UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA

"CANCELACIÓN ADAPTIVA DEL ECO EN CANALES DE COMUNICACIÓN"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MECÁNICO ELECTRICISTA

ÁREA ELÉCTRICA ELECTRÓNICA

P R E S E N T A:

GERARDO MERCADO BERNAL

DIRECTOR: Dr. BOHUMIL PSENICKA

MEXICO, D.F. **FEBRERO 2005**

Agradecimientos:

Quiero dedicar este trabajo a mi madre, gracias por su apoyo y ayuda, sin el cual nunca lo hubiera logrado.

A mis hermanas, y a todos mis amigos cuyos nombres empiecen de la A a la Z, para no omitir a ninguno. Gracias por ayudarme, estoy profundamente agradecido.

Al Dr. Bohumil Psenicka por su paciencia y ayuda para la realización de este trabajo.

Gerardo Mercado Bernal

México, 2005

Índice general

Capítulo 1

Antecedentes

1.1. Filtrado Digital

El filtrado digital es una de las operaciones mas importantes e útiles del procesamiento digital de señales. Este tipo de operaciones se pueden llevar a cabo mediante filtros analógicos o digitales, siendo los últimos los de mejor operación. Los filtros digitales son más precísos y confiables, ya que tienen un amplio margen para cambiar sus parámetros de diseño. Los filtros digitales operan con señales discretas que son muestreadas en intervalos previamente determinados. Adémas estos filtros pueden modificarse fácilmente para cambiar sus características de diseño, las cuales, pueden ser el ancho de banda, frecuencia o fase.

El filtrado digital es una operación utilizada por un procesador digital de señales (DSP); para ello utiliza un convertidor analógico digital (A/D). el cual muestrea una señal analógica para después obtener una señal digital muestreada. que será reconvertida en una analógica a la salida del convertidor D/A. La conversión digital analógica la podemos describir de manera gráfica:

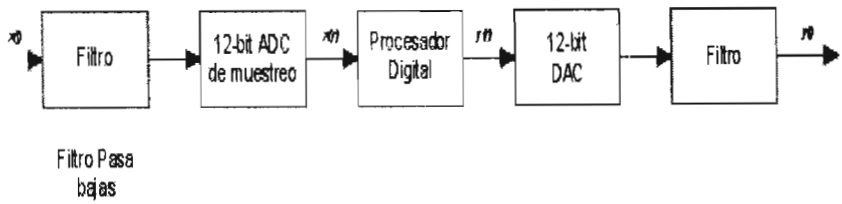


Figura 1.1: Sistema de procesamiento digital de señales en tiempo real.

- La señal primero es muestreada, convirtiéndose en una señal discreta en el tiempo.
- La amplitud de la señal es cuantizada en 2^B niveles. donde B es el número de bits usados para representar la señal en ADC.
- Los niveles de amplitud discreta son representados en palabras binarias, cada una con una logitud de palabra de B bits.

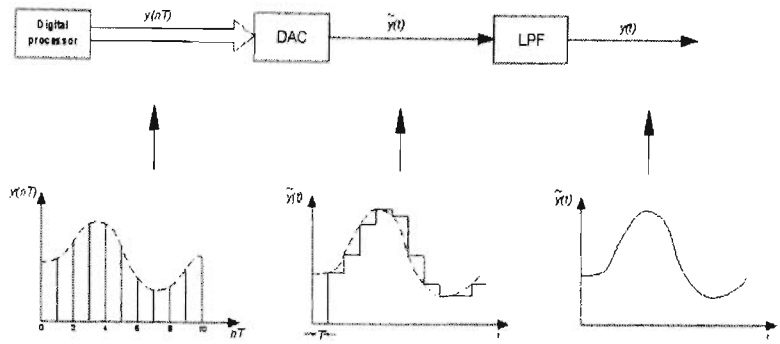


Figura 1.2: Conversión digital analógica

La conversión de estas muestras se hace a través del muestreo de la señal. Las muestras serán tomadas de una secuencia de muestras $f_m = f_0, f_1, f_2, \ldots$ que se derivan de la función $f(t)$ al instante $t = 0, T, 2T \ldots$
El número de muestras dependerá del tiempo T, pues si el número de muestras se hiciera de manera arbitrario encontraríamos fenómenos no deseados.

1.2. El teorema del muestreo.

El muestreo es la adquisición continua de datos de la señal, mediante un muestreo en un intervalo de tiempo definido. Un ejemplo de lo anterior se observa claramente en la figura 1.3. Notese que después del muestreo, la señal análoga es representada solo en el tiempo discreto, con los valores originales de la señal.

Si la frecuencia más alta de una señal es igual a $2fmax$, la señal deberá ser muestreada a una velocidad mayor o igual que $2.fmax$. entoces :

$$f_s > 2fmax \tag{1.1}$$

La ecuación anterior garantiza, que la señal no perderá información y que su reconstrucción es muy cercana a la real.

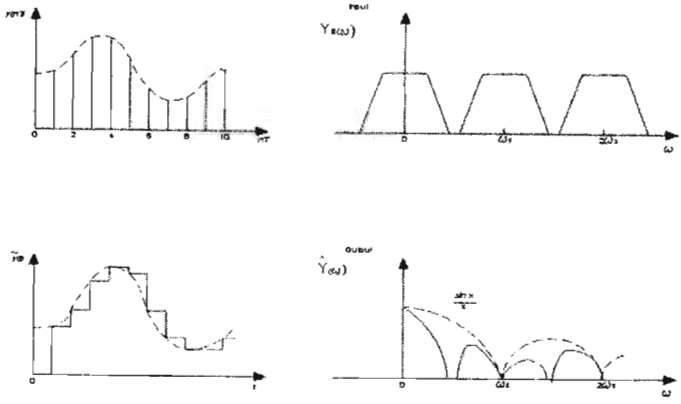


Figura 1.3: Representación en el dominio del tiempo y frecuencia de un sistema DAC, después de ser modificado por un filtro paso bajas.

El diseño de sistemas digitales requiere de filtros digitales, que efectuan las operaciones mencionadas anteriormente y es ahí donde, es importante construir alguno de ellos mediante la siguiente información :

- Aproximación. Para obtener una función de transferencia, en función del tiempo para las especificaciones deseadas.
- Realización. Para conversión de la función de transferencia mediante una tabla de coeficientes y varibles a calcular.
- Implementación. Permite construir mediante software y hardware, haciendo el programa con los coeficientes encontrados. La programación dependerá del tiempo a elegir.

Un filtro digital hace una selección de información sobre una señal digital de entrada. En el diseño de filtros digitales, es necesario encontrar los parámetros de diseño, de tal forma, que cumpla con las especificaciones.
Las especifícaciones pueden ser de banda de paso, frecuencia, de corte y de atenuación.
Con la ayuda de la transformada Z, podemos obtener los coeficientes para el cálculo de los parámetros de un filtro. Por este método resolvemos las funciones asociadas a sistemas discretos.
La clasificación de los sistemas discretos, se pueden dividir en lineales e invariantes en el tiempo, en donde los coeficientes no cambian. De aqui se derivan los filtros de respuesta finita al impulso (FIR) y los filtros de respuesta infinita al impulso (IIR).

1.3. Señales Discretas

Una señal discreta es una secuencia de valores, ésta, sólo se encuentra definida para ciertos puntos del tiempo. A su vez será continua en ciertos intervalos. La señal puede representarse como :

$$\{x(n)\} = \{x(1), x(2), x(3), ...\} \tag{1.2}$$

Donde n es el índice en el tiempo, cada muestra es tomada por separado en un intervalo fijo en el tiempo.
Ejemplos de señales discretas:

Función impulso o δ de Dirac.

$$\delta(t) = \{\infty \ t = 0 \ 0 \ , t \mid 0\} \tag{1.3}$$

Función impulso trasladada en el tiempo t_0.

$$\int_{-\infty}^{\infty} \delta(t - t_0) f(t) dt = f(t_0). \tag{1.4}$$

$$f(t) = \int_{-\infty}^{\infty} f(\delta) f(t - \delta) dt. \tag{1.5}$$

1.4. Sistemas lineales e invariantes en el tiempo.

Los sistemas digitales que trataremos son lineales e invariantes con el tiempo. Dos importantes propiedades asociadas con estos sistemas, son: la superposición y la variación, en el tiempo.

Un sistema lineal cumple con el principio de superposición si :

$$H[a_1 x_1(n) + a_2 x_2(n)] = a_1 H[x_1(n)] + a_2 H[x_2(n)]. \tag{1.6}$$

Un sistema invariante en el tiempo es aquel que cumple con las siguientes condiciones :

$$x(n) \rightarrow^{H} y(n). \tag{1.7}$$

$$x(n - k) \rightarrow^{H} y(n - k). \tag{1.8}$$

Si la señal $x(n)$ en la entrada está retardada en un tiempo k, si la señal en la salida está retardada también de mismo modo que k, el sistema es invariante en el tiempo.

1.5. Convolución

Todo sistema SLITD se puede caracterízar de tres formas:
En una ecuación en diferencias; Respuesta al impulso del sistema y en una función de transferencia.
Entonces por las propiedades anteriores podemos obtener la respuesta al impulso del sistema, que nos dará la respuesta para todo el espectro de frecuencias, por tanto la convolución de una señal se puede representar como la suma de la respuesta al impulso unitario.

$$y(n) = \sum_{i=0}^{\infty} x(n)h(n-i) \tag{1.9}$$

1.6. El sistema causal

Un sistema es causal si la señal en la salida $y(n)$ es dependiente sólo en la señal de entrada $x(n)$ y de las señales retardadas $x(n-1), x(n-2), ..., x(n-k)$; no depende a las señales $x(n+1), x(n+2), ..., x(n+k)$. Un sistema discreto tiene una salida y una entrada, como se ve en la figura 1.4.

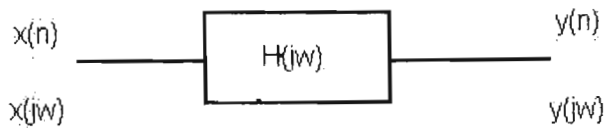


Figura 1.4: Sistema discreto en el tiempo.

- x(n) es la señal de entrada en el dominio del tiempo.
- $X(j\omega)$ es la señal de entrada en el dominio de la frecuencia.
- y(n) es la señal de salida en el dominio del tiempo.
- $Y(j\omega)$ es la señal de salida en el dominio de la frecuencia.
- $H(j\omega)$ es la función de transferencia definida por la siguiente ecuación.

$$H(j\omega) = \frac{\mathrm{Y(j\omega)}}{\mathrm{X(j\omega)}} \tag{1.10}$$

$y(n)$ es la respuesta del sistema y se determina por la convolución de $x(n)$ y $y(n)$. Entonces, en el dominio del tiempo se cálcula la respuesta del sistema, está dada por la ecuación.

$$y(n) = x(n) * h(n) = h(n) * x(n) \tag{1.11}$$

y en el dominio de la frecuencia.

$$Y(j\omega) = X(j\omega) \times H(j\omega) \tag{1.12}$$

La convolución discreta está definida por la ecuación.

$$y(n) = \sum_{k=-(\infty)}^{\infty} x(k)h(n-k) \tag{1.13}$$

pero para calcularlo necesitámos muchas operaciones. Por lo tanto transformamos la señal $x(n)$ a $X(j\omega)$ y calculamos $Y(j\omega)$ como producto de $X(j\omega)$ y $H(j\omega)$. El resultado $Y(j\omega)$ lo transformamos con la transformada inversa para obtener y(n). Para transformar una señal en el dominio del tiempo, al dominio de la frecuencia, es necesario utilizar la transformada Z o la transformada de Fourier. Por la anterior definimos la siguiente ecuación para la transformada Z.

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n} \tag{1.14}$$

donde la relación entre el dominio de s y z queda establecida por la ecuación

$$z = e^{sT} \tag{1.15}$$

donde T es el intervalo entre dos muestras (intervalo de muestreo)

La señal $x(nT)$ que también se escribe x(n), se puede representar de la siguiente forma.

$$x(nT) = x(n) = \{0.1\ 0,2\ 0,2\ 0.1\ 0,3\ 0,4\ 0,1\ 0,0\ 0,0\ ....\} \tag{1.16}$$

Capítulo 2

El Problema del Eco en los Sistemas Telefónicos

2.1. El problema del eco.

El eco en los sistemas teléfonicos es indesable y la mayoría de las veces difícil de evitar. Generalmente, el eco se produce por un desacoplamiento de impedencia en la red teléfonica, el cual genera el eco eléctrico en este tipo de sistemas. El eco acústico se produce entre el altavoz y el micrófono, en los teléfonos de manos libres. El eco es un problema que limita la calidad de la fidelidad en los sistemas de comunicación. El eco acústico se produce en los sistemas de teleconferencia y como ya se mencionó en los sistemas manos libres. En éstos últimos la fuente de sonido, en este caso el humano, está separado, del altavoz y micrófono. Por tanto al estar dentro de un cuarto se generan multiples reflexiones del sonido que también alimentan el micrófono y a su vez, también se genera eco eléctrico. Si se produce un nivel alto de sonido provocará un modo de oscilación , referido como aullido o howling. Esto limitará el volumen máximo de la salida de la bocina.

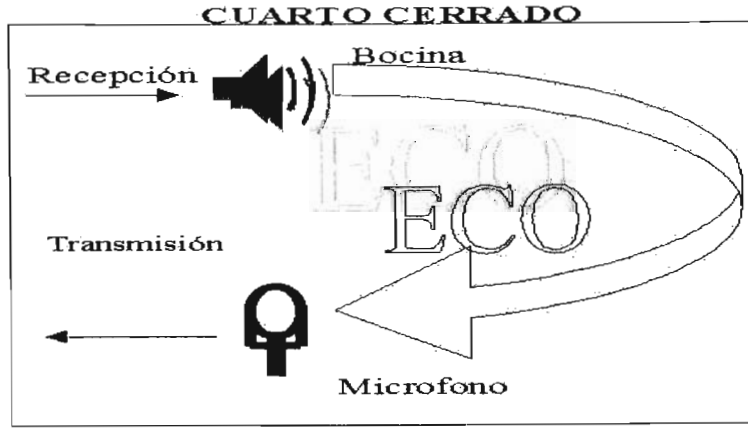


Figura 2.1: Generación de eco acústico.

El eco que tenga un retardo mayor de 40 ms deberá ser tratado con un especial procedimento, este tipo de eco es producido en algunas llamadas de larga distancia e inclusive en llamadas locales y cuando la llamada es por via satélite, es probable que se genere este retraso por la transmisión de la señal que viaja de ida y de vuelta. Produciéndo un retraso de alrededor de 600 ms, el cual excede el umbral. El eco eléctrico se produce debido al desacoplamiento de impedancias existente en las bobinas híbridas, las cuales se emplean para acoplar canales de comunicación de abonado que son de dos hilos. Un canal local del teléfono dirigido hacia la central telefónica, es un canal bidireccional de dos hilos y el canal entre oficinas centrales es de cuatro hilos. El híbrido es el dispositivo que acopla los canales de dos a cuatro hilos y viceversa. Idealmente el híbrido deberá transferir, toda la energía de la derivación, a la llegada del canal de cuatro hilos al canal de dos hilos. Esto no sucede fácilmente entre la salida y la entrada al derivador que se transfiere la salida del mismo, de tal modo que se produce el eco. En los canales de larga distancia debido a que los amplificadores empleados en los tramos son unidireccionales, se requieren de 4 hilos.

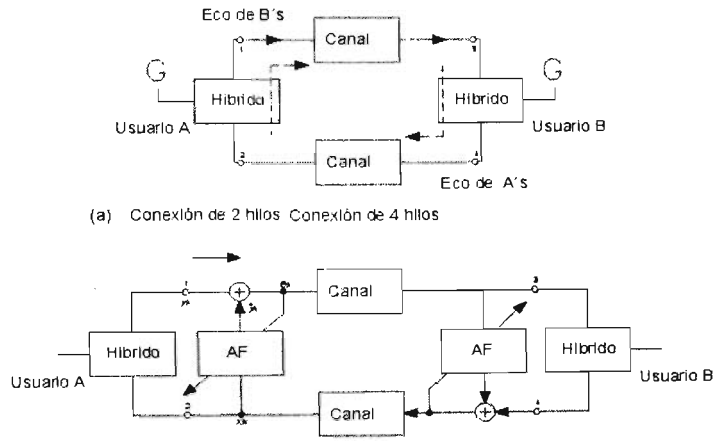


Figura 2.2: Generación del eco en una red telefónica de larga distancia.

Este tipo de problema se intenta resolver mediante la cancelación adaptiva del eco. En donde se emplean algoritmos de cancelación de eco que, por ahora, son los más eficientes propuestos a la fecha.

2.2. Métodos de control del eco

A continuación se expicará la solución que cancela el eco. Haciendo incapie en una explicación sencilla y poniendo el mayor interes en el método de la cancelación del eco.

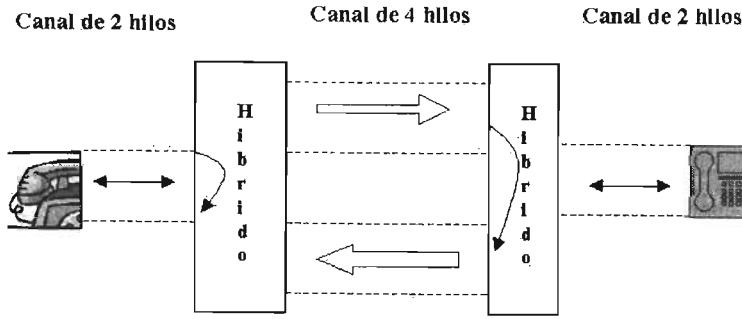


Figura 2.3: Sistema telefónico de 2H a 4H.

2.3. Balanceo de impedancia

Cada trasnformador híbrido está equipado con una red de balanceo que se diseña para minimizar el acoplamiento de impedancia. La cantidad de energía reflejada en el transformador híbrido se mide en términos de pérdida de retorno.La pérdida de retorno puede ser expresada en dB.

La pérdida de retorno, esta también en función de la frecuencia, y se puede usar el valor de la pérdida de retorno del eco (ERL), que es la cantidad promedio de la pérdida de retorno sobre una banda de frecuencia de 500 a 2500 HZ.

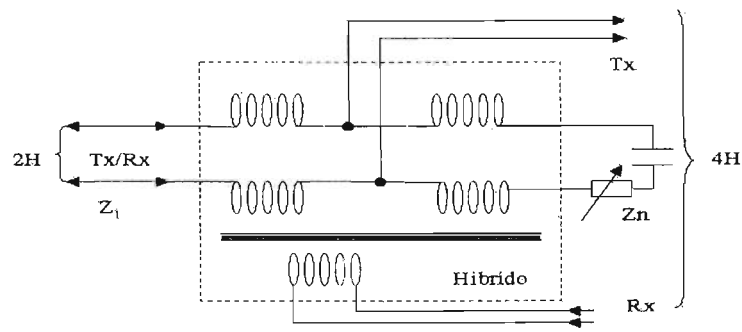


Figura 2.4: Red Híbrida de 2H-4H.

La trayectoria de dos hilos, varía de conexión a conexión ya que se utilizan diferentes tipos de télefonos en la trayectoria de 2H. Debido a este hecho, la red de balanceo del híbrido se le llama, compromiso de balanceo de redes. Se ha trabajado por medio de estructuras de filtro de tiempo continuo, de forma analógica, cancelando el eco en el híbrido, llamándole híbrido adaptable. Con este arreglo se podrá tener una variabilidad limitada de la impedancia de entrada de una línea de transmisión de 2H, por lo cual, el híbrido se conecta para reducir la complejida del cancelador. La entrada primaria al cancelador $d(t)$ es generada por la

aplicación de la entrada de referencia $y(t)$, para divisor el voltaje, a dos filtros simples $H_0(s)$ y $H_1(s)$ contruidos por un divisor de voltaje similar, al usado en sistemas de 2H, en donde las impedancias se calculan mediante aproximaciones en el extremo. La salida de cada uno de estos dos filtros son sumados para cancelar la componente del comunicador lejano en $d(t)$.

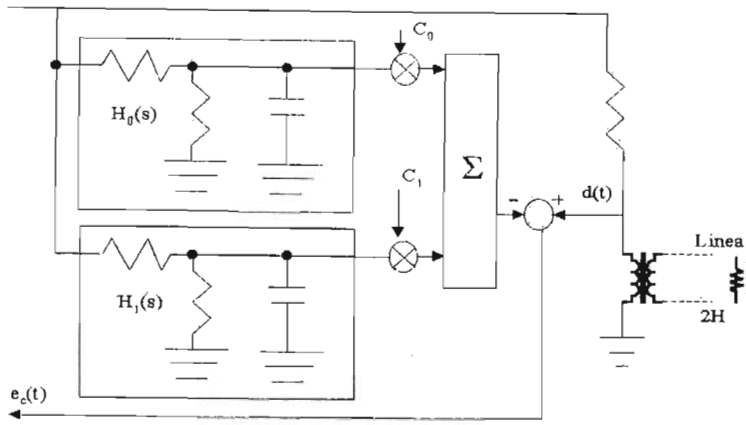


Figura 2.5: Híbrido adaptivo utilizando filtrado adaptable.

Para una adecuada cancelación puede elegirse cualquiera de las aplicaciones de esta estructura con unicamente 2 coeficientes, mientras que un filtro transversal deberá de requerir mucho mas coeficientes. Es por ello, que un filtro transversal puede tener mucho más grados de libertad que los que realmente son necesarios, para una aplicación particular.

2.4. Pérdida de inserción

La introducción de la pérdida dentro de la trayectoria del circuito de 4H, es también un método efectivo para el control del eco. La pérdida de inserción aparece una vez en la trayectoria de la señal primaria y dos veces en la trayectoria del comunicador. Cuando se incrementa un circuito de retardo, la pérdida de inserción también deberá incrementarse. La relación entre retardo, pérdida y calidad de la voz percibida, se analiza de otro modo, para este caso se consideran variaciónes directas.

La pérdida de inserción dentro de la red de voz, es algunas veces referida como un plan de pérdida y no deberá ser implementado, fuera de cuidadosas consideraciones. Un plan ideal deberá conservar una pérdida mínima, para todas las conexiones de voz, "si la pérdida no es conocida, se recomienda que se fije la cantidad a ser insertada en la parte analógica "

2.5. Supresores de eco

Los supresores y canceladores de eco, frecuentemente se emplean para controlar los problemas del eco entre el comunicador y el oyente. La solución común para evitar este efecto es colocar un supresor de eco, como el que se muestra en la figura 2.6, el cual altera dinámicamente

la ganancia en la trayectoria de transmisión y recepción. La unidad de control (detector activado por voz) vigila continuamente la señal en la que la ganacia del otro amplificador es reducida. Así el supresor de eco, inherente , permite únicamente un camino de comunicación (half-duplex) usualmente favoreciendo el nivel de la señal de la bocina.

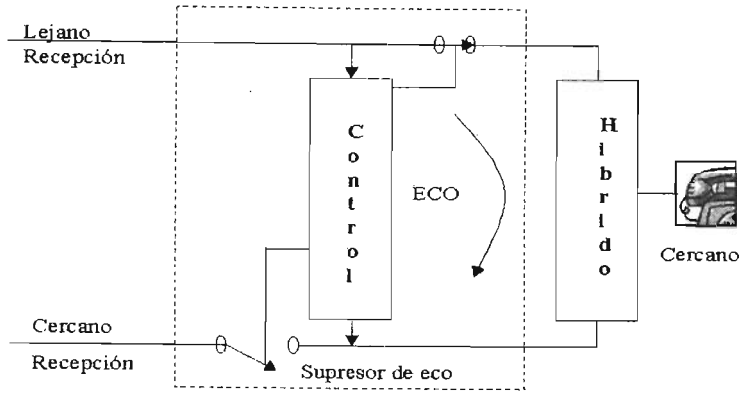


Figura 2.6: Supresor de eco.

