unidad. El termino sobrante en el numerador $[1 - e^{-cT} z^{-1}]^{-1}$ es incluido en el producto de factores del denominador. Finalmente, las potencias de z son acomodadas en orden ascendente, y las constantes son combinadas para producir la forma

$$S(z) = \frac{E(z)}{\sum_{i=0}^{M} a_i z^{-i}} ,$$

donde a_1 es real, $a_0 = 1$, $M=2k+m$ .Despejando E(z)

$$E(z) = S(z) \sum_{1=0}^{M} a_1 z^{-1} ,$$

donde ésta ecuación se conoce como el modelo de análisis.
En la siguiente sección se analiza la forma de encontrar los coeficientes a_1.

3.1.2.- ANALISIS DE PREDICCION LINEAL

Uno de los problemas más celebres en el procesamiento de señales es el predecir un valor futuro de un proceso estocástico estacionario discreto en el tiempo, dado un conjunto de valores pasados del proceso. Para ser más especificos, considérese la serie del tiempo u(n), u(n-1),..., u(n-M), representando (M+1) muestras de tal proceso, incluyendo el proceso u(n). Sea U_{n-1} el espacio de M-dimensiones generado por u(n-1),u(n-2),...,u(n-M), y sea $\hat{u}(n \mid U_{n-1})$ el valor de predicción u(n) dado este conjunto de muestras. En general se puede expresar este valor preestimado como alguna funcion θ de las muestras u(n-1),u(n-2),...,u(n-M) como a continuación:

$$\hat{u}(n \mid U_{n-1}) = \theta\,(u(n-1),\ldots,u(n-M)). \qquad (3.1)$$

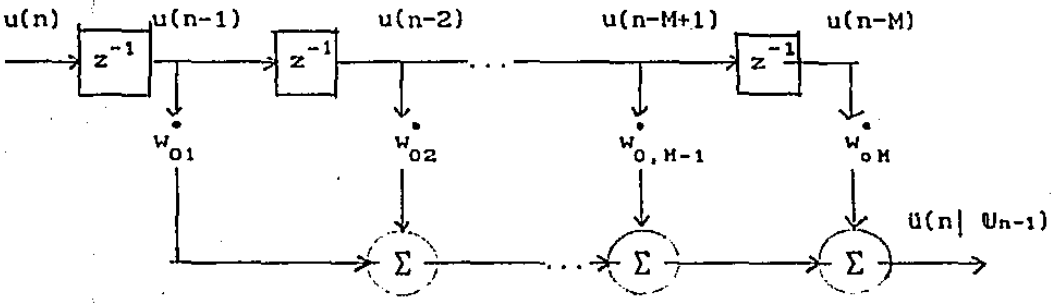
Se dice que el predictor es lineal cuando la función simplemente consiste de una combinación lineal de las muestras u(n-1), u(n-2), ...,u(n-M), como se muestra a continuación

$$\hat{u}(n \mid U_{n-1}) = \sum_{k=1}^{M} w_{ok}^{*} u(n-k) \ , \qquad (3.2)$$

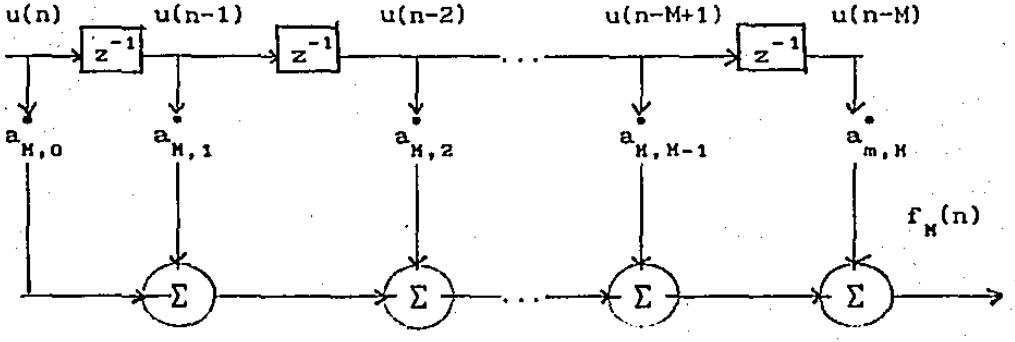
donde $w_{o1}, w_{o2}, \ldots, w_{oM}$ son coeficientes constantes.

En esta operación, el conjunto de valores u(n-1),u(n-2),...,u(n-M) es utilizado para hacer la predicción de la muestra u(n). Esta operación corresponde a la predicción de un paso en el futuro, medida con respecto al tiempo n-1. Este tipo de predicción se denomina predicción hacia adelante. En otro tipo de predicción se utilizan las muestras u(n),u(n-1),...,u(n-M+1) para hacer la predicción de la muestra u(n-M), a este tipo de predicción se le denomina predicción hacia atrás.

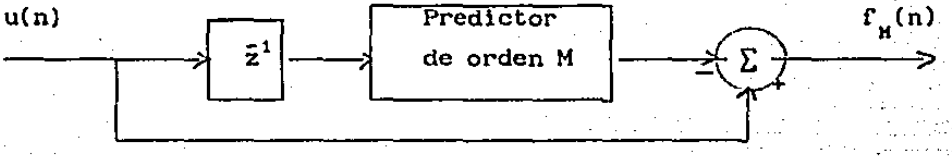
La figura 3.2 muestra un predictor hacia adelante que consiste de un filtro transversal con M pesos $w_{01}, w_{02}, \ldots, w_{0M}$ y entradas u(n-1), u(n-2),...,u(n-M), respectivamente. Se asume que estas entradas corresponden de un proceso estocástico estacionario de media cero



(a)



(b)



filtro de predicción y error

Figura 3.2.- a) Predictor de un paso; b) Filtro de predicción y error; c) Relación entre el predictor y el filtro de predicción y error.

El principal objetivo es el de diseñar el filtro de la figura 3.2.(a) de tal forma que la diferencia entre u(n) y la salida del filtro sea tan pequeña como sea posible en un sentido estadistico. La diferencia ,

$$f_H(n) = u(n) - \hat{u}(n|\,\mathcal{U}_{n-1}) \ , \qquad (3.3)$$

es llamada el error de estimación. En la teoria de Wiener, el criterio del mínimo error cuadrático medio es utilizado para optimizar el filtro. Especificamente, las constantes $w_{o1}, w_{o2}, \quad w_{oH}$ son escogidas de tal forma que minimizan el indice de ejecución, J(w), definido como el valor cuadrático medio de el error de estimación, o simplemente el error cuadrático medio :

$$J(w) = E\,[f_H(n) f_H^*(n)] \ , \qquad (3.4)$$

donde el asterisco denota complejo conjugado y w es el vector de coeficientes representado por $w_{o1}, w_{o2}, \quad w_{oH}$. El error cuadrático medio es un escalar real y positivo. Minimizando J(w), obtenemos el filtro lineal óptimo en el sentido de mínimo error cuadrático medio.

En el desarrollo de la versión discreta en el tiempo de la teoria del filtro de Wiener, es conveniente usar notación matricial. La composición del vector de los coeficientes de M por 1 elementos es representado por

$$w^T = [\ w_{o1}, w_{o2}, \quad w_{oH}\ ] \ , \qquad (3.5)$$

y el vector de las entradas u(n-1) es definido como

$$u^T(n-1) = [u(n-1), u(n-2), \ldots, u(n-M)], \qquad (3.6)$$

donde la T significa transposición. Entonces, podemos reescribir la Eq.(3.2) en la forma de un producto interno de vectores como:

$$\hat{u}(n|\,\mathcal{U}_{n-1}) = w^H u(n-1) \ ,$$

donde el indice H significa transposicion Hermitiana. Equivalentemente, tomando la transposición Hermitiana en ambos lados de la Eq.(3.6), podemos escribir

$$\hat{u}^{*}(n|\,\mathcal{U}_{n-1}) = u^{H}(n-1)w \,. \qquad (3.7)$$

Entonces, podriamos escribir el error de estimación como

$$f_{H}(n) = u(n) - w^{H}u(n-1), \qquad (3.8)$$

o, en forma compleja conjugada, como

$$f_{H}^{*}(n) = u^{*}(n) - u^{H}(n-1)w \,. \qquad (3.9)$$

Substituyendo las ecuaciones (3.8) y (3.9) en (3.4) para el cuadrático medio, obtenemos

$$J(w) = E\,[(u(n) - w^{H}u(n-1))(\,u^{*}(n) - u^{H}(n-1)w)], \qquad (3.10)$$

expandiendo el lado derecho de la ecuación (3.10) y reconociendo que el vector de w es constante, obtenemos lo siguiente:

$$J(w) = E[u(n)u^{*}(n)] - w^{H}E[u(n-1)u^{*}(n)] - E[u(n)u^{H}(n-1)]w + w^{H}E[u(n-1)u^{H}(n-1)]w \,. \qquad (3.11)$$

Asumiendo que el vector de entrada u(n-1)y u(n) son conjuntamente estacionarios, así que las cuatro esperanzas matematicas del lado derecho de la Eq.(3.11) pueden interpretarse como sigue.

1.- La esperanza $E[u(n)u^{*}(n)]$ es igual a la variancia de la entrada u(n), asumiendo que u(n) tiene media cero. Denotando ésta variancia como σ_{u}^{2} , se puede escribir

$$\sigma_{u}^{2} = E[u(n)u^{*}(n)] \,. \qquad (3.12)$$

2.- La esperanza $E(u(n-1)u^{*}(n)]$ es igual al vector de correlación cruzada entre la entrada u(n) y el vector de entradas u(n-1). Denotando este vector de correlaciones por r, podemos escribir

$$r = E[u(n-1)u^{*}(n)] \,. \qquad (3.13)$$

En forma expandida, tenemos

$$r^T = [r(0), r(-1), \ldots, r(1-M)],$$

donde

$$r(1-k) = E[u(n-k)u^*(n)], \qquad k = 1,2,\ldots,M \tag{3.14}$$

3.- La esperanza $E[u(n)u^H(n-1)]$ es igual a la transpuesta Hermitiana del vector de correlaciones cruzada r, mostrado por

$$r^H = E[u(n)u^H(n)] \; . \tag{3.15}$$

4.-La esperanza $E[u(n-1)u^*(n-1)]$ es igual a la matriz de correlaciones del vector u(n-1). Denotando esta matriz de correlación por R, podemos escribir entonces

$$R = E[u(n-1)u^H(n-1)] \tag{3.16}$$

En forma expandida, tenemos

$$R = \begin{vmatrix} r(0) & r(1) & \ldots r(M-1) \\ r(-1) & r(0) & \ldots r(M-2) \\ . & . & . \quad . \\ . & . & . \quad . \\ . & . & . \quad . \\ r(-M+1) & r(-M+2) & \ldots r(0) \end{vmatrix} , \tag{3.17}$$

donde el k-ésimo elememto, con k,i=1,2,...,M, es definido por

$$r(i-k) = E[u(n-k-1)u^*(n-i-1), \tag{3.18}$$

donde el índice k se refiere al número de renglón y el índice i se refiere a la columna en la matriz de correlaciones. Hay que notar que

$$r^*(i-k) = r(k-i) \tag{3.19}$$

En el caso de procesos estacionarios la matriz de correlaciones R es definida únicamente especificando la sucesión de autocorrelación de los valores de entrada: r(0),r(1),...,r(M-1), el cual corresponde a los retrasos de 0,1,...,M-1,

respectivamente. Substituyendo las definiciones de las ecuaciones (3.12),(3.13),(3.15),(3.16) en (3.11) se puede reescribir la expresión del error cuadrático medio como

$$J(w) = \sigma_u^2 - r^H w - w^H r + w^H R w \qquad (3.20)$$

La ecuación (3.20) especifica, que para el caso cuando el vector u(n-1) y la respuesta deseada u(n) son conjuntamente estacionarios ,la función de error cuadrático medio J(w) es precisamente una función de segundo orden. Así, se puede visualizar la dependencia del error J(w) de los elementos del vector w, como una superficie en forma de taza con un mínimo unico. Entonces el requerimiento de diseño del filtro es, que opere sobre el punto máximo o mínimo de ésta superficie de error. En éste punto, el error cuadrático medio J(w) obtiene su valor mínimo, denotado por J_{min} y, correspondientemente, el vector w obtiene su valor óptimo, denotado por w_0. El filtro transversal resultante se dice que es óptimo en el sentido del error cuadrático medio mínimo.

Para determinar el vector óptimo w_0, primero diferenciamos el error cuadrático medio J(w) en la Eq.(3.20) con respecto al vector w y ponemos el resultado igual a cero.La solución de esta ecuación nos da w_0.

La variancia σ_u^2 es una constante, y por eso su derivada es igual a cero. Usando las reglas de diferenciación definidas en [1] capitulo 3.4 encontramos que la derivada de los tres términos que quedan en el lado derecho de la Eq. (3.20) con respecto a w toman los siguientes valores :

$$\frac{d}{dw}(r^H w) = 0$$

$$\frac{d}{dw}(w^H r) = 2r$$

$$\frac{d}{dw}(w^H R w) = 2Rw$$

Entonces obtenemos la siguiente expresión para el vector gradiente, ∇, definido como la derivada del error cuadrático medio J(w) con respecto al vector w:

$$\nabla = \frac{dJ(w)}{dw}$$

$$= -2r + 2Rw \ . \qquad (3.34)$$

Denotemos el valor w_0 el valor del vector óptimo para el cual el vector gradiente ∇ es igual al vector nulo. Entonces de la ecuación (3.34) obtenemos

$$Rw_0 = r \ . \qquad (3.35)$$

La ecuación (3.35) es la forma discreta de la ecuación de Wiener - Hopf. Es también llamada la ecuación normal. Para determinar la solución, premultiplicamos la Eq.(3.35) por R^{-1}, la inversa de la matriz de correlación, obteniendo

$$w_0 = R^{-1}r \ ,$$

El cálculo del vector optimo w_0 requiere conocer:

1) La matriz de correlaciones del vector de entrada u(n-1), y (2) el vector de correlación cruzada r entre la entrada u(n) y el vector u(n-1).

El error cuadrático medio mínimo es definido por

$$J_{min}(w) = \sigma_u^2 - r^H w_0 - w_0^H r + w_0^H R w_0 \ ,$$

ya que $Rw_0 = r$, esta expresión se simplifica como

$$J_{min}(w) = \sigma_u^2 - r^H w_0 \qquad (3.36)$$

$$= P_H = r(0) - r^H w_0 \ , \qquad (3.37)$$

donde P_M es el error cuadrático medio de predicción

$$P_M = E[|f_M(n)|^2] \quad . \tag{3.38}$$

El error de predicción hacia adelante es

$$f_M(n) = u(n) - \sum_{k=1}^{M} w_{Ok} u(n-k) \quad . \tag{3.39}$$

Sean $a_{M,k}$, k= 0,1,...,M , las nuevas variables del filtro transversal, las cuales estan relacionadas con el filtro predictor hacia adelante de la siguiente forma:

$$a_{M,k} = \begin{cases} 1, & k=0 \\ -w_{Ok}, & k = 1,2,\ldots,M \; . \end{cases} \tag{3.40}$$

entonces podemos combinar los dos términos del lado derecho de la Eq.(3.40) en una sumatoria sencilla como sigue:

$$f_M(n) = \sum a^{*}_{M,k} u(n-k) \quad . \tag{3.41}$$

La relación de entrada salida de este filtro esta representada por el filtro transversal mostrado en la figura 3.2(b). Un filtro que opera con un conjunto de muestras u(n),u(n-1),...,u(n-M) para producir en su salida el error de predicción hacia adelante $f_M(n)$ es llamado filtro de error de predicción hacia adelante.
La relación entre el filtro de error de predicción hacia adelante y el predictor hacia adelante es ilustrado en el diagrama de bloques de la figura 3.2(c).
La ecuación normal (3.35) define el vector de peso del predictor hacia adelante, mientras que la ecuación (3.37) define la potencia del error de predicción hacia adelante P_M. Se pueden combinar

estas dos ecuaciones en una sola relación matricial como sigue:

$$\begin{bmatrix} r(0) & \mathbf{r} \\ \mathbf{r} & \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_o \end{bmatrix} = \begin{bmatrix} P_M \\ \mathbf{0} \end{bmatrix}, \qquad (3.42)$$

donde 0 es el vector nulo de M por 1. Se observa que la matriz de (M+1) por (M+1) del lado izquierdo de la ecuación (3.42) iguala a la matriz de correlación de las entradas u(n),u(n-1),...,u(n-M) en el filtro de predicción de error de la figura 3.2(b),esto es:

$$\left[\begin{array}{c|c} r(0) & \mathbf{r} \\ \hline \mathbf{r} & \mathbf{R} \end{array}\right] = \left[\begin{array}{c|ccccc} r(0) & r(1) & r(2) & & & r(M) \\ \hline r(-1) & r(0) & r(1) & \cdots & & r(M-1) \\ r(-2) & r(-1) & r(0) & \cdots & & r(M-2) \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ r(-M) & r(-M+1) & r(-M+2) & \cdots & & r(0) \end{array}\right], \qquad (3.43)$$

Además se puede observar que el vector de M por 1 del lado izquierdo de la ecuación 3.42 puede ser sustituido por el vector definido en la Ec.(3.40), entonces (3.42)

$$\begin{bmatrix} r(0) & r(1) & r(2) & & r(M) \\ r(-1) & r(0) & r(1) & \cdots & r(M-1) \\ r(-2) & r(-1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r(-M) & r(-M+1) & r(-M+2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} a_{M,0} \\ a_{M,1} \\ \vdots \\ a_{M,M} \end{bmatrix} = \begin{bmatrix} P_M \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \qquad (3.44)$$

Podemos también expresar la relación matricial de la ecuación (3.44) como un sistema de (M+1) ecuaciones simultáneas:

$$\sum_{l=0}^{M} a_{M,l}\, r(l-i) = \begin{cases} P, & i=0 \\ 0, & i=1,2,\ldots,M \end{cases} \quad . \tag{3.45}$$

Las ecuaciones (3.44) y (3.45) son denominadas como la ecuación normal del filtro de error de predicción adelantada de orden M.

La forma de predicción lineal considerada anteriormente se realiza hacia adelante. Hacemos uso de las M muestras u(n-1),u(n-2),..., u(n-M) para hacer la predicción de la muestra u(n). Naturalmente, también podemos operar estos datos para hacer una predicción en el pasado, es decir usamos las muestras u(n),u(n-1),...,u(n-M+1) para hacer una predicción de la muestra u(n-M). Sea U_n el espacio de M dimensiones generado por las M muestras u(n),u(n-1),..., u(n-M+1) que son usadas para hacer la predicción de la muestra u(n-M), como se muestra a continuación.

$$\hat{u}(n-M|\, U_n) = \sum_{k=1}^{M} g_k^{*}\, u(n-k+1) \quad , \tag{3.46}$$

donde $g_1, g_2, \ldots, g_M$ son los coeficientes. La Fig. 3.2(a) muestra una representación del predictor hacia atras descrito por la ecuación (3.46). Asumimos que éstos coeficientes son optimizados usando el criterio de error cuadrático medio de acuerdo a la teoría del filtro de Wiener. En la predicción hacia atrás la respuesta deseada es igual a u(n-M). El error de predicción hacia atras es

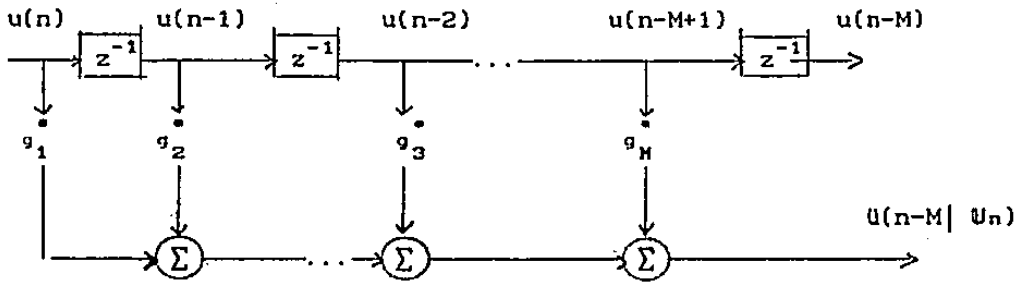
$$b_M(n) = u(n-M) - \hat{u}(n-M|\, U_n) \quad . \tag{3.47}$$

Sea P_M el minimo error cuadrático medio de predicción

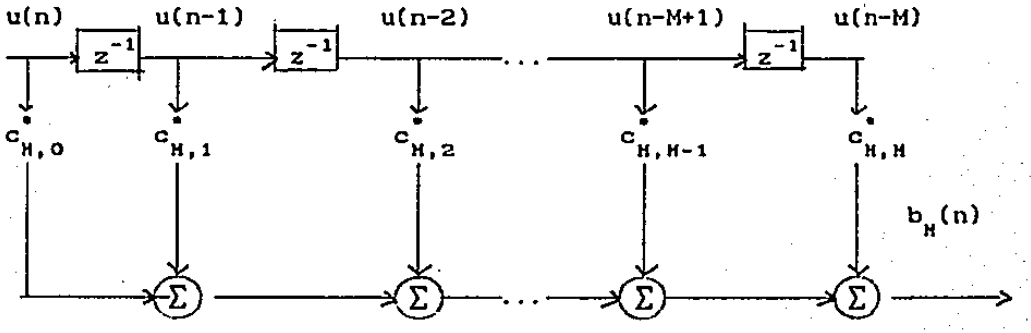
$$P_M = E[\,|\, b_M(n)\,|^2] \quad . \tag{3.48}$$

Sea g el vector optimo de M por 1 elementos del predictor hacia atras de la Fig. 3.3(a). Podemos expresar esta forma expandida como

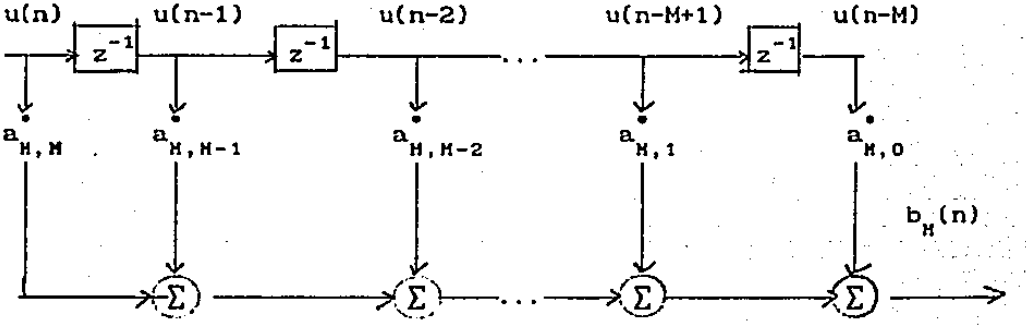
$$g^T = [g_1, g_2, \ldots, g_M] \quad . \tag{3.49}$$



(a)



(b)



(c)

Figura 3.3.- a) Predictor hacia atrás de un paso; b) Filtro de predicción hacia atrás y error; c) Filtro de predicción hacia atrás y error definido usnado los coeficientes del filtro de predicción hacia adelante y error.

