Capítulo 5

Control Activo del Ruido en el DSP TMS320C25

5.1. Introducción

El microprocesador TMS320C25 es la segunda iteración de la segunda generación de la familia de Procesadores Digitales (DSP) TMS320 de *Texas Instruments*. Desarrollados con tecnología CMOS, son capaces de manejar operaciones con punto flotante de 32 bits y fabricado en un encapsulado tipo PGA (*Pin Grid Array*) de 68 pins.

Los procesadores digitales de señales son empleados para una amplia gama de aplicaciones, desde sistemas de comunicación y control hasta procesamiento de voz e imágenes. Los DSP's de propósito general son empleados principalmente en aplicaciones de comunicaciones (sistemas de comunicación celular). Los DSP's con aplicaciones específicas son los más usados en productos dirigidos al consumidor final; como lo son teléfonos celulares, fax/modems, radios, impresoras, auxiliares de audición, reproductores MP3, televisiones de alta definición (HDTV), cámaras digitales, etc. Estos procesadores se han vuelto de común elección debido a su menor costo y a que pueden manejar diferentes tareas con solo reprogramarlos.

Los procesadores DSP's son empleados principalmente en aplicaciones en tiempo real, en donde el procesamiento debe seguir el paso de los eventos externos, siendo en varias ocasiones señales analógicas.

La arquitectura del TMS320C25; que usa un reloj de 100 ns, una longitud de palabra de 32 bits para datos y programa y un bus de direcciones de 24 bits, le permite ejecutar instrucciones a una tasa de 12.5 MIPS[1].

Una característica importante de este DSP es la capacidad del procesador de realizar (ciertas) operaciones en paralelo, esto es, la multiplicación en paralelo y la carga al acumulador, todo esto en un solo ciclo. Además, el procesador posee un conjunto de registros para

[1]MIPS=Millones de instrucciones por segundo.

llevar el control de las operaciones y un registro de 64 palabras de longitud empleado por la memoria de programa. Están presentes también, 2 unidades aritméticas que trabajan en paralelo, cada una con un registro auxiliar asociado.

Los bloques de memoria interna son de acceso dual ya que pueden accesar dos operandos en un ciclo y existe un canal destinado para acceso directo a la memoria (DMA) que soporta operaciones de entrada/salida concurrentes, disminuyendo la carga de operaciones del CPU.