El supresor de eco es simple de realizar, pero está solución deteriora la voz, debido a las interrupciones. Por ejemplo, si el comunicador cercano está inicialmente escuchando al comunicador lejano, pero éste interfiere de un momento a otro la comunicación, es probable que el interruptor impida el paso de su voz hacia la transmisión, no cerrándose lo suficientemente rápido, y el comunicador lejano no reciba los mensajes emitidos.
La conmutación, la supresión y el modo de Double -Talk puede dar lugar a una señal entrecortada. Además, en un ambiente ruidoso, la supresión de la señal lejana "Far-End"puede provocar un fenómeno repentino, entonces el ruido del medio lejano será suprimido cuando el comunicador local esté activado.
El supresor del eco se implementa con una función de detección propia de un tono de 2100 Hz. Seguido por la energía de la voz continua ó bajo de un control manual. La señal del tono de deshabilitación deberá ser 2100 Hz ± 21 Hz (2079 Hz a 2121 Hz), en un nivel de -12 dBm ± 6 dBm 0, como se especifica en CCITT V.21. El supresor deberá ser deshabilitado cuando el inhabilitador de tono es detectado en niveles entre 0 y -31 dBm= para 300 ± 100 ms. El supresor de eco será deshabilitado con el inhabilitador de tono en no menos de 100 ms.

2.6. Canceladores de eco

Un cancelador de eco primero intenta estimar la respuesta al impulso de la trayectoria del eco y entonces genera una réplica de éste. El eco estimado se obtiene de la señal recibida, como se muestra en la figura 2.7 de un cancelador de eco. Para ello se emplea un filtro adaptable obteniendo una réplica del eco desconociendo su trayectoria. El cancelador deberá estimar la respuesta al impulso de la trayectoria del eco y adaptarla rápidamente a estas variaciones. La

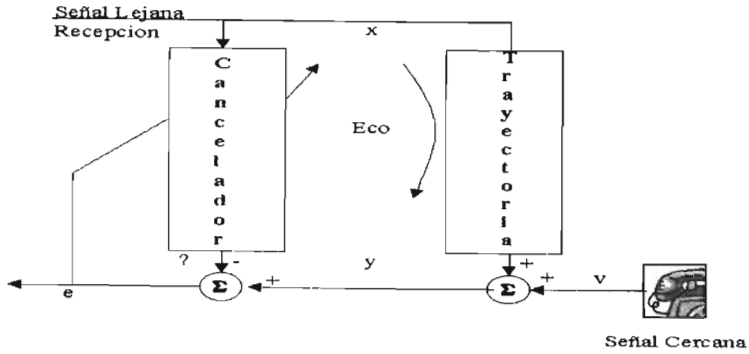


Figura 2.7: Cancelador de eco.

recomendación de la ITU-T especifica los requisitos de los dispositivos, para la cancelación del eco eléctrico y estos son :

- La pérdida de retorno de eco en una sola dirección (single talk) y los modos de la doble dirección (double talk)
- El tiempo de convergencia inicial.
- El tiempo de recuperación después de la variación de la trayectoria de eco.

El desempeño del cancelador de eco, depende de la elección del algoritmo adaptable del filtrado. Analizamos el desempeño de varios algoritmos de filtrado adaptable, ilustrando su funcionamiento, mediante la respuesta al impulso de un canal de eco típico mostrado en la figura 2.9.

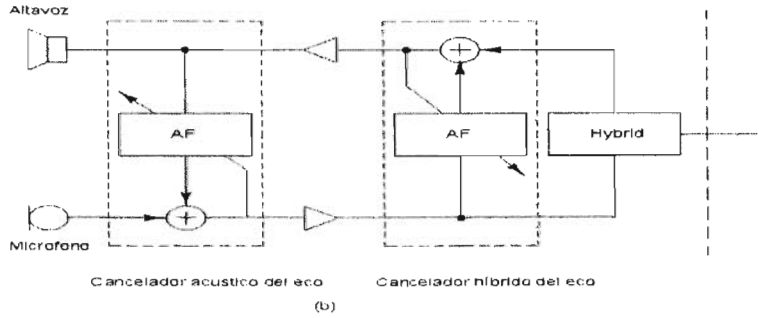


Figura 2.8: Sistema de cancelación de eco acústico e híbrido

Para ello usamos dos señales de entrada; ruido blanco, y una señal de voz, y voz. Debido a la dificultad de definir una señal de prueba de voz, los requisitos de desempeño se especifican, en las recomendaciones ITU-T para una señal de ruido blanco. Sin embargo, el desempeño

apropiado con señal de voz es una parte de los requisitos. Empleamos una señal de voz de prueba como la mostrada en la figura 2.10
Usamos la pérdida de retorno de eco como el índice de desempeño de los algoritmos. La pérdida de retorno de eco (ERL), se define, como la proporción de la energía en el eco residual $e(n)$, con respecto a la energía usando 16 puntos en movimiento promedio de las amplitudes cuadradas instantáneas. La proporción de la convergencia del algoritmo, puede estudiarse usando una gráfica que muestre ERL contra el número de la muestras. Estos gráficos normalmente se denominan las curvas de aprendizaje del algoritmo.

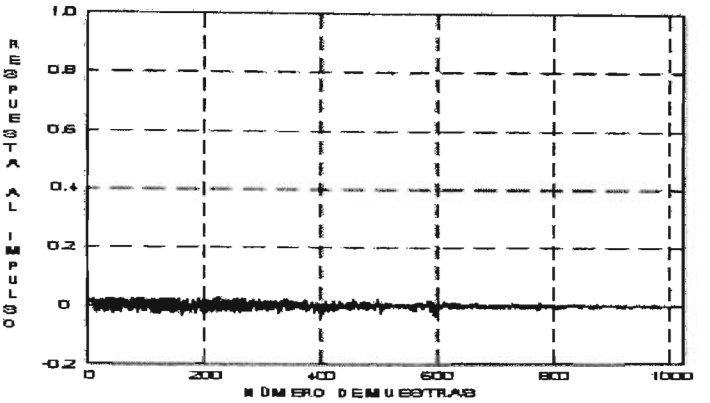


Figura 2.9: Respuesta al impulso de la trayectoria del eco para las simulaciones.

Un cancelador de eco ejecuta un mejor trabajo, en comparación con un supresor de eco, tanto digitalmente, como analógicamente. Los canceladores se instalan siempre en el extremo lejano del circuito, para cancelar el eco localmente. Cuando el cancelador del extremo lejano detecta la voz de llegada en el comunicador local, se produce un fenómeno de convergencia. Durante la convergencia, el cancelador Far-End tomará una muestra de la señal de llegada al comunicador local, almacenándola y entonces, la señal se cancela en la salida del circuito local. La capacidad de almacenaje dependerá del retardo de la trayectoria final, (longitud de la extremidad del eco), la ITU-T G.165 recomienda algún requerimento del retardo de la trayectoria final, sí la cantidad de almacenaje es pequeña, no puede elegirse para toda trayectoria de eco, si la cantidad de almacenaje es muy alta, las localidades sin uso pueden resultar en un ruido excesivo. Si la respuesta al impulso del modelo de la trayectoria canceladora de eco es mucho más grande que la respuesta de la trayectoria actual, el retorno puede ser más alto que el resultado del eco mismo, por lo tanto, el cancelador observa un retorno similar de la señal de llegada de la energía de reflejo (eco). El retorno de energía entrante deberá ser menor de 6 dB abajo de la energía de salida por ITU-T G.165 (ERL+TLP).

La máxima cantidad de tiempo requerida para converger es específicada, dentro de la G.165 es de alrededor de 500 ms. Durante este tiempo, el comunicador local puede escuchar el eco, la cantidad de información pérdida, durante la selección de este proceso es indepediente del lenguaje característico de ambos comunicadores.

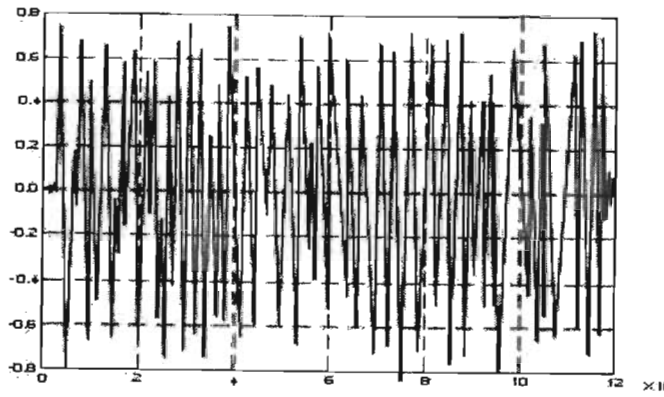


Figura 2.10: Señal de prueba de voz para simulaciones.

Una vez que la convergencia ha sido alcanzada, el cancelador inicia sustrayendo la señal de salida de la señal entrante. Este proceso de sustracción continúa, hasta que el comunicador termina de hablar, cuando el comunicador local reanuda la comunicación, el cancelador de eco reconverge nuevamente. en las condiciones double talk la calidad percibida en la cancelación del eco no cambia significativamente, aunque el cancelador tiende a diverger.

En el proceso de la cancelación del eco se detecta un eco residual, este aparece como resultado de una distorsión de la señal debido a las variaciones de las ondas de salida y de entrada. La señal de salida, convertida de digital a análógica, (D/A), también como un circuito de escape no lineal, propiedad que pueden presentar en ambos lazos del híbrido en la de 4H y en la de 2H. La cantidad del eco residual, es medida en términos de aumento de la pérdida de retorno del eco (ERLE) de las palabras en inglés, Echo Return Loss Enhancement , siendo la diferencia de medición entre el estado de cancelación activo e inactivo. Para eliminar el eco residual, el cancelador puede emplear opcionalmente una función de procesador lineal. Esto usualmente se refiere como un centro cortador. El centro cortador opera por supresión de eco residual usando atenuación. Durante la doble dirección double-talk, el centro cortador es removido del circuito, sin embargo, esta funcion la hace verdaderamente el cancelador superior y el supresor de eco, así también en este proceso, existe un cancelador de eco verdadero, y no limitado a una supresón. La recomendación de G.165 de la ITU-T, también especifíca el uso del circuito deshabilitador de tono. Este es especificado como 2100 Hz, como frecuencia característica que se especifica en la G.164. El tono de 2100 Hz se sujeta a una fase de inversión de 180 grados. como se recomienda en la V.25 de la ITU-T. El circuito deshabilitador de tono deberá reconocer la señal deshabilitadora en niveles de 6 dB y de 31 dBm, adicionalmente, la máxima cantidad de tiempo requerido para deshabilitar el cancelador (tiempo de operación) es de un segundo.

Capítulo 3

Algoritmos Adaptivos

3.1. Sistemas Adaptivos

Un sistema de control adaptivo es un sistema que verifica continuamente y automáticamente las características dinámicas, estas se ajustan a las condiciones deseadas de la planta, mediante la variación de parámetros, hasta lograr un funcionamiento estable. Generalmente los sistemas adaptivos son complejos, ya que ofrecen la posibilidad de adecuarse a las variables de las señales de entrada, que son normalmente desconocidas y variantes en el tiempo. El control adaptivo debe establecer un índice de comportamiento, el cual, debe de ser medible y selectivo hasta lograr una operación óptima.

3.2. Propiedades de Generales de los Sistemas Adaptivos

Al desarrollar un sistema adaptivo, se deben considerar todas las condiciones de entrada posibles al sistema o al menos, estadísticamente y definir bien, lo que se desea que el sistema realice, tomando en cuenta que la variables cambian bajo la operación de funcionamiento en cada una de estas condiciones. La característica de funcionamiento de un sistema adaptivo depende de sus señales de entrada, si una señal de entrada X, se aplica al sistema, entonces un sistema adaptivo se adecuará a X y causará una salida Y. Si otra señal X2 se aplica al sistema, el sistema se adaptará a X2 y producirá una segunda salida. La respuesta de los sistemas adaptivos será diferente para dos entradas diferentes. El principio de superposición no funciona para los sistemas adaptivos, en el caso de los sistemas lineales, es decir, si se aplica al sistema la suma de las dos entradas X1 + X2, el sistema se adaptará a esta nueva señal y producirá una salida diferente a Y1 + Y2.
Los sistemas adaptables son ajustables y su ajuste depende generalmente de las características promedio de tiempo finito de la señal, al desarrollar un sistema adaptable se deben considerar todas las entradas posibles al sistema, definir que se quiere que el sistema realice, para que estadísticamente se logre efectuar, es obvio que no se puede conocer todas las condiciones del sistema, pues, estas cambian con el tiempo. Los ajustes internos del sistema de los sistemas adaptivos se realizan con el fin de optimizar las medidas especificadas de funcionamiento.
En los procesos adaptivos, se emplean diferentes estructuras, tales como sistemas de malla abierta y malla cerrada, predictor adaptivo, modelado adaptivo inverso, cancelador de interferencia. Aqui solo mencionaremos dos de ellos, sistema adaptivo de malla abierta y cerrada. La

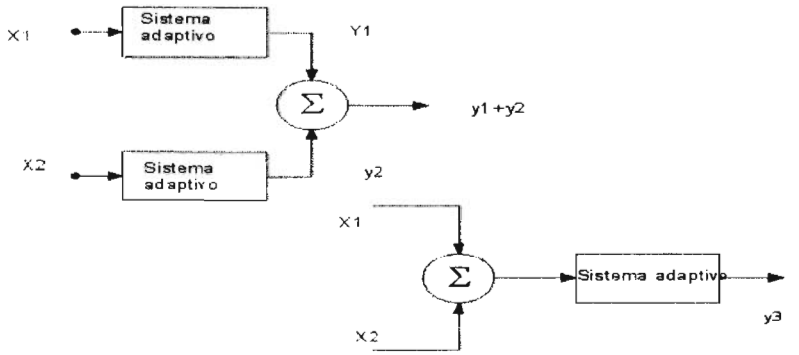


Figura 3.1: Sistema adaptivo con diferentes señales.

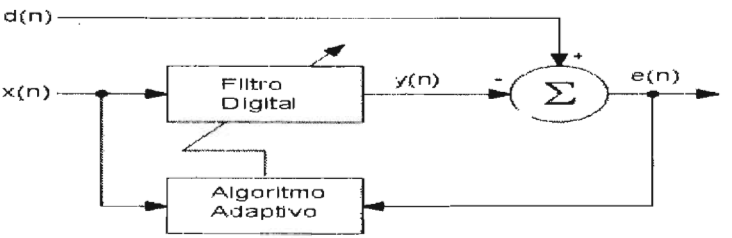


Figura 3.2: Diagrama de bloques de un filtro adaptivo

adaptación de malla abierta implica hacer mediciones de las características de entrada ó del medio ambiente y aplicar esta información a una fórmula o a un algoritmo computacional y usar los resultados para establecer los ajustes del sistema adaptivo. La adaptación de malla esencialmente es la misma que la función de transferencia de la planta y se dice que la planta ha sido identificada. La estructura y los valores del sistema adaptivo pueden o no ser semejantes a los del sistema desconocido, pero sus funciones de transferencia serán las mismas.

3.3. Algoritmos Adaptables

En este capítulo, se analizan los algoritmos de Mínimos Cuadrados Promedio (LMS), LMS Normalizado (NLMS) y se evalúa su actuación como canceladores de eco en sistemas de telecomunicaciones. Los algoritmos son comparados sobre la base de su velocidad de convergencia, la condición estable de la pérdida del eco de retorno (ERL), y la complejidad de la implementación. Mientras LMS es simple, su velocidad de convergencia es dependiente de la dispersión de los valores propios de la matriz de autocorrelación de la señal de entrada. En particular, converge lentamente con entrada de voz, este problema disminuye con el NLMS,

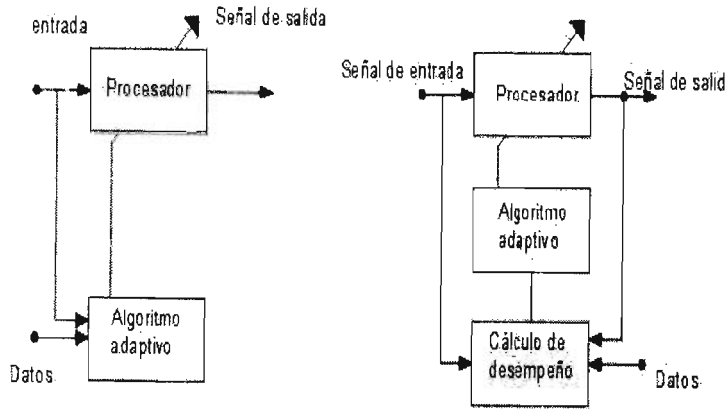


Figura 3.3: Sistema adaptivo de malla abierta y malla cerrada.

la complejidad de NLMS es comparable a la del algoritmo LMS. En la actualidad los algoritmos del tipo NLMS, se han propuesto con un número importante de variantes que emplean factores de convergencia variables con el tiempo, como una forma de satisfacer de manera simultánea los requerimientos de altas velocidades de convergencia inicial y razonablemente bajos errores después de que la convergencia ha sido obtenida.

3.4. Algoritmo de Mínimos Cuadrados promedio (LMS)

El algoritmo LMS es un algoritmo de adaptación muy usado. Debido a su relativa simplicidad al ser programado, este algoritmo, minimiza el valor esperado del error cuadrado (eco residual). Se escoje la estructura transversal para el filtro adaptable, debido a su simplicidad. El algoritmo empieza con algún valor inicial (arbitrario), para el vector de peso de derivación y los pesos son adaptados de acuerdo al algoritmo de descenso de pendiente estocástica, dicho valor mejora con el número de iteraciones. El valor final para el cómputo del vector de pesos, converge en promedio a la Solución de Wiener.
La operación del algoritmo LMS es la misma que la empleada en un sistema de control retroalimentado, en donde, se efectúa un proceso adaptable, mediante el ajuste automático del peso de la derivación y un proceso de filtrado en donde se realiza la formación de un producto interno, resultando un conjunto de la derivación de las entradas y el conjunto correspondiente del peso de la derivación, que surgen del proceso adaptable para producir una estimación de una respuesta descada. El error de la estimación se obtiene al cotejar, esta estimación con el valor real de la respuesta deseada. El error de la estimación se emplea para interactuar con el proceso adaptable, con eso se cierra el lazo de retroalimentación.
El vector u(n) es la entrada de la derivación en el tiempo n, y d(n) es la respuesta descada a la salida del filtro, ésta debe ser estimada, comparando esta estimación $\hat{d(n)}$ con la respuesta verdadera deseada d(n), con ello se genera el error de estimación e(n). Así, podemos escribir:

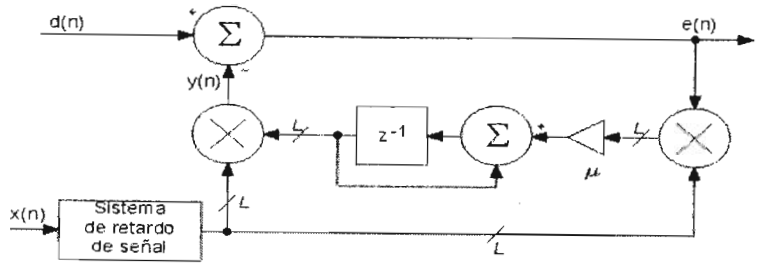


Figura 3.4: Diagrama de bloques de un filtro con algoritmo LMS.

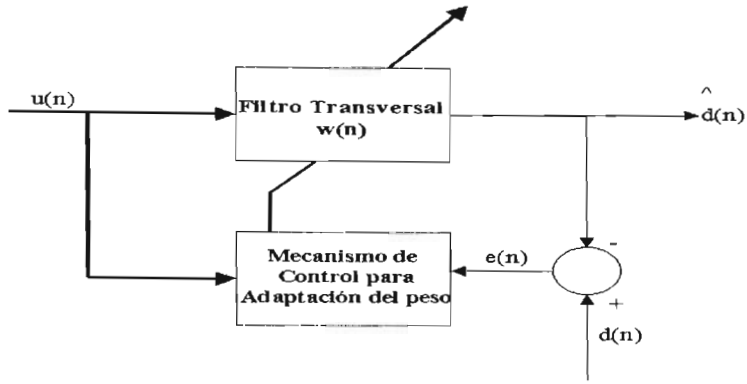


Figura 3.5: Diagrama de bloques de un filtro transversal adaptable.

$$e(n) = d(n) - \hat{d(n)} = d(n) - w^T(n)u(n) \tag{3.1}$$

donde el término $w^T(n)u(n)$ es el producto interno del vector de peso de derivación w^T y la derivación del vector de entrada u(n). La forma extendida del vector del peso-derivación se describe por:

$$w(n) = [w_0(n)w_1(n)....w_{M-1}(n)]^T \tag{3.2}$$

y el vector de entrada de derivación por:

$$u(n) = [u(n)u(n-1)....u(n-M+1)]^T \tag{3.3}$$

El criterio de la función empleado en el algoritmo de LMS es el valor esperado del error cuadrático y puede escribirse como:

$$J(n) = E[e^2(n)] \tag{3.4}$$

Asumiendo que el vector de entrada de derivación u(n) y la respuesta deseada d(n) son colectivamente estacionarias, el error cuadrático medio J(n) es una función convexa del vector de peso de derivación w^T con un mínimo único. La teoría del filtro de Wiener puede usarse para calcular el vector de peso de derivación óptimo. El proceso adaptable intenta localizar el punto óptimo usando el método de descenso de pendiente. Según este método, el valor actualizado del vector de peso de la derivación en el tiempo n + 1 se calcula empleando una simple relación recursiva.

$$w(n+1) = w(n) + \frac{1}{2}\mu[-\delta(J(n))] \tag{3.5}$$

donde $\delta J(n)$ denota el valor del gradiente del vector en tiempo n, y μ es un valor positivo real constante. El factor 1/2 es usado meramente por conveniencia, el gradiente del vector $\delta J(n)$ donde :

$$\delta(J(n)) = -2p - Rw(n) \tag{3.6}$$

donde p es el vector de correlación cruzada entre el vector de entrada de la derivación u(n) la respuesta deseada d(n) y R es la matriz de correlación del vector de entrada de la derivación u(n). Los valores de p y R se desconocen, estos son estimados por los datos de entrada. Lo más recomendable es usar estimaciones inmediatas como se muestra a continuación.

$$\hat{R}(n) = u(n)U^T(u) \tag{3.7}$$

$$\hat{p}(n) = u(n)d(n) \tag{3.8}$$

En la ecuación anterior se ha suprimido el operador de expectación.Sustituyendo (3.7) y (3.8) en el método de descenso de pendiente, como se definió por (3.9), se consigue la relación recursiva con la actualización del vector de peso de derivación como sigue:

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n)[d(n) - U^T(n)\hat{w}(n)] \tag{3.9}$$

La tilde o sombrero encima del símbolo para el vector de peso de derivación lo distingue como el valor obtenido, usando el método de descenso de pendiente, lo cual se puede escribir, en la forma de tres relaciones básicas como sigue:

1. El rendimiento del filtro:

$$y(n) = \hat{w}^T(n)u(n) = \hat{d}(n) \tag{3.10}$$

2. El error de estimación:

$$e(n) = d(n) - y(n) \tag{3.11}$$

3. La adaptación del peso de derivación:

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n)e(n) \tag{3.12}$$

Las tres relaciones anteriores representan, el algoritmo de mínimos cuadrados promedio LMS.