Para solucionar la ecuación normal para el vector g, se requieren: (1) la matriz de correlación de M por M de las entradas $u^T(n) = [u(n), u(n-1), \ldots, u(n-M+1)]$

$$R = E[u(n)u^H(n)]$$

$$R = \begin{vmatrix} r(0) & r(1) & \ldots & r(M-1) \\ r(-1) & r(0) & \ldots & r(M-2) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ r(-M+1) & r(-M+2) & \ldots & r(0) \end{vmatrix} \qquad (3.50)$$

y (2) la esperanza $E[u(n)u^*(n-M)]$ que es igual al vector de correlación cruzada entre la entrada u(n-M) y el vector de entradas u(n). Denotando este vector de correlaciones por r^{B*}, podemos escribir

$$r^{B*} = E[u(n)u^*(n-M)]. \qquad (3.51)$$

En forma espandida, tenemos

$$r^{B*T} = [r(M), r(M-1), \ldots, r(1)] ,$$

donde

$$r(M-k) = E[u(n-k)u^*(n-M)], \qquad k = 0, 2, \ldots, M-1 .$$

Para evaluar P_M es necesaria otra cantidad, la variancia, que es igual a r(0). Haciendo uso del desarrollo anterior para la predicción adelantada, para resolver la ecuación normal de la predicción hacia atras podemos escribir

$$R\, g = r^{B*} , \qquad (3.52)$$

y la potencia del error de predicción hacia atras

$$P = r(0) - \mathbf{r}^{BT}\mathbf{g} \ . \qquad (3.53)$$

El error de predicción hacia atrás $b_M(n)$ es igual a la diferencia entre la respuesta deseada u(n-M) en el tiempo n-M y la predicción de el dadas las muestras $u(n), u(n-1), \ldots, u(n-M+1)$.

$$b_M(n) = u(n-M) - \sum_{k=1}^{M} c^{*}_{M,k} u(n-k) \qquad (3.54)$$

Definanse los coeficientes de peso del filtro de predicción de error hacia atras como:

$$C_{M,k} = \begin{cases} -g^{*}_{k+1}, & k=0,1,\ldots,M-1 \\ 1, & k = M \ , \end{cases} \qquad (3.55)$$

asi podemos reescribir (3.54) como (la Fig.3.2(b))

$$b_M(n) = \sum_{k=0}^{M} c^{*}_{M,k} u(n-k) \ . \qquad (3.56)$$

Podemos escribir también el vector de constantes g como

$$g^{*}_{M-k+1} = w_{ok} \ , \quad k = 1,2,\ldots,M,$$

o equivalentemente

$$g_k = w^{*}_{o,M-k+1}, \quad k = 1,2,\ldots,M, \qquad (3.57)$$

En tal caso sustituyendo (3.57) en (3.55)

$$C_{M,k} = \begin{cases} -w^{*}_{o,M-k} \ , & k=0,1,\ldots,M-1 \\ 1, & k = M \end{cases} \qquad (3.56)$$

y usando la relación que existe entre los coeficientes del filtro de error de predicción hacia adelante y los coeficientes

del predictor hacia adelante se puede escribir

$$c_{M,k} = a^{*}_{M,M-k}, \quad k=0,1,\ldots,M . \tag{3.57}$$

Podemos expresar la relación entrada salida del filtro de error de predición hacia atras en la forma

$$b_M(n) = \sum_{k=0}^{M} a_{M,M-k} u(n-k) \tag{3.58}$$

Esta relación es representada en la figura 3.3(c). En particular podemos transformar un filtro de error con predicción hacia adelante en un filtro de error con predicción hacia atras colocando al revés las muestras en las cuales están posicionados los coeficientes y realizar el complejo conjugado en ellos.
La ecuación normal (3.52) define el vector de peso del predictor hacia atras , mientras que la ecuación (3.53) define la potencia del error de predicción hacia atras P_M. Se pueden combinar estas dos ecuaciones en una sola relación matricial como sigue:

$$\begin{bmatrix} R & r^{B*} \\ r^{BT} & r(0) \end{bmatrix} \begin{bmatrix} -g \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ P_M \end{bmatrix} \tag{3.59}$$

donde 0 es el vector nulo de M por 1. Se observa que la matriz de (M+1) por (M+1) del lado izquierdo de la ecuación (3.42) iguala a la matriz de correlación de las entradas u(n),u(n-1),...,u(n-M) en el filtro de predicción de error de la figura 3.2(b), esto es:

$$\left[\begin{array}{c|c} R & r^{B*} \\ \hline r^{BT} & r(0) \end{array}\right] = \left[\begin{array}{ccccc|c} r(0) & r(1) & r(2) & & r(M-1) & r(M) \\ r(-1) & r(0) & r(1) & \cdots & & r(M-1) \\ r(-2) & r(-1) & r(0) & \cdots & & r(M-2) \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ r(-M+1) & r(-M+2). & & . & r(0) & r(1) \\ \hline r(-M) & r(-M+1) & r(-M+2) & .. & r(-1) & r(0) \end{array}\right] \tag{3.60}$$

Además se puede observar que el vector de M por 1 del lado izquierdo de la ecuación 3.59 puede ser sustituido por el vector definido en la Ec.(3.55) y (3.57), entonces (3.59)

$$\begin{bmatrix} r(0) & r(1) & r(2) & & r(M) \\ r(-1) & r(0) & r(1) & \cdots & r(M-1) \\ r(-2) & r(-1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r(-M) & r(-M+1) & r(-M+2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} a^{*}_{M,M} \\ \vdots \\ a^{*}_{M,1} \\ a^{*}_{M,0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ P_M \end{bmatrix} \tag{3.61}$$

Podemos también expresar la relación matricial de la ecuación (3.61) como un sistema de (M+1) ecuaciones simultáneas como se muestra a continuación:

$$\sum_{i=0}^{M} a_{M,m-i} r(l-i) = \begin{cases} 0, & i = 0,1,\ldots,M-1 \\ P_M, & i=M \end{cases} \tag{3.62}$$

Las ecuaciones (3.61) y (3.62) son denominadas ecuación normal del filtro de error de predicción adelantada de orden M.

3.1.2.1.- LA RECURSION DE LEVINSON-DURBIN

Se describe a continuación un método directo para el cálculo de los coeficientes del filtro de predicción y la potencia del error de predicción resolviendo la ecuación normal aumentada. Este método es recursivo y hace uso particular de la estructura tipo Toeplitz de la matriz de correlaciones . Es conocido como la recursión Levinson - Durbin. Básicamente, este procedimiento utiliza la solución de la ecuación normal aumentada para un filtro predictor de error de orden m-1 para calcular la solución correspondiente para un filtro de predicción de error de orden M (es decir de un orden más alto), donde M es el orden final del filtro. Este procedimiento es computacionalmente muy eficiente, ya que hay un gran ahorro en el número de operaciones (multiplicaciones y divisiones) y localidades de almacenamiento. Sea el vector a_M (de tamaño m+1) el vector de coeficientes del filtro de predicción de error hacia adelante. El vector correspondiente de m+1 por 1 que corresponde al filtro de predicción de error hacia atrás es obtenido reordenando al contrario los elementos del vector a_M y obteniendo su conjugación compleja. Denotemos el efecto de estas dos operaciones como a^{B*}_{m-1}. La recursión Levinson-Durbin puede ser definida en dos formas equivalentes:

1. El vector de coeficientes del filtro de predicción-error hacia adelante puede ser ordenado como sigue:

$$a_m = \begin{bmatrix} a_{m-1} \\ 0 \end{bmatrix} + \Gamma_m \begin{bmatrix} 0 \\ a^{B*}_{m-1} \end{bmatrix} \qquad (3.63)$$

donde Γ_m es una constante escalar. La versión escalar de esta

actualización es

$$a_{m,k} = a_{m-1,k} + \Gamma_m a^{*}_{m-1,m-k}, \qquad k = 0,1,\ldots,m, \qquad (3.64)$$

donde $a_{m,k}$ es el k-ésimo coeficiente del filtro predictor-error hacia adelante de orden m; lo mismo ocurre para $a_{m-1,k}$. El elemento $a^{*}_{m-1,m-k}$ es el k-ésimo coeficiente del filtro predictor -error hacia atras de orden m-1. En la ecuación (3.64), hay que notar que $a_{m-1,0} = 1$ y $a_{m-1,m} = 0$.

2.- El vector de coeficientes del filtro de predicción-error hacia atras puede ser actualizado como sigue:

$$a^{B*}_{m} = \begin{bmatrix} 0 \\ a^{B*}_{m-1} \end{bmatrix} + \Gamma^{*}_{m} \begin{bmatrix} a_{m-1} \\ 0 \end{bmatrix} . \qquad (3.65)$$

La versión escalar de esta actualización es

$$a^{*}_{m,m-k} = a^{*}_{m-1,m-k} + \Gamma^{*}_{m} a_{m-1,k} , \qquad k = 0,1,\ldots,m \qquad (3.66)$$

donde $a^{*}_{m,m-k}$ es el k ésimo coeficiente del filtro predictor-error hacia atras de orden m.

La recursión de Levinson-Durbin es usualmente formulada en el contexto de la predicción hacia adelante, usando la forma vectorial (3.63) o (3.64) la forma escalar. Para establecer la condición que la constante Γ_m tiene que satisfacer para justificar la validez de la recursión Levinson-Durbin, se procedera en cuatro pasos como sigue:

1.- Premultiplicando ambos lados de la ecuación (3.63) por R_{m+1}, la matriz de correlación de las entradas en el filtro de predicción-error hacia adelante de orden m. Para el lado izquierdo de la ecuación (3.63) se tiene que (ver ec.(3.44)).

$$R_{m+1}a_m = \begin{bmatrix} P_m \\ 0_m \end{bmatrix}, \qquad (3.67)$$

donde P_m es la potencia del predictor-error hacia adelante.

2.- Para el primer término de la ecuación del lado derecho (3.63), se usa la siguiente forma de partición de la matriz de correlación R_{m+1}

$$R_{m+1} = \begin{bmatrix} R_m & r_m^{B*} \\ r_m^{BT} & r(0) \end{bmatrix},$$

donde r_m^{B*} es el vector de correlación cruzada entre el vector u(n) y la respuesta deseada u(n-m), entonces

$$R_{m+1}\begin{bmatrix} a_{m+1} \\ 0 \end{bmatrix} = \begin{bmatrix} R_m & r_m^{B*} \\ r_m^{BT} & r(0) \end{bmatrix}\begin{bmatrix} a_{m-1} \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} R_m\, a_{m-1} \\ r_m^{BT}\, a_{m-1} \end{bmatrix}. \qquad (3.68)$$

La ecuación normal aumentada para el filtro predictor-error hacia adelante de orden m-1 es

$$R_m a_{m-1} = \begin{bmatrix} P_{m-1} \\ 0_{m-1} \end{bmatrix} \quad . \qquad (3.69)$$

Definase el escalar

$$\Delta_{m-1} = r_m^{BT} a_{m-1}$$

$$= [r(-m), r(1-m), \ldots r(-1)] \begin{bmatrix} a_{m-1,0} \\ a_{m-1,1} \\ \\ a_{m-1,m-1} \end{bmatrix} \qquad (3.70)$$

$$= \sum_{k=0}^{m-1} a_{m-1,k} r(k-m) \qquad ;$$

substituyendo las ecuaciónes (3.69) y (3.70) en (3.68)

$$R_{m+1} \begin{bmatrix} a_{m-1} \\ \\ 0 \end{bmatrix} = \begin{bmatrix} P_{m-1} \\ 0_{m-1} \\ \Delta_{m-1} \end{bmatrix} \quad . \qquad (3.71)$$

3.- Para el segundo termino del lado derecho de la ecuacion (3.63) se usa la siguiente partición de la matriz de correlación R_{m+1}

$$R_{m+1} = \begin{bmatrix} r(0) & r_m^H \\ r_m & R_m \end{bmatrix} \qquad (3.72)$$

Donde R_m es la matriz de correlación de las entradas u(n-1), u(n-2),...,u(n-m). Entonces

$$R_{m+1}\begin{bmatrix} 0 \\ a_{m-1}^{B*} \end{bmatrix} = \begin{bmatrix} r(0) & r_m^H \\ r_m & R_m \end{bmatrix}\begin{bmatrix} 0 \\ a_{m-1}^{B*} \end{bmatrix}$$

$$= \begin{bmatrix} r_m^H a_{m-1}^{B*} \\ R_m a_{m-1}^{B*} \end{bmatrix} \qquad (3.73)$$

el escalar $r_m^H a_{m-1}^{B*}$

$$= [r(1), r(2), \ldots r(m)] \begin{bmatrix} a_{m-1,m-1}^{*} \\ a_{m-1,m-2}^{*} \\ a_{m-1,0}^{*} \end{bmatrix} \qquad (3.74)$$

$$= \sum_{k=1}^{m} a_{m-1,m-1}^{*} r(i) \quad .$$

La ecuación normal aumentada para el filtro predictor-error de orden m-1

$$R_m \, a^{B^*}_{m-1} = \begin{bmatrix} 0_{m-1} \\ p_{m-1} \end{bmatrix} \qquad (3.75)$$

Sustituyendo las ecuaciones (3.74) y (3.75) en (3.73), se podría escribir

$$R_{m+1} \begin{bmatrix} 0 \\ a^{B^*}_{m-1} \end{bmatrix} = \begin{bmatrix} \Delta^*_{m-1} \\ 0_{m-1} \\ P_{m-1} \end{bmatrix} \qquad (3.76)$$

4.- Usando las ecuaciones (3.67), (3.71) y (3.76), podemos ahora decir que premultiplicando la ecuación (3.63) por R_{m+1} resulta en

$$\begin{bmatrix} P_m \\ 0_m \end{bmatrix} = \begin{bmatrix} P_{m-1} \\ 0_{m-1} \\ \Delta_{m-1} \end{bmatrix} + \Gamma_H \begin{bmatrix} \Delta^*_{m-1} \\ 0_{m-1} \\ P_{m-1} \end{bmatrix} . \qquad (3.77)$$

Se concluye que, si la recursión de orden de la ecuación (3.63) se sostiene, entonces el resultado obtenido por la ecuación (3.77) es una conclusión directa de esta recursión. De la ecuación (3.77) se pueden obtener dos importantes conclusiones:

1. Considerando los primeros elementos de los vectores del lado izquierdo y derecho de la ecuación (3.77)

$$P_m = P_{m-1} + \Gamma_m \Delta^*_{m-1} \qquad (3.78)$$

2. Considerando los ultimos elementos de los vectores del lado izquierdo y derecho de la ecuación (3.77)

$$0 = \Delta_{m-1} + \Gamma_m P_{m-1} \quad , \qquad (3.79)$$

despejando Γ_m

$$\Gamma_m = - \frac{\Delta_{m-1}}{P_{m-1}} \quad . \qquad (3.80)$$

Además eliminando Δ_{m-1} de las ecuaciones (3.78) y (3.79) obtenemos que

$$P_m = P_{m-1}(1 - |\Gamma_m|^2) \qquad (3.81)$$

$$= P_{m-1} + \Gamma_m \Delta_{m-1} \; .$$

La forma de aplicar la recursión Levinson-Durbin para calcular los coeficientes del filtro predicción error $a_{m,k}$, k=0,1,...,M y la potencia del error P_M es la siguiente:

Se cuenta con la función de autocorrelación del proceso de entrada; en particular, se tiene r(0),r(1),...,r(M), los valores de la función de autocorrelación para retrasos de 0,1,...,M, respectivamente. Se puede obtener un estimador no sesgado de estos parametros por medio de la fórmula de promedio en el tiempo

$$r(k) = \frac{1}{N-1} \sum_{n=1}^{N-k} u(n)u^*(n-k) \; , \qquad k=0,1,\ldots,M,$$

donde N es el número de muestras, con N >> M. Existen por supuesto otros estimadores que se pueden utilizar. Dado r(0),r(1),...,r(m), la computación empieza usando la ecuación (3.70) para Δ_{m-1} y la ecuación (3.81). La recursión es inicializada con m=0, donde P_0=r(0) y Δ_0=r(-1). Hay que notar que $a_{m,0}$ es igual a 1 para toda las m, y $a_{m,k}$ es cero para toda k > m. El cálculo termina cuando m=M.

3.1.3.- CUANTIZACION VECTORIAL

Cuantización vectorial (CV) es una generalización de la cuantización escalar. Mientras que la cuantización escalar está ligada a la conversión analógico/digital, la CV está asociada con un procesamiento digital de señales sofisticado, donde en la mayoria de los casos, las señales de entrada ya tienen alguna forma de representación digital y la CV es usada, pero no exclusivamente, para el próposito de compresión de datos.

Un vector puede ser usado para describir casi cualquier tipo de patrón, ya sea la forma de onda de un segmento de voz o de una imagen simplemente formando un vector de muestras de la forma de onda o de la imagen. En nuestro caso estamos interesados en formar vectores de un conjunto de parámetros que representan la envolvente espectral de un sonido de voz. La cuantización vectorial puede ser vista como una forma de reconocimiento de patrones donde un patrón de entrada es aproximado por un conjunto predeterminado de patrones estándar, es decir, el patrón de entrada es igualado con uno de los patrones previamente guardados.

Un cuantizador vectorial Q de dimensión k y tamaño N es una transformación de un vector (o un punto) del espacio Euclidiano de dimensión k, R^k en un conjunto finito C que contiene N salidas o puntos de reproducción, llamados vectores de código (codevectors). Así,

$$Q: R^k \rightarrow C ,$$

donde $C = (y_1, y_2, \ldots, y_N)$ y $y_i \in R^k$ para cada $i \in J = \{1,2,\ldots,N\}$. El conjunto C es llamado el Alfabeto (Code Book) y tiene un tamaño de N, lo que significa que se tienen N elementos distintos, donde cada uno de ellos es un vector en R^k. La resolución de un cuantizador vectorial es $r = (\log_2 N)/k$ el cual mide el número de bits por componente vectorial usado para representar el vector de entrada y da una indicación de la

precisión que es obtenida con el cuntizador vectorial si el Alfabeto es bien diseñado.

Asociado con cada punto del cuantizador vectorial hay una partición de R^k en N regiones o celdas, R_i para $i \in J$. La i-ésima región es definida por:

$$R_i = \{ x \in R^k : Q(x) = y_i \} \qquad (3.82)$$

algunas veces llamado la imagen inversa o pre-imagen de y_i con la transformación Q y puede ser denotado más concisamente como como $R_i = Q^{-1}(y_i)$. De la definición de regiones, se concluye que

$$\bigcup_i R_i = R^k \text{ y } R_i \cap R_j = \emptyset \text{ para } i \neq j$$

así las regiones forman una partición de R^k. Una región que no tiene límites es llamada región de sobre-carga. Una región limitada, es decir, que tienen un volumen determinado es llamada una región granular.

Una importante propiedad de un conjunto en R^kes su convexidad. Un conjunto R^kes convexo si a y b $\in$ S implica que $\alpha a + (1-\alpha)b \in S$ donde $0 < \alpha < 1$. Un cuantizador vectorial es llamado regular si

a) Cada celda R_i es un conjunto convexo

b) Si $x \in R_i$, entonces $Q(x) = y_i$, esta contenido en R_i.

Un cuantizador vectorial Polytopal es un cuantizador regular cuyas celdas de partición están limitadas por segmentos de superficies de hiperplanos de dimension k. Equivalentemente un Polytopal es una intersección de un número finito de espacios medios de la forma $\{x \in R^k: u_v x + \beta_v \geq 0\}$. Un cuantizador vectorial puede ser descompuesto en dos operaciones, la codificación vectorial y la decodificación vectorial. El codificador E es una transformación de R^k al conjunto de índices J, y el decodificador D, transforma

el conjunto de indices J en un conjunto de reproducción C. Así,

$$E: R^k \rightarrow J \quad y \quad D: J \rightarrow R^k \qquad (3.83)$$

La operación total de CV puede ser interpretada como la composición de dos operaciones:

$$Q(x) = DC(x) = D(C(x)) \qquad (3.84)$$

En el contexto de un sistema de comunicaciones digitales, el codificador se encarga de seleccionar un vector de código y_i que aproxime en cierto sentido al vector de entrada x. El indice i del vector de código seleccionado es transmitido (como una palabra binaria) al receptor, donde el decodificador realiza un procedimiento de búsqueda en tablas y genera la reproducción y_i, la aproximación cuantizada del vector de entrada original.
El principal objetivo en el diseño de un cuantizador vectorial es el de encontrar un Alfabeto, una partición y una regla de decisión que maximizará la medida del desempeño total considerando la secuencia entera de vectores a ser codificados por el cuantizador. El desempeño total puede ser medido ya sea por medio de promedios estadisticos de una medida de distorsión conveniente o por medio de la consideración del peor valor de distorsión. El promedio estadistico de la distorsión de un cuantizador vectorial $Q(\cdot)$, puede ser expresado como :

$$D = Ed(X, Q(X)) = \int d(x, Q(x)) f_X(x) dx, \qquad (3.85)$$

donde $f_X(x)$ es la función de probabilidad conjunta del vector X y la integración es una integral múltiple sobre un espacio de k dimensiones. Cuando el vector de entrada tiene una distribución discreta, entonces

$$D = Ed(X,Q(X)) = \sum_i d(x_i, Q(x_i))q_X(x_i) \qquad (3.86)$$

Para un Alfabeto dado, una partición óptima es la que satisface la condición de vecino más cercano, para cada i, todos los puntos más cercanos al vector de codigo y que a cualquier otro código de vector deberá ser asignado a la región R_i. Así, para un conjunto de niveles, Y, la partición de las celdas satisface

$$R_i \subset \{ x: d(x,y_i) \leq d(x,y_j);\ \text{para toda } j \neq i$$

esto es,

$$Q(x) = y_i \text{ solo si } d(x,y_i) \leq d(x,y_j) \text{ para toda } j$$

así, dado el decodificador, el encodificador es una transformación de distorción mínima o del vecino más cercano,

$$d(x,Q(x)) = \min_{y_i \in Y} d(x,y_i) \qquad (3.87)$$

Otra de las condiciones de optimilidad es que para una partición dada $\{R_i;\ i=1,\ldots,N\}$, los vectores de código satisfacen:

$$y_i = cent(R_i) \qquad (3.88)$$

Donde el centroide cent(R) ,de cualquier conjunto $R \in R^k$, es definido como el vector y el cual minimiza la distorsión entre un punto x en R y y ,promediados sobre la distribución de X dado que X pertenece a la región R. Así

$$y^* = cent(R) \text{ si } E\{d(X,y^*)|X \in R\} \leq E\{d(X,y)|X \in R\} \qquad (3.89)$$

o equivalentemente

$$cent(R) = \min_y{}^{-1} E\{d(X,y)|X \in R\}$$

Así, el centroide es aquel vector que en cierto sentido es un representante natural o un vector central del conjunto R y la distribución de probabilidad de R.

Para una medida de distorción de error cuadrático medio, un resultado estándar es que $E(||X - y||^2)$ es minimizado cuando y es la media de X. Aplicando esto a la esperanza condicionada

$$cent(R) = E(X|X \in R) \qquad (3.90)$$

Así, si se asume que cada punto en R tiene una probabilidad igual, entonces para la medida del error al cuadrado se reduce al promedio aritmético:

$$cent(R) = \frac{1}{L} \sum_{i=1}^{L} x_i \qquad (3.91)$$

para $x_i \in R, i=1,2,3,...L$

Las condiciones necesarias para optimilidad proporcionan la base para un diseño óptimo para cuantizadores vectoriales. A continuación se presenta el algoritmo más importante de diseño de cuantizadores vectoriales, que es una versión generalizada del Algoritmo de Lloyd desarrollado para cuantización escalar. Este algoritmo se conoce como LBG (Linde,Buzo,Gray). La operación básica del algoritmo está basado en una operación de modificación iterativa del Alfabeto.

Primero se presenta el caso general cuando la función de probabilidad conjunta del vector de entrada se conoce.

a) Dado un Alfabeto, $Y_m = \{y_i\}$, encuentre la partición óptima de las celdas de cuantización, esto es, use la condición del vecino más cercano para formar las celdas.

$$R = \{ x: d(x,y_i) \leq d(x,y_j);\ \text{para toda } j \neq i\}$$

b) Usando la Condición de Centroide, encuentre el Y_{m+1}, el

alfabeto de reproducción optimo (Alfabeto) para las celdas acabadas de encontrar.

Esta forma de la iteración de Lloyd requiere que la función de densidad de probabilidad (FDP) sea conocida y la geometría de la partición sea especificada para el cálculo de los centroides. La computación de los centroides en el paso b) es generalmente imposible por métodos analíticos. En la practica una integración número ica seria necesaria para encontrar los centroides. Sin embargo una descripción analítica de la FDP generalmente no está disponible en la mayoría de las aplicaciones. En su lugar una distribución muestral basada en las observaciones empíricas es utilizada para generar las iteraciones del Alfabeto.
A continuación se presenta la iteración de Lloyd para datos empíricos:

a) Dado el Alfabeto, $Y_m = \{y_i\}$, particione el conjunto de entrenamiento en los conjuntos de agrupamientos S_i, usando la condición del vecino más cercano:

$$S_i = (x \in T : d(x, y_i) \leq d(x, y_j);\ \text{toda } j \neq i)$$

b) Usando la condición de centroide, calcule los centroides para el conjunto de agrupamientos encontrado, para hallar un nuevo Alfabeto, Y_{m+1}.

Ahora que ya se definio la forma de la iteración de mejoramiento del Alfabeto, el diseño del algoritmo puede quedar como sigue:

1.- Empezar con un Alfabeto inicial Y_1. Poner m=1.

2.- Dado el Alfabeto, Y_m, realice la Iteración de Lloyd para generar un Alfabeto mejorado Y_{m+1}.

3.- Calcular la distorsión promedio para Y_{m+1}. Si ha cambiado por una cantidad muy pequeña desde la última iteración, alto. En caso contrario poner m+1 → m y regresar al paso 1.