3.4.1. Conducta de la convergencia en el LMS

Debido a la presencia de retroalimentación en el algoritmo de LMS allí existe una posibilidad de tendencia a la inestabilidad. La estabilidad del algoritmo depende del parámetro del tamaño de paso de μ. Para que el algoritmo de LMS converja en el medio, esto es, que el valor medio esperado del vector de peso de derivación $\hat{w}(n)$ se acerque a la solución óptima (Wiener) w_0 como el número de iteraciones n que se aproxime al infinito, μ debe satisfacer :

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{3.13}$$

donde λ_{max} es el valor propio ó eigenvalor más grande de la matriz de correlación R. Usando resultados del álgebra de matrices, se tiene que $TR_r > \lambda_{max}$, entonces, se puede definir la condición de estabilidad como:

$$0 < \mu < \frac{1}{\mathrm{u(n)}^2} \tag{3.14}$$

Otra forma de convergencia que nos interesa, es la convergencia del cuadrado medio. Esto significa, que el final medio (condición - estable) del valor del error cuadrado medio es finito $J(\infty)$. Esto ocurre sí, y sólo sí:

$$\sum_{i=1}^{M} \frac{\mu\lambda_1}{2 - \mu\lambda_1} < 1 \tag{3.15}$$

donde λ_1, i = 1, 2....., M, son los valores propios (eigenvalores) de la matriz de la correlación, R y M es el número de derivaciones. El vector del peso de la derivación $\hat{w(n)}$ no converge exactamente al punto mínimo de $J(n)$. En cambio, el algoritmo ejecuta un movimiento aleatorio alrededor del punto mínimo, debido a la presencia de ruido en el gradiente. Esto produce un error de condición estable que siempre es $J(\infty)$ mayor que el mínimo error cuadrado medio J_{min} que corresponde a la solución de Wiener. La diferencia entre el valor final $J(\infty)$, el valor mínimo que el J_{min}, es llamado el exceso del error cuadrado medio $J_{ex}(\infty)$. La relación de $J_{ex}(\infty)$ a J_{min} es llamado desajuste M, que es una medida de la desviación de la solución, calculada por el Algoritmo LMS de la solución de Wiener. El desajuste M se da mediante la siguiente ecuación:

$$M \approx \frac{\mu \mathrm{M} \lambda_{\mathrm{av}}}{2} \tag{3.16}$$

donde λ_{av} es el valor propio promediado de la matriz de correlación subyacente R de las entradas de derivación.

$$\lambda_{av} = \frac{1}{\mathrm{M}} \sum_{i=1}^{M} \lambda_i \tag{3.17}$$

Suponga que el conjunto promedio de la curva de aprendizaje del algoritmo de LMS se aproxima a una simple exponencial con la constante tiempo $\tau_{mse,av}$. La constante de tiempo promedio para el algoritmo de LMS se caracteriza por:

$$\tau_{mse,av} \approx \frac{1}{2\mu\lambda_{\mathrm{av}}} \tag{3.18}$$

Las ecuaciones (3.6) y (3.17) muestran que el desajuste M es directamente proporcional al parámetro de tamaño del paso μ, considerando que la constante de tiempo promedio es inversamente proporcional a $\tau_{mse,av}$. Por consiguiente si μ es pequeña se reducirá el desajuste M, el tiempo de asentamiento se incrementa y viceversa.

3.5. Conveniencia de la cancelación de eco mediante el algoritmo LMS.

Debido a su simplicidad, el algoritmo de LMS es fácil programarlo en la mayoría de Procesadores de Señales Digitales (DSP), toma aproximadamente tres ciclos de instrucción para su ejecución. El algoritmo se conoce por ser robusto, contra la aplicación más allá (numérico) de errores introducidos por la longitud del registro finito. Estas propiedades hacen al algoritmo LMS atractivo para las aplicaciones de cancelación de eco.
Sin embargo, la velocidad de convergencia del error cuadrado medio depende en lo disperso de los valores propios de R. Cuando la dispersión de los valores propios es grande, la velocidad de convergencia se reduce. Para el ruido blanco, los valores propios son iguales y el algoritmo de LMS tiene una velocidad de convergencia razonablemente alta. En la mayoría de las aplicaciones de cancelación de eco, la voz es la señal de entrada, la señal de voz tiene una gran dispersión en los valores propios y esto retarda la convergencia. La curva de aprendizaje del LMS se muestra en la figura 3.6 esta muestra que la convergencia del algoritmo LMS es mucho más lenta con la entrada de voz, que con ruido blanco como señal de entrada. Aquí la pérdida de retorno de eco se calcula usando el movimiento promedio de 16 puntos. Notese que la curva de aprendizaje depende del número de puntos usados en promedio en la figura 3.7 se muestra la curva de aprendizaje de la señal usando un movimiento promedio de 128 puntos para calcular ERL. Con un promedio mayor el estado de ERL es estable, sabemos que el ERL instantáneo real es el mismo en ambos casos. A continuación, la pérdida de retorno de eco se calcula usando el movimiento promedio de 16 puntos.
La dependencia indeseable de la razón de convergencia del LMS en la dispersión de los valores propios ha motivado la investigación de otros algoritmos de filtrado adaptable, con señales características independientes de la convergencia independientes de la señal de entrada para la cancelación de eco.

3.6. Algoritmo Normalizado de Mínimos Cuadrados Promedio NLMS

El algoritmo Normalizado de Mínimos Cuadrados Promedio es una versión modificada del algoritmo LMS. En el algoritmo de LMS, el factor de Corrección para el vector de peso de derivación w(n) es calculado como μ(n)e(n). Entonces esta cantidad es directamente proporcional al vector de entrada de la derivación u(n)por el error de estimación. Así el error de el gradiente estimado se magnífica para una valor mayor de u(n), siendo esto un factor potencial de inestabilidad. Este problema puede evitarse normalizando el factor de corrección por la norma del cuadrado Euclidiano del vector de entrada de derivación u(n). Esta variante del algoritmo LMS, con el factor de corrección normalizado, se llama el algoritmo LMS Normalizado (NLMS).

3.7. Descripción del algoritmo

La actualización producida por el algoritmo de NLMS puede interpretarse como la solución a la optimización el problema como se muestras en las siguiente ecuación :

$$\frac{\min}{\mathrm{w(n)}}\{d(n) - w^H(n)u(n)]^2 \sum_{i=1}^{M} \lambda_i\} \tag{3.19}$$

Así, para $\mu \in [0, 1]$, la nueva estimación del vector de peso de derivación producida por el algoritmo NLMS, es un compromiso entre el acceso a la nueva señal deseada y la desviación de la estimación anterior. Los pasos siguientes definen el algoritmo de NLMS:

1. Filtro de salida:

$$y(n) = \hat{w}^H(n)u(n) \tag{3.20}$$

2. El error estimado:

$$e(n) = d(n) - y(n) \tag{3.21}$$

3. La adaptación del peso de derivación.

$$\hat{w}(n+1) = \hat{w}(n) + \frac{\mu \mathrm{u(n)e(n)}}{[\mathrm{u(n)}]^2} \tag{3.22}$$

donde : $\alpha = \mu \;\; y \;\; 0 < \alpha < 1$
La división por $[u(n)]^2$ en la ecuación (3.22) puede conllevar a problemas numéricos, cuando la entrada de la señal es pequeña. Este problema se puede resolver agregando un valor positivo pequeño de α para la norma $[u(n)]^2$. De, la ecuación (3.22) de adaptación del peso de derivación la que se modifica como sigue. Usaremos una a= 0.001 en nuestras simulaciones e implementación.

3.8. Conducta de la convergencia del algoritmo de NLMS

Bajo ciertas premisas simplificando la condición necesaria y suficiente para la convergencia del algoritmo de NLMS es :

$$\mu \in (0, 2) \tag{3.23}$$

Al contrario de la condición de convergencia para el algoritmo de LMS, dada en la (3.13), la condición anterior es independiente de las características de la señal. La convergencia más rápida ocurre cuando

$$\mu = 1 \tag{3.24}$$

El algoritmo de LMS no garantiza el tamaño del paso constante óptimo. Un tamaño del paso, óptimo para una cierta clase de señales, no puede ser el óptimo para otras, o puede igualar el resultado en la divergencia del algoritmo. Esto obliga a un valor conservador del tamaño

del paso, resultando constante en una convergencia más lenta. El algoritmo de NLMS puede ser interpretado como el algoritmo de LMS con un tamaño del paso variante en tiempo. Equivalentemente, el algoritmo de NLMS escoge el tamaño del paso óptimo que depende de la señal de entrada.
La tasa de convergencia de los algoritmos NLMS y de LMS es comparable si se emplea ruido blanco u(n), sin embargo para señales coloreadas como la voz, el algoritmo de NLMS converge más rápidamente que el algoritmo de LMS. La curva de aprendizaje de los algoritmos LMS y de NLMS se muestra en Figura 3.7, para una señal de voz como entrada. Vemos que el algoritmo de NLMS tiene una velocidad de convergencia mayor y una condición estable de ERL mejor que el algoritmo de LMS.

3.9. Conveniencia para la cancelación de eco

La complejidad del algoritmo de NLMS es comparable con el algoritmo LMS. La convergencia es más rápida en el algoritmo de NLMS para señales de voz como entrada, haciéndolo más conveniente que el algoritmo de LMS para la cancelación de eco.

Aunque el algoritmo NLMS es muy simple, el potencial computacional de DSPs actuales se limita al orden máximo dado por los filtros adaptables NLMS de unos centenares. Mientras para canceladores de eco en la red telefónica de larga distancia el orden es suficiente, aunque es insuficiente para canceladores de eco acústicos que operan en mucho mayor tiempo, en cuartos con constantes de tiempo de reverberación de unos cientos de milisegundos. El camino de eco acústico normalmente más prolongado en tiempo; va de varios centenares de derivaciones para los teléfonos de manos-libres, de unos miles para sistemas modernos del teleconferencias. Esto ha incitado una investigación de algoritmos computacionalmente más eficaces que el algoritmo de NLMS.

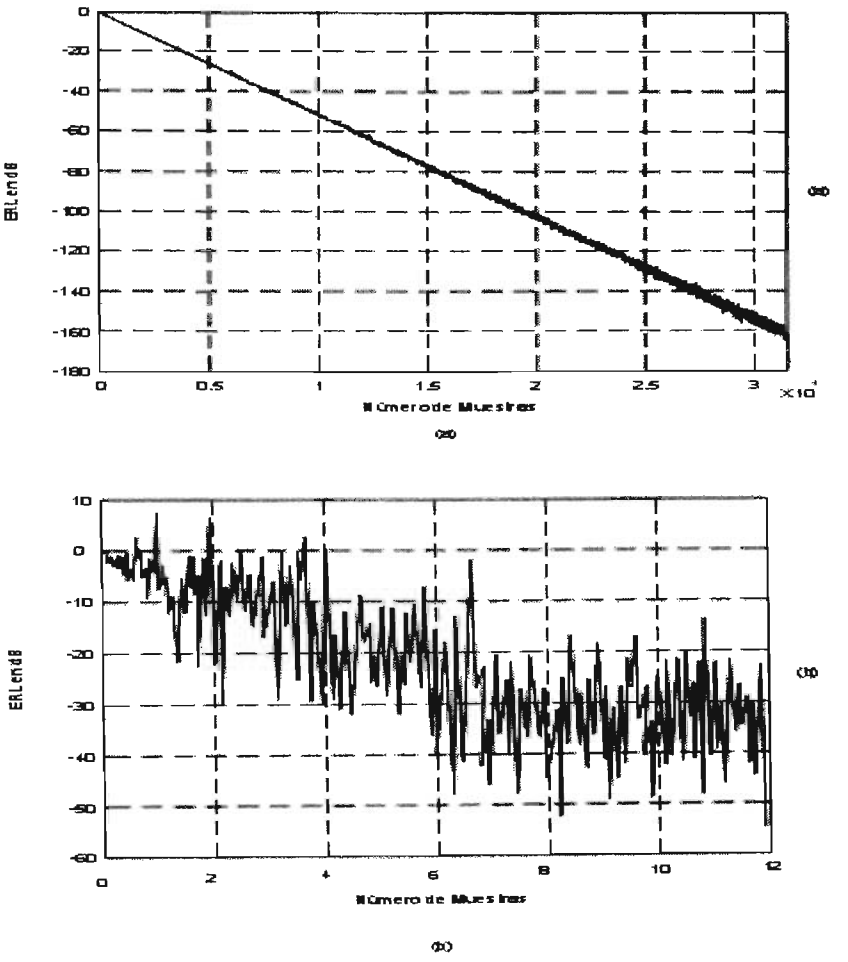


Figura 3.6: Curvas de aprendizaje de algoritmo LMS con entrada de ruido blanco usando señales de SNR 30

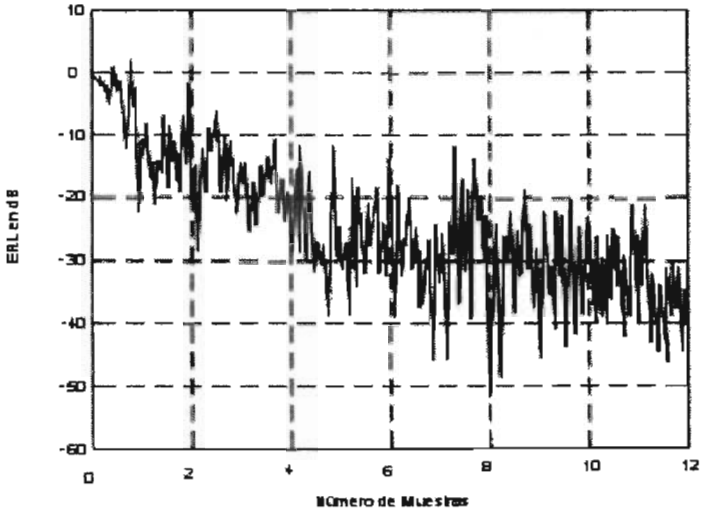


Figura 3.7: Curva de aprendizaje de algoritmo LMS. Usando 128 puntos en movimiento promedio.

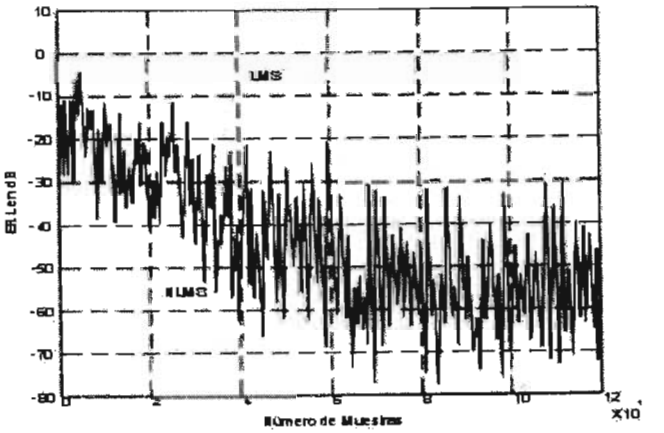


Figura 3.8: Curvas de Aprendizaje de los Algoritmws LMS (u 0.08)

Capítulo 4

Fundamentos de control activo

4.1. Estructuras para la cancelación de eco.

Los sistemas adaptables se pueden dividir en dos grupos principales :

- los filtros adaptables lineales.
- Las redes neuronales artificiales que son sistemas no lineales.

Los filtros adaptables son de gran utilidad en sistemas para la cancelación de eco, estos sistemas son eficientes y operan satisfactoriamente a las variaciones de tiempo, con señales de entrada de forma estadística, por lo tanto es un filtro ideal para el procesamiento digital de señales y para aplicaciones de control. Este tipo de filtros son usados en varios campos de la ingeniería, como : en comunicaciones, geología, en radares e ingeniería biomédica.

La diferencia esencial entre varias aplicaciones del filtrado con filtros adaptables, esta en la forma de la respuesta obtenida. Las aplicaciones de este tipo de filtro varían de acuerdo a su configuración como se muestra en la figura(4.1). A continuación se describen cuatro aplicaciones básicas del filtrado adaptivo

1. Identificación. El comportamiento de un filtro adaptable para el reconocimiento de una señal, es usado para proporcionar un modelo lineal que respresenta la mejora del acceso de datos a un sistema desconocido. El sistema y el filtro adaptable son excitados por la misma entrada. La fuente de salida dará la respuesta deseada por el filtro adaptable, si el sistema es de naturaleza dinámica, el modelo será variante con el tiempo.

2. Modelación inversa. En esta aplicación, la función del filtro adaptable es proporcionar un modelo inverso que represente el acceso de la señal de ruido para el sistema. Para un sistema lineal el modelo inverso tiene una función de transferencia igual al recíproco de la función de transferencia del sistema tal que la combinación de ambos, forman un medio ideal para la transmisión de señales. Una función retardada a la entrada del sistema constituye la respuesta deseada por el filtro adaptable. En algunas aplicaciones, la entrada es usada sin retardo como respuesta deseada.

3. Predicción. En este sistema el filtro adaptable hace por un pronóstico del valor presentado de una señal aleatoria. El valor presente de la señal sirve al propósito de la respuesta deseada para el filtro adaptable, así también el valor anterior de la señal

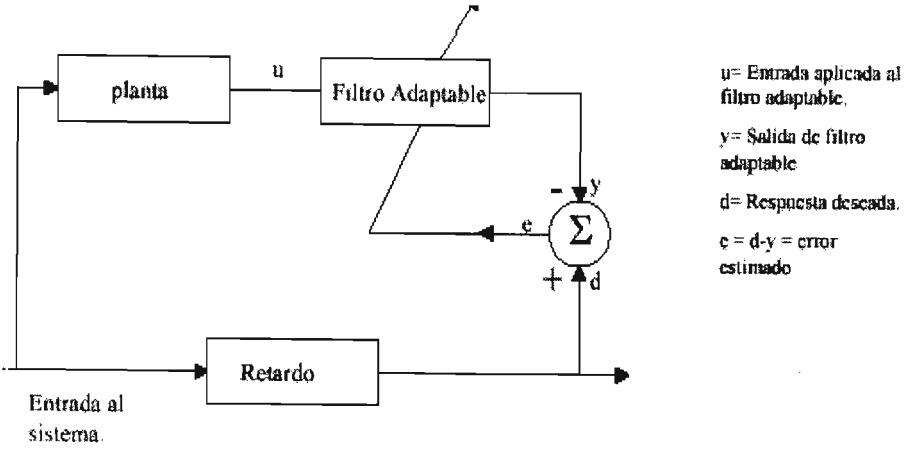


Figura 4.1: Diagrama de un filtro adaptivo aplicado a la identificación

suministrada a la entrada del filtro adaptable. La salida del filtro adaptable en este caso la estimación de error puede servir como la salida del sistema, que funcionará como un predictor de error y como un filtro de predicción de error.

4. Cancelación de interferencia. El filtro adaptable es usado para cancelar una interferencia desconocida contenida a lo largo de una componente de la señal de información en una señal primaria, con la cancelación optimizada. En algún sentido, la señal primaria sirve como respuesta deseada para el filtro adaptable. Una señal de referencia (auxiliar) es empleada como la entrada al filtro adaptable. La señal de referencia es derivada de un sensor o de un conjunto de senores localizados en relación con el sensor ó sensores que suministran la señal primaria, de tal forma que la presencia de la componente de la señal de información deteriorada, sea débil o esencialmente indetectable.

4.2. Filtros con la respuesta finita al impulso FIR

La función de transferencia de los filtros digitales con la respuesta finita al impulso tiene la forma siguiente:

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_k z^{-k} \tag{4.1}$$

$$H(z) = \sum_{i=0}^{k} b_i z^{-i} \tag{4.2}$$

Si se utiliza la transformada z inversa a la ecuación anterior se obtiene la ecuación de diferencias

$$y(n) = \sum_{i=0}^{k} b_i x(n-i) \tag{4.3}$$

De esta función se puede dibujar la estructura directa del filtro FIR y se muestra en la figura (4.2). Esta estructura se llama la estructura transversal. En la figura (4.3) es la estructura del filtro FIR en primera forma canónica.

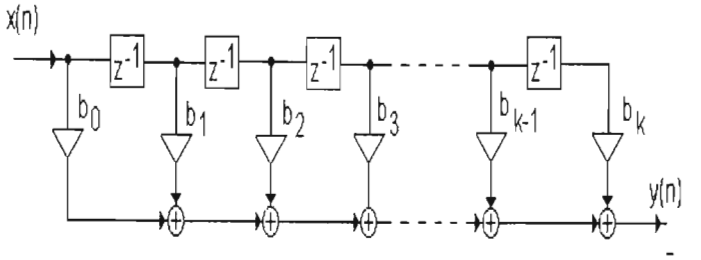


Figura 4.2: Estructura directa del filtro FIR

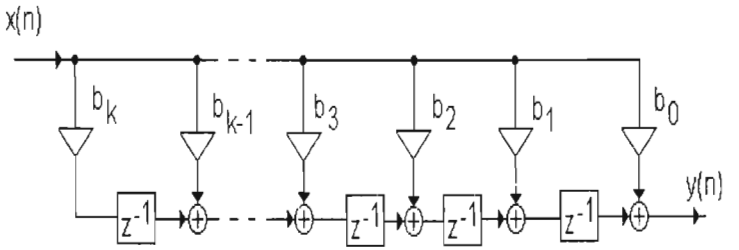


Figura 4.3: Forma canónica del filtro FIR

De la función de transferencias, ecuación (4.3) se ve, que los coeficientes de la función de transferencia son simultáneamente muestras de la respuesta al impulso unitario.

Las ventajas del filtro FIR

- Los filtros con la respuesta finita son siempre estables.
- La fase de la función de transferencia es lineal en todo dominio de las frecuencias.
- Las muestras de la salida son dependientes sólo a las muestras en la entrada y a las muestras de entrada retrasadas. Por eso no son tan sensibles a errores de redondeo como los filtros con la respuesta infinita.

Desventajas del filtro FIR

- Para cumplir las especificaciones de la plantilla se necesitan muchos sumadores, amplificadores y elementos de retardo en comparación con los filtros IIR.
- El retraso de las señales es mayor que en los filtros con la respuesta infinita IIR.

Los ceros de la función de transferencia que no están ubicados en el círculo de radio unitario deben ser recíprocos $1/z_i$. Entonces se puede escribir la función de transferencia en la forma

$$H(z) = H(0)\frac{1}{z^k}(z-1)(z+1)^b \prod_{i=1}^{c}(z - z_{0i}).(z - \frac{1}{z_{0i}}) \tag{4.4}$$

La respuesta al impulso puede ser dividida en cuatro grupos, como se puede ver en la figura (4.4)

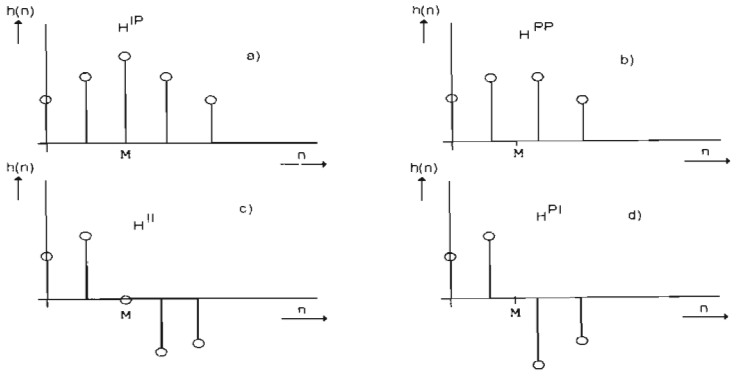


Figura 4.4: Diferentes posibilidades de respuesta al impulso

- En la figura (4.4a) el número de impulsos es impar, y la simetría con respecto al punto M es par, y por eso podemos expresar la función de transferencia en la ecuación (4.5)

$$H(\omega)^{IP} = h\left(\frac{n}{2}\right) + \sum_{i=0}^{\frac{n}{2}-1} 2h(i)\cos(\frac{n}{2} - i)\omega \tag{4.5}$$

- En la figura (4.4b) el número de los impulsos es par y la simetría con respecto al punto M es también par. La función de transferencia se calcula mediante la ecuación (4.6)

$$H(\omega)^{PP} = \sum_{i=0}^{\frac{n-1}{2}} 2h(i)\cos(n - 2i)\frac{\omega}{2} \tag{4.6}$$

- En la figura (4.4c) el número de los impulsos es impar y la simetría con respecto al punto M es también impar. La función de transferencia se puede expresar mediante la ecuación (4.7)

$$H(\omega)^{II} = \sum_{i=0}^{\frac{n-1}{2}} 2h(i) \sin \frac{(n-2i)}{2}\omega \tag{4.7}$$

- En la figura (4.4d) el número de los impulsos es par y la simetría con respecto al punto M es impar. La función de transferencia se puede calcular mediante la ecuación (4.8)

$$H(\omega)^{PI} = \sum_{i=0}^{\frac{n}{2}-1} 2h(i) \sin(\frac{n}{2} - i)\omega \tag{4.8}$$

El filtro transversal de respuesta finita, también se puede expresar mediante tres elementos básicos, los cuales son, elemento de retardo unitario, multiplicador y sumador. El número de elementos de retardo utilizados en un filtro determina la duración de la respuesta al impulso, esto se muestra como M-1, esto nos indica el orden del filtro. En la figura (4.5) los elementos de retardo son identificados por operadores de retardo unitario Z^{-1}. Cuando (Z^{-1}) opera en la entrada $x(n)$, el resultado de la salida es $x(n-1)$, que a su vez es multiplicador a la entrada de la derivación, a la cual es conectada, para un coeficiente del filtro referido como peso de derivación. Así al conectar el multiplicador $W_k(n)$ a la entrada de derivación $x(n-k)$ produce una versión escalar del producto anterior, $w_k x(n-k)$, donde w_k es el peso de derivación respectivo y $K = 0.1....M-1$. La suma de las salidas multiplicadas individualmente produce la salida del filtro :

$$y(n) = \sum_{K=0}^{m-1} w_k x(n-k) \tag{4.9}$$

La anterior ecuación es la suma de convolución entre la repuesta al impulso de una duración finita del filtro, w_k, y la entrada del filtro $x(n)$ para producir la salida $y(n)$.
En resumen, un filtro FIR es regularmente estable ya que su función de transferencia contiene q polos en el origen. Un filtro FIR no requiere del conocimiento de valores pasados de la salida por tanto son relativamente faciles de diseñar, garantizando la linealidad de fase. Estos filtros tienen algunas desventajas, una de ellas es que requieren de gran cantidad de coeficientes para una aproximación deseada.

4.3. Diseño del Filtro FIR

La característica en las figuras 4.7a o en la figura 4.8 se aproxima mediante la serie de Fourier

$$|H| = A(f) = \sum_{m=-M}^{M} c_m e^{j2\pi f m T} \tag{4.10}$$

donde los constantes c_m y c_{-m} se pueden calcular mediante la ecuación (4.11)

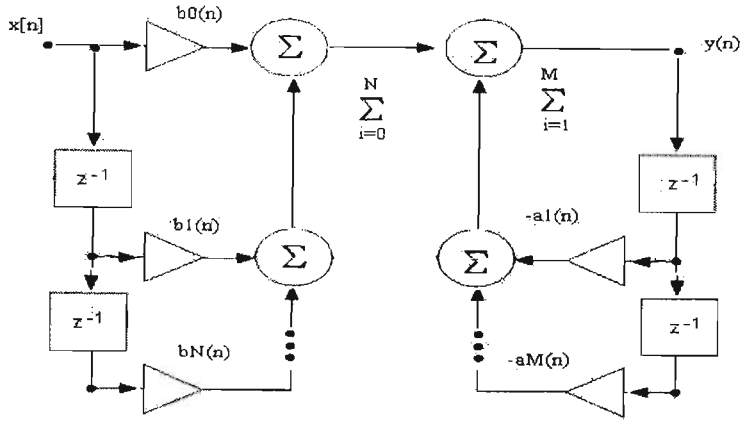


Figura 4.5: Estructura de un filtro IIR, de la forma Directa I

$$c_m = c_{-m} = \frac{2}{f_m} \int_0^{\frac{f_m}{2}} A(f) \cos(2\pi f T m) df \tag{4.11}$$

Si se sustituye en la ecuación (4.10) por $e^{j2\pi fT} = z$ se obtiene

$$H_1(z) = \sum_{m=-M}^{M} c_m z^m \tag{4.12}$$

Esta función es la función de transferencia de un sistema no causal. Para $a_i = c_{M-i}$ la última ecuación se puede escribir en la forma

$$H(z) = \frac{1}{z^M} \sum_{i=0}^{2M} a_i z^{-i} \tag{4.13}$$

Esta función es una función de transferencia de un filtro causal FIR. El término $2M$ significa que el filtro FIR tiene $2M$ elementos de retardo. Si la frecuencia de muestreo normalizada es 2, los coeficientes c_m se calculan mediante la ecuación (4.14)

$$c_m = \int_0^1 \cos(m\pi f) df \tag{4.14}$$

4.4. Filtro de respuesta al impulso infinito. IIR

Este filtro en más eficiente con respecto al FIR, debido a la que tiene la capacidad de retroalimentarse pueden utilizar ambos polos y ceros de la función del sistema, mientras que un filtro FIR, solo realiza ceros. Los filtros IIR operan rápidamente debido a que tienen menores operaciones de cálculo. Pero, no cuenta con buena estabilidad y su fase es no lineal, mientras que el FIR es simpre estable y puede tener fase lineal. Aqui se mencionan las diferentes estructuras para la realización de un filtro IIR.

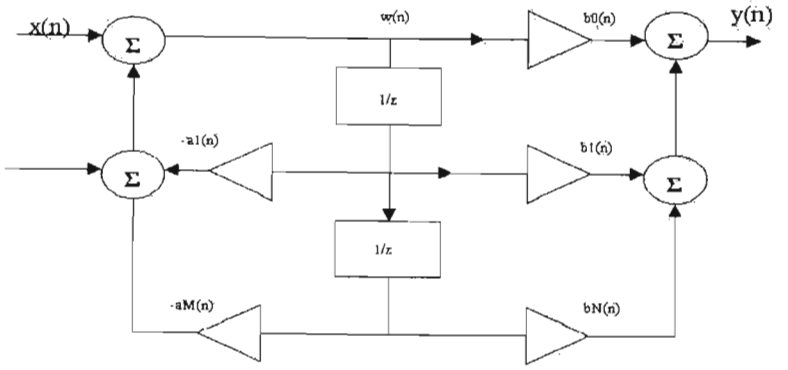


Figura 4.6: Estructura de un filtro IIR, de la forma Directa II

El filtro IIR de forma directa se muestra en la figura, la salida queda representada como :

$$y(n) = \sum_{i=1}^{M} -a_i(n)y(n-i) + \sum_{j=0}^{N} b_j(n)u(n-j) \tag{4.15}$$

En la figura 4.6 se presentan dos bloques en la mitad derecha e izquierda, cambiando el símbolo usado para un bloque de multiplicación constante para un triángulo que indica la dirección del flujo de la señal, por medio de lo anterior se indica la operación en cada diagrama de bloque. La operación de una suma separada se muestra para cada par de número que es sumado, este número es importante debido a que indica el número de sumas realizadas por el programa ha implementar, el cual hará una ejecución en paralelo, o para la ejecución secuencial.

La forma de la función de transferencia que corresponde a la diferencia de la ecuación 4.15 es :

$$H(Z) = \frac{\sum_{i=0}^{l} b_i z^{-1}}{1 + \sum_{i=1}^{k} a_i z^{-1}} \tag{4.16}$$

En función de la ecuación anterior se obtiene la Forma directa II de la figura 4.5
A continuación se describe la estuctura de cascada y paralelo, estos se obtienen de la función de transferencia, este método realiza la interconexión de subsistemas de un orden bajo. Estos subsistemas frecuentemente son menos sensitivos para reducir los parámetros de la señal causada, por una restricción de longitud de palabra finita. El subsistema se utiliza en un sistema de primer orden pero, también es utilizado en sistemas de segundo orden siempre y cuando la función de transferencia, contenga polos y ceros conjugados complejos. La forma general del sistema de primer orden y de segundo orden se muestra en la figura 4.9 y 4.10

Implementación en cascada. Para el método de cascada se requiere descomponer la función de transferencia, obteniendo el factor de la función de transferencia del numerador y de-

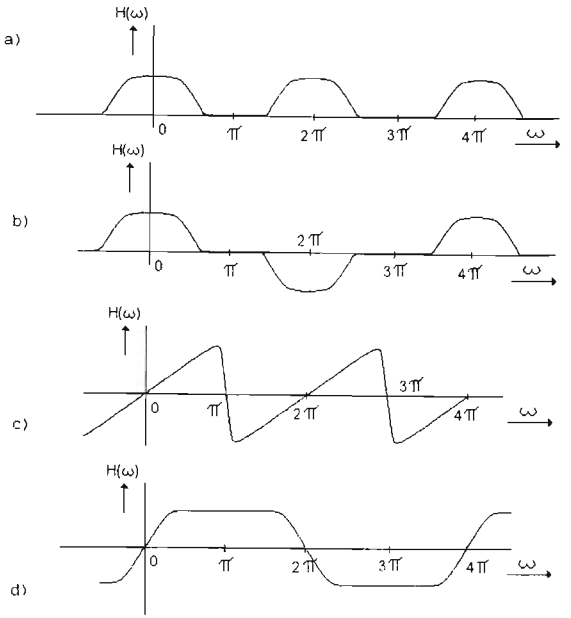


Figura 4.7: Respuestas de diferentes filtros FIR

nominador dentro de un producto lineal y cuadrático. Estos factores lineales y cuadrático corresponden a polos y ceros reales conjugados complejos, respectivamente. Los factores pares del numerador y denominador definen al subsistema que esta en cascada para producir el sistema deseado.

$$H(z) = K H_1(z).....H_M(z) \quad M = Max \tag{4.17}$$

La transformación corresponde del diagrama de bloques de la figura 4.12. Cada subsistema en cascada es realizado con la estructura De la Forma Directa II, La función de transferencia del subsistema tiene la forma de factores de primer y segundo orden. Algunos de los coeficientes(A_{0i}, ó A_{1i}, por ejemplo) pueden ser cero.
Cuando los factores del numerador y el denominador son pares se produce la función de transferencia del subsistema, entonces los factores pares son tantos como sean posibles. Cuando se usan pares máximos, el número de elementos de almacén requerido para la realización de cascada en el mismo como para la realización de la Forma Directa II.
El factor de pareja del orden del subsistema no es único; por lo tanto se puden realizar diferentes estructuras en cascada. Con una señal de precisión infinita y de valores de parámetros, todas las realizaciones de cascada dan como resultado idénticas señales de salida para la misma entrada. En un sistema físico tenemos longitudes de palabras finitas, el cambio de la señal de salida debido al truncamiento del parámetro o el circundante, son menos para algunos factores pares y orden del subsistema que para otros. Un método para seleccionar los mejores valores pares y de orden no existe; sin embargo pueden usarse subsistemas para

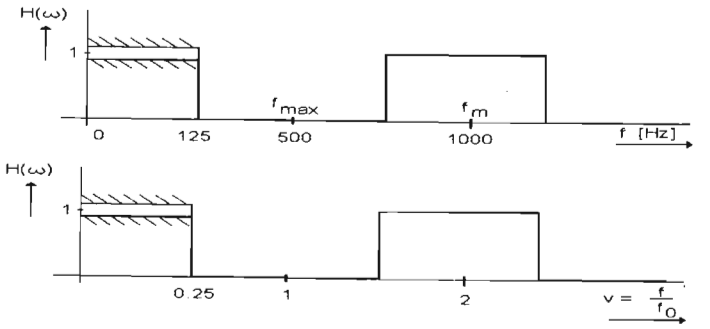


Figura 4.8: Especificaciones para un filtro FIR normalizado y no normalizado

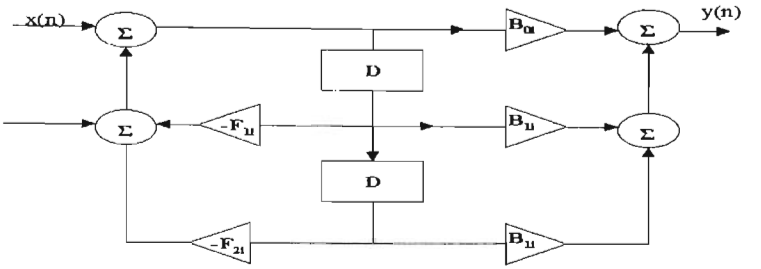


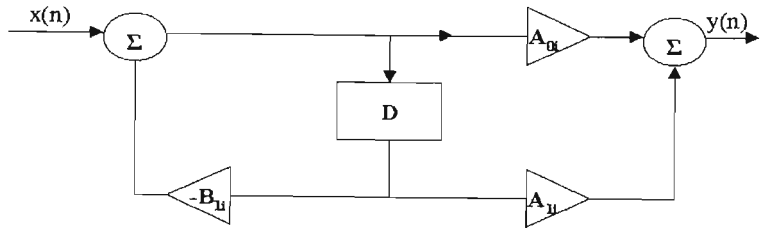
Figura 4.9: Sistema de segundo orden, de la realización de la forma directa II

encontrar la selección.

La realización paralela : El segundo método de descomposición de la función de transferencia para la realización de una función paralela, se realiza mediante la descomposición dentro de una subsección de orden bajo aditivo por medio de la expansión de fracciones parciales. Las subsecciones son realizadas y conectadas en paralelo, Por tanto iniciamos una función de transderencia en forma estándar

4.5. Filtrado adaptable en Subbandas

El filtrado adaptable en subbandas, las señales son divididas en N bandas mas bien subbandas por medio de bancos de filtros pasa banda estos se traslapan con filtros que se encuentran en el mismo arreglo. Las señales son submuestreadas por un factor M, el cual es menor al número de sub bandas, en cada una de estas subbandas es colocado un filtro transversal adaptable operando a Fs/M Hz, en donde los coeficientes generados se adaptan para minimizar el valor



(a) Sistema de primer orden

Figura 4.10: Sistema de primer orden, de la realizacón de la forma directa II

cuadrático medio del error de salida en cada subbanda. Los errores de salida son enviados a otro bancos de filtros pasa banda en donde sintetizan la señal.

La frecuencia de muestreo es reducida por le factor M y este valor es muy cercano al número de subbandas, los procesos de filtrado y adaptación se pueden llevar a cabo a una frecuencia de fs/M, donde fs es la frecuencia de muestreo. Por lo anterior la complejidad computacional se reduce. La convergencia puede ser incrementada ajustando los valores de convergencia en cada una de las subbandas de acuerdo a la energía de la señal de entrada. La correlación entre muestras sucesivas de las señales submuestreadas se debilita si se incrementan los factores de submuestreo. Para factores relativamente grandes, la velocidad de convergencia llega a ser independiente de las estadísticas de las señales de entrada. Una señal con factores de submuestreo mayores o iguales a 16 la velocidad de convergencia del sistema con ruido blanco o señales de voz, como señales de entrada, es prácticamente las misma.

En un sistema en subbandas se pueden especificar las bandas para obtener una mejora en la utilización de dispositivos auxiliares, como detectores Double Talk.

4.6. Filtrado adaptable en el dominio de la frecuencia.

Para aumentar la velocidad de convergencia de los filtros transversales por medio de algoritmos basados en la búsqueda del gradiente, se implementa en los sistemas computacionales la transformada rápida de Fourier, esta por sus características calcula las convoluciones y correlaciones para la obtención de los coeficientes del filtro.

A partir de la prueba del algoritmo LMS se derivan un conjunto de factores de convergencia óptimos, los cuales son obtenidos posteriormente, en forma similar usa una transformación ortogonal como la transformada discreta de Fourier. Este algoritmo efectúa velocidades de convergencia mayores que otro tipo de algoritmos.

Para evaluar el funcionamiento de un cancelador de eco es el llamado ERLE (echo return loss ehancement) el es un referente para una medida de atenuación del eco. Por ello el ERLE indica el funcionamiento del cancelador del eco.

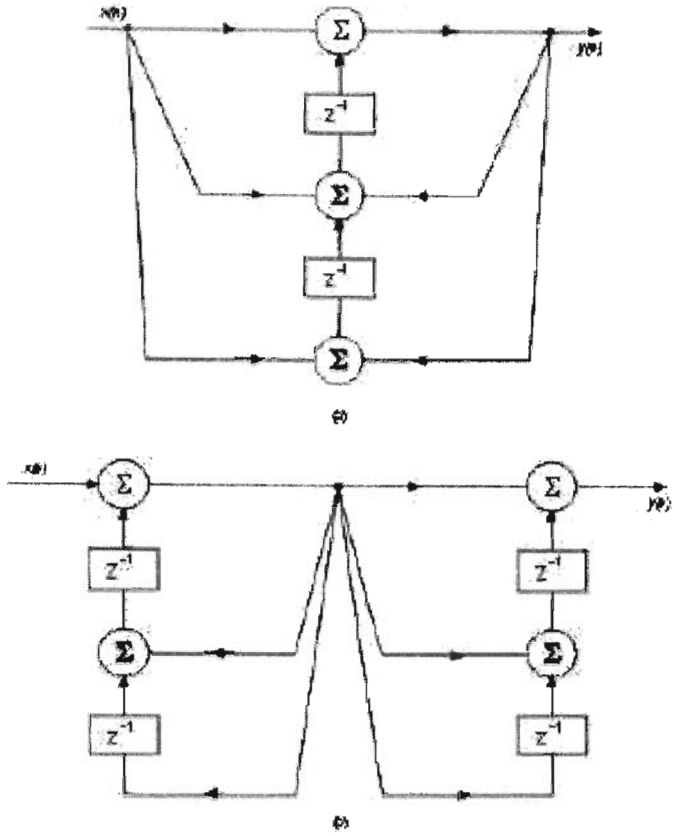


Figura 4.11: Diagrama de bloque de la representación de un sistema de segundo orden de la forma directa II (a) y un sistema no canónico (b)

4.7. Método 1: Detector de Double-Talk.

Este detector registra el inicio y final del periodo de double talk e inhibe la adaptación del cancelador, evitando así un funcionamiento inadecuado del sistema. En este método se mide la potencia de las señales de entrada y de eco y se declara double -talk figura 4.19 cuando

$$x^2(n) < A_{min} d^2(n)$$

Donde A_{min} es la atenuación introducida por el canal de eco, es importante señalar que se necesita conocer le factor de pérdida por lo menos de manera aproximada es por ello que este método solo podrá funcionar en circuitos híbrido donde el valor de esta perdida es de 6 dB aproximadamente. Entonces este sistema resulta de poco interés.

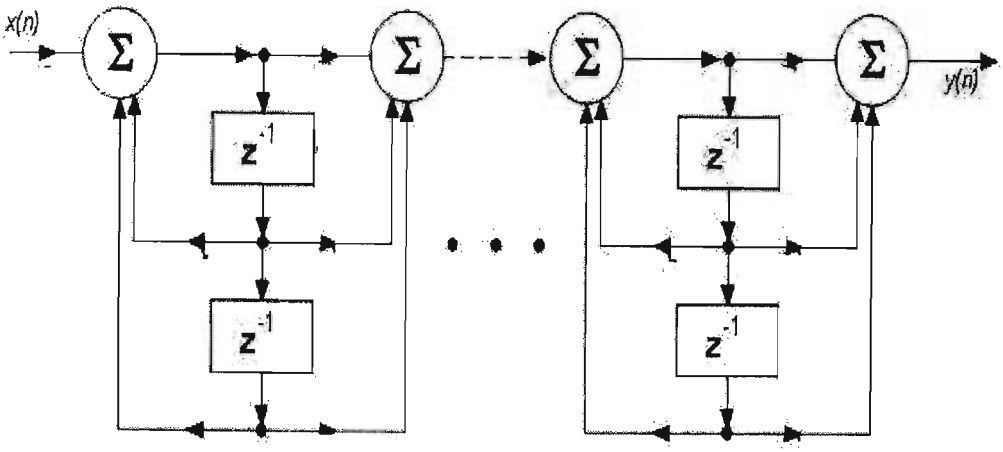


Figura 4.12: Diagrama de bloque de la representación de un sistema de cascada

4.8. Método 2

La atenuación introducida al sistema a la señal de eco, ERLE este se obtiene dividiendo la potencia de la señal de eco antes de su cancelación, d(n), entre la potencia del eco residual, e(n). El double talk se declara si cumple con los siguiente:

$$ERLE < ERLE_{op}$$

$ERLE_{op}$ es el umbral determinado previamente, la selección de este paráinetro depende de la capacidad para adaptarse a la detección y a la varaciones de los canales en el tiempo. En umbral muy alto tiene una buena detección, mientras que uno bajo no detectará eco aún cuando sea muy grande.

4.9. Método 3

Debido a las características espectrales de las señales de voz, cuya energía se localiza entre los 100 Hz. y 1000 Hz, se propone un detector de double talk para canceladores de eco con estructuras en subbandas, SBEC
En las figuras 4.20 y 4.21 el principio de operación del detector propuesto el cual tiene un filtro adaptivo de orden N/M donde es el orden del cancelador si opera en toda la banda y M el factor de submuestreo. Un algoritmo de decisión se emplea para inhibir la adaptación del SBEC, cuando el período de double talk es detectado, en este proceso se usa el mismo algoritmo de adaptación que le usado por los filtros adaptables que operan en subbandas, por medio de esto se estima el tipo de señal para decidir cancelar o no la señal.

Por otro lado la degradación de la señal depende al sistema double talk o a cambios en

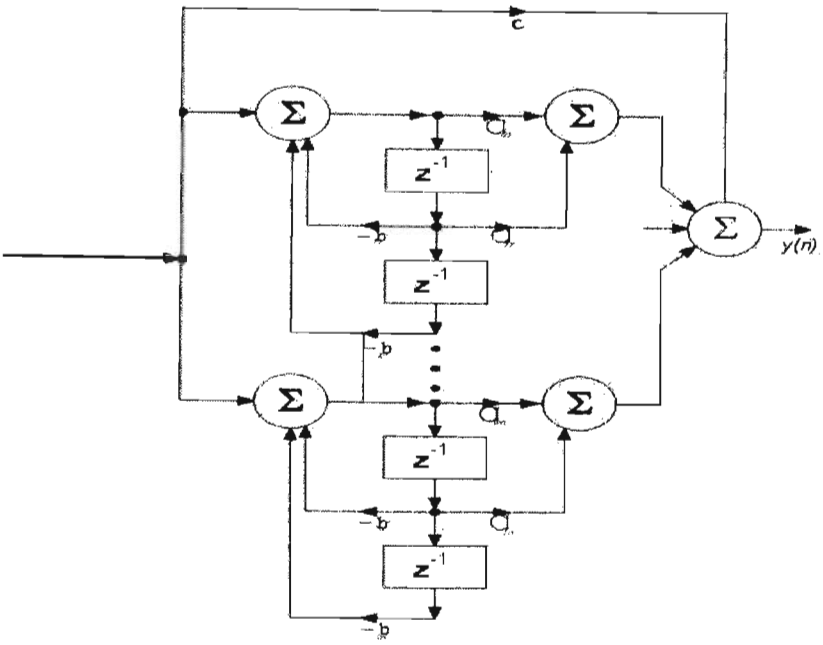


Figura 4.13: Diagrama de bloque de la representación de un sistema en paralelo

las características del canal de eco. En ambos casos el nivel de filtro usado en el detector decrecerá. En estos casos el nivel de cancelación permanecerá por debajo de cierto umbral durante toda la duración del double talk, esto ocurre pues la potencia de la voz del interlocutor cercano, el cuál será interpretado por el algoritmo como ruido adaptivo. También después de cierto aprendizaje el filtro podrá adaptarse a los modificaciones y el nivel de cancelación aumentará. Esto detectara los periodos de double talk, en forma rápida y precisa, sin restringir las variaciones en las características del canal de eco

El algoritmo de detección registra un fenómeno de double-talk, $ERLE_{DT}$ cruza en sentido descendente de un alto umbral a una inhibición del SBEC y este cruzará en sentido descendente. Cuando el algoritmo aprende las modificaciones necesarias el $ERLED_{DT}$ cruzará, en sentido ascendente alguno de los dos umbrales, con ello entrará en un ciclo la adaptación de SBEC.

Por la anterior es conveniente usar ambos métodos y estos no restringen la capacidad del sistema y algoritmo. Experimentos realizados muestran que algunos valores que podrían ser adecuados se encuentran entre 30 y 35 dB para el umbral alto, y entre 20 y 25 para el umbral bajo.