Varios criterios para parar pueden ser usados, una versión particular que es común y efectiva es probar si $(D_m - D_{m+1})/D_m$ esta abajo o arriba de un nivel apropiado.
Existen varias técnicas para la inicialización de los Alfabetos, la que nosotros utilizamos es la llamada técnica de separación. Esta técnica fue introducida por Linde y compañía, la idea básica es la siguiente: El Alfabeto globalmente óptimo de resolución cero de una secuencia de entrenamiento es el centroide de la secuencia entera. El código unico , y_0,en este Alfabeto puede ser separado en dos palabras de código, $y_0+\epsilon/2$ y $y_0-\epsilon/2$, donde ϵ es un vector de norma euclidiana pequeña. Este nuevo Alfabeto tiene dos palabras de codigo, el algoritmo de Lloyd puede ser ejecutado en este Alfabeto para producir un código de resolución 1. Cuando estan completos, todos los códigos del nuevo Alfabeto pueden ser separados, formando una suposición inicial para un Alfabeto de resolución igual a 2. Asi se continua de esta manera, usando un Alfabeto de resolución r para formar las condiciones iniciales de un Alfabeto de resolución r+1 por medio de separación.

3.1.3.- MEDIDAS DE DISTORSION

Idealmente una medida de distorsión deberá ser manejable para que permita el análisis y el diseño, además que debe ser evaluable de tal forma que sirva en el proceso de codificación para la selección del vecino más cercano o la salida con menor distorsión. La más conveniente o más ampliamente usada es el error cuadrado o la distancia Euclidiana al cuadrado entre el vector de entrada X y el vector cuantizado $\hat{X} = Q(X)$, definida como:

$$d(X,X) = ||X - \hat{X}||^2 = \sum_{k=1}^{N} (Xi - Xi)^2 \quad . \qquad (3.92)$$

El promedio de la distorsión de error cuadrático o también llamada la distorsión promedio se define como

$$D = Ed(X,X) = E||X - \hat{X}||^2 \quad . \qquad (3.94)$$

Esta medida es frecuentemente asociada con la energía o potencia de la señal de error por esta razón tiene cierto encanto en adición de ser fácil de manejar. Otras medidas de distorsión pueden ser definidas para medir la disimilitud entre la entrada y los vectores de reproducción. Muchas de las medidas de interés en CV tienen la forma

$$d(X,\hat{X}) = \sum_{i=1}^{k} d_m(Xi,\hat{X}i) \quad , \qquad (3.95)$$

donde $d_m(x,\hat{x})$ es una medida de distorsión escalar. De particular interés es el caso cuando la medida de distorción escalar esta dada por $d_m(x,\hat{x}) = |x - \hat{x}|^m$ para valores enteros positivos de m. Cuando m=1, se convierte en la norma l_1 del vector de error, $X-\hat{X}$. Cuando m=2, se obtiene la medida de error discutida anteriormente. Otra medida de distorsión de particular interés es la medida de error cuadrático con ventana

$$d(x,y)=(x-y)^T W(x-y), \qquad (3.96)$$

donde W es una matriz de peso simétrica y definida positiva .Esta medida incluye la distorsión usual de error cuadrático en el caso especial cuando W = I, la matriz identidad. En el caso que se tenga que W es una matriz diagonal con valores diagonales $w_{11} > 0$ se tiene

$$d(x,y) = \sum_{i=1}^{k} w_{11}(x_i - y_i)^2 \quad . \qquad (3.97)$$

Todas las medidas de distorción previamente consideradas tienen la propiedad que dependen de los vectores x y $\hat{x}$ solamente a través del vector de error $x - \hat{x}$. Tales medidas de distorción que tienen la forma $d(x,\hat{x}) = L(x-\hat{x})$ son llamadas medidas de distorción de diferencias. Medidas de distorsión que no tienen esta forma pero que dependen de x y $\hat{x}$ en una forma más complicada han sido propuestas para sistemas de compresión de datos. En sistemas de compresión de voz la medida de distorsión de Itakura y Saito tiene gran importancia.

$$d(x,\hat{x}) = (x-\hat{x})R(x)(x-\hat{x})^t , \qquad (3.98)$$

donde por cada x, $R(x)$ es una matriz de $k \times k$ definida positiva. Esta medida de distorsión es de la misma forma que la definida por la ecuación (3.96).

En los sistemas tradicionales de LPC, los diferentes parametros son cuantizados separadamente, pero es natural pensar en cuantizar estos parámetros usando las técnicas de cuantización vectorial. Los parámetros que describen el modelo normalizado de LPC son cuantizados juntos como vector. Ya que el primer término es igual a 1, quisiéramos cuantizar el vector $(a_1, a_2, \ldots, a_k)$. Una medida de distorsión $d(a,\hat{a})$ entre a y su reproducción $\hat{a}$, puede ser vista como una medida de distorción entre dos filtros o modelos inversos normalizados (ganancia unitaria). Tal medida de distorsión ha sido proporcionada por Itakura-Saito y tiene la forma de la ecuación (3.98) con $R(a)$ la matriz de autocorrelación $\{r_a(k-j); k=0,1,\ldots,k-1; j=0,1,\ldots,k-1\}$. Todas las medidas de distorsión consideradas dependen de la forma de onda de la voz solamente a través de sus propiedades de segundo orden, es decir, usando la autocorrelación o sus modelos espectrales. Estas medidas de distorsión son más fácilmente definidas en el dominio espectral, aunque su evaluación es mas fácil de instrumentar sin

hacer referencia a tal dominio. Estas medidas de distorsión pueden ser usadas entre procesos aleatorios, lo mismo que para el caso deterministico. Una medida de distorsión espectral es una función de dos densidades espectrales, f y $\hat{f}$, el cual asigna un numero no negativo $d(f,\hat{f})$ que representa la distorsión de usar $\hat{f}$ en lugar de f. La más común de tales medidas es la medida de distorsión de diferencias donde se utiliza una norma L_p en la diferencia $f-\hat{f}$. Estas son medidas o distancias en el sentido que ellas satisfacen requerimientos de simetria $d(f,\hat{f})=d(\hat{f},f)$ y la desigualdad del triángulo

$$d(f,g) \leq d(f,h)+d(h,g)$$

Las medidas de distorsión que nosotros utilizamos dependen del logaritmo de la diferencia de los espectros, como resultado, la división de espectros es utilizada y asi medidas de distorsión usando divisiones son utilizados

$$d(f,\hat{f}) = d(1,f/\hat{f}) = d(f/\hat{f},1)$$

En la generación de los Alfabetos para codificación de voz dos medidas de distorsión son efectivas: La medida de distorsión de Itakura-Saito (d_{IS}) y la medida normalizada de Itakura-Saito (d_{GN}). Para dos espectros en potencia f(v) y $\hat{f}(v)$, la distorsión de Itakura-Saito es

$$d_{IS}(f,\hat{f}) = \int_{-\pi}^{\pi} \frac{dv}{2\pi} \left[\frac{f}{\hat{f}} - \ln \frac{f}{\hat{f}} - 1\right], \qquad (3.99)$$

y la distorsión normalizada de Itakura-Saito es

$$d_{GN}(f,\hat{f}) = d_{IS}\left(\frac{f}{\sigma^2}, \frac{\hat{f}}{\hat{\sigma}^2}\right) .$$

La aplicación de d_{IS} a la predicción lineal se hace aparente si f es una muestra de densidad espectral de voz y $\hat{f}$ es un modelo de

reproducción del espectro de la forma

$$\hat{f}(v) = \frac{\hat{\sigma}^2}{|A(z)|^2} \quad , \tag{3.100}$$

donde

$$A(z) = \sum_{k=0}^{M} a_k z^{-k} \qquad \text{con } a_0 = 1 \tag{3.101}$$

y $z = \exp(iv)$

$$d_{IS}(f, \frac{\hat{\sigma}^2}{|A(z)|^2}) = \int_{-\pi}^{\pi} \frac{dv}{2\pi} \left[\frac{f|A(z)|^2}{\hat{\sigma}^2} - \ln \frac{f\ |A(z)|^2}{\hat{\sigma}^2} - 1 \right] , \tag{3.102}$$

para el primer termino de la integral

$$\int_{-\pi}^{\pi} \frac{dv}{2\pi} \frac{f|A(z)|^2}{\hat{\sigma}^2} = \int_{-\pi}^{\pi} \frac{dv}{2\pi\hat{\sigma}^2} f \left| \sum_{k=0}^{M} a_k e^{-jkv} \right|^2 \tag{3.103}$$

$$= \int_{-\pi}^{\pi} \frac{dv}{2\pi\hat{\sigma}^2} \left[f \sum_{k=0}^{M} a_k e^{jkv} \sum_{l=0}^{M} a_l e^{-jlv} \right]$$

$$= \frac{1}{\hat{\sigma}^2} \left[\sum_{l=0}^{M} \sum_{k=0}^{M} a_k a_l \int_{-\pi}^{\pi} \frac{dv}{2\pi} f(v)\, e^{j(k-l)v} \right]$$

$$= \frac{1}{\hat{\sigma}^2} \left[\sum_{l=0}^{M} \sum_{k=0}^{M} a_k a_l r(k-l) \right] \tag{3.104}$$

$$= \frac{1}{\hat{\sigma}^2}\Big[a_0a_0r(0)+ a_0a_1r(-1)+\ldots+ a_0a_m r(-M)$$
$$+ a_1a_0r(1)+ a_1a_1r(0)+\ldots+ a_1a_m r(1-M)$$
$$\vdots$$
$$+ a_ma_0r(0)+ a_ma_1r(-1)+\ldots+ a_ma_m r(-M)\Big]$$

$$= \frac{1}{\hat{\sigma}^2}\Big[\sum_{k=0}^{M} a_k a_k r(0) + 2a_0a_1r(1) + 2a_0a_2r(2) + \ldots + 2a_0a_m r(m)$$
$$+ 2a_1a_2r(1) + 2a_1a_3r(2) + \ldots + 2a_1a_m r(m-1)$$
$$+ 2a_2a_3r(1) + 2a_2a_4r(2) + \ldots + 2a_2a_m r(m-2)$$
$$+\ldots+$$
$$+ 2a_{m-1}a_m r(1)\Big]$$

$$= \frac{1}{\hat{\sigma}^2}\Big[\sum_{k=0}^{M} a_k a_k r(0) + 2\sum_{n=1}^{M}\sum_{k=0}^{M-n} a_k a_{k+n}\, r(n)\Big]$$

$$= \frac{1}{\hat{\sigma}^2}\Big[r_a(0)r(0) + 2\sum_{n=1}^{M} r_a(n)\, r(n)\Big] , \qquad (3.105)$$

donde

$$r_a(n) = \sum_{k=0}^{M-n} a_k a_{k+n} . \qquad (3.106)$$

Para el segundo termino de la ecuación (3.99)

$$\int_{\pi}^{\pi} \ln(f/\hat{f})\, \frac{dv}{2\pi} = \ln(\sigma^2/ \hat{\sigma}^2) . \qquad (3.107)$$

Esta propiedad es debida a Grenader y Szegó [2] la cual ha sido utilizada en la literatura de predicción lineal.
Entonces, sustituyendo en la ecuación (3.99) las ecuaciones (3.105) y (3.107) se obtiene los siguiente

$$d_{IS}(f,\hat{f}) = \int_{-\pi}^{\pi} \frac{dv}{2\pi} \left[\frac{f}{\hat{f}} - \ln \frac{f}{\hat{f}} - 1\right] =$$

$$= \frac{1}{\hat{\sigma}^2} \left[r_a(0)r(0) + 2\sum_{n=1}^{M} r_a(n)\, r(n) \right] - \ln(\sigma^2/\hat{\sigma}^2) - 1 \ , \quad (3.108)$$

y para la distorsión de Itakura-Saito normalizada

$$d_{GN}(f,\hat{f}) = d_{IS}\left(\frac{f}{\sigma^2}, \frac{\hat{f}}{\hat{\sigma}^2}\right) =$$

$$= \left[r_a(0)r(0) + 2\sum_{n=1}^{M} r_a(n)\, r(n) \right] - 1 \ . \quad (3.109)$$

Esta ecuación es una de las mas importantes cuando se realiza el reconocimiento de comandos hablados en tiempo real. La mayor parte del programa de reconocimiento, usando el microprocesador de procesamiento digital de señales el μPD7720, tiene programada esta ecuación. En la sección 4.1.3. se explica la forma en que fue programada esta ecuación.

3.1.4.- RECONOCIMIENTO

En la codificación de la voz usando cuantización vectorial, un solo Alfabeto es diseñado para una secuencia larga de entrenamiento, que representa toda la voz que puede ser codificada por el sistema. En el caso de reconocimiento de comandos hablados, para cada palabra se generan Alfabetos separados .Se diseña cada Alfabeto de una secuencia de entrenamiento conteniendo repeticiones de una de las palabras del vocabulario.

Por ejemplo,para una palabra determinada,se genera un secuencia de Alfabetos, para cada parte en que fue dividida la palabra, usando el algoritmo de diseño de un cuantizador vectorial, en una secuencia de entrenamiento con varias repeticiones de la palabra determinada. Para clasificar la palabra desconocida, primero se códifica cada una de las secciones de la palabra usando cada uno de los Alfabetos de secciónes múltiples y la distorsión promedio para cada sección de Alfabeto es guardada. La palabra desconocida es entonces clasificada de acuerdo al Alfabeto de secciones múltiples que da un promedio de distorsión minimo.

Para ser más precisos, sea V el número de palabras en el vocabulario de reconocimiento, sea T_k el número de palabras pronunciadas en la secuencia de entrenamiento usadas para diseñar el Alfabeto C_k para la k-ésima palabra del vocabulario, donde $k=1,\ldots,V$. También sea F_{qk} el número de bloques en la q-ésima palabra de entrenamiento para C_k donde $q=1,\ldots,T_k$, y finalmente ,sea U_{mqk} el bloque el m-ésimo bloque de la q-ésima palabra de entrenamiento para C_k donde $m=1,\ldots,F_{qk}$. Entonces hay V Alfabetos de secciones múltiples C_k. Cada uno de ellos teniendo Alfabetos por sección C_{kj}. La sección de Alfabeto C_{kj} es diseñada usando usando n bloques por cada palabra de entrenamiento de la k-ésima palabra del vocabulario. Esto es, C_{kj} es diseñada usando los bloques U_{mqk}, donde $m=(j-1)n+1,\ldots,jn$, and $q=1,\ldots,T_k$. En

particular C_{k1} es diseñado usando los primeros n bloques de cada una de las palabras pronunciadas en el entrenamiento de la k-ésima palabra, C_{k2} usando los segundos n bloques, etc. Finalmente, sea C_{kji}, $i=1,\ldots,N_{kj}$ el numero de palabras de código en la sección de Alfabeto C_{kj}.

Supongamos que una nueva palabra va a ser clasificada conteniendo L bloques, y P_l es el conjunto de estimadores de la autocorrelación del l-ésimo bloque ($l=1,\ldots,L$). Ahora, sea D_k la distorsión promedio resultado de codificar la palabra desconocida con el Alfabeto C_k.

$$D_k = \frac{1}{L} \sum_{j=1}^{S_k} d_{kj} \quad . \tag{3.110}$$

Donde S_k es el número de secciones de Alfabetos en C_k, y

$$d_{kj} = \sum_{l=(j-1)n+1}^{min[jn,L]} \min_{i} d(P_l, C_{kji}) \quad ,$$

es la distorsión total de codificar la j-ésima sección de la entrada con la j-ésima sección del Alfabeto C_{kj} de C_k, donde n es el número de bloques por sección.

Entonces la palabra es clasificada como la r-ésima palabra del vocabulario de reconocimiento, donde

$$D_r = \min_{k} D_k \quad .$$

Para nuestro caso cada palabra es dividida en cuatro secciones. Cada sección cuenta con un Alfabeto de cuatro palabras de código. Se hace una detección de inicio y final tanto para la secuencia de entrenamiento como para las secuencias de clasificación. Basicamente, la detección de inicio, se hace cuando la señal excede ciertos umbrales de energía y la detección del final se realiza cuando habiéndose detectado el inicio se encuentra que la energía

de la señal esta por debajo de cierto umbral de energía. Se toman ciertas precauciones, para evitar, que cuando el ruido ambiental exceda el umbral de inicio no se determine que una palabra nueva se este pronunciando. La forma de hacer esto es contando el número de bloques que hay entre el inicio y el final de la palabra, si este número es menor a cierto número preestablecido anteriormente se decide que no hay palabra nueva y que sólamente se trataba de ruido. El úmbral de energía E_{min} es calculado de la siguiente forma

$$E = \sum_{i=1}^{W} x_i^2 \quad ,$$

donde W es el ancho de la ventana de análisis y x_i son las muestras en el dominio del tiempo, de un convertidor de 14 bits a una tasa de muestreo de 8000 Hz, además de haberles efectuado las operaciones de preénfasis y la aplicación de una ventana de Hamming.

En el capitulo 4.1.3 se explica la forma en que se realizarón los experimentos y las diferentes consideraciones que se tomaron para realizar estos.

3.1.4.1.- PRUEBA DE HIPOTESIS

Cuando se realiza el reconocimiento de comandos aislados por medio de la voz, donde se tiene un número determinado de palabras del vocabulario ,puede ser posible que se diga una palabra que no este dentro del vocabulario o que simplemente se halla captado sólamente ruido, en este caso el algoritmo de reconocimiento especificará cual es la palabra dentro del vocabulario que se parece más a esta señal de entrada. En este caso se esta cometiendo un error, puesto que se está decidiendo algo incorrecto. Cuando sucediera esta situación sería deseable que el algoritmo fuera capáz de decidir si la palabra dicha esta dentro del vocabulario o no. Para realizar esta operación se realiza una prueba de hipótesis, donde una de las hipótesis, es que la medida de distorsión global menor que se obtuvo, correspondiente a una de las palabras del vocabulario, esta dentro de un cierto rango de valores de distorsión, cuando se asume que la palabra que se reconocio es la misma palabra que fue dicha; es decir que la palabra dicha sí esta dentro del vocabulario de reconocimiento. Denotemos esta hipótesis como H_0. Por otra parte se tiene la hipótesis contraria, denotada H_1, donde se especifica que la medida de distorsión global menor, esta fuera del rango de distorsiones de la palabra reconocida; es decir la palabra dicha esta fuera del vocabulario de reconocimiento. Basados en una observación sencilla, denotada y, debemos escoger entre estas hipótesis. Un criterio para hacer esta selección debe ser escogido. Un criterio que es rasonable es escoger la hipótesis la cual tenga mas probabilidad de haber ocurrido basados en un observación sencilla. Es decir basados en y ¿ cual de las hipótesis es más probable que sea verdad ?. Las dos probabilidades condicionales son denotadas

$$P(H_0|y) \text{ y } P(H_1|y)$$

Estas son las probabilidades que H_0 es verdad dado y; y la

probabilidad que H_1 sea verdad dado y. Estas probabilidades son llamadas probabilidades a posteriori, así que este criterio es llamado criterio de probabilidad a posteriori máximo. La regla de decisión es la de escoger Ho si

$$P(Ho|y) > P(H_1|y) \quad o \quad \frac{P(Ho|y)}{P(H_1|y)} > 1 \quad ,$$

o escoger H_1 en caso contrario.

La regla de decisión puede tambien ser expresada en términos de funciónes de densidad de probabilidad, que frecuentemente es una forma más conveniente. Así, la regla de decisión puede ser expresada : escójase Ho si

$$P(Ho|y \leq Y \leq y + dy) > P(H_1|y \leq Y \leq y + dy)$$

y escójase H_1 en caso contrario. Usando la definición de probabilidad condicional:

$$P(Ho|y\leq Y \leq y + dy) = \frac{P(y\leq Y \leq y + dy)P(Ho)}{P(y\leq Y \leq y + dy)} \quad ,$$

donde P(Ho) es la probabilidad que la hipótesis Ho es verdadera. La probabilidad de que H_1 sea verdad es entonces 1-Ho. Estas probabilidades son llamadas probabilidades a priori. La función de densidad de probabilidad de la variable y es p(y). Así, $P(y \leq Y \leq y + dy) = p(y)dy$. Similarmente, $P(y\leq Y\leq y + dy|Ho)$ puede ser reemplazado por $po(y)dy$. Entonces

$$P(Ho|y\leq Y\leq y+dy) = \frac{po(y)dyP(Ho)}{p(y)dy} \quad .$$

En el límite para una dy pequeña

$$P(Ho|y)=\frac{po(y)P(Ho)}{p(y)} \quad ,$$

similarmente

$$P(H_1|y) = \frac{p_1(y)[1-P(H_0)]}{p(y)} \quad .$$

La regla de decisión puede ser entonces escrita: escójase Ho si

$$\frac{p_0(y)P(H_0)}{p_1(y)[1-P(H_0)]} > 1$$

o equivalentemente

$$\frac{p_0(y)}{p_1(y)} > \frac{1 - P(H_0)}{P(H_0)} \quad ,$$

el recíproco de esta ecuación puede servir también como regla de decisión: escójase H1 si

$$\frac{p_1(y)}{p_0(y)} > \frac{P(H_0)}{1 - P(H_0)} \quad . \qquad (3.111)$$

La razón $p_1(y)/p_0(y)$ es de particular importancia y es llamada likelihood ratio. Conociendo la función de densidad de probabilidad de $p_1(y)$ y $p_0(y)$, podríamos evaluar la ecuación (3.113). Si se asume que las funciones de densidad $p_0(y)$ y $p_1(y)$ son Gausianas con medias μ_0 y μ_1 y variancias σ_0^2 y σ_1^2 respectivamente, es decir

$$p_0(y) = \frac{1}{(2\pi)^{1/2}\sigma_0} e^{-(y-\mu_0)^2/2\sigma_0^2}$$

y

$$p_1(y) = \frac{1}{(2\pi)^{1/2}\sigma_1} e^{-(y-\mu_1)^2/2\sigma_1^2} \quad .$$

La razón de verosimilitud es entonces

$$\frac{p_1(y)}{p_0(y)} = \frac{\sigma_0 \, e^{-(y-\mu_1)^2/2\sigma_1^2}}{\sigma_1 \, e^{-(y-\mu_0)^2/2\sigma_0^2}}$$

$$= \frac{\sigma_0}{\sigma_1} \; e^{(y^2(\sigma_1^2-\sigma_0^2)+y(2\mu_0\sigma_0^2-2\mu_0\sigma_1^2)+(\sigma_1^2\mu_0^2-\sigma_0^2\mu_1^2))/2\sigma_1^2\sigma_0^2} ,$$

asi la regla de decisión es: escójase H1 si

$$\frac{\sigma_0}{\sigma_1} \; e^{(y^2(\sigma_1^2-\sigma_0^2)+y(2\mu_0\sigma_0^2-2\mu_0\sigma_1^2)+(\sigma_1^2\mu_0^2-\sigma_0^2\mu_1^2))/2\sigma_1^2\sigma_0^2} \geq \frac{P(H_0)}{1-P(H_0)}$$

tomando el logaritmo a esta ecuación, se tiene la siguiente regla

$$(y^2(\sigma_1^2-\sigma_0^2)+y(2\mu_0\sigma_0^2-2\mu_0\sigma_1^2)+(\sigma_1^2\mu_0^2-\sigma_0^2\mu_1^2))/2\sigma_1^2\sigma_0^2 \geq \ln\left(\frac{\sigma_1 \, P(H_0)}{\sigma_0(1-P(H_0))}\right) .$$

(3.112)

Así, para decidir si la palabra reconocida pertenece al conjunto de palabras del vocabulario, es necesario evaluar la expresión (3.112), si el resultado es menor que el valor del lado derecho, se decide que la palabra reconocida si pertenece al vocabulario, en caso contrario se rechaza. Para calcular la expresión (3.112) es necesario contar con los valores de las medias μ_1 y μ_0 y de las variancias σ_1^2 y σ_0^2, en nuestro caso sólamente se cuenta con estimadores de estos valores. La probabilidad a priori, P(Ho), también se conoce.

3.2.- TRANSFORMADA DISCRETA DE FOURIER CON APLICACIONES AL PROCESAMIENTO DIGITAL DE SENALES

En los siguientes capítulos se describen tres técnicas para calcular la Transformada Discreta de Fourier sobre una imágen de 256 por 256 puntos. Los algoritmos que serán descritos son: El algoritmo de la Transformada Rápida de Fourier, el algoritmo de Goertzel y el algoritmo de Descomposición por Sub-Bandas. Estos tres algoritmos fueron programados usando un sistema de cómputo tipo Pc, el cual utilizaba el microprocesador de procesamiento digital de señales el TMS32020. La descripción de este sistema así como la forma en que fué programado cada algoritmo estan descritos en el capitulo 4.2.