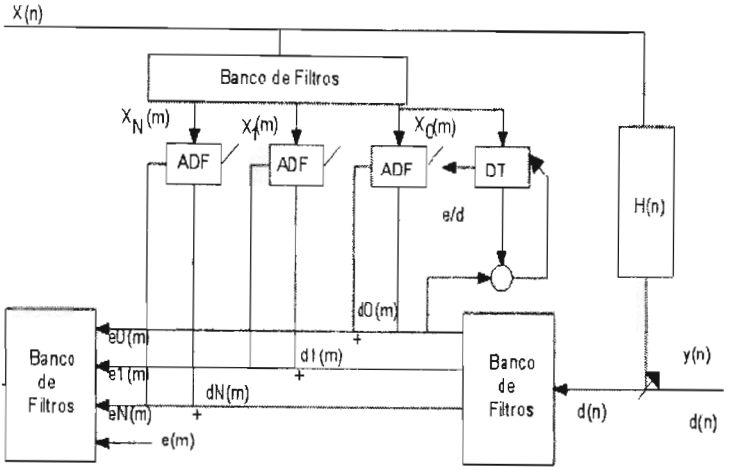


Figura 4.14: Estructura básica para cancelación en subbanda, protegida en situaciones double

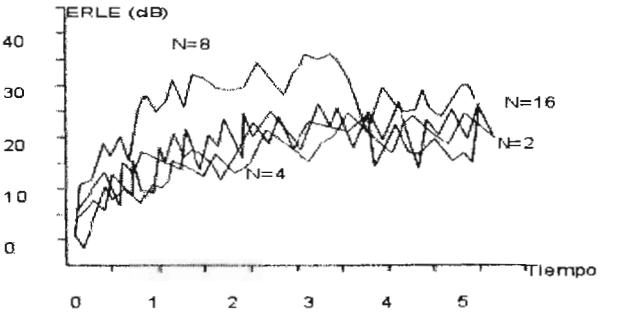


Figura 4.15: Convergencia del cancelador de eco en subbandas empleando 2,4,8 y 16 subbandas

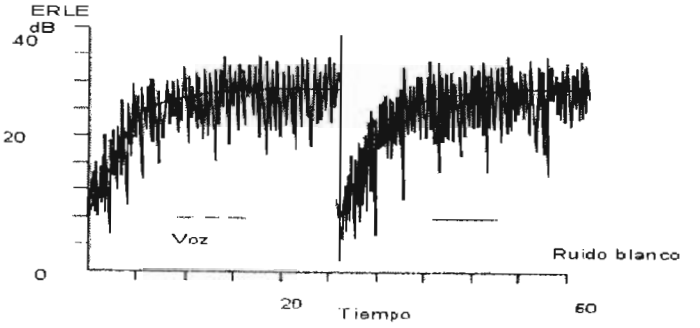


Figura 4.16: Convergencia de un cancelador de eco con 16 subbandas. Las señales de entrada fueron de voz real y ruido blanco

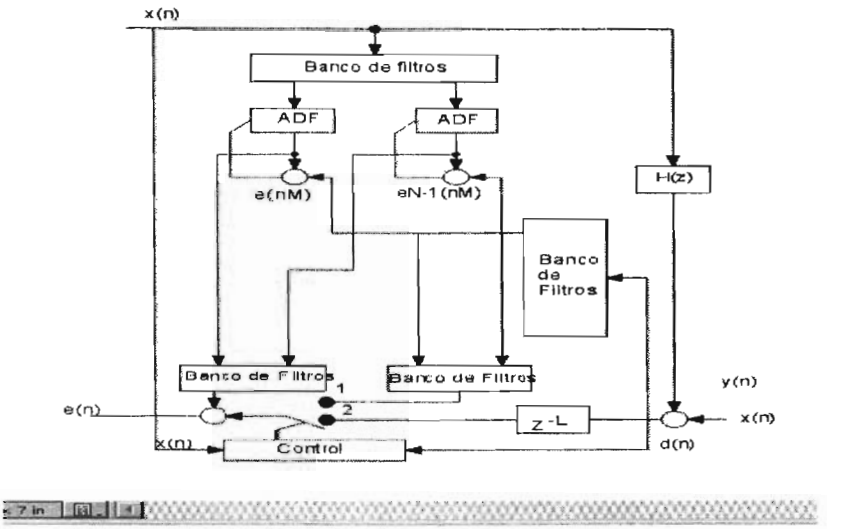


Figura 4.17: Cancelador de eco en subbandas con tiempo de retardo menor

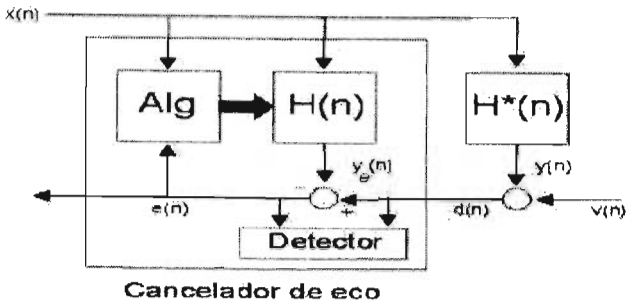


Figura 4.18: Detector de Double talk usando la potencia de las señales de error, x(n) y referencia d(n)

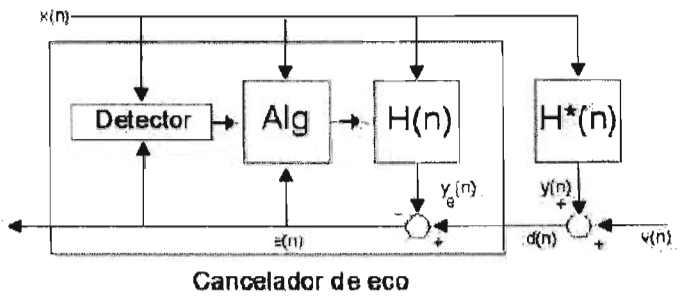


Figura 4.19: Detector de Double talk usando la potencia de las señales de entrada, x(n) y referencia d(n)

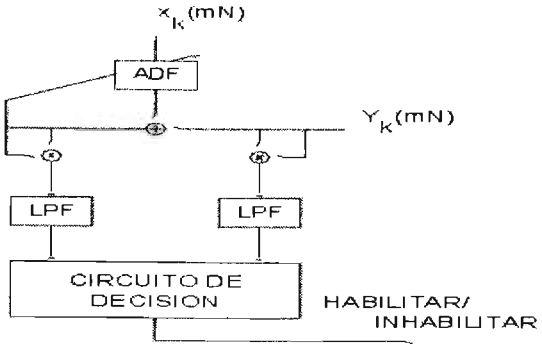


Figura 4.20: Detector Double-talk para cancelaciones de eco usando configuraciones en sub-bandas

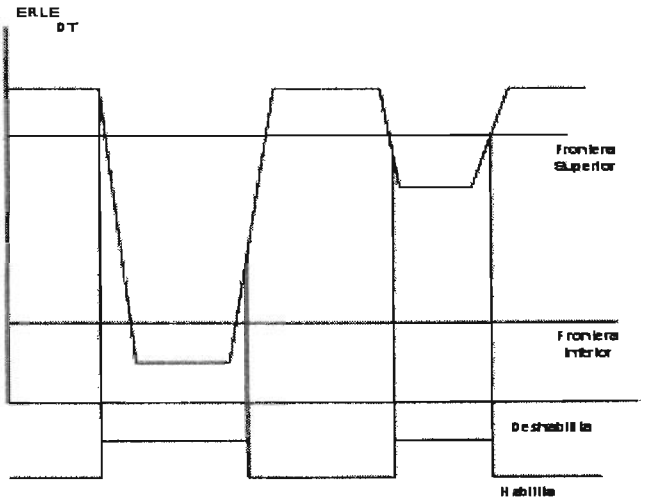


Figura 4.21: Principio de detección del método propuesto

Capítulo 5

La Transformada Rápida de Fourier

La transformada discreta de Fourier está definida mediante la siguiente ecuación :

$$x(n) = \sum_{k=0}^{N-1} x_0(k) W^{nk} \tag{5.1}$$

Donde tenemos que $W = e^{j\frac{2\pi}{N}}$, describe la parte de integración, y k es un número binario, considerando que N = 4 y de $N = 2^c$ entoces c=2 y representando k y n como números binarios:

$$k = 0, 1, 2, 3 \;\; o \;\; k = (k_1, k_0) = 00, 01, 10, 11$$
$$n = 0, 1, 2, 3 \;\; o \;\; n = (n_1, n_0) = 00, 01, 10, 11$$

Rescribiendo el anterior método en función de n y de k tenemos que :

$$k = 2k_1 + k_0 \quad n = 2n_1 + n_0 \tag{5.2}$$

Donde k_1, k_0, n_0, yn_1 pueden tomar valores de cero y uno, usando la representación (5.2) podemos rescribir para el caso de que N = 4 como :

$$x(n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} X_0(k_1, k_0) W^{(2n_1+n_0)(k_1+k_0)} \tag{5.3}$$

Factorizando desde W, y considerando que W entonces :

$$W^{(2n_1+2n_0)(2k_1+2k_0)} = W^{(2n_1+n_0)2k_1} W^{(2n_1+n_0)k_0}$$
$$= W^{4n_1k_1} W^{2n_0k_1} W^{(2n_1+n_0)k_0}$$
$$= W^{(2n_0k_1)} W^{(2n_1+n_0)k_0}$$

Notamos que el siguiente término se simplifica como:

$$W^{4n_1k_1} = \left[W^4\right]^{n_1k_1} = \left[e^{-j\frac{2\pi 4}{4}}\right]^{n_1k_1} = [1]^{n_1k_1} = 1 \tag{5.4}$$

Entonces sustituyendo la ecuación (5.3) rescribimos la ecuación de la siguiente forma :

$$X(n_1, n_0) = \sum_{k_0=0}^{1} \left[\sum_{k_1=0}^{1} X_0(k_1, k_0) W^{(2n_0k_1} \right] W^{(2n_0+n_0)k_0} \tag{5.5}$$

Esta ecuación representa la notación del algoritmo de la FFT, que demuestra cada uno de los parámetros individualmente. Primero rescribimos la sumatoria un partes enumerando la ecuación mediante la ecuación (5.5), obtenemos :

$$X_1(n_0, k_0) = \sum_{k_0=0}^{1} x_0(k_1, k_0) W^{2n_0 k_1} \tag{5.6}$$

Enumerando la ecuación representada en (5.6) obtenemos :

$$\begin{aligned} X_1(0,0) &= X_0(0,0) + X_0(1,0)W^0 \\ X_1(0,1) &= X_0(0,1) + W^0(1,1)W^0 \\ X_1(1,0) &= X_0(0,1) + X_0(1,0)W^2 \\ X_1(1,1) &= X_0(0,1) + X_0(1,1)W^2 \end{aligned} \tag{5.7}$$

Rescribiendo la ecuación (5.7) en notación matricial.

$$\begin{bmatrix} X_1(0,0) \\ X_1(0,1) \\ X_1(1,0) \\ X_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \times \begin{bmatrix} X_0(0,0) \\ X_0(0,1) \\ X_0(1,0) \\ X_0(1,1) \end{bmatrix} \tag{5.8}$$

Simplificando y rescribiendo según la ecuación (5.6).

$$X_2(n_0, n_1) = \sum_{k_0=0}^{1} X_1(n_0, k_0) W^{(2n_1+n_0)K_0} \tag{5.9}$$

transformando los resultados en forma matricial obtenemos :

$$\begin{bmatrix} X_2(0,0) \\ X_2(0,1) \\ X_2(1,0) \\ X_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \times \begin{bmatrix} X_1(0,0) \\ X_1(0,1) \\ X_1(1,0) \\ X_1(1,1) \end{bmatrix} \tag{5.10}$$

De las ecuaciones (5.6) y (5.10) tenemos que :

$$X(n_1, n_0) = X_2(n_0, n_1) \tag{5.11}$$

Esta función resulta de los parámetros obtenidos anteriormente. La simplificación de estos parámetros se obtenido el algoritmo de las FFT. Si combinamos (5.6) (5.10) y (5.11).

$$\begin{aligned} X_1(n_0, k_0) &= \sum_{K_0=0}^{1} x_0(k_1, k_0) W^{2n_0 k_1} \\ X_2(n_0, n_1) &= \sum_{k_0=0}^{1} x_1(n_0, k_0) W^{(2n_1+n_0)k_0} \\ X(n_1, n_0) &= x_2(n_0, n_1) \end{aligned} \tag{5.12}$$

La ecuación (5.12) representa el algoritmo de COOLEY-TUKEY para un algoritmo de la FFT N = 4.

Ejemplo:

Para ilustrar la notación asociada con la formula De COOLEY-TUKEY para representar la FFT. Consideramos la Ecuación (5.1) para el caso $N = 2^3 = 8$, entonces :

$$n = 4n_2 + 2n_1 + n_0 \quad n_1 = 0 \; o \; 1$$

$$k = 4k_2 + 2n_1 + n_0 \quad k_i = 0 \; o \; 1 \tag{5.13}$$

Por lo tanto (5.1) se convierte

$$X(n_2, n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} x_0(k_2, k_1, k_0) W^{(4n_2+2n_1+n_0)(4k_2+2k_1+k_0)} \tag{5.14}$$

Rescribiendo

$$W^{(4n_2+2n_1+n_0)(4k_2+2k_1+k_0)} = W^{(4n_2+2n_1+n_0)4k_2} W^{(4n_2+2n_1+n_0)2k_1} W^{(4n_2+2n_1+n_0)k_0} \tag{5.15}$$

Como :

$$W^8 \left[e^{-j\frac{2\pi}{8}}\right]^8 = 1 \tag{5.16}$$

entonces:

$$W^{(4n_2+2n_1+n_0)4k_2} = W^{4n_0k_1} \tag{5.17}$$

Sustituyendo en la ecuación (5.13)

$$X(n_2, n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} x_0(k_2, k_1, k_0) W^{4n_0k_1} W^{(2n_1+n_0)(2k_1)} W^{(4n_2+2n_1+n_0)k_0} \tag{5.18}$$

Si definimos

$$X_1(n_0, k_1, k_0) = \sum_{k_1=0}^{1} x_1(K_2, k_1, k_0) W^{4n_0k_2} \tag{5.19}$$

$$X_1(n_0, n_1, k_0) = \sum_{k_1=0}^{1} x_1(n_0, k_1, k_0) W^{(2n_1+n_0)2k_1} \tag{5.20}$$

$$X_3(n_0, n_1, n_2) = \sum_{k_1=0}^{1} x_1(n_0, k_1, k_0) W^{(2n_2+n_0)2k_1} \tag{5.21}$$

$$X_2(n_2, n_1, n_0) = X_3(n_0, n_1, n_2) \tag{5.22}$$

determinamos la matriz de factorización o el equivalente para una señal de N = 8. Esta grafica esta desarrollada de las ecuaciones (5.18),(5.19),(5.20) y (5.21)

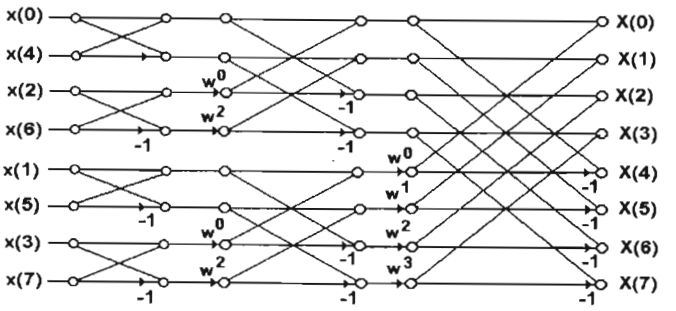


Figura 5.1: Mariposa Básica para FFT=8

5.1. Definimos la transformada de Fourier :

$$X(n) = \sum x_0(k) e^{-j2\pi \frac{nk}{N}} \quad n = 0, 1, 2.....N-1 \tag{5.23}$$

De la ecuación anterior sistutuyendo para n=k y N=4 y si $W = e^{-j2\pi \frac{\pi}{N}}$, obtenemos las siguientes ecuaciones :

$$X(0) = x_0(0)W^0 + x_0(1)W^0 + x_0(2)W^0 + x_0(3)W^0$$
$$X(1) = x_0(0)W^0 + x_0(1)W^1 + x_0(2)W^2 + x_0(3)W^3$$
$$X(2) = x_0(0)W^0 + x_0(1)W^2 + x_0(2)W^4 + x_0(3)W^6$$
$$X(3) = x_0(0)W^0 + x_0(1)W^3 + x_0(2)W^6 + x_0(3)W^9$$

Escribiendo en forma matricial

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{5.24}$$

Sabiendo que $X(n) = [W^{nk} X_0]^k$
sustituyendo en (5.23) con W y multiplicando N(N-1) veces hará necesariamente una reducción de los cálculos. El algoritmo de la FFT disminuye el número de multiplicaciones y adiciones requeridas para su cálculo. Para demostrar el algoritmo de la FFT, es conveniente cerrar los números de las relaciones encontradas para, donde c es una integral que proviene de la formula (5.22). Resultando como elección los valores de N = 4 y c = 2. La primera definición para el desarrollo de la FFT lo podemos escribir mediante la matriz.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{5.25}$$

Evaluando (5.24) con lo desarrollado en la forma matricial (5.22), podemos determinar que N=4, n=2 y K=3, por lo tanto :

$$W^6 = W^{-2}$$

$$W^{nk} = W^6 = e^{-j6\frac{2\pi}{4}} = e^{-j3\pi}$$

$$= e^{-j2\frac{2\pi}{2}} = W^2 = W^{NK}$$

De esta segunda definición desarrollamos los factores para ser representados de manera matricial, como a continuación se muestra.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 1 & 0 & 0 & W^6 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{5.26}$$

De las columnas 1 y 2 tenemos una integración, que se puede transformar en un vector representado como:

$$X(n) = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} \tag{5.27}$$

Con la expresión anterior, logramos factorizar algunos términos, con ellos hacemos eficente las operaciones de la FFT. La expresión (5.26), se utiliza para hacer la multiplicación requerida para la siguiente matriz.

$$\begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \tag{5.28}$$

El vector $X_1 = k$ es igual al producto de mas matrices de la ecuación (5.26), el elemento x_10 es el complemento complejo de la siguiente adicción.

$$X_1(0) = X_0(0) + W^0 X_0(2)$$

El elemento X_1 es también determinado por el producto complejo de la adición solo un número complejo se necesita para realizar la operación. Es por esto que $W^0 = -W^2$

$$X_1(2) = X_0(0) + W^2 X_0(2) = X(0) - W^0 X_0(2)$$

Donde la multiplicación compleja $W_0 X_0(1)$ ha sido determinada de la ecuación (5.27). El vector intermedio es determinado por la forma compleja de la adición y de las dos multiplicaciones. Operando y sustituyendo las operaciones anteriores obtenemos :

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} \tag{5.29}$$

El término es determinado de la multiplicación y adición ambas complejas.

$$X_2(0) = X_1(0) + W^0 X_1(1)$$

El elemento $X_2(1)$, es obtenido de una adición porque, $W^0 = -W^2$ por los que $X_2(2)$ es determinado por un número complejo a partir de una multiplicación y una adición.
Operando los términos de la ecuación (5.26) se han requerido del total de 4 líneas complejas de la matriz y de cada una de las adiciones. Si sustituimos en $X(n)$ la ecuación (5.22), se harán 16 operaciones complejas. De lo anterior, la FFT require de $\frac{NC}{2} = 4$ multiplicaciones complejas y NC adicciones complejas. Entonces determinados que el método de decimación realiza N^2 multiplicaciones complejas y $N(N-1)$ adicciones complejas

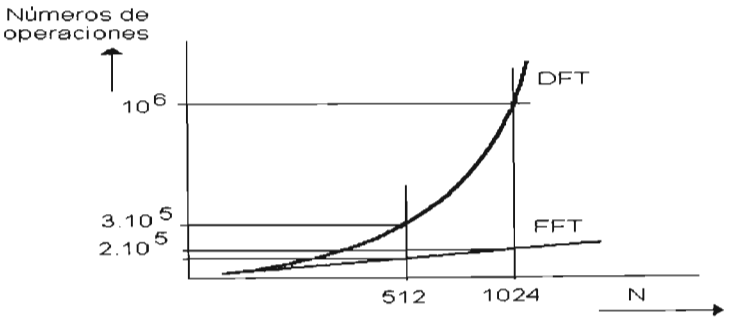


Figura 5.2: Comparación en el número de operaciones ralizadas entre la FFT y la DFT

La FFT explota al máximo las propiedades de simetría y periodicidad de la DFT, por ello el número de operaciones se redeuce en un orden de $N^2 a N log_2(N)$, con ello el tiempo de operación en un procesador digital de señales disminuye considerablemente.

Capítulo 6

Cancelador de eco propuesto

La identificación del tiempo real de la trayectoria del eco es un problema difícil de resolver debido al comportamiento del eco. En primer lugar la trayectoria del eco es no estacionaria y el espectro de voz no es plano, con gran dispersión en sus valores propios, lo que resulta una velocidad lenta de convergencia. Los canceladores de eco transversales son adecuados al caso pues utilizan algoritmos adaptables basados en la búsqueda del gradiente.

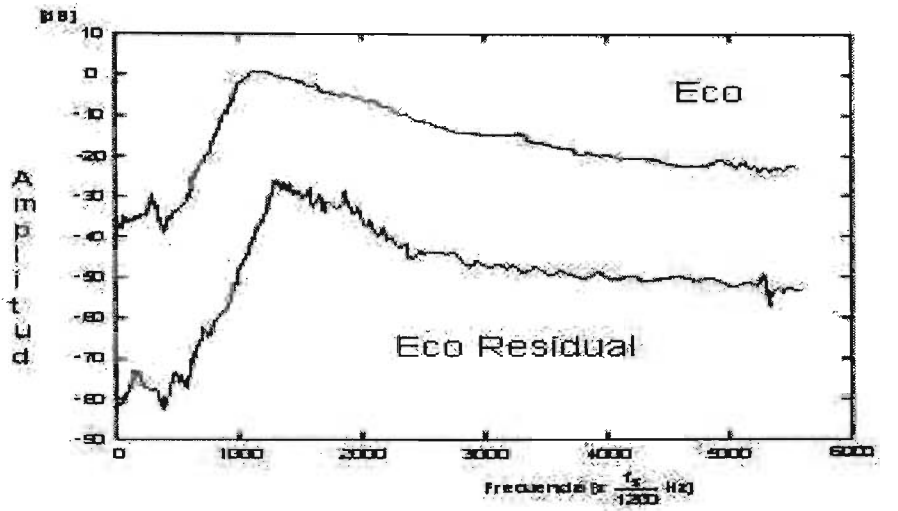


Figura 6.1: Espectro de densidad de potencia del algoritmo propuesto

Para ayudar a resolver este problema se propone la modulación de ruido pseudo aleatorio (pr), el cual consiste en multiplicar la señal de entrada del filtro adaptable por una secuencia pseudo-aleatoria, con esto se reduce lo disperso de los valores propios y se puede utilizar el algoritmo NMLS. En aplicaciones para la cancelación de eco, la señal recibida se distorsiona, esto puede evitarse si la señal de la pseudo secuencia es cancelada en el extremo cercano. Esto es complicado ya que la señal de recepción es desconocida y es de tiempo variable, por lo tanto debe estimarse la señal de entrada. Se propone utilizar una señal aleatoria que se

agrega a la entrada, y en lugar de modularla se extrae del extremo cercano usando un filtro adaptable de menor orden.

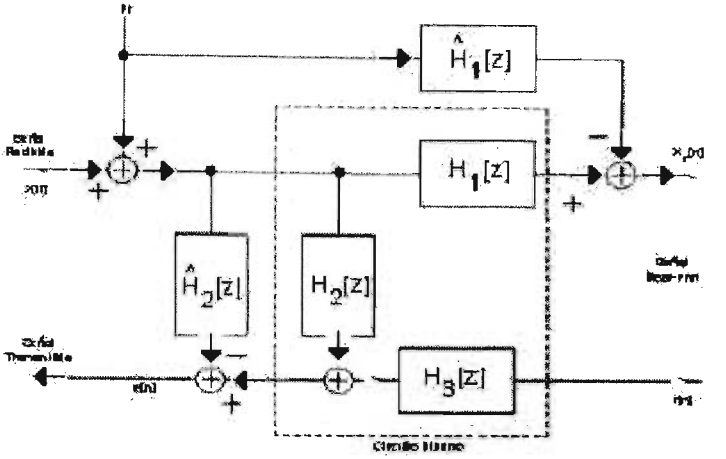


Figura 6.2: Estructura de Cancelador de eco propuesto

6.1. Estructura del cancelador de eco propuesto

La estructura para la cancelación de eco, como ya se menciono permite reducir lo disperso de los valores propios o eingenvalores de entrada de la matriz de autocorrelación, mediante una secuencia pseudo aleatoria a la señal de entrada.
Habrá que considerar que si una secuencia aleatoria se agrega a la señal de recepción, la señal del extremo cercano se distorsiona, esto puede evitarse si secuencia aleatoria se cancela en el extremo cercano. Si tómanos en cuenta que la función de transferencia entre el punto de recepción y el extremo lejano, H(z) es desconocida y su tiempo variable, la señal aleatoria no puede obtenerse directamente en el punto del extremo cercano, entonces el filtro adaptable se usa para cancelar la secuencia aleatoria. Con ello se ortogonaliza la señal de entrada sin distorsionar la señal del extremo cercano. Para ello se requieren dos filtros adaptables como se muestra en la figura (4.3)
Estos dos filtros transversales adaptables, se usarán en la estructura del cancelador de eco propuesto, uno de ellos se usará para estimar la señal de eco. El orden del filtro deberá ser relativamente grande y utilizaremos una versión normalizada del algoritmo LMS, que cuenta con una baja complejidad al ser programado. Siendo este una opción para actualizar los coeficientes del cancelador de eco.

6.1.1. Programa de Cancelación de eco

```
Clear;
% Inicialización del sistema a ser identificado

N2=128;                              % Orden del sistema no conocido
Z2 =zeros (1,N2);                    % línea  de retardo del sistema no conocido
A= zeros (1,N2);                     % vector de coeficientes
% vector de coeficientes del sistema no conocido
for k = 1: N2
theta =2.0*pi*i/3.0;
A(k)= exp(-0.2*k)*cos(2*pi*k/ 10+sin (2*pi*k/30));
end

% Inicialización del filtro FIR

N = 128;  % orden del filtro
W = zeros(1,N); % coeficientes
Y = zeros(1,N);           % línea de retardo

% Simulación

Num_iter = 128;                      %Número de iteraciones por bloque
Num_block=150;                              %Número  total de bloques
add=100.0;                                 %relación  señal a ruido
psig =0.0;
pxin=0.0;
erlop=0.9;

for k 1 = 1.: Num-block
k1
pot_sig=0.0;
pot_err=0.0;
pot_in=0.0;
if  k1>50 & k 1 < 1 00
add=100;
else
add= 100;
else
add=100;
end
for k2=1:Num_iter

x = 2.0*(rand (1)-0.5);                    %señal de entrada
xn=2.0*(rand (1)-0.5 ) / add;              %ruido aditivo
```

```
% FIR unknown
for k=N2:-1:2
Z2(k)=Z2(k-1);                    %recorrer línea de retardo
 End
Z2                                %cargar nuevo dato
yI=Z2*A;                         %cálculo de la señal de salida
yd=yl+xn;

%FIR adaptivo
for k=N:-1:2
Y (k) =Y (k- l);                   %recorrer línea de retardo
end
Y(1)=x;                           %cargar nuevo dato
y0=Y*W;                               %cálculo de la señal de salida

%Cálculo del error
error = yd-y0;
%Adaptación
%Detección de double talk
psa=psig;
pxa=pxin;
psig=0.99*psa+0.01*(yd*yd);
pxin=0.99*pxa+0.01*(x*x);
erl=psig / pxin;
if erl<erlop
alfa= 1.0;                      %factor de convergencia
else
alfa=0.001;                             %factor de convergencia
end
miu=alfa/(Y*Y');
W=W+miu*error*Y;

%Promedios
pot_sig=pot_sig+(y1*y1);
pot_err=pot_err+((y1-y0)*(y1-y0));
pot_in=pot_in+(x*x);
end
MSE(K1)= 10.0*LOG10(pot_err/Num--block);
PSG(k1)=10.0*LOG10(psig);
end
figure(2);
subplot(2,1,1); plot(MSE,'-k');
subplot(2,1,2); plot(PSG,'-k');
```

B.2 Programa que generó las figuras que se muestran en el capítulo V, que muestran las
superficies de error (ersurf.m):

```
% Inicialización del sistema desconocido
N2=2;                    %Orden del sistema no conocido
Z2=zeros(1,N2);              %línea de retardo del sistema no conocido
A=zeros(1,N2);                 %vector de coeficientes

%vector de coeficientes del sistema desconocido
A(1)=9
A(2) =12

%lnicialización de los Filtros FIR
N=2;                      %orden  del filtro
AI=zeros(1,N);                  %coeficientes
ZI=zeros(1,N);                  %línea de retardo

%SIMULACION

Num_block1=20;                  %Número de iteraciones por bloque
Num_block2=20;
xd=0.0;
Num-iter=2000;
MSE=zeros(20,20);
for k3=1:Num_iter
k3
for k1 =1:Num_block1
for k2=1:Num_block2
AI(1)=k1;
AI(2)=k2;
xpr=2.0*(rand (1)-0.5);          %señal de entrada
xprl=4.0*(rand(1)-0.5);       %señal de entrada
xra=xd;                    %señal de entrada
xd=0.7*xra+-xpr;
x=xpr1;

%sistema desconocido
Z2(2)=Z2(1);
Z2 (1) =x;                       %cargar  nuevo dato
yd=Z2*A';                       %cálculo de la señal de salida

% filtro adaptable
ZI(2)=ZI(1);                      %recorrer línea  de retardo
ZI(1)=x;                          %cargar nuevo dato
y0=ZI*AI';                          %cálculo  de la señal de salida

%Cálculo de error
error =yd-y0;
MSE(k1,k2)=MSE(k1,k2)+(error*error/Num_iter);
```

```
end
end;
end;
figure(1);
mesh(MSE);
figure(2);
contour(MSE,20);
```

B.3 Programa que genera las figuras que se presentaron en el capitulo V, que muestra las comparaciones de velocidad de convergencia entre el NLMS y la estructura propuesta.
Prnlms.m):

```
clear;
%Captura de la señal de entrada
[yin,fr]=wavread('voz2');
NO=length(yin)
xmax=max(yin)
yin=,Yin(2000:NO);
NO=length(yin)
xav=sum(yin,/NO
xmax=max(yin)
for k= 1: NO
yin(k)=(yin(k)-xav)/xmax;
end
figure(1)
plot(yin)
pause

% Inicialización del canal de eco

N2= 128;                %Orden del sistema no conocido
Z2=zeros(1,N2);               %línea de retardo del sistema no conocido
A=zeros(1,N2);               %vector de coeficientes

%vector de coeficientes del canal de eco
for k=1:N2
theta=2.0*pi*i/3.0;
A(k)=exp(-0.2*k)*cos(2*pi*k/ 10+sin(2*pi*k/30));
end

% Inicialización del canal de entrada
NI=40;                    %Orden del sistema no conocido
ZI=zeros(1,NI);                %línea de retardo del sistema no conocido
AI=zeros(1,NI);                %vector de coeficientes
ZHI=zeros(1,NI);              %línea de retardo del híbrido de entrada
AHI=zeros(1,NI);             %Coeficientes del Híbrido de entrada
```

```
%vector de coeficientes del canal de entrada
AI(20)= 1.0;
AHI(20) =1.0;

%Inicialización de los filtros FIR_1 y FIR_2

N= 128;                    %orden del filtro
W=zeros(1,N);                  %coeficientes
Y=zeros(1,N);                  %línea  de retardo
NI1=40;                    %orden del filtro
WI=zeros(1,NI1);               %coeficientes
YI1=zeros(1,NI1);              %línea de retardo

%SIMULACIÓN
Num_iter=128;                  %Número de iteraciones por bloque
%Num_block=fix(NO/12)          %Número total de bloques
Num_block=100;
add= 100.00;                   %relación señal a ruido
psig=0.0;
pxin=0.0;
erlop=0.8;
for kl=1:Num_block
kl
pot_sig=0.0;
pot_err=0.0;
pot_in=0.0;
pot_errl=0.0;
pot_yI=0.0;
if k1>100 & kl<150
add=100;
else
add=100;
end
for k2=1::Num_iter
xpr=2.0*(rand(1)-0.5);                %señal de entrada
xd=yin((kl-1)*Num_iter+k2);           %señal de entrada
x=xd+xpr;
xn=2.0*(rand(1)-0.5)/add;             %ruido aditivo

% canal de eco

for k=N2:-1:2
Z2 (k) =Z2 (k- 1);                    %recorrer línea de retardo
End
Z2(1)=x;                              %cargar nuevo dato
y1=Z2*A';                             %cálculo de la señal de salida
yd=y1 +xn;
```

```
%canal de entrada

for k=NI:-1.2
ZI(k) =ZI(k- 1);                %recorrer línea de retardo
end
ZI(1)=xd;                      %cargar nuevo dato
yI=ZI*AI';                      %cálculo de la señal de salida
for k=NI:-1.2
ZHI(k)=ZHI(k-1);                 %recorrer línea de retardo
end
ZHI(1)=xpr;                      %cargar nuevo dato
yHI=ZHI*AHI';                    %cálculo de la señal de salida
yHI1=yl+yHl:
% FIR adaptivo 1

for k= N: -1.2
Y(k) =Y(k-1);                   %recorrer línea de retardo
end
Y(1)=x;                          %cargar nuevo dato
yO=Y*W';                         %cálculo de la señal de salida

%Cálculo del error

error=yd-yO;
resec((k-1)*Num_iter+k2)=error;  %señal  de error
echo((kl- 1)*Num_iter+k2)=yd;      %señal de eco

%Adaptación
%Detección de double talk
psa=psig;
pxa=pxin;
psig=0.99*psa+0.01*(yd*yd);
pxin=0.99*pxa+0.01*(x*x);
erl=psig/pxin;
if erl<erlop
alfa=0.10;              %factor de convergencia
else
alfa=0.001;           %factor de convergencia
end
miu=alfa/(Y*Y');
W=W+miu*error*Y;

% FIR adaptivo 2
for k=NI: -1:2
Yl1(k) =YI1(k- 1);               %recorrer  línea de retardo
end
```

```
Yl1(1) =xpr;                    %cargar nuevo dato
YI0=YI1l*WI;                        %cálculo de la señal de salida


%Calculo del error
errorI=yHIl-YI0;
xst((k1-1)*Num_iter+k2)=errorIl;    %señal de entrada
xfes((k1-1)*Num_iter+k2)=yI:    %señal de entrada

%Adaptación
miu=0.01/(YI1*YIl');
WI=WI+miu*errorI*YI1;

%Promedios

pot_sig=pot_sig+(y1*yl);
pot_ err=pot_err+((yl-y0)*(y1-y0));
pot_in=pot_in+(x*x);
pot_ errl=pot_ errl+((yI-errorl)*(yI-errorI));
pot_yI=pot_yI+(yHIl)*(yHI1);
end
MSE(kl)=10.0*LOG10(pot_err,);
PS(kl)=10.0*LOG10(pot_sig);
MSEI(kl)=10*LOG10(pot_yI);
End
Fig2
figure(2);
%subplot(2,1,1); plot(MSE);
%subplot(2,1,2); plot(PS,);
X_H=1:Num_block;
plot(X_H,MSE,X_H,PS);
NF=512;
NF1=-256;
wh=hamming(NF);
[Pe,F]=spectrum(resec,NF,NFI1,wh,fr)
[Pr,F]=spectrum(echo,NF,NF1,wh,fr);
Pedb(:,1)=20*log10(Pe(:,1));
Prdb(:,1)=20*Iog10(Pr(:,l));
figure(3);
%subplot(2,1,1); plot(F,Pedb(:,1));
%subplot(2,1,2); plot(F,Prdb(:,1));
plot(F,Pedb(:, l):plot(F,Prdb(:,1));
figure(4)
plot(X_H,MSEI,X_H,PSHI);
figure(5)
subplot(2,1,l); plot(xst);
subplot(2,1,2); plot(xfes);
```

```
figure(6)
subplot(2, 1, 1); plot(echo);
subplot(2,1,2); plot(resec);
```

B.4 Programa que genera las figuras que muestran el espectro de densidad de potencia (pdf.m):

```
clear;

%Captura de la señal de entrada
[yin,fr]=wavread('vozl');
NO=length(yin)
xmax=max(yin)
yin=yin(2000:NO);
NO=length(yin)
xav=sum(yin)/NO
xmax=max(yin);
for k=1:NO
yin(k) =(yin(k) -xav) /xmax;
end
plot(yin)
pause
NF=512;
NF1=256;
wh=hamming(NF);
xiw=zeros(1,NF) +i*zeros(1, NF);
for k=1:NF
xic(k)=yin(k);
end
for k=1:NF1
xiw(k)=xic(k)*wh(k)+i*0;
end
xf=fft(xiw,NF)
NF1=fix(NF/2);
for k= I:NF1
xf2d(k,1)=xf(k);
end
NO1=fix((NO-NF) / NF 1);
for kl=l:NO1
for k= 1:NF1
xic(k)=xic(NF1+k);
xic(NF1+k)=yin(NF1*kl+k)+i*0;
end
for k= 1: NF
```

Capítulo 7

Microprocesador TMS320C30

7.1. Introducción

En este capítulo se presenta la estructura básica del microprocesador TMS320C30. Este microprocesador forma parte de la tercera generación de la familia TMS320 de procesadores digitales de señales (DSP) desarrollados con tecnología 1μm CMOS, capaces de manejar operaciones con punto flotante de 32 bits. Se fabrica en un encapsulado de tipo PGA (*pin grid array*) de 181 pins.

El ciclo de reloj es de 60 ns, palabra de datos y programa de 32 bits, con *bus* de direcciones de 24 bits, lo que resulta en un espacio de memoria de 2^{24} = 16,7 megapalabras. Ejecuta instrucciones a una tasa de 33.3 MFLOPS [1] y 16.7 MIPS [2]. Funciona con un reloj externo de 66 MHz.

Otra característica importante de un DSP, con respecto a los demás microprocesadores, es la multiplicación en paralelo y carga al *acumulador*, todo esto en un solo ciclo (instrucción tipo MAC, *multiply and accumulate*).

Este microprocesador posee un conjunto de registros, algunos de propósito general denominados *archivos de registros*, que son utilizados por la CPU, *unidad central de procesamiento*, para llevar el control de sus operaciones. La CPU tiene un depósito de 64 palabras de longitud empleado por la memoria de programa. Para realizar operaciones aritméticas, usa dos unidades aritméticas que trabajan en paralelo, cada una de las cuales posee un registro auxiliar asociado (ARAU0 y ARAU1). Los bloques de memoria internos son de acceso dual (pueden accesar dos operandos en un ciclo de máquina). Existe un canal destinado para DMA (*acceso directo a memoria*) que disminuye la carga a la CPU, el cual soporta operaciones de entrada/salida concurrente.

En cuanto a los periféricos desarrollados dentro del circuito integrado, se consideran dos temporizadores y dos puertos seriales independientes, los cuales debido a su arquitectura basada en registros, facilitan la implantación de compiladores de alto nivel, como es el caso del lenguaje C. El bloque de memoria ROM es de 4k-palabras y existen además dos bloques de memoria RAM de 1k. La unidad aritmético-lógica y el multiplicador son de 40bits para punto flotante y de 32 bits para números enteros. La CPU tiene un registro de corrimiento de hasta

[1] MFLOPS = millones de operaciones de punto flotante por segundo.
[2] MIPS = millones de instrucciones por segundo.

32 bits, ocho registros o acumuladores de precisión extendida, así como dos generadores de direcciones con ocho registros auxiliares. Para las unidades aritméticas se tienen dos registros auxiliares. Asimismo, la CPU puede ejecutar instrucciones de dos o tres operandos, y posee dos banderas externas de propósito general y cuatro interrupciones externas. El controlador de DMA se utiliza para leer o escribir en la memoria sin que se interrumpa a la CPU.

7.2. Arquitectura

La arquitectura interna del TMS320C30 se estructuró con la finalidad de poder ejecutar grandes volúmenes de operaciones aritméticas, utilizando tanto hardware como software. Posee una extensa memoria interna y un alto grado de paralelismo en las operaciones, lográndose así una reducción significativa en el tiempo de ejecución de un bloque de instrucciones.

7.3. Unidad central de procesamiento (CPU)

La administración de los recursos de la CPU se realiza a través de registros agrupados en un archivo de registros que contiene los siguientes componentes:

1. Multiplicador de números enteros y de punto flotante.
2. Unidad aritmético-lógica (ALU), la cual ejecuta operaciones de punto flotante, de enteros y operaciones lógicas.
3. Registro de corrimiento de hasta 32 bits.
4. *Buses* internos (CPU1/CPU2 y REG1/REG2).
5. Unidades aritméticas con registros auxiliares asociados (ARAU).
6. Conjunto de 28 registros con diversos propósitos (*register file*).

Multiplicador de números enteros y de punto flotante

Ejecuta instrucciones con una duración de un ciclo de máquina de la siguiente manera:

- Multiplicación entera de 24 bits con resultado de 32 bits.
- Multiplicación de punto flotante de 32 bits con resultado de 40 bits.

Unidad aritmético-lógica (ALU)

Aquí se ejecutan operaciones con una duración de un ciclo de máquina para datos enteros y lógicos de 32 bits, y datos de punto flotante de 40 bits. La extensión de la palabra de datos que resulta tiene la misma extensión que los operandos.

Registro de corrimiento de 32 bits

Es utilizado para realizar desplazamientos de hasta 32 bits hacia la izquierda o a la derecha en un solo ciclo.

Buses internos CPU1/CPU2 y REG1/REG2

La concepción separada de estos buses permite extraer dos operandos de memoria y dos operandos desde los registros para que se realicen multiplicaciones paralelas, así como sumas y restas de cuatro operandos en un solo ciclo de máquina.

Unidades aritméticas con registros auxiliares asociados ARAU0 y ARAU1

Es posible generar dos direcciones en un solo ciclo que opera en paralelo con el multiplicador y la ALU, esto permite usar un modo de direccionamiento indexado, circular y de inversión de bit.

7.3.1. Archivo de registros de la CPU

Éstos se pueden utilizar como registros de propósito general, aunque también cada uno de ellos realiza una función específica; por ejemplo, se tienen ocho registros de precisión extendida y ocho registros auxiliares, entre otros, que se presentan a continuación.

Registros de precisión extendida R0-R7

Almacenan operandos y realizan operaciones con enteros de 32 bits y números de punto flotante de 40 bits. Se asume que para operaciones con enteros, usan los 32 bits menos significativos y en operaciones de punto flotante, 40 bits.

Registros auxiliares AR0-AR7

Se accesan por la CPU y por la ARAU. Su función principal es la generación de direcciones de 24 bits, pero se pueden usar también como contadores de cuenta cerrada o como registros de propósito general.

Apuntador a página de datos DP

Registro de 32 bits, donde los ocho menos significativos se utilizan para direccionamiento directo como un apuntador a la página de datos direccionada. Las páginas son de 64 k-palabras, teniéndose un total de 256 páginas.

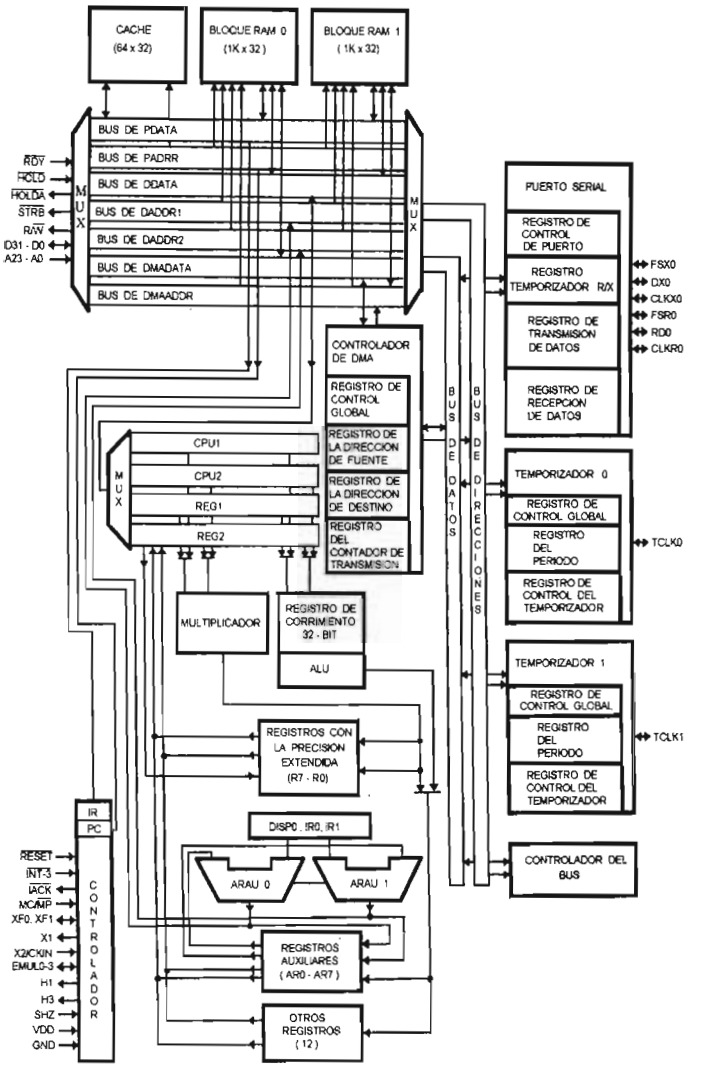


Figura 7.1: Arquitectura del microprocesador TMS320C30

Nombre del registro	*Función*
R0	Registro de precisión extendida 0
R1	Registro de precisión extendida 1
R2	Registro de precisión extendida 2
R3	Registro de precisión extendida 3
R4	Registro de precisión extendida 4
R5	Registro de precisión extendida 5
R6	Registro de precisión extendida 6
R7	Registro de precisión extendida 7
AR0	Registro auxiliar 0
AR1	Registro auxiliar 1
AR2	Registro auxiliar 2
AR3	Registro auxiliar 3
AR4	Registro auxiliar 4
AR5	Registro auxiliar 5
AR6	Registro auxiliar 6
AR7	Registro auxiliar 7
DP	Apuntador a página de datos
IR0	Registro índice 0
IR1	Registro índice 1
BK	Registro de tamaño de bloque
SP	Apuntador al paquete
ST	Registro de estado
IE	Registro de habilitación de int. CPU/DMA
IF	Registro de banderas de int. del CPU
IOF	Registro de banderas de E/S
RS	Registro de dir. inicial de repetición
RE	Registro de dir. final de repetición
RC	Contador de repetición
PC	Contador de programa

Cuadro 7.1: Registros del TMS320C30

Registros de índice IR0, IR1

Contienen el valor usado por la correspondiente ARAU para calcular una dirección indexada.

Registros de tamaño de bloque BK

Son utilizados por la ARAU para especificar el tamaño del bloque en el direccionamiento circular.

Apuntador al paquete del sistema SP

Registro de 32 bits que contiene la dirección guardada en la localidad más alta del paquete. Este registro siempre apunta al último elemento que entra. El paquete es intervenido por interrupciones, llamadas y retornos de subrutinas, y por las instrucciones **PUSH** y **POP**.

Registro de estado ST

Guarda información global sobre el estado de la CPU, es decir, contiene las banderas que indican si el resultado de una operación fue cero o negativo. Mediante software, es posible salvar y restaurar este registro.

Registro de habilitación de interrupciones de CPU/DMA IE

Registro de 32 bits donde un "1"en la posición adecuada habilita la interrupción correspondiente. Las interrupciones para DMA están en los bits 26 a 16 de este registro. Un "0"deshabilita la interrupción.

Registro de banderas de interrupciones de CPU IF

Registro de 32 bits en el cual, de la misma manera que el anterior, con un "1" habilita y con un "0" deshabilita la interrupción correspondiente.