3.2.1.- TRANSFORMADA RAPIDA DE FOURIER

En este capitulo damos una revisión de los conceptos básicos de la Transformada Rápida de Fourier (FFT, siglas en inglés). La Transformada Discreta de Fourier (TDF) es la versión discreta en el tiempo de la transformada de Fourier en el tiempo continuo. La transformada de Fourier de una señal analógica x(t) es

$$X(w) = \int_{-\infty}^{\infty} x(t)\, \exp(-jwt)dt \quad , \tag{3.113}$$

donde en general, las funciones x(t) y X(w) son funciones complejas de la variable continua en el tiempo t y la variable w en el dominio de la frecuencia. La señal continua en el tiempo x(t) es convertida a una señal discreta en el tiempo x(nT) muestreandola cada T segundos. La TDF esta dada por

$$X(w) = \sum_{-\infty}^{\infty} x(n)\exp(-jwn) \quad , \tag{3.114}$$

donde w representa la frecuencia normalizada y toma valores entre 0 y 2Π. X(w) es periódica con periodo 2Π, como resultado solo es necesario considerar sus valores entre 0 y 2Π. En general, muestreo en el dominio del tiempo es asociado con periodicidad en el dominio de la frecuencia, y al revés, muestreo en el dominio de la frecuencia corresponde a periodicidad en el dominio del tiempo. Esta propiedad es un resultado básico de la teoria de Fourier, y forma el fundamento para la transformada discreta de Fourier. Asúmase que la señal consiste de N muestras. Ya que no existe

restricción de qué es lo que pasa fuera de los N puntos, es conveniente asumir que la señal es repetida periodicamente. Bajo esta suposición, y debido a la correspondencia mencionada anteriormente entre muestreo y periodicidad, la transformada de Fourier se convierte discreta con una distancia entre muestras sucesivas de igual a la frecuencia fundamental de la señal en el dominio del tiempo. Esta distancia es $2\Pi/N$ en unidades de frecuencia normalizada. Entonces la DFT, esta dada como :

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \quad , \; k=0,1,\ldots,N-1 \qquad (3.115)$$

donde $W_N = \exp(-J2\Pi/N)$, y W_N es conocido como la fase o factor de giro (Twiddle). La ecuación (3.115) es generalmente referida como una TDF de N puntos. Debido a que el numero de multiplicaciones y adiciones complejas requerido es aproximadamente N**2 para un valor de N grande, el número total de operaciones aritméticas requerido para una N dada se incrementa rápidamente si este valor cambia ligeramente. El excesivo número de cálculos requerido para calcular la TDF directamente cuando N es grande ha hecho que se busquen métodos alternativos para calcular eficientemente la TDF. La mayoria de estos métodos hacen uso de la simetria inherente y periodicidad del factor de giro.
Se tienen las siguientes relaciones de periodicidad y simetria:

Propiedad de Simetria : $W_N^k = - W_N^{k+(N/2)}$ (3.116)

Propiedad de Periodicidad : $W_N^k = W_N^{N+k}$ (3.117)

Un método más eficiente para calcular la TDF que significativamente reduce el número requerido de operaciones aritméticas es el algoritmo de decimación en el tiempo llamado

FFT. En este algoritmo, N es un número factorizable que permite que todos los puntos de la TDF sean descompuestos en transformaciones mas pequeñas. El tamaño de la transformación más pequeña así derivado es conocido como el radix del algoritmo. Así para un algoritmo de FFT con radix 2, la transformada o mariposa más pequeña usada es la TDF de dos puntos. Generalmente, para una FFT de N puntos, hay como resultado N muestras en el dominio de la frecuencia que corresponden a N muestras en el tiempo de la señal de entrada x(n). Para una FFT de radix 2, N es una potencia de 2. El número de operaciones aritméticas puede ser reducido inicialmente descomponiendo la TDF de N puntos en dos trasformadas de N/2 puntos cada una. Esto significa que la secuencia en el tiempo x(n) es descompuesta en dos subsecuencias de N/2 puntos, las cuales consisten de las muestras numeradas impares y las muestras numeradas pares con los índices expresados matemáticamente como 2n y 2n+1, respectivamente. Substituyendo estos índices en el tiempo en la ecuación original de la TDF da como resultado

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1)\, W_N^{(2n+1)k} \qquad (3.118)$$

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, W_N^{2nk} + W_N^{k} \sum_{n=0}^{N/2-1} x(2n+1)\, W_N^{(2n+1)k} . \qquad (3.119)$$

Ya que

$$W_N^2 = (\exp(-j(2\pi/N))^2 = \exp(-j\pi/(N/2))^2 = W_{N/2} .$$

Entonces la ecuación (3.119) puede ser escrita como

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\, W_{N/2}^{nk} + W_N^{k} \sum_{n=0}^{N/2-1} x(2n+1)\, W_{N/2}^{nk} , \qquad (3.120)$$

$$= Y(k) + W_N^k Z(k) \qquad k=0,1,\ldots,M-1,$$

donde Y(k) es el primer término de la sumatoria y Z(k) es el segundo término de la sumatoria. Y(k) y Z(k) pueden ser vistos como una TDF de N/2 puntos de las muestras pares y las muestras impares, respectivamente. En este caso el número de multiplicaciones y sumas complejas es aproximadamente N+2(N/2)**2 porque de acuerdo a (3.120), la TDF de N puntos es separada en dos TDF de N/2 puntos, los cuales son combinados por medio de N multiplicaciones y sumas complejas. De este modo, separando la TDF de N puntos en dos TDF de N/2 puntos, el número total de operaciones aritméticas ha sido reducido. A pesar de que (3.120) puede ser usado para evaluar X(k) para $0 < k < N-1$, se pueden hacer todavía más reducciones en el cuando la propiedades de simetría (3.116) y periodicidad (3.117) del factor de giro son utilizadas para calcular X(k) separadamente sobre los siguientes rangos:

Primer medio del espectro en frecuencia: $0 < k < (N/2) - 1$

Segundo medio del espectro en frecuencia: $(N/2) < k < (N-1)$

la ecuación (3.120), para $N/2 < k < N-1$, puede ser escrita como

$$X(k+N/2) = \sum_{n=0}^{N/2-1} x(2n)\, W_{N/2}^{n(k+N/2)} + W_N^{k+N/2} \sum_{n=0}^{N/2-1} x(2n+1)\, W_{N/2}^{n(k+N/2)} , \qquad (3.121)$$

donde $0 < k < (N/2)-1$

Ya que

$$W_{N/2}^{n(k+N/2)} = W_{N/2}^{n(N/2)} W_{N/2}^{nk} = e^{-j2\pi n} W_{N/2}^{nk} = W_{N/2}^{nk}$$

y

$$W_N^{k+N/2} = e^{j\pi} W_N^k = - W_N^k ,$$

entonces la ecuación (3.121) puede ser escrita como

$$X(k+N/2) = \sum_{n=0}^{N/2-1} x(2n)\, W_{N/2}^{nk} - W_N^k \sum_{n=0}^{N/2-1} x(2n+1)\, W_{N/2}^{nk} \qquad (3.122)$$

$$= Y(k) - W_N^k Z(k) \quad . \qquad k=0,1,\ldots,(N/2)-1$$

De este modo la ecuación (3.120) puede ser utilizada para calcular la primera parte del espectro en frecuencia de X(k) para el índice $0 < k < (N/2)-1$, mientras que la ecuación (3.122) puede ser utilizada para calcular la segunda parte del espectro en frecuencia X(k+N/2). El proceso de decimación y la explotación de la simetría puede reducir el calculo de la TDF tremendamente. Si se vuelve a decimar las muestras numeradas como pares y como impares de la misma forma que se hizo anteriormente cuatro TDF de N/4 puntos son obtenidas,resultando tener todavía más reducción en el cómputo. Consecuentemente, para llegar al algoritmo final de radix 2, este proceso de decimación es repetido hasta que eventualmente una TDF de N puntos pueda ser evaluada como una colección de TDF en mariposa de 2 puntos, como se muestra en la figura (3.4). En este algoritmo, el proceso de decimación pasa a través de un total de M estados donde N = 2**M con N/2 FFT en forma de mariposa de dos puntos por etapa, dando un total de (N/2)log2(N) mariposas para una FFT de N puntos. En la figura 3.4 la mariposa de dos puntos es dibujada, donde P y Q son las entradas, que en general son datos complejos tanto como el factor de giro.

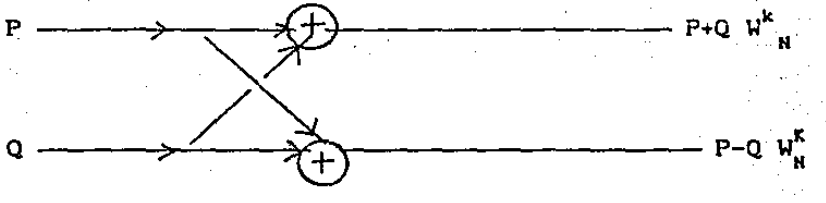


Figura 3.4.- Diagrama de flujo de una mariposa de Radix 2

Como se muestra en la figura 3.4 la salidas P'y Q' de la mariposa de radix 2 esta dada por

$$P' = P + Q\,W_N^k$$

$$Q' = P - Q\,W_N^K \quad .$$

La transformada rápida de Fourier fué llevada a cabo de acuerdo al algoritmo de Cooley Tukey [4]. En este algoritmo las multiplicaciones estan organizadas en una mariposa genérica de un arreglo de números complejos y un arreglo de cálculos de salida. En general para un arreglo de N puntos de datos de entrada hay L arreglos de cómputo donde L = log2(N). Para cada arreglo de calculo hay nodos duales situados en un espaciamiento de nodo dual. Porque este nodo dual es independiente de los otros nodos, el calculo de otros nodos pueden ser llevados al mismo tiempo. Así, los requerimientos de almacenamiento son limitados por el arreglo de los datos únicamente y los resultados de cada arreglo de cómputo son colocados en el mismo lugar que en las entradas.
El cálculo de un nodo y su correspondiente nodo dual esta dado por

$$\text{nodo} \qquad x_l(k) = x_{l-1}(k) + W^p\, x_{l-1}(k+N/2^{**}1)$$

$$\text{nodo dual} \qquad x_l(k+N/2^{**}1) = x_{l-1}(k) - W^p\, x_{l-1}(k+N/2^{**}1),$$

donde: l es el número de arreglo computacional
k es el índice del dato en la localida k
N es el tamaño de la FFT
W^p es el factor de giro

El factor de giro para el coseno(x) y seno(x) donde $x=2\pi k/N$ es precalculado y guardado en un tabla de busqueda. Hay que notar que para datos de entrada de N puntos sólamente se necesita una tabla

de 3N/4. El diagrama de flujo de la figura (3.5) se refiere al siguiente algoritmo.

1.- Los datos son introducidos como un número complejo. La parte real y la parte compleja son guardados en localidades consecutivas .

2.- Inicialización del contador de los cálculos de los arreglos. El espaciamiento entre los nodos duales es calculado N2=N/2.

3.- Se revisa si todos los arreglos computacionales han sido calculados, si es así empieza el procedimiento de reacomodo.

4.- I es inicializado, I monitorea los pares de nodos duales. Esto es para asegurar que que los nodos previos considerados no son encontrados por segunda ocasión.

5.- Realiza el cálculo de los valores de nodo y nodo dual como se discutió anteriormente.

6.- K es incrementado.

7.- Revisa si todos los nodos han sido han sido calculados, para evitar que la recalculación de los nodos duales, si es así ve al punto 9.

8.- Se mueve hacia el siguiente nodo en el arreglo incrementándose con el espaciamiento de offset. Ve al paso 5.

9.- Antes de repetir el calculo del arreglo se revisa si no excedimos el tamaño del arreglo. Si es así regresa a 11.

10.- Si k > N-1 se procede con el siguiente arreglo, es decir el nuevo espaciamiento es recalculado. K es puesto a cero y se va al paso 3.

11.- Cuando todos los nodos del último arreglo son calculados se realiza el reacomodo usando la técnica de ordenamiento cambiando el orden de los bits. En el capitulo 4.2 se indica la forma en que se diseño este algoritmo usando un sistema para procesamiento digital de imágenes usando el TMS32020 en la Universidad de California en Santa Barbara.

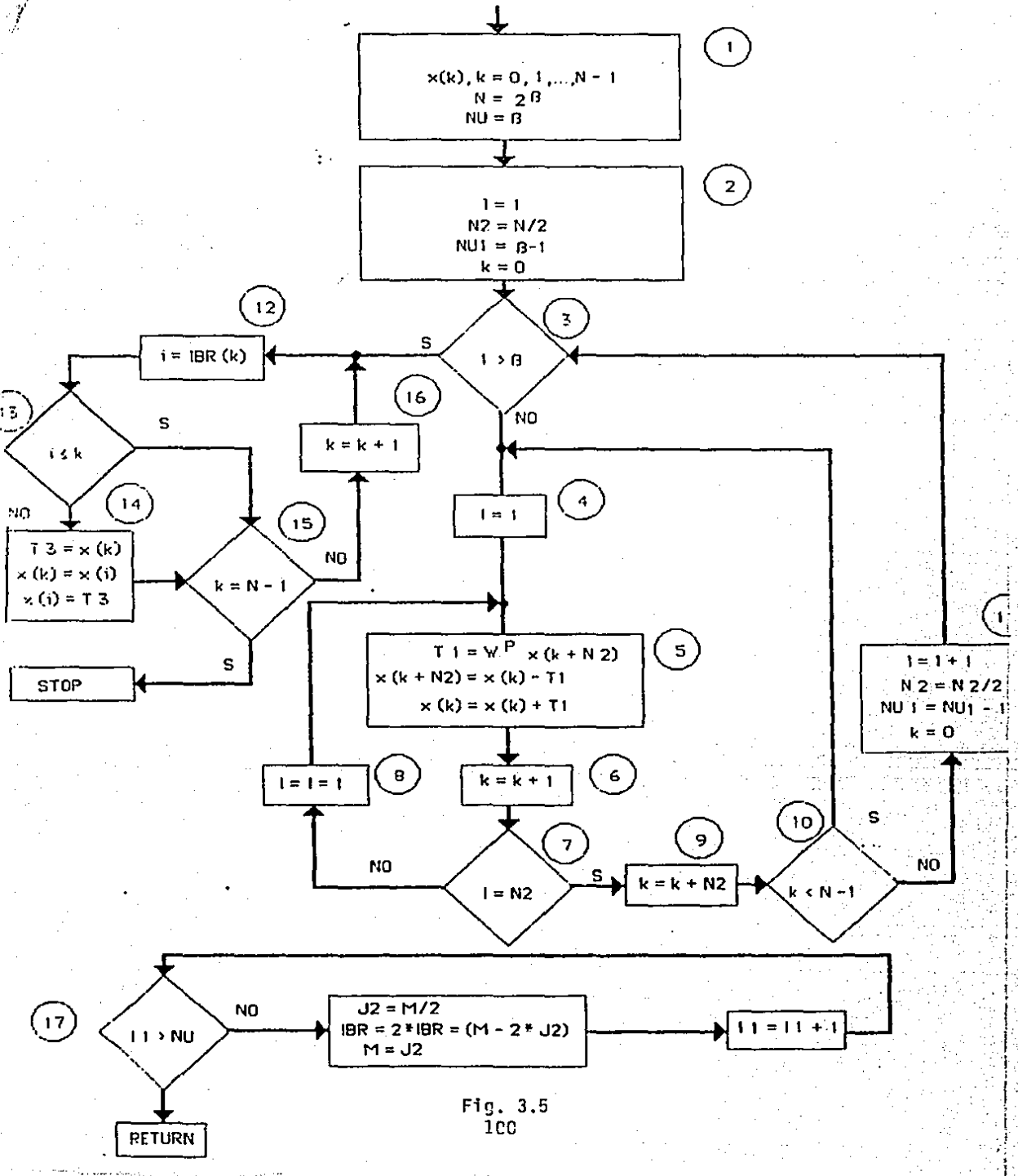

1
x(k), k = 0, 1, ..., N - 1
N = 2^β
NU = β
2
l = 1
N2 = N/2
NU1 = β-1
k = 0
3
l > β
S
NO
12
i = IBR (k)
13
i ≤ k
S
NO
14
T 3 = x (k)
x (k) = x (i)
x (i) = T 3
15
k = N - 1
NO
S
STOP
16
k = k + 1
4
l = 1
5
T 1 = W^P x (k + N 2)
x (k + N2) = x (k) - T1
x (k) = x (k) + T1
8
l = l + 1
6
k = k + 1
7
l = N2
NO
S
9
k = k + N2
10
k < N - 1
S
NO
1
l = l + 1
N 2 = N 2/2
NU 1 = NU1 - 1
k = 0
17
l l > NU
NO
J2 = M/2
IBR = 2*IBR = (M - 2* J2)
M = J2
l l = l l + 1
RETURN

Fig. 3.5

3.2.2.- EL ALGORITMO DE GOERTZEL

Este método es un ejemplo de como la periodicidad de la secuencia W_N^{kn} puede ser usada para reducir los cálculos necesarios para la transformada discreta de Fourier. Especificamente veremos que esta transformada puede ser vista como la respuesta de un filtro digital el cual ha sido estructurado en una manera que reduce el numero de operaciones aritméticas.

Para derivar el algoritmo de Goertzel, empezamos notando que

$$W_N^{-kN} = e^{j(2\pi/N)Nk} = e^{j2\pi k} = 1 \ . \qquad (3.123)$$

Por otra parte, la definición de transformada discreta de Fourier

$$X(k) = \sum_{r=0}^{N-1} x(r) W_N^{kr} \ , \qquad (3.124)$$

multiplicando esta ecuación por W_N^{-kN} ,no se afecta en nada a esta

$$X(k) = W_N^{-kN} \sum_{r=0}^{N-1} x(r) W_N^{kr} \qquad (3.125)$$

$$= \sum_{r=0}^{N-1} x(r) W_N^{-k(N-r)} \ ,$$

por conveniencia definamos la secuencia

$$y_k(n) = \sum_{r=0}^{N-1} x(r) W_N^{-k(n-r)} \ , \qquad (3.126)$$

de las ecuaciones (3.125) y (3.126) se concluye que

$$X(k) = y_k(n) \Big|_{n=N} \ .$$

Evidentemente la ecuación (3.126) es una convolución discreta de una secuencia de duración finita x(n), $0 \leq n \leq N-1$, con la secuencia W_N^{-kn}. Asi $y_k(n)$ puede ser vista como la respuesta de un sistema, con respuesta impulso igual W_N^{-kn}, a una entrada x(n). En particular, X(k) es el valor de la salida cuando n=N. Un sistema con respuesta unitaria igual W_N^{-kn} es dibujado en la figura 3.6.

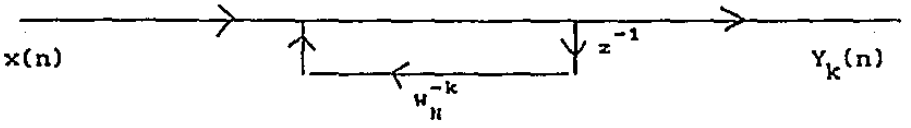


Figura 3.6.- Diagrama de flujo de la computación recursiva de primer orden de X(k).

Este sistema se obtiene tomando la transformada Z de W_N^{-kn}

$$H_k(z) = \sum_{n=0}^{\infty} W_N^{-kn} z^{-n} = \sum_{n=0}^{\infty} (W_N^{-k} z^{-1})^{-n} = \frac{1}{1 - W_N^{-k} z^{-1}} \qquad (3.127)$$

Ya que las entradas x(n) y W_N^k son complejas, el cálculo de cada nuevo valor de $y_k(n)$ requiere de cuatro multiplicaciones reales y cuatro sumas reales. Ya que los valores intermedios $y_k(1)$, $y_k(2)$, $\ldots, y_k(N-1)$ deben ser calculados para $y_k(N) = X(k)$, usando la figura 3.6 requiere de 4N multiplicaciones y 4N sumas reales para X(k), para un valor particular de k. Así, éste esquema es ligeramente menos eficiente que el método directo. Sin embargo, hay que notar que este esquema no requiere el cálculo, o el almacenamiento de los coeficientes W_N^{kn}, ya que estas cantidades son calculadas implícitamente en el proceso de recursión de la figura 3.6. Es posible reducir el número de operaciones por un factor de 2 mientras que se puede seguir trabajando en este esquema. Multiplicando el numerador y el denominador de $H_k(z)$ por el factor $(1-W_N^k z^{-1})$, se obtiene

$$H_k(z) = \frac{1 - W_N^{-k} z^{-1}}{(1 - W_N^{-k} z^{-1})(1 - W_N^{-k} z^{-1})}$$

$$\frac{1 - W_N^{-k} z^{-1}}{1 - 2\cos((2\pi/N)k)\ z^{-1} + z^{-2}}\ , \qquad (3.128)$$

la figura 3.7 muestra el diagrama de flujo de esta función de transferencia. Sólamente dos multiplicaciones son necesarias para evaluar los polos de esta función de transferencia, ya que los coeficientes son reales y (-1) no es preciso contarlo como una multiplicación. Como antes cuatro sumas son necesarias para implementar los polos. Ya que sólamente es necesario obtener un resultado de este sistema cuando $y_k(N)$, la multiplicación compleja de $-W_N^k$ requerida para instrumentar el cero no es necesario realizarla cada iteración, sólamente hasta la N-ésima iteración. Así, el número total de operaciones son 2N multiplicaciones reales y 4N sumas reales para los polos más 4 multiplicaciones y sumas reales para el cero. Así, el número total de operaciones 2(N+2) multiplicaciones reales y 4(N+1) sumas reales, cerca de la mitad del número de multiplicaciones reales por el método directo.

En este método de Goertzel no es necesario evaluar todos los N valores diferentes de X(k). En general podemos evaluar X(k) para cualesquiera M valores de k, y no es necesario tener guardada toda la tabla Wn. En este caso el número de operaciones es proporcional a NM. El cálculo de X(N-k) tiene el mismo número de polos que en la ecuación 3.128, pero el coeficiente del cero es el complejo conjugado del cero en 3.128. Ya que el cero sólamente es calculado en la última iteración, las 2N multiplicaciones y 4 adiciones requeridas para los polos pueden ser utilizadas para el cálculo de dos valores de la TDF. Así el número de multiplicaciones requerido para el cálculo de una TDF de N puntos es igual a N**2 y el número de sumas es 2*N**2. Este esquema es atractivo cuando N es pequeño. Debido a que la estructura de este algoritmo tiene la forma de un filtro IIR se puede programar directamente en cualquier microprocesador de PDS. Además,si sólamente nos interesan

algunos componentes de frecuencia, usando este algoritmo no es necesario tener la tabla de senos y cosenos guardada en memoria, que en algunos microprocesadores DSP es una limitación.
La forma en que fué diseñado éste algoritmo así como los problemas que se presentaron en su diseño son presentados en el capítulo 4.2.2.

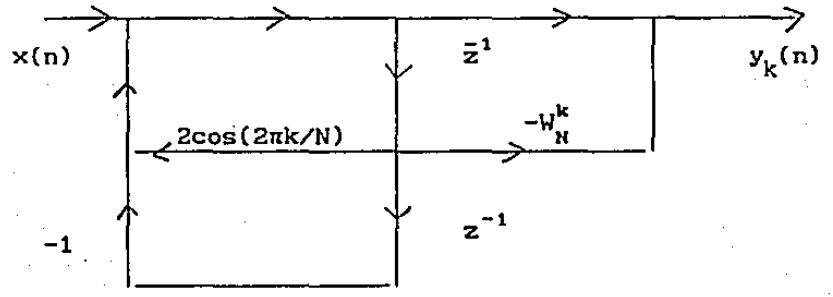
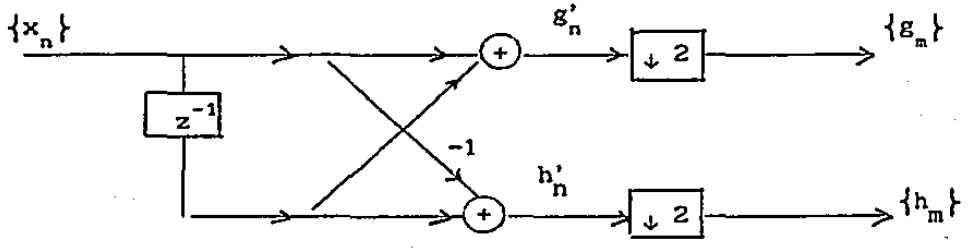


Figura 3.7.- Diagrama de la función (3.128)

3.2.3.- ALGORITMO POR DESCOMPOSICION DE SUB-BANDAS

A continuación se presenta otro algoritmo para calcular la transformada discreta de Fourier. En este algoritmo se separa en dos partes la señal de entrada, una que cuenta con los componentes de baja frecuencia y la otra que cuenta con los componentes de alta frecuencia. De esta forma usando microprocesadores DSP se podrían utilizar diferentes números de bits de precisión para cada banda cuando se efectúa el cálculo de la TDF. Así se podrían utilizar mas bits en la banda en que se tenga más interés.
Considérese el análisis de la siguiente estructura de malla (Lattice).