Registro de banderas de E/S IOF

Controla los pines externos XF0 y XF1, los cuales pueden ser configurados como entradas o salidas.

Registro de dirección inicial de repetición RS

Cuando el procesador se encuentra operando en el modo de repetición, este registro contiene la dirección inicial del bloque de instrucciones que van a ser repetidas.

Registro de direccionamiento final de repetición RE

Contiene la última dirección del bloque de programa que va a ser repetido.

Contador de repetición RC

Registro de 32 bits que especifica el número de veces que un bloque de código se repetirá.

Contador de programa PC

Registro de 32 bits que contiene la dirección de la siguiente instrucción que será accesada por la CPU. Aunque el PC no es parte de los registros de la CPU, sí es un registro que puede ser modificado por instrucciones que alteran el flujo del programa.

7.4. Organización de la memoria

El espacio total de memoria del procesador definido por un bus de direcciones de 24 bits da por resultado 16M-palabras. El programa, datos y espacio de E/S están contenidos dentro de este espacio de 16M-palabras, lo cual permite que tablas, coeficientes, código de programa o datos puedan estar guardados, ya sea en RAM o en ROM.

Memorias RAM, ROM y depósito

El procesador tiene dos bloques de memoria RAM (bloque 0 y bloque 1), cada uno de ellos es de 1k-palabras, y un bloque de ROM de 4k-palabras. Cada bloque permite dos accesos en un solo ciclo de máquina.

Dado que los buses de programa (PADDR, PDATA), de datos (DADDR1, DADDR2) y de DMA (DMAADDR, DMADATA) están separados, pueden realizarse paralelamente lecturas y escrituras de datos, acceso al código del programa y operaciones de DMA. Existe un depósito de instrucciones de 64 palabras que almacena las secciones de código más utilizadas, reduciendo así el número necesario de accesos.

Mapa de memoria

La configuración del mapa de memoria depende del estado del pin $MC/\overline{MP}$.

En el *modo de microcomputadora* ($MC/\overline{MP} = 1$), el espacio en ROM de 4k-palabras está mapeado de las localidades 0h a la 0FFFh; las primeras 192 localidades se destinan a vectores de interrupción, vectores de "trampa", y a un espacio reservado. Las localidades de la 1000h a la 7FFFFFh son para memoria externa, cuando está activo el pin $\overline{STRB}$.

En el *modo de microprocesador* ($MC/\overline{MP} = 0$) no existe el bloque de 4k de ROM. Las primeras 192 localidades están destinadas a interrupciones, trampas y algunas localidades reservadas. En este espacio se accesa a memoria externa cuando está activo el pin $\overline{STRB}$. A partir de la localidad C0h, el espacio de memoria restante corresponde a memoria externa.

Para ambos modos de operación, microcomputadora y microprocesador, las localidades 800000h a la 801FFFh están mapeadas al bus de expansión cuando se activa el pin $\overline{MSTRB}$. Los registros de control de los periféricos se encuentran entre las localidades 808000h a la 8097FFh para ambos modos de operación. El bloque 0 de RAM está mapeado de la 809800h a la 809BFFh y el bloque 1 de la 809C00h a la 809FFFh, y las localidades 80A000h a la 0FFFFFFh están destinadas a memoria externa.

Modos de direccionamiento

Este microprocesador soporta seis tipos de direccionamiento:

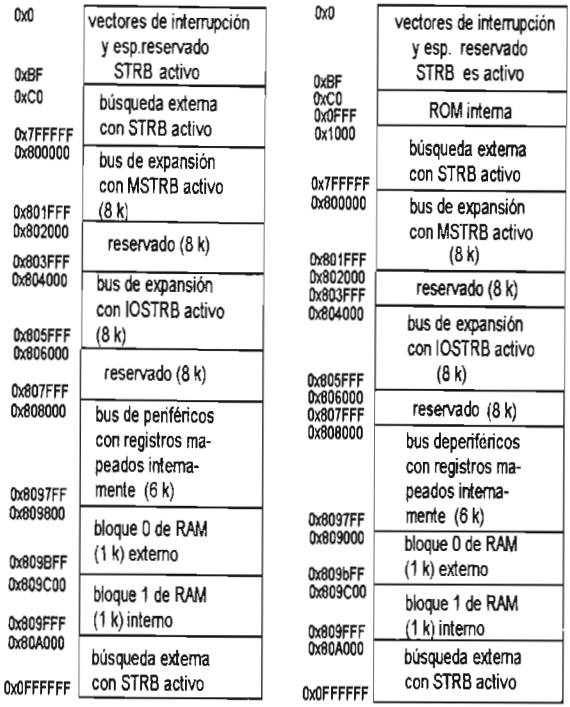


Figura 7.2: Mapa de memoria

- *Por registro:* el operando es siempre un registro de la CPU, pudiendo ser un registro de precisión extendida.
- *Corto-inmediato:* el operando es un valor inmediato de 16 bits.
- *Largo-inmediato:* el operando es un valor inmediato de 24 bits.
- *Directo:* el operando es el contenido de una dirección de 16 bits.
- *Indirecto:* un registro auxiliar indica la dirección del operando.
- *Relativo al PC:* se suma al PC un desplazamiento signado de 16 bits.

7.5. Operación del bus interno

El contador de programa (PC) se encuentra conectado al bus de direcciones del programa, y el registro de instrucción (IR) al bus de datos de programa; ambos buses pueden accesar sólo una palabra cada ciclo de máquina. En cuanto al bus de direcciones de datos, éste permite dos accesos en un ciclo de máquina.

El controlador de DMA posee sus propios buses de direcciones (DMAADDR) y de datos (DMADATA), lo cual permite al controlador accesar a memoria en paralelo con los accesos a memoria de los buses de programa y datos.

7.6. Operación del bus externo

El bus externo está constituido por un bus primario de direcciones de 24 bits y por uno de expansión de 13 bits. Este bus accesa memoria de programa, datos y espacio de E/S. Se utiliza una señal externa $\overline{RDY}$ para generar estados de espera, cuando se accesan memorias o dispositivos lentos.

Interrupciones externas

La CPU soporta cuatro interrupciones externas ($\overline{INT3} - \overline{INT0}$). Maneja también interrupciones internas y una señal externa de $\overline{RESET}$ no protegida. Estas señales pueden interrumpir, ya sea a la CPU o al controlador de DMA. Cuando la CPU responde a la interrupción, el pin $\overline{IACK}$ puede utilizarse como una contraseña externa a la interrupción.

Señalización para ejecución de instrucciones sincronizadas

La CPU utiliza dos banderas externas XF0 y XF1 configurables mediante software como entradas o salidas. Se utilizan por las operaciones de sincronización del procesador, las cuales soportan comunicación entre microprocesadores, así como también es posible establecer tiempos de espera, entre otras aplicaciones.

7.7. Control de periféricos

Los periféricos incluidos en el circuito integrado son dos puertos seriales y dos temporizadores. El control de ellos se realiza a través de registros mapeados en memoria, para lo cual se utiliza un bus periférico específico. Dichos registros de control se encuentran en las localidades 808000h hasta la 8097FFh.

Temporizadores

Los dos módulos de temporización de 32 bits pueden funcionar como temporizadores, o bien, como contadores de eventos con dos modos de señalización y una señal de reloj generada, ya sea interna o externamente. Asociado a cada módulo, hay un pin que puede ser configurado como entrada de reloj para el temporizador o como una salida controlada por el mismo procesador.

Puertos seriales

Ambos puertos son completamente independientes uno del otro, pero idénticos en su configuración de registros y también configurables para operar con palabras de datos de 8, 16, 24 o 32 bits.

La señal de reloj para cada puerto serial puede ser generada en forma interna o externa, contándose con un divisor de frecuencia.

7.8. Acceso directo a memoria (DMA)

El controlador de DMA, como ya se mencionó, puede leer o escribir en la memoria sin interferir con la operación de la CPU. Esto es de gran utilidad cuando se interactúa con dispositivos externos de baja velocidad, evitando disminuir el desempeño de la CPU. El controlador de DMA contiene sus propios generadores de direcciones, registro fuente y destino, y contadores de transferencia. Dado que el controlador posee sus propios buses de datos y direcciones, se minimizan los conflictos con la CPU. Una operación de DMA consiste en la transferencia de una palabra o un bloque de datos desde o hacia la memoria.

7.9. Operaciones con ductería (*pipeline*)

El microprocesador TMS320C30, al accesar una instrucción de memoria y proceder a ejecutarla, realiza básicamente cuatro operaciones, las cuales son ejecutadas en el modo ductería como sigue:

Unidad de búsqueda (FETCH) (F)

Obtiene las palabras de instrucción de memoria y actualiza el contador de programa.

Unidad de decodificación (D)

Realiza la decodificación de la instrucción y genera una dirección; también controla las modificaciones realizadas a los registros auxiliares y al apuntador del apilado.

Unidad de lectura (R)

En caso de ser necesario, realiza lectura de operandos.

Unidad de ejecución (E)

De requerirse, esta unidad lee los operandos del archivo de registros, ejecuta la operación necesaria y actualiza el archivo de registros. En ocasiones también escribe resultados previos a memoria.

Cuando estas cuatro operaciones se superponen perfectamente en el mismo ciclo de ductería, operando en paralelo, la potencialidad del modo ductería se utiliza plenamente (figura 7.3). Para el programador, la ejecución de instrucciones en ductería es transparente, sin embargo, éste puede buscar que el acomodo de ellas favorezca a que el programa se realice óptimamente.

Durante la ejecución de operaciones en ductería pueden generarse conflictos debidos a los diferentes accesos que realiza una instrucción en particular, o bien, al uso del apilado que pueda realizar, entre otras causas.

Prioridad de las unidades de ductería

Se asigna una prioridad a las operaciones, esto es, el orden en que se llevan a cabo:

- Ejecución (E)
- Lectura (R)
- Decodificación (D)
- Búsqueda (F)
- Canal de DMA (en caso de presentarse)

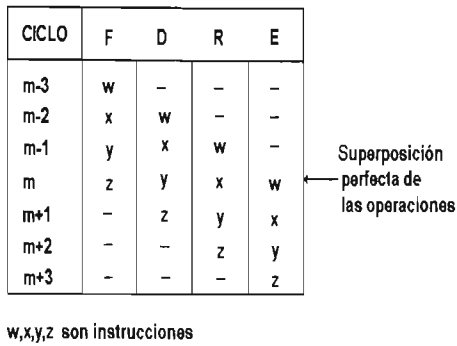


Figura 7.3: Ductería

Cuando una instrucción está lista para entrar al siguiente nivel de ductería, pero ese nivel no está listo para aceptar a dicha instrucción se presenta un conflicto. En este caso, la unidad de más baja prioridad espera hasta que la siguiente unidad en prioridad complete su ejecución.

Los conflictos entre las operaciones de DMA y la estructura de ductería se minimizan debido a que el controlador de DMA posee sus propios buses de datos y direcciones.

Conflictos en ductería

Pueden agruparse en tres categorías principales:

1. *Conflictos de saltos.* Se generan cuando alguna instrucción lee o modifica el contador de programa (PC).
2. *Conflictos con registros.* Involucran retardos generados al leer o escribir los registros cuando sean usados para generar direcciones.
3. *Conflictos con memoria.* Ocurren cuando las diferentes unidades del microprocesador compiten por los recursos de memoria.

7.10. Uso de los recursos del sistema

El microprocesador soporta aritmética entera y de punto flotante, lo que permite al programador concentrarse en el algoritmo que debe programar y preocuparse lo menos posible de problemas como escalamiento, rango dinámico o sobreflujos.

Al efectuarse un *reset*, el microprocesador finaliza la ejecución del programa en curso y coloca el valor del vector de reset en el contador de programa. El reset por hardware (pin externo) también inicializa varios registros y bits de estado.

Después del reset deben inicializarse modos de operación, apuntadores de memoria, interrupciones y especificaciones propias de la aplicación. Dentro de la configuración inicial, deben incluirse las primeras instrucciones a los registros mapeados en memoria y en la estructura de interrupciones.

Control de flujo dentro de un programa

El modelo de programación de este microprocesador posee diversas estructuras que facilitan la programación de algoritmos de procesamiento digital de señales, como son:

- Llamadas a subrutinas (llamadas, trampas y retornos).
- Uso del apilado por software.
- Saltos retardados.
- Operaciones en modo multiprocesador.
- Interrupciones.
- Modo de repetición.

Llamadas a subrutinas (llamadas, trampas y retornos)

Una vez diseñada la subrutina correspondiente, ésta se ejecuta mediante las instrucciones **CALL, CALLcond** y **TRAPcond**. A través de las instrucciones **RETScond** y **RETIcond** se regresa de la subrutina invocada, lo que restaura automáticamente el apilado (*stack*).

Uso del apilado por software

El microprocesador tiene un apilado controlado por software, cuya localización se determina por el contenido del apuntador al apilado (SP). Dicho apilado se accesa no solamente por las instrucciones **CALL** y **RETS**, sino también como una forma de almacenamiento temporal con las instrucciones **PUSH** y **POP** para números enteros, y **PUSHF** y **POPF** para números de punto flotante. El apuntador del apilado no se inicializa por hardware durante el reset, por lo que es importante asignarle un valor con una dirección de memoria predeterminada.

Saltos retardados

Este tipo de saltos funcionan como los saltos normales, sin embargo, no omiten el uso de la estructura de ductería, por lo que es conveniente procurar utilizarlos. Cuando un salto retardado es encontrado dentro de un programa, las tres instrucciones siguientes son ejecutadas. Las restricciones, en este caso, son que ninguna de estas tres instrucciones sean un salto, llamada a subrutina, retorno de subrutina o de interrupción, instrucción de repetición, instrucción trampa o instrucción de espera.

Operaciones en modo multiprocesador

Estas operaciones, auxiliadas mediante señalización externa, garantizan la integridad de la comunicación e incrementan la velocidad de ejecución.

Para mantener el acceso a una memoria global y compartir datos de una manera coherente, es necesario un protocolo para arbitrar este tipo de procesamiento. Este es el propósito de un pequeño conjunto de instrucciones que ayudan a la sincronización (**LDFI, LDII, SIGI, STFI, STII**).

Interrupciones

Se encuentran en forma vectorizada y existe un orden de prioridad entre ellas. Cuando una interrupción ocurre, se activa el correspondiente bit en el registro de banderas de interrupción (IF). Si en el registro de habilitación de interrupciones (IE), el bit correspondiente se encuentra activo, y las interrupciones a su vez se han habilitado por el bit GIE en el registro de estado, se ejecuta la rutina de interrupción respectiva. Existe también la posibilidad de escribir en el IE y de esta manera forzar la interrupción por software, o bien, deshabilitarla para que no se lleve a cabo.

Exceptuando rutinas de interrupción muy simples, es importante asegurar que el contexto del procesador (es decir, el archivo de registros) se haya respaldado previamente al dar el salto a la subrutina. Este conjunto de registros debe almacenarse en el apilado con anterioridad al brinco de subrutina y recuperarse de manera inversa (el apilado puede verse como una memoria LIFO), conservándose entonces el contexto original.

Modo de repetición

Se consideran dos modos de repetición: *repetición de un bloque de código* (**RPTB**) y *repetición de una sola instrucción* (**RPTS**). El control del número de repeticiones, así como las direcciones inicial y final del bloque que se van a repetir, se realizan manipulando los registros RS, RE y RC.

Capítulo 8

Herramienta de desarrollo EVM

8.1. Introducción

En este capítulo se describe brevemente la herramienta de desarrollo para ejecutar algunos de los algoritmos que se proponen más adelante. El *módulo de evaluación* (EVM) es una tarjeta diseñada para usarse con el software depurador de código C (*C source debugger*), el cual ofrece al usuario la posibilidad de realizar una emulación en pantalla, a la vez que los algoritmos se ejecutan en la tarjeta. La intervención del *controlador de bus de prueba* (TBC *test bus controller*) permite realizar la emulación directa en la tarjeta, parar la ejecución del programa, y visualizar registros y memoria. Mediante el uso de un puerto de emulación MPSD y el TBC se carga el programa en el EVM.

Se define un protocolo de comunicación entre el *host* (una computadora IBM PC/AT compatible) y la tarjeta objetivo, es decir, la EVM. Dicho protocolo basa su operación en el intercambio de información entre registros mapeados en memoria, lo que permite desarrollar software de propósito específico para una aplicación dada, utilizando un lenguaje de alto nivel para el acceso a memoria, como lo es el lenguaje C.

El programa debe estar previamente formateado, según la especificación COFF *(common object file format)*, para generar el código objeto a partir de segmentos de programa escritos, tanto en ensamblador como en ANSI C. De este código objeto se genera el código ejecutable en un archivo **.out**, que debe cargarse en la tarjeta ya sea mediante el programa **evmload.exe**, o bien, con el ambiente del depurador.

8.2. Características generales

Interfaz con el bus anfitrión

La *tarjeta emuladora* tiene las especificaciones necesarias para insertarse en una ranura libre del bus IBM PC/AT de tarjetas de 8 bits. En la figura 8.1 se esquematiza la tarjeta EVM y sus componentes principales.

La diferencia más notoria sobre otros emuladores existentes consiste en lo que sus diseñadores denominan "emulación en el sistema mismo "(*embedded in-system emulation*). Esto significa que la emulación se efectúa dentro del sistema objetivo (EVM), realizándose la interfaz con

el host mediante un circuito destinado a ello: el 74ACT8990, (TBC). Dicho circuito controla el intercambio de información entre el TMS320C30 y el bus de la computadora anfitriona.

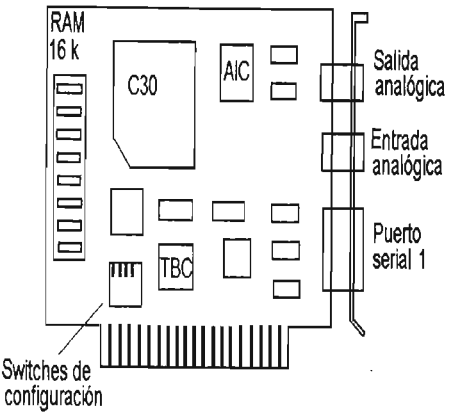


Figura 8.1: Módulo de evaluación (EVM) para el TMS320C30

El programa, que se ejecuta desde la memoria de la computadora personal, se comunica en realidad con el TBC a través del poleo (monitoreo) de seis direcciones de 16 bits. El TBC interactúa con el TMS320C30 mediante interrupciones por hardware por medio de los pines INT0, INT1 e INT2. Esto quiere decir que la comunicación desde un programa anfitrión en la computadora opera con poleo, mientras que el TMS320C30 se comunica con el TBC a través de interrupciones de hardware, lo que le proporciona a este último total autonomía del anfitrión mientras no se realice una requisición de intercambio de información.

La configuración del sistema, como se ha referido, provee una tasa moderada de intercambio de información (alrededor de 200 kbytes por segundo), sin embargo, garantiza la sincronización de dicha transferencia, ya que si un dato tiene que ser escrito o leído por el host (PC) es necesario realizar un poleo y borrado de banderas en los registros mapeados en memoria. En ciertas condiciones, es posible evitar dicho poleo, que eleva la eficiencia (*throughput*) [1] hasta 400 kbytes por segundo.

Como hardware adicional en la generación de señales de protocolo (*glue logic*) se incluyen un par de PAL (*dispositivos lógicos programables de uso generalizado*). y dos transreceptores 74ALS652. El formato de interfaz entre el TBC y el puerto de emulación del TMS320C30 sigue las especificaciones MPSD (*modular port scan device*) de Texas Instruments, *User's Guide*.

[1] *Throughput* es un término generalmente aceptado para referir la cantidad de información digital que puede obtenerse como salida de un sistema, se utiliza como un parámetro de desempeño.

Memoria externa

El módulo de evaluación posee un banco de memoria SRAM de 16k-palabras con cero estados de espera y se halla directamente conectado al espacio de direcciones del bus primario. Las memorias utilizadas son de la marca Cypress, modelo CY7C164-35VC, con un tiempo de acceso de 35 ns, lo que requiere un reloj de 30 MHz como base de tiempo en el C30 para satisfacer los requerimientos de la SRAM. Este banco de memoria provee al módulo de cierta autonomía en cuanto a capacidad de almacenamiento se refiere. Es posible almacenar aproximadamente dos segundos de una señal muestreada a 8 kHz sin ser necesario el intercambio de información con el host.

Interfaz analógica

De los dos puertos seriales de que dispone el TMS320C30, el primero de ellos (puerto 0) se ha utilizado para comunicarse con un controlador de interfaz analógica: el TLC32044, *analog interface controller* (AIC). Dicha interfaz necesita de una base de tiempo, por lo que se utilizó para ello el primer temporizador, además del pin XF0 como $\overline{RESET}$ del AIC. Este controlador provee de una interfaz A/D y D/A con 14 bits de resolución, tasa de muestreo y filtrado programables, que cubren satisfactoriamente los requerimientos de una arquitectura para procesar señales de voz.

La entrada analógica acepta voltajes en un rango de $\pm 1{,}5$ V y de ± 3 V. Se incluye también una etapa de acondicionamiento de la señal proveniente del exterior que consta de un seguidor y un amplificador con ganancia 2, basados en amplificadores TL072 con entrada JFET. La salida analógica acepta una carga mínima de salida de 300Ω, sin embargo, para poder ser manejada por un equipo amplificador comercial, es necesario acoplarla para cargas de hasta 4Ω. Con una salida típica de 1.5 Vpp, el AIC se conecta a una etapa de atenuación con una ganancia de x0.2 para posteriormente ser amplificado por un factor de x20 en el amplificador de potencia LM386, de manera que la ganancia de voltaje resultante sea de x4. Por ejemplo, para una carga de 8Ω con una salida inicial de 1.5 Vpp, el voltaje resultante a la salida del LM386 debe ser de 6 Vpp.

Puerto serial externo

El puerto serial 1, completamente independiente de aquel que se usa como interfaz A/D y D/A, se ha previsto como un medio de comunicación al exterior del EVM. Se incluye además en el conector de salida de 10 pins la señal XF1. Los pins disponibles en dicho conector se presentan en la tabla 8.1.

Pin	*Señal*	*Pin*	*Señal*
1	GND	2	FSR1
3	CLKX1	4	DR1
5	DX1	6	TCLK1
7	FSX1	8	XF1
9	CLKR1	10	GND

Cuadro 8.1: Señales de puerto serial

Los usos que pueden darse a dicho conector son diversos: conexión con otras tarjetas EVM para ejecutar procesamiento distribuido, conexión a otro dispositivo serial para recibir información, de un PBX, (*private branch exchange*).

8.3. Configuración y generación de código

Configuración del mapa de memoria de la tarjeta

Es posible configurar algunas características de la tarjeta mediante microinterruptores. Como se muestra en la tabla 8.2, con los interruptores marcados SW1, SW2, se selecciona la dirección base de los registros utilizados por el programa de aplicación en el host para la comunicación hacia la tarjeta EVM. La primera dirección base debe ser la opción por *default*, a menos que exista otro periférico (por ejemplo, un cursor) en ese espacio de memoria.

SW1	SW2	*Dirección base*
ON	ON	$0 \times 0240 - 0 \times 025F$
ON	OFF	$0 \times 0280 - 0 \times 029F$
OFF	ON	$0 \times 0320 - 0 \times 033F$
OFF	OFF	$0 \times 0340 - 0 \times 035F$

Cuadro 8.2: Microinterruptores en EVM

Los interruptores SW3 y SW4, por lo general, se mantienen en ON para permitir el modo de microprocesador y la salida SBM en alta impedancia, respectivamente.

Generación de código

Una vez que se ha compilado o ensamblado cualquier programa para el TMS320C30 se genera un archivo **.obj**, el cual contiene ya el código de máquina en formato COFF que es posible ejecutar en un sistema basado en este microprocesador.