Si $\{x_n\}$ es una sucesión causal definida para n=0,1,2,..., entonces

$$g'_n = x_n + x_{n-1} \quad ; G'(z) = X(z)(1-z^{-1}) = X(z)H_1(z)$$
$$h'_n = x_n + x_{n-1} \quad ; H'(z) = X(z)(z^{-1}-1) = X(z)H_2(z)$$

$$g_m = \frac{1}{2}\left(x_{2m} + x_{2m+1}\right)$$
$$h_m = \frac{1}{2}\left(x_{2m} - x_{2m+1}\right) \qquad (3.129)$$

m=0,1,2,...,

donde el 1/2 se utiliza con propósitos de normalización, y H_1 y H_2 son filtros paso bajas y paso altas respectivamente.
Para el cálculo de la TDF, $\{x_n\}$ es una sucesión finita de tamaño N. Asumir que $N=2^{\gamma}$. La TDF de $\{x_n\}$ está dada por

$$X_k = \sum_{n=0}^{N-1} x_n \, W_N^{nk} \qquad , \; k = 0,1,\ldots,N-1 \; .$$

De la ec. (3.129)

$$x_{2m} = g_m + h_m$$

$$x_{2m+1} = g_m - h_m \qquad\qquad m=0,1,2,\ldots,N/2-1,$$

$$X_k = \sum_{m=0}^{N/2-1} \left(g_m + h_m \right) W_N^{2mk} + \sum_{m=0}^{N/2-1} \left(g_m - h_m \right) W_N^{(2m+1)k}$$

$$X_k = \sum_{m=0}^{N/2-1} g_m \left(1 + W_N^k \right) W_N^{2mk} + \sum_{m=0}^{N/2-1} h_m \left(1 - W_N^k \right) W_N^{2mk}$$

$$X_k = \left(1+W_N^k \right) \sum_{n=0}^{N/2-1} g_m \, W_{N/2}^{mk} + \left(1-W_N^k \right) \sum_{n=0}^{N/2-1} h_m \, W_{N/2}^{mk} \qquad (3.130)$$

Las sumatorias del lado derecho son transformadas discretas de Fourier de N/2 puntos:.

$$X_k = \left(1 + W_N^k \right) G_{<k>_{N/2}} + \left(1-W_N^k \right) H_{<k>_{N/2}} \; ,$$

aunque el rango del índice k es sobre N valores, k=0,1,...,N-1, cada una de las sumas en (3.130) necesitan ser calculadas para k entre 0 y N/2-1, ya que G_k y H_k son periódicas en k con período N/2. La fig. (3.8) muestra la TDF para 8 puntos.

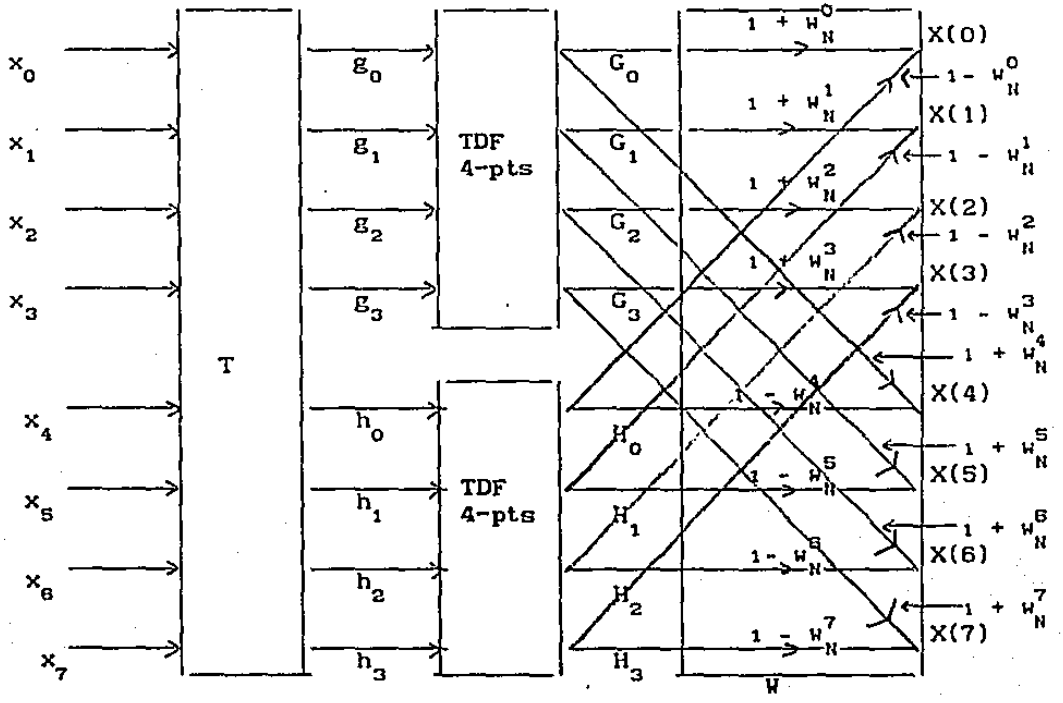


Figura 3.8.- TDF para 8 puntos usando el método de descomposición de Sub-bandas.

La matríz T esta dada por :

$$T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Se tiene entonces que

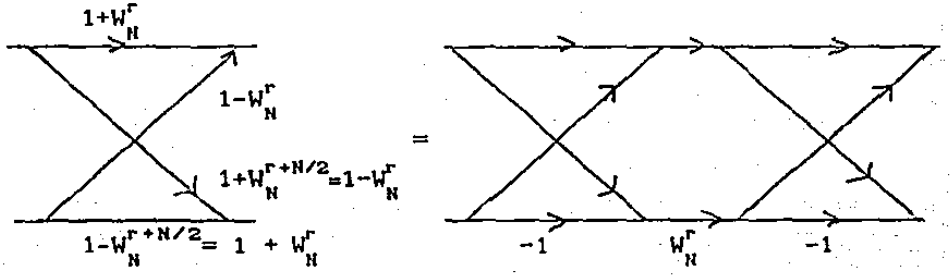
$$\begin{bmatrix} g \\ h \end{bmatrix} = T\,x \qquad (3.131)$$

Por otra parte la matríz W es definida como

$$W = \begin{bmatrix}
1+W & 0 & 0 & 0 & 1-W_N^0 & 0 & 0 & 0 \\
0 & 1+W_N^1 & 0 & 0 & 0 & 1-W_N^1 & 0 & 0 \\
0 & 0 & 1+W_N^2 & 0 & 0 & 0 & 1-W_N^2 & 0 \\
0 & 0 & 0 & 1+W_N^3 & 0 & 0 & 0 & 1-W_N^3 \\
1+W & 0 & 0 & 0 & 1-W_N^4 & 0 & 0 & 0 \\
0 & 1+W_N^5 & 0 & 0 & 0 & 1-W_N^5 & 0 & 0 \\
0 & 0 & 1+W_N^6 & 0 & 0 & 0 & 1-W_N^6 & 0 \\
0 & 0 & 0 & 1+W_N^7 & 0 & 0 & 0 & 1-W_N^7
\end{bmatrix}$$

entonces $X = W \begin{bmatrix} g \\ h \end{bmatrix}$ (3.132)

Para el cálculo de los valores de salida de cada una de las mariposas de este esquema se puede hacer la siguiente simplificación:



En la mariposa del lado izquierdo se efectúan cuatro multiplicaciones y 2 sumas, mientras que en la del lado derecho se efectúan 1 multiplicación y 4 sumas. La mariposa del lado

derecho puede ser entendida más claramente utilizando notación matricial.

$$\begin{bmatrix} 1+W_N^r & 1-W_N^r \\ 1-W_N^r & 1+W_N^r \end{bmatrix} = \begin{bmatrix} 1 & W_N^r \\ 1 & -W_N^r \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -W_N^r \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Así en la figura 3.9 se muestra el esquema de una TDF para ocho puntos usando descomposición de Sub-bandas. Aplicando iterativamente el filtrado y decimación de cada una de las etapas se puede llegar a transformaciones sencillas de dos puntos.
Se puede concluir que el numero de operaciones necesarias para calcular una TDF de N puntos es el siguiente:

$\frac{N}{2} \log_2 N$ multiplicaciones complejas

$3N \log_2 N$ sumas

Comparándola con la FFT : Se necesitan el mismo número de multiplicaciones y 3 veces más el número de sumas.
Se podría buscar un esquema mas sofisticado para el cálculo de la multiplicación de la matriz **T** por **x**, como por ejemplo usando un método de búsqueda en tablas para encontrar esta transformación.
En el capítulo 4.2.2 se describen la forma en que fué programado éste algoritmo.

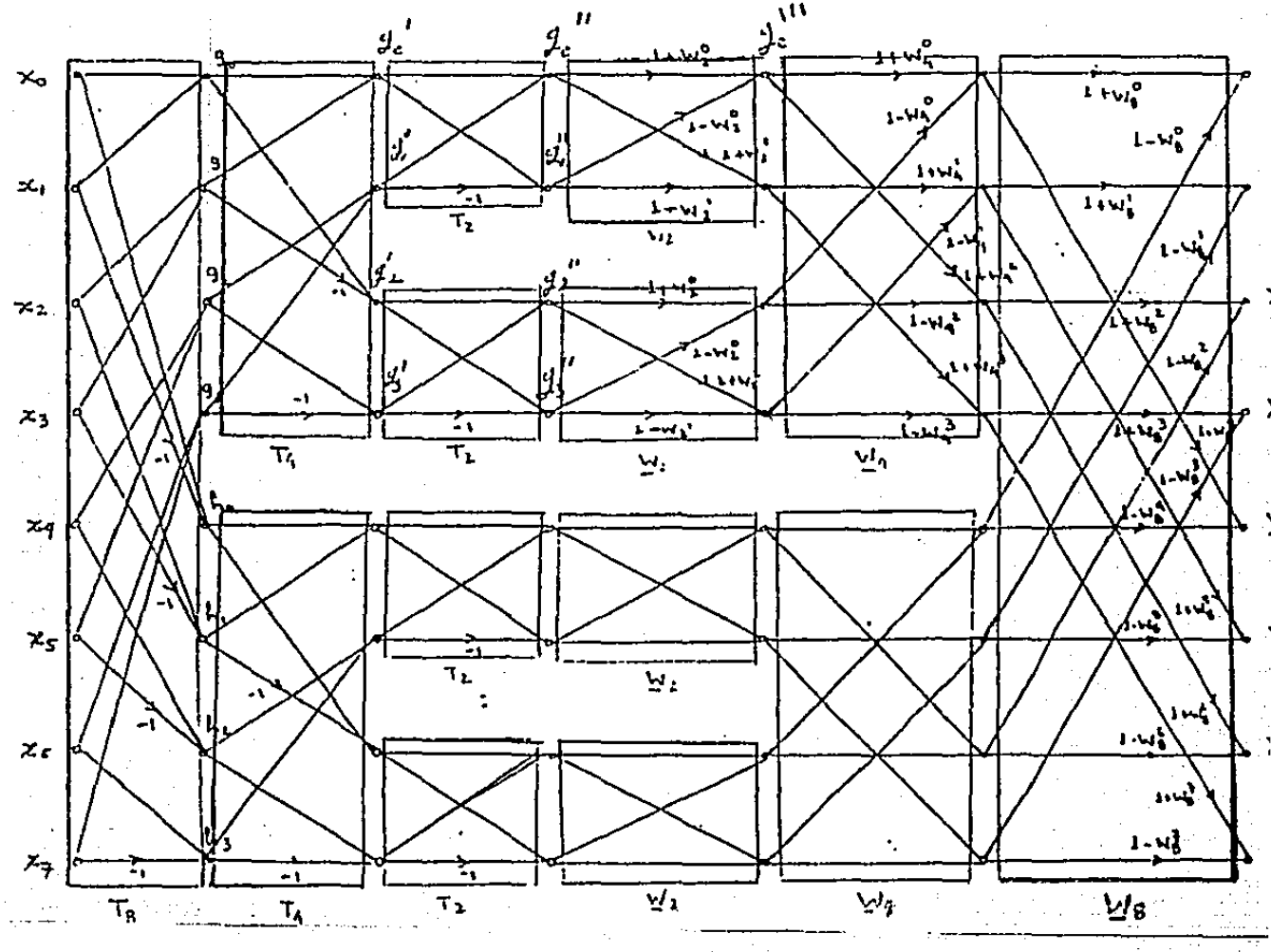


Figura 3.9.- Calculo completo de una TDF de 8 puntos por medio del método de descomposición de Subbandas

3.3.- FILTRADO ADAPTABLE

En el inciso 3.1.2. se hizo un análisis de un filtro, donde su salida estimaría un valor deseado d(n), en ese caso el valor deseado era la estimación de la entrada u(n). La solución de este filtro nos llevó a la ecuación llamada Wiener-Hopf o ecuación Normal (3.35). El diseño del filtro de Wiener requiere una información a priori de los datos a ser procesados. El filtro es óptimo solamente cuando las características de la estadística de la entrada son iguales a la información a priori en la cual el diseño del filtro está basado. Cuando la información no es conocida completamente, entonces no es posible diseñar el filtro de Wiener o el diseño en este caso ya no es óptimo. Una forma de solucionar este problema es el de estimar los parámetros de la estadística de la señal de entrada y después utilizar estos resultados en una fórmula no recursiva para calcular los parámetros del filtro. Para una operación en tiempo real, este procedimiento requiere de Hardware muy costoso y complejo. Un método más efeciente es el uso de filtros adaptables. Tal dispositivo se diseña a sí mismo, usando para esto un algoritmo de recursión, el cual hace posible que el filtro trabaje en una forma satisfactoria en un medio ambiente en donde las principales características de la señal son desconocidas. El algoritmo empieza con ciertas condiciones iniciales, representando una completa ignorancia del sistema. Así, en un medio ambiente estacionario el algoritmo converge a la solución óptima de Wiener en un sentido estadístico. En un medio ambiente no estacionario el filtro es capaz de seguir las variaciones de la estadística de la señal de entrada, si estas variaciones son lo suficientemente lentas. Como una consecuencia directa del algoritmo de recursión, los parámetros del filtro son actualizados en cada iteración, los parámetros se vuelven dependientes de los datos, esto significa que el filtro adaptivo

es un dispositivo no lineal. Además por otra parte se dice que estos filtros son lineales porque para estimar la función de interés es obtenida adaptivamente (a la salida del filtro) como una combinación lineal del conjunto de observaciones aplicadas al filtro de entrada. Se han desarrollado una variedad de estos filtros recursivos, en este trabajo nos centraremos solamente en el metodo del mínimo error cudrático medio (Least Mean Square, LMS) En la siguiente figura se presenta el diagrama de bloques de filtro un filtro adaptable (3.10).

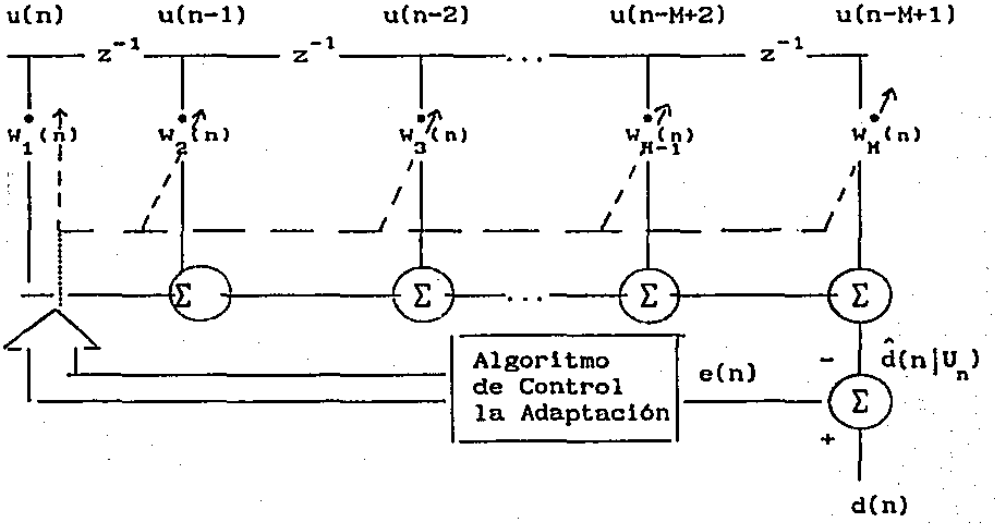


Figura 3.10..- Estructura de un filtro adaptable

Esta estructura consiste básicamente en dos partes, un filtro transversal con los coeficientes adaptables y un mecanismo para adaptar tales coeficientes.

3.3.1.- ERROR CUADRATICO MEDIO MINIMO (LMS)

El requerimiento que un filtro adaptable tiene que satisfacer es el de encontrar un vector de coeficientes que satisfaga la ecuación normal (3.35). Una forma de hacer esto es encontrar la solución de esta ecuación por metodos analiticos. Ya que esto es muy complejo, una forma de alternativa de encontra la solución es utilizar un método llamado Descenso de Pendiente Maximo (Steepest Descent). Para encontrar el valor mínimo del error cuadrático medio, J_{min} ec. (3.36), usando este método, se procedería como sigue:

1.- Se empieza con un valor inicial arbitrario del vector de coeficientes w(0), por lo general se trata del vector nulo.

2.- Se calcula el vector de gradiente, el cual es definido como el gradiente del error cuadrático medio J(n), evaluado con respecto al vector w(n) en el tiempo n.

3.- Se calcula el siguiente valor del vector w(n+1) haciendo un cambio del vector anterior hacia una dirección contraria a la del vector de gradiente.

4.- Se va al paso 2 y se repite el proceso.

Es intuitivamente razonable que correcciones sucesivas del vector de coeficientes en la dirección contraria al vector de gradiente eventualmente llevará al error cuadratico medio J_{min}, donde en este punto el vector de coeficientes toma su valor óptimo, w_0.
De acuerdo al método de Descenso de Pendiente Maximo el valor del vector de coeficientes en el tiempo n+1 es calculado

$$w(n+1) = w(n) + \frac{1}{2}\mu\,[-\nabla(n)], \qquad (3.132)$$

donde $\nabla(n)$ es definido en la ecuación (3.34) y que aquí es reescrito por conveniencia

$$\nabla(n) = \frac{\partial J(n)}{\partial w(n)} = -2r + 2Rw(n),$$

sustituyendo esta ecuación en (3.132) se obtiene

$$w(n+1) = w(n) + \mu [r - Rw(n)] . \qquad (3.133)$$

Hay que observar que el parámetro μ controla el tamaño de corrección del vector de coeficientes de una iteración o otra.
Si fuera posible contar con medidas exactas del vector de gradiente en cada iteración, y el parámetro μ es convenientemente bien escogido, entonces usando este algoritmo convergerá a la solución óptima del filtro de Wiener. En realidad no es posible contar con una medida exacta del gradiente y es necesario utilizar estimadores del gradiente con los datos que estan arribando. La mejor estrategia para obtener un estimador del gradiente $\nabla(n)$ es sustituir los estimadores de la matriz de correlaciones R y el vector de correlaciones cruzada r ,en la fórmula (3.34). La forma más sencilla de escoger los estimadores de R y r es usar los estimadores instantáneos, definidos como

$$\hat{R}(n) = u(n)u^{H}(n) \qquad (3.134)$$

y

$$r(n) = u(n)d^{*}(n) , \qquad (3.135)$$

entonces substituyendo estos valores en (3.34) se obtiene

$$\hat{\nabla}(n) = -2u(n)d^{*}(n) + 2u(n)u^{H}(n)\hat{w}(n). \qquad (3.136)$$

Este estimador es sesgado con repecto al valor esperado. Substituyendo esta ecuacion en la ecuación del algoritmo de descenso de pendiente maximo (3.133)

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n)[d^{*}(n) - u^{H}(n)\hat{w}(n)] , \qquad (3.137)$$

equivalentemente se podria escribir este resultado como

$$e(n) = d(n) - \hat{w}^{H}(n)u(n) \qquad (3.138)$$

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n)e^{*}(n) . \qquad (3.139)$$

La ecuación (3.138) define el error de estimación, mientras que el

segundo término de la ecuación (3.139) del lado derecho $\mu u(n)e^{*}(n)$determina la corrección que debe ser aplicada al vector de coeficientes w(n).

3.3.2.- APLICACIONES DEL FILTRADO ADAPTABLE

Las características de los filtros adaptables hacen que estos sean utilizados en diversas aplicaciones, ya que su habilidad para poder operar en sistemas donde no se conoce o está cambiando la estadística de la señal de entrada lo hacen una herramienta muy poderosa. Estas técnicas han sido utilizadas muy satisfactoriamente en las áreas de control, comunicaciones ,sismología ,radar, reconocimiento de patrones, sonar y procesamiento de imágenes. En esta parte se presentan brevemente una de las tantas aplicaciones del filtrado adaptable.

3.3.2.1.- IDENTIFICACION DE SISTEMAS

Supóngase que tenemos un sistema dinámico desconocido, con un conjunto de muestras discretas en el tiempo, definiendo la variación de la señal de salida del sistema en respuesta una señal conocida estacionaria aplicada a la entrada del sistema. Se asume que este sistema es lineal e invariante. Se requiere entonces modelar un sistema en la forma de un filtro transversal consistente en un conjunto de elementos de retraso y un correspondiente conjunto de coeficientes ajustables, los cuales están interconectados como se muestra en la figura 3.11. En el tiempo n, la señal de entrada consiste en un conjunto de muestras $u(n), u(n-1), \ldots, u(n-M+1)$. Estas muestras son multiplicadas por un conjunto de coeficientes ajustables $w_1(n)$, $w_2(n), \ldots,$ $w_M(n)$ para producir una señal de salida denotada por $y(n)$. Esta salida es comparada con la salida actual del sistema desconocido $d(n)$ para producir un error de estimación $e(n)$. Usando este error de estimación $e(n)$ como la entrada al algoritmo de control de adaptación de los coeficientes, así los coeficientes modifican sus valores con cada iteración. Esta operación es repetida por un gran número de iteraciones ,

hasta que la medida de error entre la planta desconocida y nuestro modelo de éste, , sea lo suficientemente pequeño en algún sentido estadístico. Cuando el sistema desconocido es variante en el tiempo, la salida de este sistema es no estacionaria. En este caso el filtro adaptable seguirá las variaciones estadísticas del sistema.

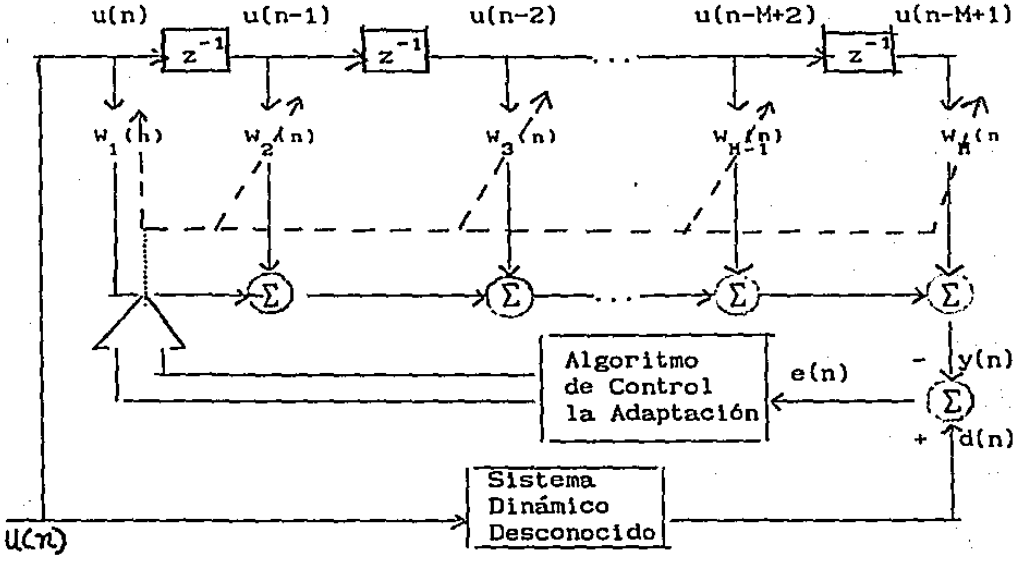


Figura 3.11.- Modelado de un sistema dinámico desconocido

En el capítulo 4.3 se describe la forma en que fué realizado un sistema de procesamiento digital de señales para la identificación de un sistema desconocido en un sistema de cómputo basado en el microprocesador de procesamiento digital de señales el 56000 de Motorola.

C A P I T U L O 4

4.- DESCRIPCION DE LOS EXPERIMENTOS

En este capítulo se presenta la forma en que fue instrumentada cada una de las aplicaciones del capítulo 3, usando microprocesadores de procesamiento digital de señales.

4.1- RECONOCIMIENTO DE VOZ

4.1.1.- DESCRIPCION DEL EQUIPO SIEMENS

El sistema de reconocimiento de comandos usando la voz se instrumentó en un sistema desarrollado por Siemens, el cual está basado en el microprocesador de procesamiento digital de señales el μPD7720. Este sistema cuenta con dos microprocesadores de este tipo, los cuales están interconectados en serie y que comparten memorias RAM y EPROM. Además se cuenta con un microcomputador el 8751 de Intel, el cual cuenta con memorias de datos y programa tanto internas como externas. Tanto los microprocesadores μPD7720 como el 8751 se comunican a través de memorias compartidas y registros de status. Este sistema tiene convertidores analógico digital (A/D) y digital analógico (D/A) de 14 bits cada uno, con sus repectivos filtros paso bajas.
Además, debido a que la forma de entrada y salida de los datos en el μPD7720 es serial, usando los puertos SO y SI , el sistema cuenta con una interface paralelo serie y serie paralelo para convertir los datos de salida del A/D y de entrada del D/A respectivamente. Además se cuenta con un UART y USART que permiten la comunicación externa del sistema con otros dispositivos (PC, terminal,etc). Los microprocesadores μPD7720 son los encargados de ejecutar los algoritmos de procesamiento digital de señales.

Mientras que el microcomputador 8751 es el encargado de realizar las tareas de administrador del sistema, como son, la de establecer la comunicación con el mundo exterior; establecer la frecuencia de muestreo de los convertidores A/D y D/A y de los filtros paso bajas; además el es encargado de decidir quien es el que tiene acceso a las memorias compartidas. En la figura 4.1 se muestra el diagrama de bloques de este sistema.

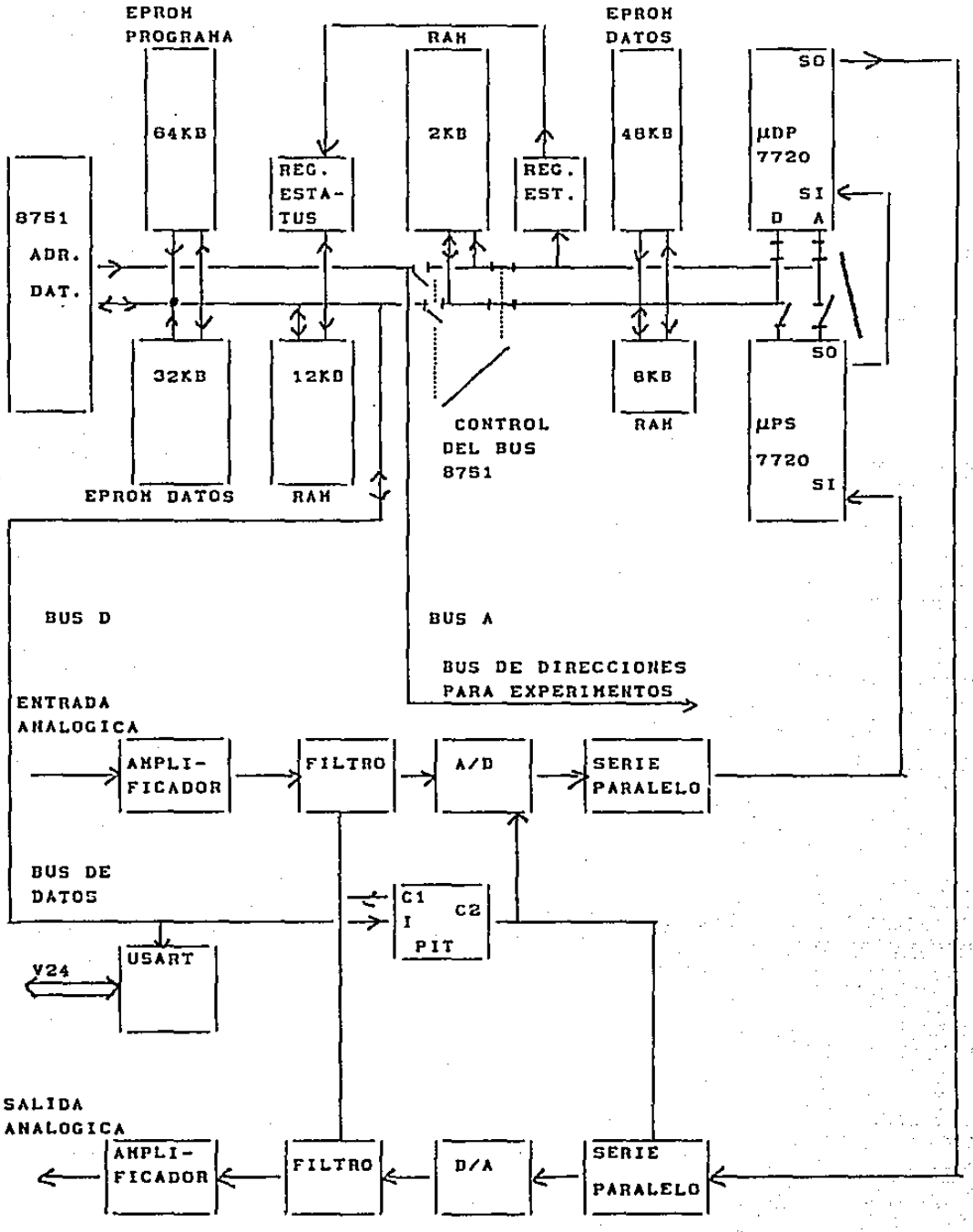


Fig.4.1. Diagrama de bloques del sistema de Siemens (SMS).

4.1.2.- DESCRIPCION DE LOS PROGRAMAS

En esta parte del reporte se presenta una aplicación donde se puede explotar al máximo la arquitectura del equipo desarrollado por Siemens. La aplicación que se desarrolló es la de un sistema de reconocimiento de comandos hablados. Los sistemas de reconocimiento de comandos son sistemas capaces de reconocer palabras aisladas (tomadas de un vocabulario preestablecido) en forma automática. Debido a los adelantos tecnológicos y al desarrollo de algoritmos de reconocimiento de patrones, estos sistemas trabajan en tiempo real con probabilidades de éxito (reconocer el comando o palabra correcta) del orden de 99%, dependiendo del vocabulario preestablecido. De entre las aplicaciones de este tipo de sistemas se eligió una que presenta una complejidad intermedia permitiendo medir las posibilidades de éxito de futuros proyectos en esta área. El sistema seleccionado maneja una serie de menús, que aparecen en la pantalla de una microcomputadora tipo Pc, donde se selecciona una de las opciones que aparecen en el menú; y una vez seleccionada trae otro menú que corresponde a la palabra seleccionada. En el inciso 4.1.3.1 se da una descripción detallada de éste sistema. Otra de las aplicaciones en las cuales se trabajó usando el equipo de Siemens, fué la de desarrollar un conmutador telefónico capáz de comunicar una llamada telefónica externa a una de las extensiones que maneje el conmutador. La forma de indicar a que extensión se desea comunicar es por medio de la voz. La descripción y avance de este sistema se encuentra en el inciso 4.1.3.2. Este sistema es una de las tantas aplicaciones de un sistema de reconocimiento de comandos. Siguiendo esta misma filosofía se podría tener diferentes aplicaciones, como por ejemplo, la comunicación que se podría hacer entre cuentahabiente

de una institución bancaria y esta. En la actualidad para hacer operaciones bancarias por teléfono se necesita un generador de tonos que conectado al teléfono se indica la operacion que se quiere realizar. Usando el sistema de reconocimiento de comandos hablados el generador de tonos podría ser eliminado y en su lugar indicar la operación que se quisiera realizar utilizando la voz. Otras aplicaciones podrían ser utilizadas en el campo de la medicina, donde se podrían tener sistemas que ayudaran a personas minusválidas. Por ejemplo, se podría tener en un cuarto de hospital un sistema de reconocimiento que pudiera efectuar diferentes acciones para un enfermo inmovilizado, tales como tener el control de la iluminación del cuarto, tener el control de la posición de la cama donde se encuentra el paciente,etc. Otro ejemplo, sería una silla de ruedas con motor donde el paciente le indicaría con voz la dirección que debe tomar esta (derecha, izquierda, adelante, atrás). Como se ve el número de aplicaciones podría ser muy variado pero lo importante que hay que hacer notar es que todas los sistemas seguirían con ligeras variaciones la misma filosofía de diseño presentado en este reporte.

4.1.3.- SOLUCION INSTRUMENTADA

A continuación se presenta una descripción detallada de la forma de funcionamiento del sistema de reconocimiento de comandos por medio de la voz que fue desarrollado utilizando el sistema de Siemens.

4.1.3.1.- MANEJO DE MENUS POR MEDIO DE LA VOZ

Aquí se describe detalladamente la forma en que se realiza el reconocimiento de comandos por medio de la voz. El funcionamiento

de este sistema puede ser descrito de la siguiente forma:

1.- En la pantalla de la microcomputadora tipo Pc aparece un menú formado con cuatro palabras.

2.- Una persona pronuncia, por de un medio micrófono una de la palabras que aparece en el menú.

3.- El sistema reconoce la palabra dicha y cambia el menú de acuerdo a ésta.

4.- Se repite el proceso hasta que se llega a un menú donde la seleccion de una palabra realiza un comando especifico.

A continuación se describe la forma de operación del sistema para realizar las funciones antes decritas. El sistema se puede separar en tres partes en cuanto a la forma de operación del Software: el programa que ejecuta el microcontrolador 8751, el programa que ejecuta el 7720 y los programas que ejecuta la Pc. El programa PATRON.A51, que esta escrito en lenguaje ensamblador del microprocesador 8751, es el encargado de realizar las operaciones de control de la arquitectura del SMS. Es decir, es el encargado de ejecutar los comandos, que entran por el puerto serie, y que son enviados por la Pc. Los comandos que podria ejecutar son por ejemplo: recibir los patrones de reconocimiento de un conjunto de palabras de un menú, inicializar la frecuencia de muestreo del convertidor A/D y D/A, indicar la frecuencia de corte del filtro analógico a la entrada y salida de los convertidores, indicarle a los microprocesadores 7720A y 7720B la función a realizar (la forma de indicarlo es a través de interrupciones y comparticion de memorias comunes), enviar correlaciones por el puerto serie a la Pc, enviar el resultado de cual fue la palabra reconocida, etc.

El programa PATRON.NEC esta escrito en lenguaje ensamblador del 7720, y básicamente ejecuta el algoritmo de reconocimiento, además como parte de este algoritmo se calcula la correlación de las muestras que entran por el convertidor A/D; con estas correlaciones se pueden calcular los patrones de una palabra. El programa PATRON.C esta escrito en lenguaje C y es ejecutado por la Pc. Las funciones de este programa son las de establecer la comunicación del SMS con la Pc a través del puerto serie, desplegar en la pantalla los menús de las palabras que pueden ser reconocidos, enviar los patrones de estos menús al SMS, recibir las correlaciones que son calculadas en tiempo real en el 7720, enviar las intrucciones a ejecutar por el SMS, etc. El programa UNION.FTN es un programa fortran que corre en la Pc. Este programa es la unión de diferentes programas, que básicamente la función que realizan es la de crear los patrones de reconocimiento a partir de un archivo de correlaciones. En la computadora Pc aparece un menu donde se indican las opciones que se pueden ejecutar: reconocimiento ,entrenamiento. A continuación se revisan cada una de estas opciones.

4.1.3.1.1.- RECONOCIMIENTO

En la opción de reconocimiento se trabaja en dos modos: uno donde al momento de efectuar el reconocimiento, se calcula la media y la desviación estándar de la medida de distorción de Itakura-Saito de una palabra, repetida varias veces, del menú; Este cálculo se hace para evitar que si se dice una palabra que no se encuentra dentro del menú, no exista la posibilidad de que se confunda con una que si esta en este. En el otro modo se realiza el reconocimiento de una palabra del menú y se trae otro menú dependiendo de la palabra reconocida. La forma de operar en este modo es el

siguiente: La Pc lee de disco los patrones de reconocimiento del primer menú y le indica al 8751 por medio del puerto serie que le enviará los patrones. El 8751 recibe los patrones y los acomoda en la memoria compartida entre el y el 7720 . Una vez que se han recibido los patrones de reconocimiento, que son los patrones de 4 palabras y que ocupan la totalidad de los 2k bytes de la memoria compartida entre el 7720 y el 8751, la Pc le indica al 8751 que realize la función de reconocimiento. Hasta este momento, el control sobre la memoria compartida lo ha tenido el 8751, a partir de que la Pc le indica al 8751 que se realizará el reconocimiento el 7720 será el que tenga control sobre esta memoria. La forma de decidir quien tiene el control sobre la memoria compartida, así como indicarle al 7720A la forma de trabajar, la realiza el 8751. La forma de comunicación entre los microprocesadores 8751 y 7720 se realiza a través de la memoria compartida, un registro, e interrupciones del 8751 al 7720. Al recibir la orden de reconocimiento el 8751 interrumpe al 7720, el cual se encontraba en un estado de espera, y le indica la función a realizar a través de la memoria compartida. El 8751 entonces se queda esperando a que el 7720A le indique la palabra reconocida, la forma de indicárselo es por medio del registro de status.

El conversor digital / analógico se encarga de convertir la señal analógica del micrófono amplificada en una señal digital. El conversor D/A funciona con una frecuencia de muestreo de 8 Khz. Este dato digital entra directamente al microprocesador μPD7720 (A). La primera función del microprocesador de procesamiento digital de señales es la de aplicar una ventana de Hamming en los datos y hacer el preénfasis. Después se calcula la función de autocorrelación de la señal, usando la autocorrelación cero para decidir el inicio de la palabra.

Cuando esta rebasa el umbral se empieza a considerar que se esta diciendo una palabra. La función de autocorrelación es de orden 15 con 128 muestras por bloque y es guardada en la memoria RAM de 8k bytes. Para determinar el final de la palabra se compara de nuevo la función de autocorrelación de orden cero con el umbral, si esta se mantiene por debajo de umbral durante varios bloques, entonces se toma la decisión de que se ha terminado una palabra. Si el número de bloques de la función de autocorrelación es menor que un número determinado de bloques se toma la decision de que la señal que había entrado era solamente ruído y no se toma en cuenta para realizar el reconocimiento y se vuelve a repetir el procedimiento anterior. Una vez que se decidió que la palabra dicha tiene las características antes mencionadas se procede a realizar el reconocimiento. Usando la función de autocorrelación se encuentra la medida de distorción de Itakura-Saito entre la palabra dicha y los patrones que se encuentran en la memoria RAM compartida, y se decide por el patrón de reconocimiento que halla tenido la medidad de distorcion menor a lo largo de toda la palabra. Una vez que ya se sabe cual fue la palabra reconocida el microprocesador 7720 envía el número de la palabra reconocida al registro de status y la distorción a la memoria compartida. Entonces el 8751 lee este número y la distorción y los envia a través del puerto serie a la Pc. La Pc recibe estos números y compara si la distorción calculada esta dentro del rango de distorciones de la palabra reconocida. Si la palabra esta fuera de este rango, esto quiere decir, que probablente se pronunció mal la palabra o que se dijo una palabra que no esta dentro del menú, entonces sale un mensaje en donde se indica que se repita la palabra de nuevo; y se repiten los pasos anteriores que ejecutarón los microprocesadores 8751 y 7720, solamente que en esta ocasión no es necesario enviar de nueva cuenta los patrones de

reconocimiento, si no que se trabaja con los patrones anteriores que ya estan cargados en la memoria compartida. Si la palabra sí esta dentro del rango entonces se envia un mensaje indicando que palabra se reconocio y se lee el archivo que contiene los patrones asociados a esta palabra y se repiten los pasos que anteriormente fueron ejecutados. Es decir se envían al SMS los patrones nuevos a ser reconocidos. En la pantalla de la Pc aparecen los patrones del menú nuevo.

4.1.3.1.2.- ENTRENAMIENTO

En esta parte se describe la forma de realizar el entrenamiento. Se efectúan los mismos pasos que se indicaron en la etapa de reconocimiento para cargar y ejecutar los programas. Cuando en la pantalla de la Pc aparece el menú principal se tiene que escoger la opción de Entrenamiento. Una vez escogida esta opción se pregunta por el nombre de la palabra a crear los patrones. Este nombre servirá para crear una serie de archivos en donde se encontrarán las correlaciones y los patrones de reconocimiento. Después de poner el nombre de la palabra se indica que se pronuncie la palabra. La Pc le indica al SMS que realize la operación de entrenamiento, el microprocesador 8751 interrumpe al 7720A indicándole que efectúe el cálculo de las correlaciones. El cálculo se efectúa tomando en cuenta las mismas consideraciones que se tomaron durante el reconocimiento con respecto al inicio y final de la palabra pronunciada. Estas correlaciones son colocadas en la memoria ram compartida, y cuando se encuentra el final de la palabra pronunciada se le indica al 8751 el número de bloques de correciones calculado, asi este toma control sobre la memoria compartida y envía las correlaciones hacia la Pc. La Pc

toma estas correlaciones y las va guardando en un archivo con el nombre indicado anteriormente. Para crear correctamente los patrones de reconocimiento de una palabra es necesario repetir varias veces ésta. Así, que cada vez que se calculan los patrones de una palabra, se pregunta si se acaba el entrenamiento, si se contesta negativamente se vuelve a repetir el procedimiento del cálculo de las correlaciones. Si se contesta positivamente se cierra el archivo con todas las correlaciones de la palabra que fué repetida varias veces. Una vez hecho lo anterior se pregunta si el cálculo de las correlaciones fue correcto, esta pregunta se hace porque en el cálculo de estas pudo ocurrir que no se haya pronunciado bien la palabra y se quiera eliminar todo el entrenamiento. Si el cálculo es correcto entonces el archivo de correlaciones se utiliza en la siguiente etapa. Después este archivo es leído por los programas, que estan escritos en Fortran, que son los encargados de calcular los patrones de reconocimiento a partir de un archivo de correlaciones. Después de haber sido creados los patrones de reconocimiento, se pregunta si se quiere crear un nuevo menú de comandos usando los patrones de reconocimiento que han sido calculados anteriormente. Se pueden seleccionar cuatro palabras para construir los menús.

4.1.3.2.- SIMULACION DE UN CONMUTADOR TELEFONICO CON ACCESO A LA EXTENCION POR MEDIO DE LA VOZ

En esta parte se describe la forma en que se simula la operación de un conmutador que permite el acceso a una de las extensiones que maneja el conmutador por medio de la voz. La forma de operación es muy parecida a la aplicación que se tiene en el inciso 4.1.3.1. La diferencia principal es que los patrones de

reconocimiento se encuentran grabados en la memoria EPROM. Se cuenta con los patrones del 0,1,2,3,4,5,6,7,8,9, si y no. Los programas que se encargan de realizar todas estas operaciones son los programas PATROM.NEC, RECROM.A51 y RECROM.C. El programa RECROM.C envia los mensajes de operación al sistema SMS, despliega en la terminal los números que estan siendo reconocidos. Este programa es el que realiza la simulación del conmutador. El programa RECROM.A51 es el encargado de recibir los mensajes que le envía la Pc, interpretarlos y enviarle las instrucciones de operación al 7720A; También envia el número de patron que ha sido reconocido. El programa PATROM.NEC realiza el algoritmo de reconocimiento. Este programa efectúa exactamente los mismos pasos que realiza el programa PATRON.NEC en el inciso 4.1.3.1.1. ,pero con la diferencia que los patrones de reconocimiento no se encuentran en la memoria RAM compartida sino se encuentran en la memoria EPROM.

4.1.3.3.- RECONOCIMIENTO DE COMANDOS Y RESPUESTA DEL EQUIPO POR MEDIO DE LA VOZ

En esta parte se presenta la aplicación en donde al mismo tiempo que aparece el menú con las palabras que se pueden reconocer se reproduce un mensaje con voz, indicando lo mismo que aparece en la pantalla de la Pc, en la bocina que esta conectada en el SMS. Además cada vez que se reconoce una palabra del menú se indica cual palabra fué la reconocida por medio de un mensaje de voz a través de la bocina. Los programas encargados de realizar estas funciones son los siguientes PATVOZ.NEC, VOZPAT.A51 y PATVOZ.C . El programa PATVOZ.C corre en la Pc y es el encargado de enviar las instrucciones de operacion al SMS a través del puerto serial. Los patrones de reconocimiento solamente se envian una

sola vez al equipo SMS. Las palabras que se pueden reconocer en esta aplicación son las siguientes: SUBE, BAJA, DERECHA, IZQUIERDA. El programa VOZPAT.A51 es el encargado de recibir los comandos que envia la Pc, además de indicarle al 7720A la operación a realizar, también modifica la frecuencia de muestreo del los convertidores A/D Y D/A. Esta frecuencia tiene que ser modificada porque cuando se esta realizando reconocimiento la señal se muestrea a 8 Khz, y cuando se quiere que se reproduzca un mensaje la frecuencia de muestreo cambia a 2 Khz. La función del programa PATVOZ.NEC es la de realizar el algoritmo de reconocimiento de la misma forma como se indico en el inciso 4.1.3.1.1., además se encarga de reproducir los mensajes que se encuentran grabados en la memoria EPROM . La forma de reproducción es muy sencilla, simplemente se va leyendo la voz digital que ha sido guardada en la memoria EPROM. Estos datos fueron muestreados a 2 Khz y cuantizados utilizando solamente 8 bits. La frecuencia de muestreo fué muy baja debido principalmente al tamaño de la memoria EPROM con la que se contaba. Actualmente se estaba trabajando para tener voz codificada usando técnicas de LPC y así poder tener una mejor calidad de mensajes y un mayor cantidad en tiempo de estos. El mensaje que sale depende de la función que se este realizando así por ejemplo si se realizó el reconocimiento de una palabra el microprocesador 7720A leerá el mensaje que corresponde a indicar que palabra fué la que se reconoció. El microprocesador 7720A leé un Byte del mensaje a reproducir y se lo envia a través delpuerto serie al 7720B. El microprocesador 7720B ya tiene grabado el programa a ejecutar y básicamente este se encarga de leer un dato que viene por el puerto serie y lo envia por el puerto serial de salida hacia el convertidor D/A . Del convertidor D/A sale el mensaje hacia la bocina.

4.2.- TRANSFORMADA DISCRETA DE FOURIER

En esta parte se consideran las cuestiones prácticas que plantea la ejecución de la transformada discreta de Fourier en imágenes usando el microprocesador de procesamiento digital de señales el TMS32020.

La TDF de Fourier de dos dimensiones de una imágen de NxN {u(m,n)} es una transformación separable definida como

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) W_N^{km} W_N^{ln} , \qquad 0 \le k,l \le N-1 \tag{4.1}$$

y la transformada inversa es definida como

$$u(m,n) = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} v(k,l) W_N^{-km} W_N^{-ln}, \qquad 0 \le k,l \le N-1 \tag{4.2}$$

en notación matricial

$$V = FUF \tag{4.3}$$

y

$$U = F^{*} V F^{*} . \tag{4.4}$$

Si U y V son formados, de tal forma que sus renglones sean los vectores u y v, respectivamente. Entonces

$$v = \mathcal{F} u , \qquad u = \mathcal{F}^{*} v \tag{4.5}$$

$$\mathcal{F} = F \otimes F . \tag{4.6}$$

La matriz $\mathcal{F}$ de $N^2 x N^2$ representa la transformada discreta de Fourier de dos dimensiones de NxN. Ya que la transformada discreta de Fourier es separable, la transformada de (4.3) es equivalente a tener dos transformadas discretas unitarias de tamaño N cada una. Cada una de éstas transformadas puede ser calculada en $N\log_2 N$ operaciones usando la FFT. Así, el número total de operaciones es $N^2 \log_2 N$. A continuación se describen las consideraciones practicas, para llevar a cabo la instrumentación de

los diferentes algoritmos que se describieron en el capitulo 3.2, así como una breve semblanza del equipo utilizado en la Universidad de California en Santa Barbara.

4.2.1.- EQUIPO DE PROCESAMIENTO DIGITAL DE IMAGENES DE LA UNIVERSIDAD DE CALIFORNIA

El sistema con el cual estuve desarrollando la aplicaciones que se mencionarón en los capitulos anteriores, estaba basado en una microcomputadora tipo Pc-AT, el cual contaba con lo siguiente: una tarjeta de despliegue de imágenes (FG100), un monitor de alta resolución, una tarjeta de procesamiento digital de señales (PDS) basado en el microprocesador de procesamiento digital de señales el TMS32020.