Ensamblador

Si se trata de un programa escrito exclusivamente en lenguaje ensamblador, se utiliza el programa **asm30.exe** como sigue:

asm30 [**archivo de entrada** [**archivo objeto** [**archivo de lista**]]] [**-opciones**]

Las opciones más comunes son:

-v especifica la versión, puede ser -v30 para el TMS320C30 o -v40 para el TMS320C40.
-l produce un archivo de listado (**.LST**).
-i especifica un directorio para búsqueda de archivos especificados en las directivas **.copy**, **.include** o **.mlib**.
-c realiza la compilación con sensibilidad a la mayúsculas.

-x produce una tabla de referencias cruzadas al final del archivo de listado.

Una opción es invocar al programa sin ningún argumento, lo que ocasiona que el programa pida entonces los argumentos al usuario.

Ligador

Es necesaria una última etapa de ligado en la cual se asignan las direcciones definitivas para que el código ejecutable sea localizado dentro del mapa de memoria del sistema objetivo. En este caso, dicho sistema objetivo es la tarjeta EVM, y es necesario realizar esta etapa de ligado con información adicional sobre la localización de los datos, del código de programa, memorias RAM y ROM, etc. Tal acción se ejecuta con el programa **lnk30.exe**, cuya sintaxis es la siguiente:

lnk30 [-opciones] archivo 1 ... archivo n

Las opciones más usuales son:

-m produce un archivo de mapeo (**.MAP**)
-o especifica a continuación el archivo de salida (por default **A.OUT**)
-q al momento de ligar no aparece el rótulo de TI
-x obliga a que relea las librerías y detecte referencias no encontradas en la primera pasada
-a produce código con direcciones absolutas
-r produce código con direcciones relocalizables

Un ejemplo común de ligado es el siguiente:

lnk30 archivo, archivo de comandos, -o archivo de salida

El ligador también tiene la opción de ser invocado sin argumentos, lo cual ocasiona que dicho programa pida los datos necesarios al usuario.

Archivos de comandos

Para que el ligador conozca la localización de la memoria dentro de un sistema objetivo, es necesario especificarle tal información en un archivo de texto con extensión **.cmd** (archivo de comandos). En dicho archivo de comandos se especifica la naturaleza de los bloques de memoria usados (lectura y escritura, sólo lectura), así como su localización y extensión dentro del mapa de memoria. A cada bloque se le da un nombre y a su vez se localizan en ellos las diferentes secciones del código. La división en secciones del código al momento de ligar obedece a que tal código puede tener diferentes características, siendo en algunas ocasiones código de programa, una tabla de datos fija o una tabla de datos variable.

En el formato **COFF** se definen tres secciones por default:

sección .text contiene código ejecutable.
sección .data contiene datos inicializados o invariables.

sección .bss es un espacio reservado para variables no inicializadas (por ejemplo, los stados internos de un filtro).

Además, el ligador permite que el usuario pueda crear otras secciones con características similares, y básicamente las clasifica en dos categorías: inicializadas y no inicializadas.

Secciones inicializadas

Contienen datos o código ejecutable: **.text** y **.data** son secciones inicializadas. El usuario puede crear nuevas secciones inicializadas con las directivas **.sect** y **.asect**.

Secciones no inicializadas

Se utilizan para reservar espacio en memoria para datos no inicializados. La sección **.bss** es de este tipo y el usuario puede definir nuevas secciones con la directiva **.usect**. Pueden encontrarse varios ejemplos de secciones no inicializadas creadas por el usuario con los nombres "**stack** ", "**vectors** " y "**comdata** ".

A continuación, se presenta el contenido de un archivo de comandos utilizado para ligar el código de inicialización listado en la sección anterior.

```
MEMORY
{
INT_V : org = 0x000000, len = 0x40
SRAM  : org = 0x000040, len = 0x1FC0
a     : org = 0x002000, len = 0x2000
RAM0  : org = 0x809800, len = 0x400
RAM1  : org = 0x809C00, len = 0x400
}
SECTIONS
{
 vectors: {} > INT_V
 comdata: {} > SRAM
.text   : {} > SRAM
 buffer : {} > a
 stack  : {} > RAM1
}
```

8.4. Comunicación entre host y EVM

La microcomputadora anfitriona. es decir, el host, tiene comunicación con EVM dentro de un esquema maestro-esclavo, donde obviamente el host ejecuta el papel de maestro. Como ya se ha mencionado, la interfaz entre ambos la realiza el circuito TBC, el cual es direccionado desde el bus AT mediante los PAL que actúan como decodificadores de memoria, y que también se utilizan entre el TBC y el C30 para sincronizar adecuadamente la producción de las interrupciones con la base de tiempo propia del microprocesador C30. Un esquema de la interconexión del bus PC/AT, el TBC y el TMS320C30 se muestra en la figura 8.2.

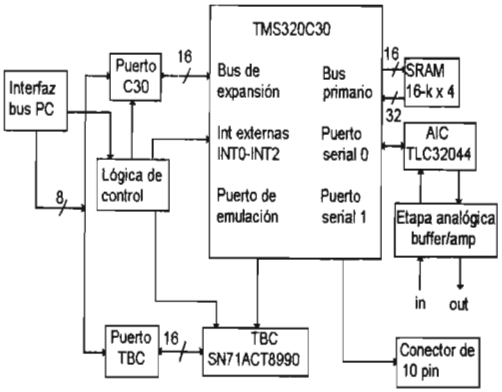


Figura 8.2: Interconexión entre el bus anfitrión, el TBC y el TMS320C30

Operaciones de lectura/escritura desde el host hacia el TBC

Se direccionan seis registros desde el bus PC/AT, los cuales se localizan física y lógicamente en el TBC, y a través de los cuales un programa de aplicación interactúa con el TMS320C30. Aunque la mayor parte son de 16 bits, uno de ellos en realidad sólo existe como una dirección y al realizarse una escritura, se ejecuta un reset por software (registro SOFT RESET). En la tabla 8.3 se presentan los desplazamientos a partir de una dirección base (véase configuración) de los registros mencionados, así como su mnemónico asociado y su tamaño en bits.

Registro	*Desplazamiento*	*Tamaño (bits)*	*Acceso*
RESERVADO	$0 \times 0000 - 0 \times 0008$	-	-
CONTROL5	$0 \times 000A$	16	L/E
RESERVADO	$0 \times 000C - 0 \times 0012$	-	-
MINOR CMD	0×0014	16	L/E
RESERVADO	$0 \times 0016 - 0 \times 001E$	-	-
STATUS 0	0×0400	16	L
RESERVADO	$0 \times 0402 - 0 \times 041F$	-	-
COM CMD	0×0800	8	L/E
COM DATA	0×0808	16	L/E
SOFT RESET	0×818	0	E

Cuadro 8.3: Registros desde el bus PC/AT

Manejo de eventos a través del TBC

El TBC considera cuatro eventos provenientes del bus PC/AT, los cuales a su vez disparan las correspondientes interrupciones al TMS320C30. Se enumeran de EVT0 a EVT3 y su utilización se describe a continuación.

EVT0: embedded simulation support

Evento no modificable, se utiliza como parte del protocolo entre el TBC y el puerto de emulación del TMS320C30.

EVT1: host read acknowledge

Se activa (EVT1=1) cuando el TMS320C30 escribe al registro de comunicaciones. Entonces el host puede leer el dato de dicho registro y poner en cero esta bandera para que una futura escritura del C30 pueda ser detectada.

EVT2: host write acknowledge

Se activa (EVT2=1) cuando el TMS320C30 efectúa una lectura del registro de comunicaciones. Esto quiere decir que el C30 ha recogido el dato. Acto seguido, el host debe poner de nuevo en cero a esta bandera y puede escribir otra vez.

EVT3: TMS320C30 reset source

Mientras esta bandera sea mantenida en 1, el C30 permanece en estado de reset. Existe un registro cuyo direccionamiento provoca dicho estado.

Escritura de datos

Para realizar la escritura de un dato localizado en una variable en la memoria del host hacia el TMS320C30, se llevan a cabo una serie de pasos, los cuales se ejemplifican en el siguiente segmento de un programa en *Microsoft C*.

```
#define IOBASE      0x0240
#define MINOR_CMD   0x0014
#define COM_DATA    0x0808
#define STATUS0     0x0400
#define MASK1       0x0004
#define MASK2       0x6044

/* la variable 16_bit_data contiene el valor que va a ser escrito */
unsigned short 16_bit_data;

do {

/* 1. borra la bandera EVT2 localizada en el reg. MINOR_CMD */

outpw(IOBASE + MINOR_CMD, MASK1);
```

```
/* 2. escribe el dato en el registro de comunicaciones */

outpw(IOBASE + COM_DATA, 16_bit_data);

/* 3. actualiza el registro de estado del TBC */

outpw(IOBASE + MINOR_CMD, MASK2);

/* 4. verifica que el dato ha sido leido del lado del TMS320C30 */

}while( inpw(IOBASE + STATUS0) & MASK1);

/* actualiza el registro de estado */

outpw(IOBASE + MINOR_CMD, MASK2);

/* 5. una vez abandonado el ciclo anterior se puede empezar de
nuevo desde 1 */
```

Lectura de datos

Para recoger un dato del registro de comunicaciones se debe polear la bandera EVT2 hasta que se indique la presencia de un dato en el registro de comunicaciones, como se muestra en el siguiente segmento de código:

```
#define IOBASE      0x0240
#define MINOR_CMD   0x0014
#define COM_DATA    0x0808
#define STATUS0     0x0400
#define MASK0       0x0002
#define MASK2       0x6044

unsigned short 16_bit_data;

do {

/* actualiza el registro de estado del TBC */

outpw(IOBASE + MINOR_CMD, MASK2);

/* si la bandera EVT1 es activa, se borra, en otro caso
regresa a actualizar el registro de estado */

if (inpw(IOBASE + STATUS0) & MASK0)
outpw(IOBASE + MINOR_CMD, MASK0);
```

```
}while(!(inpw(IOBASE + STATUS0) & MASK0));

/* procede a leer el dato */

16_bit_data = inpw(IOBASE + COM_DATA);
```

Reinicialización por software

La inicialización del TMS320C30 se da de manera automática cuando se le aplica energía por primera vez. Sin embargo, el TBC provee la manera de aplicarle un reset por software, a través del registro CONTROL5, de la siguiente manera:

```
#define IOBASE      0x0240
#define CONTROL5    0x000A
#define RESET_ON    0x0808
#define RESET_OFF   0x0800

outpw(IOBASE + CONTROL5, RESET_ON);

outpw(IOBASE + CONTROL5, RESET_OFF);
```

Existe una tercera manera de reinicializar al TMS320C30 sin la intervención del TBC. Como se había mencionado, el registro SOFT RESET no existe físicamente, pero su direccionamiento provoca que una señal activa en bajo se genere en el PAL UA5, el cual aplica el reset al microprocesador.

```
#define IOBASE      0x0240
#define SOFT_RESET  0x0818

outpw(IOBASE + SOFT_RESET, 0);
```

Capítulo 9

Conclusiones

Un cancelador de eco adaptable, se genera una réplica del eco que existe en un sistema telefónico, el eco es retransmitido a la salida del sistema (near- end). En este proceso se genera eco residual que es usado para calcular los coeficientes del cancelador de eco. Con ello el valor cuadrático medio de error de aproximación sea mínimo.

Si la estimación de eco es calculada con precisión, entonces la señal del extremo cercano se transmitirá sin distorsión y con una cantidad baja de eco en la red. Los canceladores de eco transversales son usados ampliamente, debido a su baja complejidad. En esta tesis se analizaron los algoritmos LMS y NLMS, y se propone un método para aumentar la velocidad de convergencia de los canceladores de eco, empleando algoritmos tipo LMS. Por medio del programa MATLAB se evaluó en desempeño de estos algoritmos, tomando en cuenta su velocidad de convergencia con el estado estable ERLE y su complejidad computacional. El algoritmo LMS es fácil de programar pero, su velocidad de convergencia depende de los valores de dispersión de los eigenvalores de la señal. El LMS converge lentamente con la voz como señal de entrada. Con el uso del algoritmo NLMS el problema de la dispersión es menor y tiene una convergencia mas rápida. Aquí se propone la mejor estructura para la cancelación de eco, la configuración en subbandas para la cancelación de eco acústico y las cancelaciones en el dominio de la frecuencia, estas degrada los valores propios de la matriz de autocorrelación de la señal de entrada. Las simulaciones se realizaron en un orden de 128 con el uso de ruido blanco en voz como señal de entrada. El algoritmo de cancelación de eco converge establemente con el ruido blanco pero, con voz de entrada presenta oscilaciones debido a variaciones de la potencia de la señal de voz. La estructura para la cancelación de eco aquí propuesta agrega ruido a la voz, para controlar los valores de dispersión de los eigenvalores de la señal y el ruido agregado se elimina con un filtro operado como un cancelador de ruido.

Bibliografía

[1] BORGHESANI C. and Ambardar, A.: *Mastering DSP Concepts Using Matlab.* New Jersey, Prentice Hall, 1998.

[2] BRIGHAM E.: *The Fast Fourier Transform*, New York, Prentice-Hall, 1983.

[3] BURRUS C. and Parks T.: *DFT / FFT and Convolution.* John Wiley and Sons, 1985.

[4] BUTTERWORTH S.: *On the Theory of Filter Amplifiers*, Wireless Engr., vol 7.,pp 536-541, October 1930.

[5] CAUER, W.: *Siebschaltungen.* V.D.I. Verlag G.m.b.H, Berlin 1951.

[6] CAUER, W.: *Synthesis of Linear Communacationn Networks (Translated from the German)* McGraw-Hill Book Company, New York 1958.

[7] CHASSAING R and Horning D.: *Digital Signal Processing with the TMS320C25* John Wiley and Sons, Toronto1990.

[8] CHEN, H.: *Linear Network Design and Synthesis* McGraw-Hill Book Company, New York 1964.

[9] CHEN, H.: *The Analysis of Linear Systems* McGraw-Hill Book Company, New York 1963.

[10] CHEN W.: *The circuits and Filters Handbook.* CRC Press/IEEE, ISBN 0-8493-8341-2.

[11] CHING P. and Wu S.: *Realtime Digital Signal Processing System Using a Parallel Processing Architecture.* Microprocessors and Microsystems, No.10, December 1989, pp. 653-658.

[12] McCLELLAN J., Schafer R. and Yoder M.: *DSP FIRST A Multimedia Approach.* Prentice Hall 1998.

[13] McCLELLAN J.: *The design of two-dimensional digital filter by transformation.* Proc. 7th. Annu. Princenton Conf. Information Sciences and Systems, 1973, pp. 247- 251.

[14] McCLELLAN J., Schaffer R and Yoder M.: *DSP First a Multimedia Approach.* New York, Prentice Hall, 1998.

[15] DARLINGTON, S.: *Synthesis of Reactance 4-Poles, which Produce Prescribed InsertionLoss Characteristics.* J. Math. Phys. Vol. 18, pp 257-353, September 1939.

[16] ELLIOTT D. aud Rao, K.: *Fast Transforms. Algorithms, Analyses, Applications.* New Jersey, Academic Press, 1982.

[17] FETTWEIS A.: *Digital Filters Structures Related to Clasical Filter Networks.* Arch. Elek. Uebertragung, vol. 25, pp. 79-89, Feb.1971.

[18] N.J. Fliege.: *Multirate Digital Signal Processing.* John Wiley and Sons, ltd, New York, Reprinted September 2000.

[19] FOSTER R.M.: *A Reactance Theorem.* Bell System Tech. Journal., vol 3, pp 259-267, April 1924.

[20] FRANTZ G.:*The Texas Instruments TMS320C25 Digital Signal Microcomputer.* IEEE Micro 6, No.6, 1986, pp. 10–28.

[21] GÓMEZ S. et al.: *An Application-specific FFT processor.* Electronic Engineering, No.6, June 1988, pp. 99–106.

[22] GUILLEMIN E.: *Communication Networks.* Vol 4. John Wiley and Sons. New York 1935.

[23] GUILLEMIN E.: *Introductory Circuit Theory.* John Wiley and Sons New York 1953.

[24] GUNN L.: *Chips Try Teaching Computers To Speak, Listen.* Electronic Design, No.11, May 1989, pp. 33–36.

[25] HONIG M. and Messerschmitt, C.G.: *Adaptive filters Structures, Algorithms and Applica tions.* New York, Kluwer Academic Publishers, 1986.

[26] HUELSMAN L.: *Active and Pasive Anslog Filter Design.* McGraw Hill, ISBN 0-07-112519-1 New York 1993.

[27] HUMPHREYS L.: *The Analysis, Design and Synthesis of Electrical Filters.* Prentice Hall, ISBN 0-13-032904-3, Toronto, 1977.

[28] INGLE V.: *Digital Signal Processing Using Matlab V.4* PWS Publishing Company. ITP. Boston 1997.

[29] KUH E.: *Special Synthesis Techniques for Driwing Point Impedance Function* IRE Transactions PGCT, Vol CT2, no 4, pp, 302-308, December 1955.

[30] LACROIX A.:*Digitale Filter.* 3. Auflage, Wien, Munchen, Oldenburg, 1988.

[31] LACROIX A. and Witte K.: *Zeitdiskrete normierte Tiefpasse.* Heidelberg, Dr.Alfred Huthig Verlag, 1980.

[32] LARIMORE M.: *An Algorithm for Adapting IIR Digital Filters.* IEEE Trans. on ASSP, No.4, August 1980, pp. 428–440.

[33] LARRY E.:*Manual de laboratorio de procesamiento digital de señales.* México, UNAM, Facultad de Ingeniería, 2000.

[34] LARRY E.:*Arquitecturas de DSP's, Familia TMS320 y El TMS320C50.* México, UNAM, Facultad de Ingeniería, 2000.

[35] LUIKUO G., Fleming M. and Magar M.: *A 500 MOPS DSP Chip Set.* Electronic Engineering, No.6, June 1988, pp.106–113.

[36] MALLAT S.:*Multifrecuency Chanel Decomposition of Images and Wavelet Models.* IEEE Transiction ASSP 37 (1989), s. 2091-2110.

[37] MARSHALL D., Jenkins W. and Murphy, J.: *The Use of Orthogonal Transforms for Improving Performance of Adaptive Filters.* IEEE Trans. on CAS, No.4, April 1989, pp.474–484.

[38] MARVEN G. and Ewers G.: *A simple approach to Digital Signal Processing.* Oxford, Alden Press Limited, Texas Instruments, 1994.

[39] MILT L.: *Signal–Processing Circuits.* Electronic Design. No.4, February 1990, pp. 101–108.

[40] MITRA S.: *Digital Signal Processing. A computer-Based Approach* MCGraw-Hill Companies,Inc 1998.

[41] MITRA S.: *Digital Signal Processing. A computer-Based Approach.* New York, McGraw-Hill Companies, 1998.

[42] MITRA S., Chacrabarti S. and Abreu E.: *The nonuniform Discrete Fourier Transform and its Signal Processing Applications.* Proc European Signal Processing Conference-EUSIPCO, Brussels, Belgium, August 1992, pp. 909-912.

[43] MITRA S.K. and Sherwood R.: *Canonic realizations of digital filters using the continued fraction expansion.* IEEE Trans. Audio Electroacoustics, AU-20: August 1972, pp. 185-194.

[44] _____ .: *Digital Ladder Networks* IEEE Trans. Audio Electroacoustics, AU-21: February

1973, pp.30-36.

[45] MITRA S., Hirano K., Furano K. and Sherwood R.: *Digital sine-coseno generator.* International Conference on Digital Signal processing, Florence, Italy, September 1975, pp. 142-149.

[46] OPPENHEIM A. and Schafer W.: *Discrete – Time Signal Processing.* Prentice Hall, Inc., 1989.

[47] OPPENHEIM A. and Schafer R.: *Discrete-Time Signal Processing.* New Jersey, Prentice Hall, 1989.

[48] _____.: *Digital Signal Processing.* New Jersey, Prentice Hall, 1985.

[49] ORFANIDIS S.: *Introduction to Signal Processing* Prentice Hall, New Jersey 1996.

[50] PARKS T and Burrus C.: *Digital Filter Design.* John Wiley Sons, Inc., New York, 1987.

[51] RAMÍREZ F. and Moreno J.: *Prácticas de procesamiento digital de Señales.* Centro de Investigación y de Estudios Avanzados del IPN, Departamento de Ingeniería Eléctrica, Octubre 1996.

[52] RAO K. and Yip P.: *Discrete Cosene Transform.* San Diego, Academic Press, 1990.

[53] SAAL R. and E.Ulbrich.: *On the design of Filters by Synthesis.* Proc. IRE , vol. CT-5, no.4, pp 284-327, December, 1958

[54] STORCH L.: *Synthesis of Constant-time-delay Ladder Networks Using Bessel Polynomials.* Proc. IRE, vol. 42, no. 11, pp 1666-1675, N0vember 1954.

[55] STORER J.E.: *Relationship between the Bott-Dufin and Pantel Impedance Synthesis.* Proc. IRE, vol 42, no. 9, p. 1451, September, 1954.

[56] STRANG G. , NGUYEN T.: *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Wellesley, 1996.

[57] SU K.: *Analog Filters.* Chapman and Hall, ISBN 0-412-63840-1.

[58] SCHAFER R. and Rabiner L.: *Digital Representation of Speech Signal.* Proceedings of the IEEE, Vol.63, No.4, June 1975.

[59] SHYNK J.: *Adaptive IIR Filtering.* IEEE ASSP Magazine, No.2, April 1989.

[60] SORENSEN H. and Chen J.: *A Digital Signal Processing Laboratory Using The TMS320C30.* New Jersey, Prentice Hall, Upper Sadle River, 1997.

[61] STANLEY W. and Dougherty G.: *Digital Signal Processing.* San Diego, Reston Publishing corp., 1984.

[62] STEARNS S. and David R.: *Signal Processing Algorithms In Matlab.* New Jersey, Prentice Hall, 1996.

[63] STRAHLE W.: *Adaptive Nonlinear Filter Using Fractal Geometry.* Electronic Letters, No.19, September 1988, pp. 1248–1250.

[64] TEXAS INSTRUMENTS: *TMS320C25 Digital Signal Processor. Product Description.* Printed in USA, 1986.

[65] _____.: *Digital Signal Processing with the TMS320C30.* Toronto, John Wiley and Sons, 1992.

[66] _____.: *Second-Generation TMS320 User's Guide.* Printed in USA, 1987.

[67] _____.: *TMS34020 User's Guide.* Printed in USA, 1987.

[68] _____.: *TMS320C1x/TMS320C2x Assembly Language Tools User's Guide.* Printed in USA, 1987.

[69] _____.: *TMS320C25 C Compiler Reference Guide.* Printed in USA, 1988.

[70] _____.: *TMS320C5X Starter Kit Users Guide.* Printed in USA, 1995.

[71] _____.: *TMS320 Family Development Suport-Reference Guide.* Printed in USA, 1994.

[72] _____.: *TMS320C2X DSP Starter Kit Users Guide.* Printed in USA, 1993.

[73] Reference Guide. Printed in USA, 1988.

[74] P.P. Vaidynathan.: *Pasive Cascadcd Latice Structures for Low Sensitivity FIR Filter Design. with Application to Filter Banks.* IEEE Trans Cas, vol 33, pp 1054-1Macro 1064, November 1986.

[75] P.P. Vaidynathan.: *A tutorial on Multirate Digital Filter Banks.* Proc IEEE ISCAS'88, pp 2241-2248, 1988.

[76] P.P. Vaidynathan.: *Multirate System and Filter Banks.* Prentice Hall, Englewood Cliffs, New Yersey 1993.

[77] VAN DEN ENDEN A. and Verhoeck N.: *Discrete-time Signal Processing.* New Jersey, Prentice Hall, 1989.

[78] WHITE M. et al.: *New Strategies For Improving Spech Enhancement.* Int. Journal on Biomedical Computing, No. 25, 1990, pp. 101–124.

[79] WEINBERG L.: *Network Analysis and Synthesis* McGraw-Hill, New york 1962.

[80] WIDROW B. and Stearns S.: *Adaptive Signal Processing.* Prentice–Hall, Inc., 1985.

[81] ZVEREV A.: *Handbook of Filter Synthesis.* John Wiley and Sons, New York, 1967.

Índice alfabético