La tarjeta de PDS esta localizada dentro de la Pc y es direccionada como si fuera una memoria por el microprocesador principal de la Pc. La tarjeta de PDS consiste del TMS32020 con 4k bytes de memoria de programa y 4k bytes de memoria de datos. Estas memorias son compartidas entre el microprocesador de la Pc y el TMS32020. La figura 4.2 muestra el diagrama de bloques de este sistema.

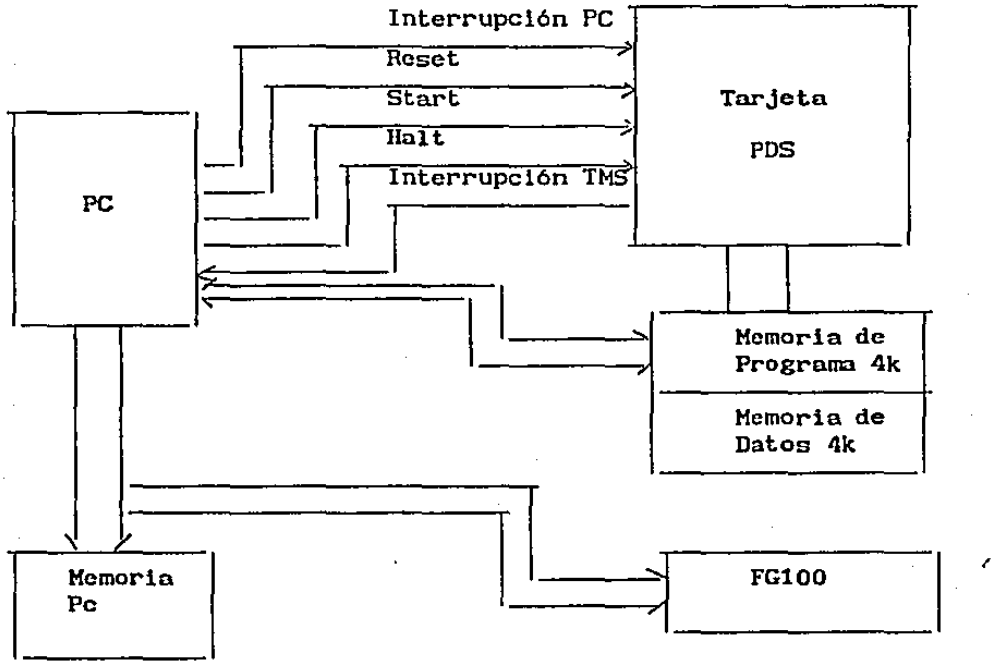


Figura 4.2.- Diagrama de Bloques

Para correr una aplicación donde se utilice el TMS32020, es necesario ejecutar los siguientes procedimientos de comunicación:

1.- La PC inicializa la tarjeta de PDS , enviando señales de control, las cuales dan un reset al microprocesador TMS32020 y limpian todas las memorias de datos. También se inicializa la tarjeta de despliegue de imagenes el FG100, así que una nueva imágen pueda ser desplegada en la pantalla.

2.- La PC carga el código de programa del TMS en la memoria de programa.

3.- Una imagen es entonces cargada en el FG100 y desplegada en el monitor.

4. Usando rutinas de comandos de comunicación del FG100 , que pueden ser invocadas desde un programa en C, la PC es capaz de transferir una columna o un renglón de la imágen a la memoria de datos de la tarjeta PDS.

5.- La PC transfiere entonces el control de ejecución del programa al TMS, enviándole una señal de interrupción, el cual le indica al TMS que empiece a ejecutar su programa de aplicación. Durante este tiempo el TMS tiene el control sobre el la memoria de datos de 4k compartida.

6.- La PC se espera hasta que el TMS la interrumpa, indicándole de esta forma que ya terminó de ejecutar su programa de aplicación.

7.- El control es transferido de nueva cuenta a la PC, y el resultado es leído de la memoria compartida, el cual a su vez es transferido a la memoria del FG100 para que así pueda ser desplegado en la pantalla de alta resolución.

4.2.2.- DESCRIPCION DE LOS PROGRAMAS

La transformada rápida de Fourier fue realizada usando el algoritmo de Cooley Tukey [10]. En el cálculo de la FFT el escalamiento de resultados intermedios es necesario para prevenir sobresaturación. El TMS32020 tiene características para optimizar el procesamiento digital de señales y un una serie de registros de corrimiento para realizar el escalamiento. Así, para cada etapa en el cálculo de la FFT los resultados son divididos entre dos. Así para un vector de 256 puntos todas las salidas seran escaladas por 1/2**8.

Se escribierón dos programas (con varias subrutinas) para el cálculo de una FFT de dos dimensiones, un programa fue escrito en lenguaje de programación C y el otro en lenguaje de ensamblador

del TMS32020. Los programas principales son FFT256.C y FT256.ASM. A continuación se describen los pasos necesarios que efectúan estos programas para calcular la FFT. Hay que hacer notar que para ejecutar los Algoritmos de Goertzel y Descomposición de Sub-Bandas se efectuaban los mismos pasos que se efectuaban para calcular el algoritmo de la FFT. La única diferencia era cuando se cargaba el programa que ejecutaba el TMS, dependiendo del algoritmo que se quería ejecutar, es decir, si se quería correr el algoritmo de Goertzel se cargaba entonces el código del programa del TMS32020 con este algoritmo.

1.- Primero el programa FFT256.C carga en la memoria de programa de la tarjeta de PDS el código del programa del FFT256.ASM. Entonces la PC envía un reset al TMS32020 así que futuras interrupciones a este puedan ser manejadas.

2.- Al usuario entonces se le pide el nombre de la imágen ha ser procesada. Esta imágen es desplegada usando la tarjeta de despliegue el FG-100.

3.- Usando el Mouse el usuario selecciona la región ha ser procesada, el tamaño de la región es de 256 por 256 puntos.

4.- Entonces cada renglón de la región es enviado a la memoria de datos de la tarjeta de PDS, la posición de inicio de estos datos es la localidad 800 hex. Los datos de entrada tienen que ser multiplicados por un factor, esto se hace para tener una escala apropiada para el TMS32020. El formato de los datos deberan estar en formato Q15, es decir , 1 es igual a 7FFF (hex); el valor más grande de los datos de la imágen es 255 y como este valor corresponde al valor de 1, entonces debemos multiplicar este valor por 128 para tener un formato de Q15. La PC interrumpe al TMS para que este empiece el cálculo de la FFT y se queda en un estado de espera hasta que TMS termine, la forma de indicar el TMS que ya termino es por medio de una interrupción a la PC.

5.- La PC entonces lee el resultado de la memoria de datos y

forma una nueva matriz de renglones de FFTs de una dimensión. Si el número de renglones procesados es menor que 256 va al paso 4.

6.- Despues de terminar el cálculo de los renglones, empieza a calcular la FFT en las columnas de la matriz de renglones de la primera FFT. El paso 4 y 5 es repetido solamente que ahora se utilizan las columnas.

7.- Entonces se calculan la magnitud del resultado de las dos FFTs. Se salvá en una matriz y se encuentra la primera y segunda magnitudes mas grandes.

8.- Debido a que el valor de DC es muy grande comparado con los otros valores se escala las magnitudes con el valor mas grande y despues se efectúa la siguiente transformación para poder tener un buen despliegue $u' = \log(1+|u|)$.

9.- La magnitud es entonces centrada y despues desplegada.

Para verificar que ésta aplicación funcionará bien , se hicierón pruebas con imágenes de prueba, tales como circulos, cuadrados y líneas, obteniéndose los mismos resultados que en la bibliografía [11].

Las siguientes consideraciones harian el sistema mas veloz:

1) Debido a que no podemos guardar una matriz de 256 por 512 en la PC debido principalmente por que el compilador no acepta ese tamaño de matriz, se sugirio usar un mejor compilador para solucionar este problema. En este momento se estaban utilizando cuatro matrices de 64 por 512 para solucionar este problema.

2) Expansión de memoria de 128k en la tarjeta del DSP ayudaria a mejorar el tiempo de procesamiento.

3) La velocidad tambien mejoraría si más de una columna de datos es enviada a la tarjeta del TMS. Esto significaría que transferencia de Entrada\Salida disminuyera.

Comparando la operación de este sistema con otros sistemas, por ejemplo las estaciones de trabajo SUN, con la misma aplicación, se observó una ganacia significativa en tiempo de procesamiento.
Así, se puede concluir que el uso de microprocesadores de procesamiento digital de señales veloces combinados con algoritmos rápidos permite la ejecución de trabajos que consumen mucho tiempo en un sistema tipo PC barato, eliminando la necesidad de tener equipo mas poderoso y al mismo tiempo más caro.

4.3.- FILTRADO ADAPTABLE

4.3.1.- DESCRIPCION DE LOS PROGRAMAS

En esta parte se presenta una aplicación de filtrado adaptable con el propósito de identificar la función de transferencia de un sistema desconocido. El propósito de este experimento era probar la operación del microprocesador de procesamiento digital de señales, el 56001 de motorola, usando el algoritmo LMS. El sistema que se quería identificar fué simulado usando este microprocesador. El diagrama de bloques de todo el sistema es presentado en la figura 4.3. El filtro IIR H(z), con polos solamente, con M coeficientes fijos es seguido por un retrazo z^{-1}, y después se tiene el filtro adaptable $C_n(z)$ (donde el índice n indica que los coeficientes de este filtro varían con el tiempo). La entrada u(n) de este sistema es una sucesión de ruido gausiano blanco, es decir, una sucesión de números aleatorios tomados de una función de densidad de probabilidad Gausiana. Los números aleatorios tienen una media igual a 0 y una variancia (potencia) igual a σ_u^2. La salida de d(n) de H(z) puede ser representada por la siguiente función de transferencia:

$$d(n) = u(n) - \sum_{k=1}^{M} a_k \, d(n-k) \quad , \tag{4.7}$$

donde $\{a_k\}$ son los coeficientes del denominador. Se observa que d(n) es simplemente la diferencia entre la entrada y una combinación lineal de las salidas pasadas. Debido a la retroalimentación, H(z) tiene memoria infinita, en el sentido que la salida en el tiempo n es una función de todas las salidas anteriores. La sucesión {d(n)} es frecuentemente llamada un proceso autorregresivo (AR, siglas en inglés).

Filtrando u(n) con H(z), se pueden modificar las propiedades

de la correlación de la entrada . Antes de ser filtrada la sucesión no esta correlacionada, es decir

$$E\ [\ u(n)u(n+m)\ ] = \begin{cases} \sigma_u^2 , & m = 0 \\ 0 , & m \neq 0 , \end{cases} \qquad (4.8)$$

donde E es el operador de esperanza. Después de filtrar,

$$E\ [\ d(n)d(n+m)\ ] = \begin{cases} \sigma_d^2 , & m = 0 \\ c_m(a_k)\sigma_d^2 , & m \neq 0 \end{cases} \qquad (4.9)$$

donde $\{\ c_m(a_k)\}$son funciones de los coeficientes de H(z), y σ_d^2 es la variancia de d(n).

Asumir que la entrada del filtro adaptivo es x(n) = d(n-1). Además asumir que la tarea del filtro es estimar d(n) de sus valores pasados, es decir usando x(n). La salida del filtro es

$$y(n) = \sum_{k=1}^{M} c_k(n)x(n-k) \ . \qquad (4.10)$$

La función del filtro adaptivo es la de ajustar sus coeficientes $\{C_k\}$ por medio de la minimización del error cuadrático medio (MSE) tal que la salida y(n) se aproxime a d(n):

$$MSE \equiv E\ [e^2(n)]\ ,$$

donde

$$e(n) = d(n) - y(n)$$

$$= d(n) - \sum_{k=1}^{M} c_k(n)d(n-k-1) \ . \qquad (4.11)$$

Se puede demostrar que los coeficientes del filtro adaptable convergen a los coeficientes de -A(z), es decir

$$c_{k'conv} = \begin{cases} -a_{k+1} & , \ k = 0,\ldots,N-1 \\ 0 & , \ k \geq N \ . \end{cases} \qquad (4.12)$$

El denominador 1 + A(z) es llamado el filtro inverso de H(z) , y -A(z) es el filtro de predicción óptimo para la estimación de d(n), usando una combinación lineal de sus valores pasados {d(n-1),...,d(n-N)}.

El filtro de adaptación LMS es definido por el siguiente conjunto de ecuaciones,

salida: $$y(n) = \sum_{k=1}^{M} c_k(n)x(n-k)$$

error: $$e(n) = d(n) - y(n) \qquad (4.13)$$

Actualización de los coeficientes:

$$c_k(n+1) = c_k(n) + \mu e(n)x(n-k), \qquad k = 0,\ldots,N-1 \qquad (4.14)$$

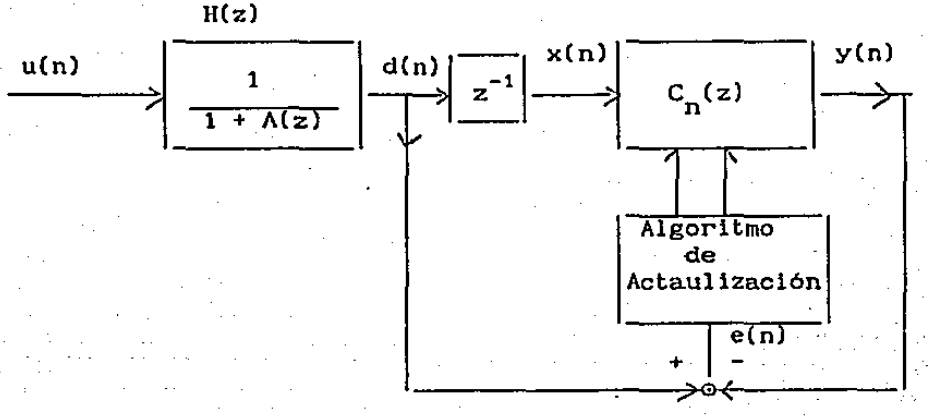


Figura 4.3. Diagrama de bloque de un sistema de identificación de un sistema desconocido

Se planteó la identificación del siguiente sistema H(z)

$$H(z) = \frac{1}{1 - 3z^{-1} + 4.1875\, z^{-2} - 3.4688\, z^{-3} + 1.8203z^{-4} - .58594z^{-5} + .097653z^{-8}} \qquad (4.15)$$

Este sistema fue descompuesto en tres filtros de segundo orden interconectados en cascada para tener el mismo filtro, como se especifica a continuación

$$H(z) = H_1(z)H_2(z)H_3(z),$$

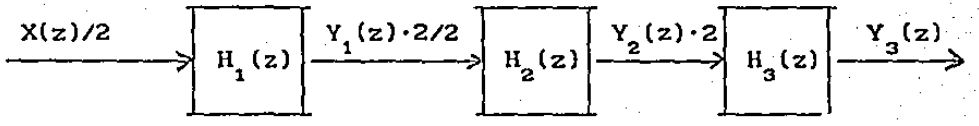
donde

$$H_1(z) = \frac{1}{1 - 1.5z^{-1} + 0.625z^{-2}}$$

$$H_1(z) = \frac{1}{1 - z^{-1} + 0.5z^{-2}}$$

$$H_1(z) = \frac{1}{1 - .5z^{-1} + 0.3125z^{-2}}$$

Para instrumentar estos filtros usando el 56001 de Motorola se tuvierón que escalar los coeficientes de los filtros primero y segundo para poder representar los coeficientes mayores que uno usando 24 bits; por lo tanto fue necesario hacer el escalamiento de los datos de entrada a los diferentes filtros.



Ya que algunos de los coeficientes del filtro H(z) son mayores que

uno, es necesario efectuar escalamiento a los coeficientes del filtro adaptivo, para que estos no se saturen durante la convergencia. Una solución es multiplicar y(n) por un factor g ≥ 1 antes de ser sustraido de d(n) para calcular e(n). El sistema modificado es descrito por las siguientes ecuaciones :

$$e(n) = d(n) - gy(n) \tag{4.16}$$

$$\frac{y'(n)}{g} \equiv y(n) = \sum_{k=1}^{M} c_k(n)d(n-k-1) \ . \tag{4.17}$$

Como resultado,

$$d(n)= gy(n)+e(n) = g \sum_{k=1}^{M} c_k(n)d(n-k-1) + e(n) = \sum_{k=1}^{M} c'_k(n)d(n-k-1) + e(n), \tag{4.18}$$

donde $c'_k(n) = gc_k(n)$. La ecuación 4.18 tiene la misma forma que la ecuación 4.11 del sistema original (g=1). Asi que los coeficientes del filtro $c'_k(n)$ convergerian a $-a_k/g$.

Ya que el valor máximo de $|a_k| = 4.1875$, $g = 8$ se hara que los coeficientes converjan a magnitudes menores a uno. Se escogió un número múltiplo de dos porque asi el escalamiento se puede hacer muy fácilmente con corrimientos.

Escribí un programa instrumentando el algoritmo de LMS en lenguaje ensamblador del 56001 de Motorola. Usando los valores de $\mu = .025$ y $\mu = .25$. Usando .25 la razón de convergencia es más rapida que .025, pero la variancia de los coeficientes es mayor. La convergencia también depende de las condiciones iniciales de las c_k's. Los valores de los coeficientes del filtro adaptable eran muy cercanos a los valores esperados $-a_k$. Si se quisiera tener mejores resultados, seria necesario correr varias realizaciones y obtener un promedio de los coeficientes.

4.4.- CONSIDERACIONES NUMERICAS

En esta parte se presentan algunas consideraciones numéricas que tienen que ser tomadas en cuenta cuando se utilizan los microprocesadores de procesamiento digital de señales de punto fijo. Los cálculos en esta clase de microprocesadores estanbasados en una representación de complemento a dos con punto fijo. Cada número de N-bits (donde N = 16 en los microprocesadores μ7720, familia TMS320; y N = 24 en el 56000) es evaluado con un bit de signo, i bits para la parte entera y N-i bits para la parte fraccionaria. Asi, por ejemplo, el número

0 000 0110 1110 0000

|_____ punto binario

tiene el valor de 6.875 . Este número en particular se dice que esta representado en un formato Q8, ya que se estan utilizando 8 bits para tener la parte fraccionaria. El rango de este número esta entre (1000 0000 0000 0000) -128 y (0 111 1111 1111 1111) 127.996. La precisión de un número en formato Q8 es igual a 1/2**8, (cerca de 0.004.)

Aunque se puede trabajar con números que tengan diferente formato, es común que se trabaje con números con el mismo formato. Esta clase de microprocesadores cuenta con una serie de instrucciones en Software y dispositivos en Hardware que permiten un manejo apropiado del formato que se escoja. Aunque existen estas facilidades, frecuentemente se presentan situaciones en donde es más fácil trabajar con un formato que con otro. Por ejemplo, si se tuviera que hacer una multiplicación de dos números, uno con un formato de Q15 (15 bits para representar la parte fraccionaria) y otro con formato Q8, el resultado correcto quedaría en formato Q23. Por lo tanto, si quisiéramos poner el resultado en alguno de los formatos anteriores, tendríamos que hacer un corrimiento de 8 bits a la derecha para tener un formato

q15; un corrimiento de 15 bits a la derecha para tener un formato de Q8. Pero debido, a que algunos de los microprocesadores solo permiten los corrimientos a la derecha o izquierda de cierto número de bits, tal vez no exista una instrucción que permita hacer los corrimientos deseados, si no que tal vez se tengan que ejecutar varias instrucciones para realizar estos corrimientos. Por esta razón, es muy importante que la persona encargada del diseño de sistemas, usando esta clase de microprocesadores, tenga una idea clara, tanto de la arquitectura, como del conjunto de instrucciones que se puedan ejecutar, para así poder seleccionar el formato más adecuado a sus necesidades. Es muy importante que el diseñador tenga una idea clara de como se manejan los formatos, porque en ocasiones sucede que un algoritmo este entregando una respuesta correcta, pero como no se sabe en que formato esta o si se esté considerando un formato incorrecto, esta respuesta parece errónea o no se la respuesta deseada. Desde mi punto de vista, un manejo y entendimiento de los formatos, es de vital importancia en el diseño de un sistema de procesamiento digital de señales usando esta clase de microprocesadores.

4.4.1.- MULTIPLICACION DE DOS OPERANDOS CON DIFERENTE NUMERO DE BITS PARA SU REPRESENTACION

Usando los microprocesadores de NEC el μ7720 y la Familia TMS320, yo escogí trabajar con la representación de los números con un formato Q15, debido principalmente a que se cuenta con instrucciones poderosas para poder manejar este tipo de formatos. En los diferentes algoritmos que he programado, se ha presentado la situación donde es necesario hacer la multiplicación de dos números, los cuales aunque estan representados usando un formato Q15, utilizan diferente número de bits, ese fue el caso en la parte de reconocimiento, así como en el algoritmo de Goertzel. Por

ejemplo se ha tenido que realizar la multiplicación de un número representado por 32 bits, por ún número representado usando 16 bits. A primera vista pareciera ser un problema trivial, pero debido a la forma en que se manejan los números internamente en las diferentes arquitecturas, este se convierte en un problema difícil, y solamente después de tener un claro entendimiento tanto de la arquitectura como del conjunto de instrucciones, y después de hacer varios experimentos de prueba y error se llega a una solución.

El problema surge principalmente, porque en el número que esta representado con dos palabras de memoria (32 bits), estamos usando la primera palabra como un número, en el cual el bit más significativo ya no representa el signo, si no que representa parte del numero. Pero el Hardware que realiza las operaciones toma este bit como signo, es necesario hacer algunas manipulaciones en software para solucionar este problema. (el TMS320C25 soluciona este problema por Hardware). Por ejemplo usando 32 bits con 15 bits para representar la parte fraccionaria (Q15) se tendria la siguiente relación:

0000 0000 (hex)	= 0.	(dec)
0000 4000 (hex)	= .5	(dec)
0000 7fff (hex)	= .999969482422	(dec)
0000 8000 (hex)	= 1.	(dec)
0000 ffff (hex)	= 1.99996948242	(dec)
0001 0000 (hex)	= 2.	(dec)
7fff 0000 (hex)	= 65535.999969482422	(dec)
8000 0000 (hex)	= -65535.999969482422	(dec)
ffff ffff (hex)	= -0.000030517578	(dec)

Los números entre los rangos 8000H y FFFFH son considerados como negativos, en los microprocesadores, si solamente se utiliza una palabra, y como los registros que se utilizan para la

multiplicación solamente aceptan de 16 bits, entonces los números que caen dentro de este rango son considerados como negativos y no su representación real usando 32 bits.

A continuación se presenta un listado de una rutina en lenguaje ensamblador del TMS32020 solucionando este problema.

```
*
        IDT 'MUL'
*       EN ESTA SUBRUTINA SE MUESTRA LA MULTIPLICACION DE DOS
*       NUMEROS CON REPRESENTACION DE FORMATO Q15.EL PRIMER NUMERO
*       UTILIZA 32 BITS PARA SER REPRESENTADO; EL SEGUNDO NUMERO
*       SOLAMENTE
*       UTILIZA 16 BITS, EL RESULTADO QUEDA EN 32 BITS.
*
*                 RH        RL          0001 8000 H      3.0
*              x        DATO1            x   4000 H     x .5
*              ------------------       -----------     -----
*                RESULTH RESULTL        0000 BFFF H      1.5
*
******************************************************************************
*
*
RH       EQU 110           * PARTE ALTA DEL OPERANDO DE 32 BITS
RL       EQU 111           * PARTE BAJA DEL OPERANDO DE 32 BITS,
*                          * EN FORMATO Q15
DATO1    EQU 112           * OPERANDO DE 16 BITS EN FORMATO Q15
ONE      EQU 113           *
TEMP1    EQU 114
TEMP2    EQU 115
BIGEF    EQU 116
ABSRN1   EQU 117
ABSRNL   EQU 118
ABSDAT   EQU 119
SIGN     EQU 120
RESULH   EQU 121           * PARTE ALTA RESULTADO DE 32 BITS
RESULL   EQU 122           * PARTE BAJA RESULTADO DE 32 BITS,
*                          *  EN FORMATO Q15
*
*
*
******************************************************************************
*
*
```

```
        AORG 0
        LDPK  0
        LACK  1
        SACL  ONE
        LALK  >FFFF
        SACL  BIGEF             * BIGEF <- 2
        LAC     DATO1
        ABS
        SACL    ABSDAT          * ABSDAT <- ABS(DATO1)
        ZAC                     * ACC <- 0
NLOOP   SPM     0               * CORRIMIENTO DEL REGISTRO P=0
SIGNE   ZALS    RH              * DETERMINA EL SIGNO DEL PRODUCTO
        XOR     DATO1
        SACH    SIGN,1          * SE SALVA EL SIGNO DEL PRODUCTO
*                               *  SIGN 0=+, 1=-
ABSRN   ZALH    RH              * SE TOMA EL VALOR ABSOLUTO DE
*                               * RH Y RL
        ADDS    RL
        ABS
        SACH    ABSRN1
        SACL    ABSRNL
        ZAC                     * ACC <- 0
        BIT     ABSRNL,0
        BBZ     POSITO          * SI EL BIT MAS SIGNIFICATIVO DE
*                               * ABSRNL ES IGUAL A 0 SE REALIZA
*                               * LA MULTIPLICACION NORMALMENTE
*                               * ,GOTO POSITO.
        ADDS    BIGEF           * ACC <- VALOR MAXIMO EN Q16 (2)
        ADD     ABSRNL          * ACC <- 2-ABSRNL
        SFR                     * CORRIMIENTO A LA DERECHA PARA
*                               * HACER QUE EL ACUMULADOR SEA <=1
        SACL    TEMP1
        LT      TEMP1           *(2-R(PARTE BAJA))*ABSDAT
        MPY     ABSDAT
        PAC
        SFL                     * SE MULTIPLICA POR DOS PARA
*                               * COMPENSAR LA DIVISION POR 2
*                               * PREVIA
        SACH    TEMP1,1         * SE SALVA EL RESULTADO EN Q15
        ZAC
        ADDS    TEMP1
        LT      ABSRN1
        B       SALTO
POSITO  LT      ABSRNL          * T <- R(PARTE BAJA)
        MPY     ABSDAT          * P <- R(PARTE BAJA)*ABSDAT
```

```
        LTA     ABSRN1          * T <- R(PARTE ALTA)
        SACH    TEMP1,1         * SE SALVA EL RESULTADO EN Q15
        LAC     TEMP1           * ACC <- R(PARTE BAJA)*ABSDAT
SALTO   SPM     1               * CORRIMIENTO DEL REGISTRO P
*                               * IGUAL A 1
        MPY     ABSDAT          * R(PARTE ALTA)*ABSDAT
        SACH    TEMP2
        SACL    TEMP1
        PAC
        ADDS    TEMP1
        ADDH    TEMP2       * ACC <-(R(PARTE ALTA)+R(PARTE BAJA))*DATO1
        SACH    RESULH
        SACL    RESULL
        LAC     SIGN            * SE REVISA EL SIGNO;SE NIEGA
*                               * SI ES NEGATIVO
        BZ      FINAL
        ZALH    RESULH
        ADDS    RESULL
        CMPL
        ADD     ONE
        SACH    RESULH          * RESULTADO PARTE ALTA
        SACL    RESULL          * RESULTADO PARTE BAJA
FINAL   IDLE
        END
```

Una rutina usando el mismo principio se hizo para las rutinas de reconocimiento, usando el microprocesador µ7720.

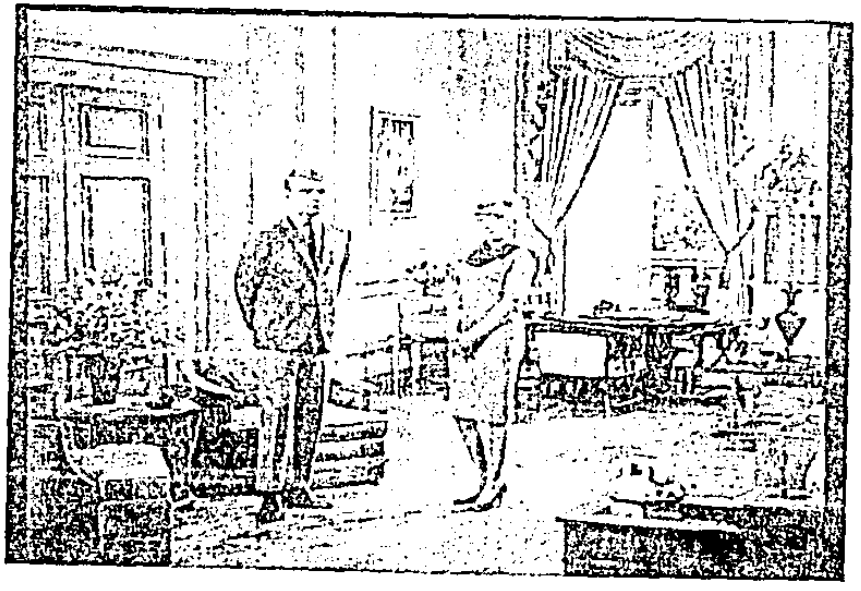
5.- Concluciones.

En este capitulo se presentan las conclusiones y resultados de cada uno de los experimentos vistos anteriormente.

Para el caso de reconocimiento de patrones se obtuvierón resultados que nos permitierón evaluar el desempeño de los algoritmos de reconocimiento y algunas de las fallas que se cometierón en la implantación de estos. Por ejemplo se encontro que para hallar el principio de una palabra usando solamente umbrales de energia, en ocaciones resultaba que no se detectaba el inicio en algunas palabras que empezaban con consonantes con bajo contenido energetico, como por ejemplo, la consonante "s". Para solucionar este problema se podria pensar en un esquema donde se combinara tanto umbrales de energia, como cruces por cero. Se tuvierón algunos problemas cuando se implanto el algoritmo de reconocimiento, debido principalmente a la forma del sistema Siemens: El tamaño de la memoria externa era muy reducido y sólamente nos permitia poder guardar cuatro patrones de reconocimiento ; la comunicación entre este sistema y la PC se hacia a través de un puerto serial que hacia muy lenta la comunicación; sólamente uno de los microprocesadores µPD7720 tenia acceso a las memoria externas, haciendo que el otro microprocesador solamente trabajara con sus datos internos. Los resultados que obtuvimos fueron muy satisfactorios y nos motivan a seguir con esta linea de investigación. Por ejemplo, en algunas palabras se tuvo una probabilidad de exito de hasta un 90 porciento. Se pueden hacer muchas mejoras en nuestro sistema, tanto en Hardware como en Software. Por ejemplo en el Hardware se podria pensar en un sistema el cual contara con dos o mas microprocesadores, de la segunda o tercera generación, los cuales podrian estar conectados en paralelo y compartiendo memorias de

datos. Asi mientras que uno estuviera calculando correlaciones el otro estaria calculando las medidas de distorción de los patrones con las correlaciones calculadas previamente. Asi usando este esquema se podria tener un sistema de reconocimiento capaz de reconocer voz continua, en donde los patrones ya no serian palabras aislados, si no que serian los fonemas que forman las palabras. Usando estos fonemas y un metodo con cadenas de Markov se podrian reconstruir las palabras.

Se analizan a continuación los resultados obtenidos en los experimentos de la transformada discreta de Fourier usando el microprocesador TMS32020. Usando el algoritmo de la FFT el tiempo de ejecución fue de 9.92 ms para 256 datos; es decir que para una imagen de 256x256 se tuvo un tiempo de ejecución de 5.079 segundos. Utilizando el algoritmo de Goertzel se tuvo un tiempo de 339 ms, y para el mismo tamaño de imagen que el anterior algoritmo 173.56 segundo. Para el algoritmo de subbandas se tuvo un tiempo de 20.46 ms para 256 datos y para la imagen 10.4704. Hay que hacer notar que estos tiempos se ven incrementados considerablemente debido principalmente a que la traida de una imagen de disco y el despliegue de la imagen en la computadora personal es un poco lenta. Asi por ejemplo el tiempo de ejecución real, usando el equipo descrito en el capitulo 4.2, llevo un tiempo aproximado de 1.45 segundos para una imagen de 256x256. Se muestran a continuación dos fotografias con sus respectivas transformadas discretas de Fourier con su magnitud centrada.



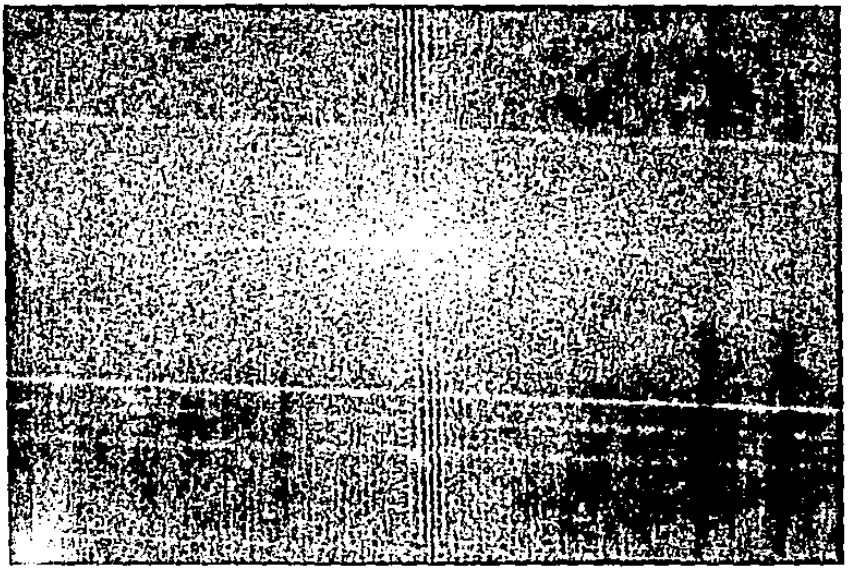


Fig. 5. 1.- Imagen de 256x256 puntos con sus respectiva magnitud en la frecuencia contenida.

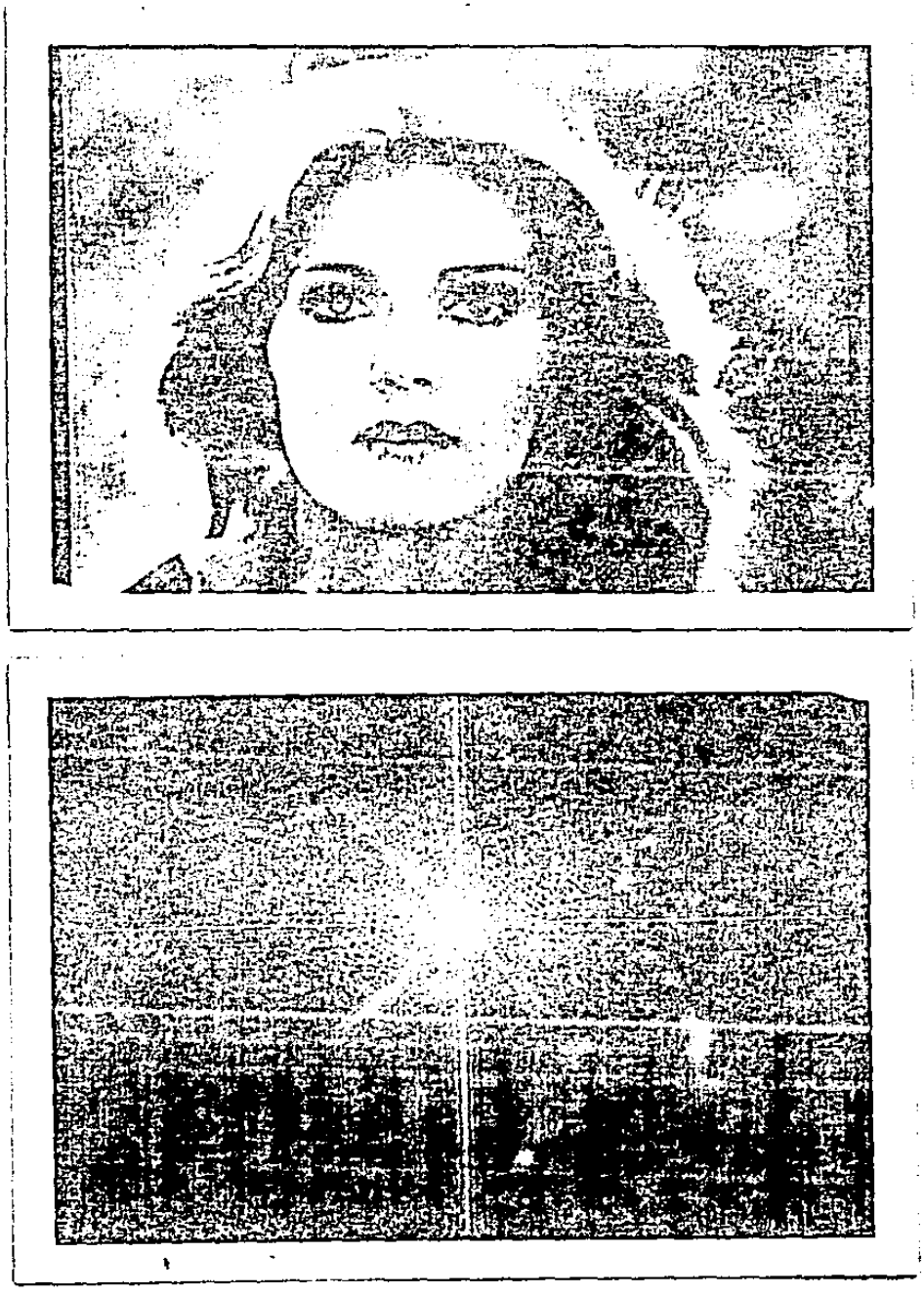


Fig. 5. 2. – Imagen de 256x256 puntos con su respectiva magnitud en la frecuencia centrada.

En los experimentos de filtrado adaptable se obtuvieron los siguientes resultados: cuando los valores iniciales del filtro de adaptación estaban cercanos a los valores del filtro desconocido ,los coeficientes del filtro adaptable convergian a los coeficientes del filtro desconocido en una forma mas rápida. Además, su repuesta en frecuencia en la fase y magnitud era la misma.Cuando los coeficientes del filtro de adaptación eran iguales a cero en las condiciones iniciales, estos no convergian en ocaciones a los valores de los coeficientes del filtro desconocido, pero la magnitud en el dominio de la frecuencia si correspondia a la magnitud del filtro desconocido, no ocurriendo lo mismo que con la fase. A continuación se muestran las graficas de la magnitud y de la fase tanto del filtro desconocido como de los filtros adaptables. En estas graficas se observa en el dominio de la frecuencia como el filtro con las condiciones iniciales diferentes decero,se adapto para tener una respuesta en frecuencia parecida al filtro desconocido.

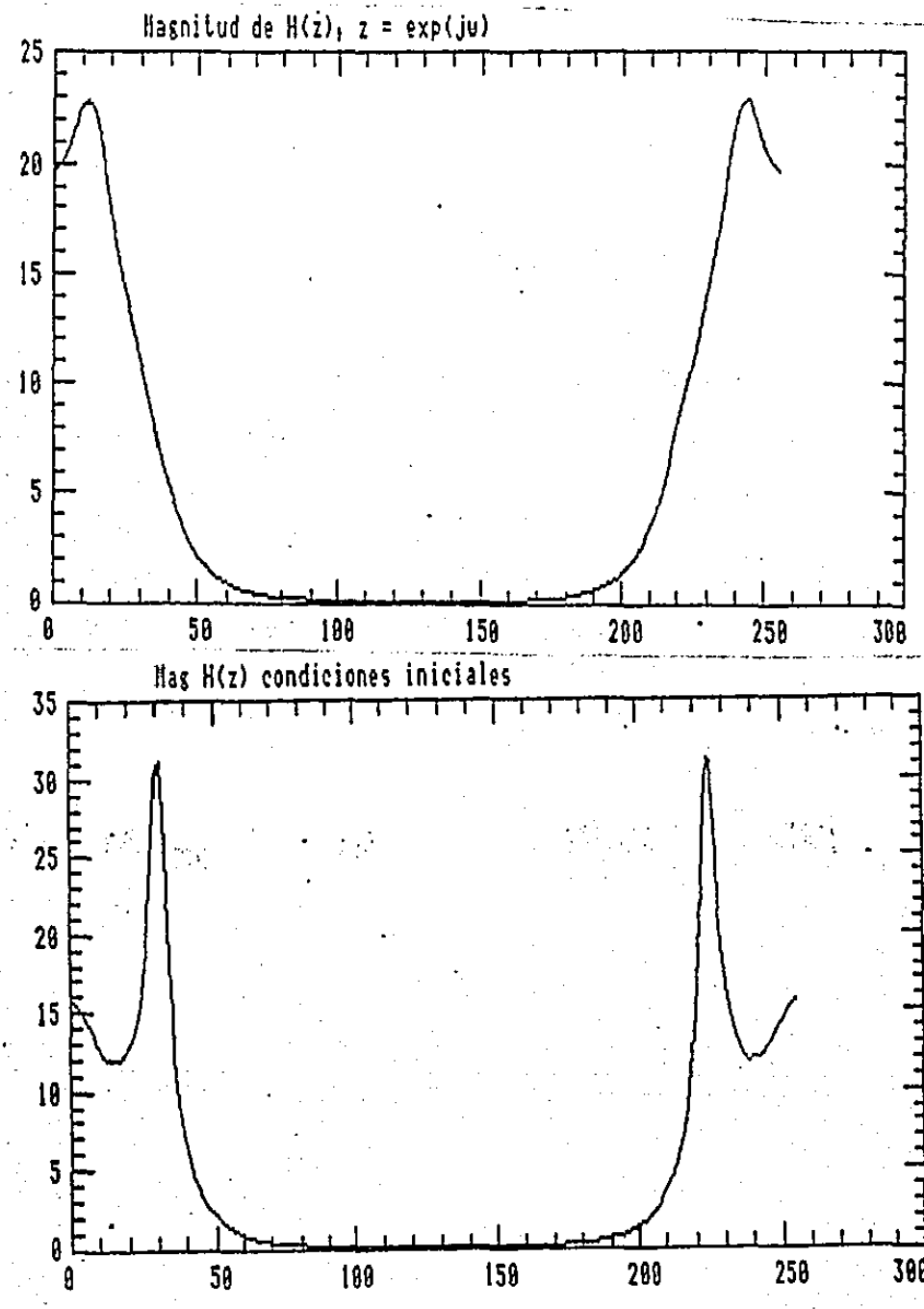


Figura 5.3 .- Magnitud del filtro desconocido y del filtro con los valores de los coeficientes cercanos al filtro desconocido.

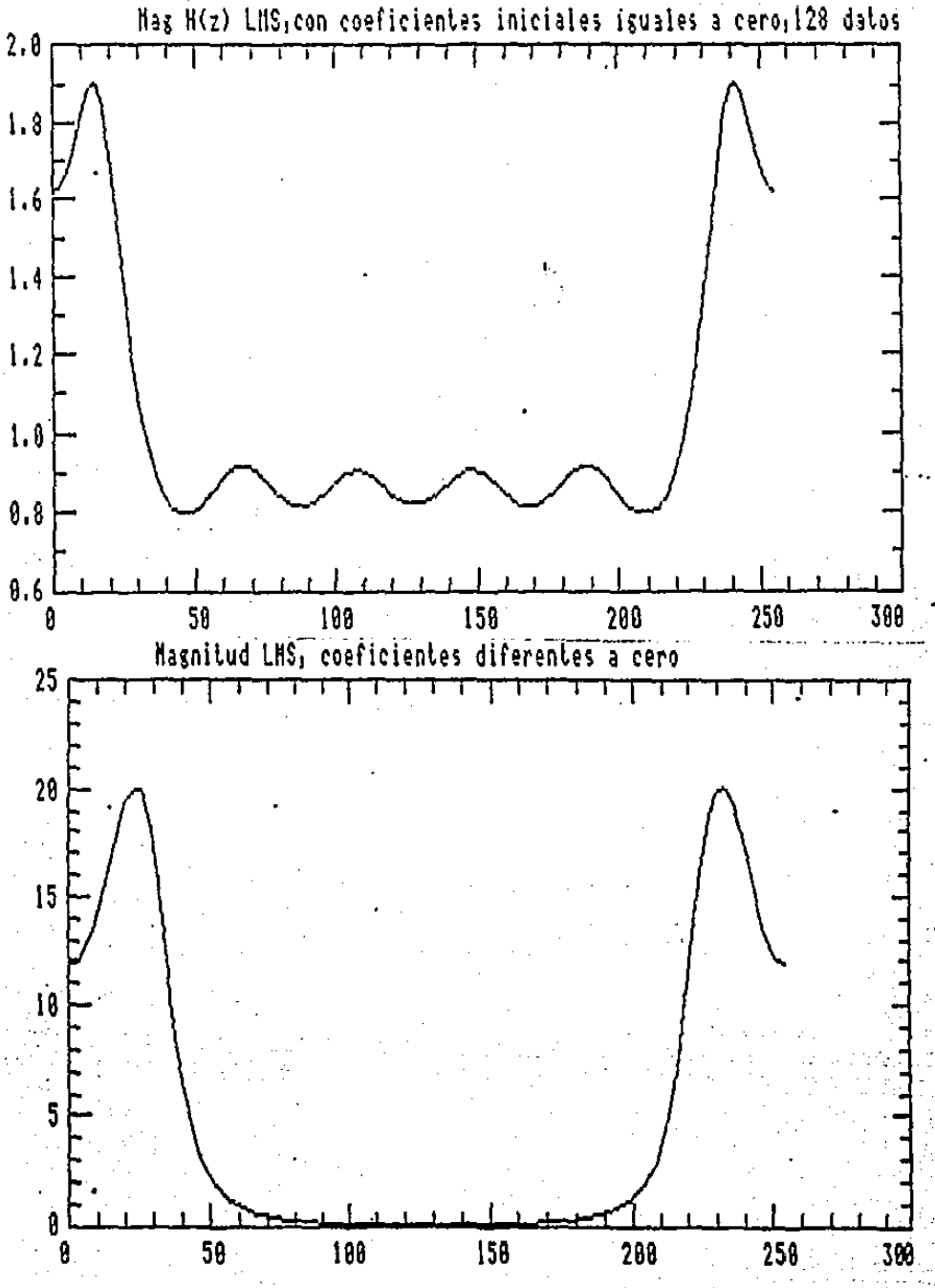


Fig.5.4.- Magnitud de los filtros adaptables con condiciones iguales y diferentes que cero.

En conclusión se puede decir que se cumplió con el objetivo que se había planteado al principio de este trabajo, el cual era instrumentar los algoritmos mas característricos de procesamiento digital de señales en microprocesadores comerciales de procesamiento digital de señales.

Es importante hacer notar, que aunque existen ciertas diferencias entre los microprocesadores DSP de la primera generación y la segunda generación no hay que dejarse llevar por esta sociedad consumista que su unico fin es el mercantil y desechar microprocesadores, suspuestamente obsoletos, por otros que sólamente tiene ligeras variaciones o que están de moda.

Bibliografía

1.- Haykin, Adaptive Filter Theory, Prentice Hall 1986.

2.- Szego, Orthonormal Polynomials, 4th edition Providence,RI: American Mathemathical Society, 1975.

3.- R.M.Gray, A.Buzo, Distortion Measures for Speech Processing, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-28, No. 4, August 1980.

4.- S.B.Davis, Computer Evaluation of Laryngeal Pathology Based on Inverse Filtering of Speech, Speech Communication Research Laboratory, Inc. Santa Barbara, California, 1976.

5.- A.V.Oppenheim, editor, Applications of Digital Signal Processing. Prentice Hall, Englewood Cliffs, N.J. 1978.

6.- A.K.Jain, Fundamentals of Digital Image Processing, Prentice Hall and System Sciences Series 1989.

7.- A.Buzo, A.H.Gray, Speech Coding Based Upon Vector Quantization, IEEE Trans. Acoust. Speech and Signal Processing, Vol.Assp-28, pag.562-574, October 1980.

8.- D.K.Burton, J.K.Shore, and J.T.Buck, Isolated Word Speech Recognition Using Multi-Section Vector Quantitation Code-Books, Computer Sciences and Systems Branch Information Technology Division, Naval Research Laboratory, Washington D.C, July 1984.

9.-A.Gersho, R.M.Gray, Signal Coding: Quantitation and Compression, with Emphasis on Vector Quantitation, University of

California, Santa Barbara and Standford University, October 1988.

10.- E.O.Brigham, The Fast Fourier Transform, Prentice Hall, 1974.

11.- R.C.Gonzalez, Digital Images Processing, Prentice Hall,1984.

12.- Whalen. Detection of Signals in Noise. Academic Press. Inc 1971.

13.- J.Hayes, Diseño de Sistemas Digitales y Microprocesadores Mc Graw Hill, 1986.

14.- Digital Signal Aplications with the TMS320 Family Texas Instruments, 1986.

15.- C.Rivera, Reconocimiento de Voz por Computadora, Tesis de Licenciatura Facultad de Ingenieria U.N.A.M. México, Abril.1985.

16.- TMS32020 User's Guide, Texas Instruments, 1986.

17.- FG-100-AT User's Manual Imaging Technology Inc, 1986.

18.- DSP56000 Digital Signal Processor, User's Manual, 1986.

19.- µPD7720 NEC Signal Processing Interface (SPI), 1981.

20.- Treibersystem fúr den Mikroprozessor 8051 eines Signalverarbeitungs-Mikroprozessor Systems, Siemens, Múnchen 1983.