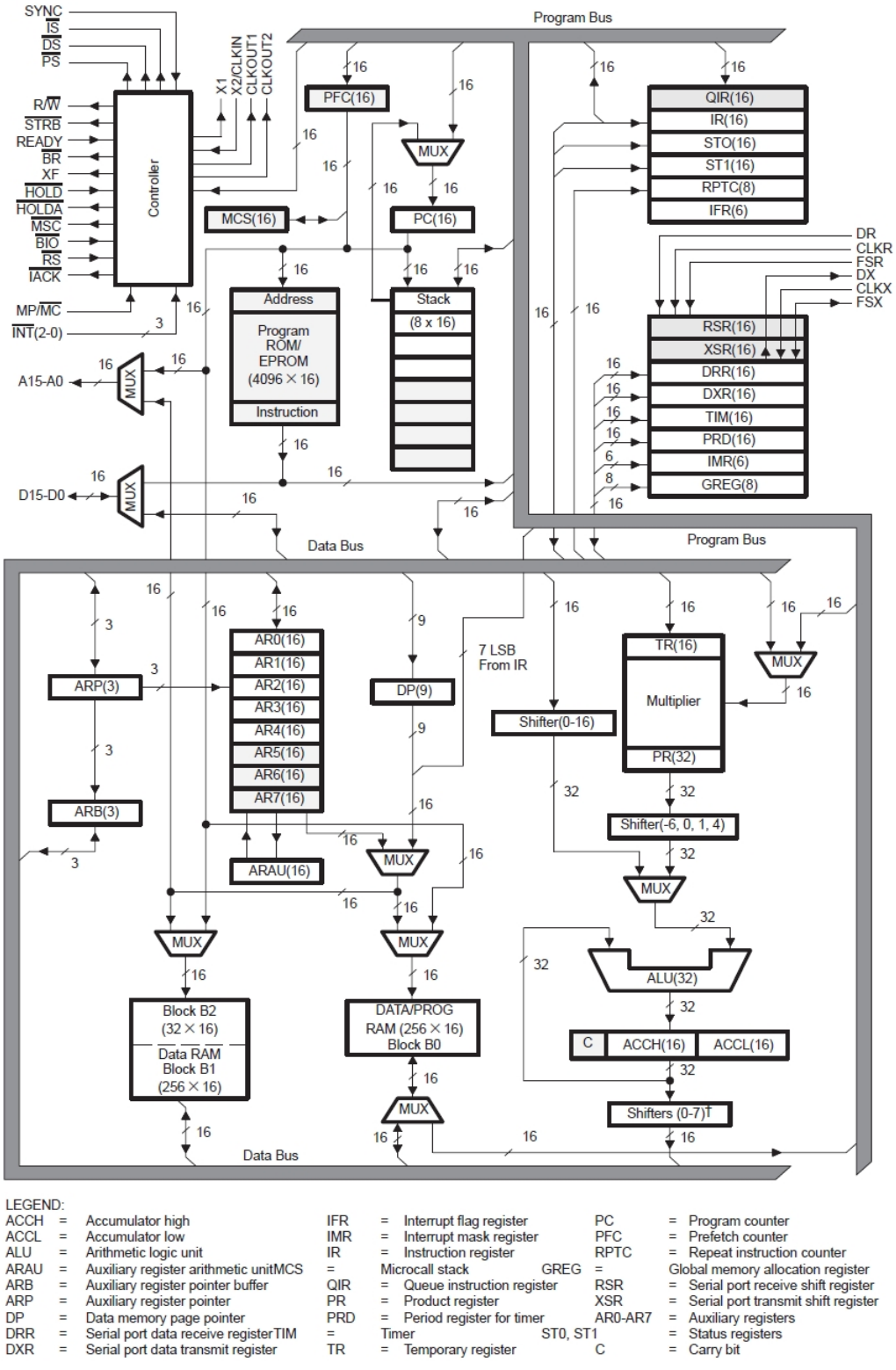


Figura 5.1: Arquitectura del DSP TMS320C25

5.2. Arquitectura del DSP TMS320C25

La familia del procesadores TMS320C2x utiliza una arquitectura Hardvard que le otorga flexibilidad y velocidad. En esta arquitectura, la transferencia entre espacios de programa y datos incrementa le flexibilidad del DSP. Esta modificación permite que los coeficientes sean almacenados en la memoria del programa para ser leídos por la memoria RAM, eliminando la necesidad de una memoria ROM dedicada a los coeficientes. Esto permite también que las instrucciones y subrutinas basadas en los valores calculados estén disponibles en cualquier momento.

La capacidad de los dispositivos TMS320C2x para obtener un mejor desempeño en aplicaciones de DSP se logra a través de instrucciones que multiplican/almacenan en un solo ciclo con opción de mover los datos, además de 8 registros auxiliares con una unidad aritmética dedicada y alta velocidad en la entrada/salida (I/O) de datos para procesamiento de señales con uso intensivo de captura de datos. La arquitectura del TMS320C25 esta optimizada para realizar funciones de filtrado en estructuras FIR[15].

5.2.1. Acumulador y ALU de 32 bits

El C25 cuenta con una Unidad de Aritmética Lógica (ALU) y acumulador que realiza un amplio rango de funciones lógicas y aritméticas en un solo ciclo de máquina. El ALU ejecuta una variedad de instrucciones las cuales proveen las siguientes capacidades:

- Dirigir hacia una dirección especificada por el acumulador
- Normalizar números de punto fijo contenidos en el acumulador
- Probar un bit específico de una palabra contenida en memoria

Una entrada al ALU siempre proviene del acumulador, mientras la otra entrada puede provenir del registro de producto (Product Register - PR) del multiplicador o la entrada de datos que han sido recogidos de la memoria RAM a través del bus de datos. Después de que el ALU ha realizado las operaciones lógicas o aritméticas, el resultado es guardado en el acumulador. El acumulador de 32 bits es divididos en dos segmentos de 16 bits para almacenamiento de datos.

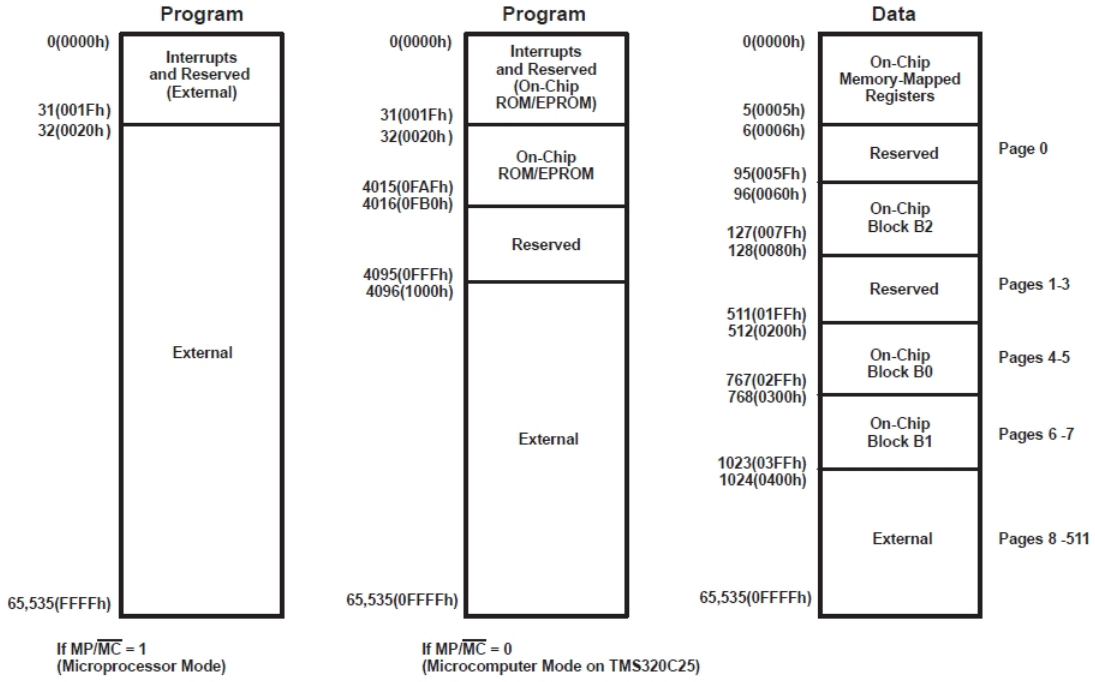
5.2.2. Multiplicador en paralelo de 16 $\times$ 16

El multiplicador es capaz de calcular productos de 32 bits (con o sin signo) en un solo ciclo de máquina. El multiplicador tiene los siguientes registros asociados:

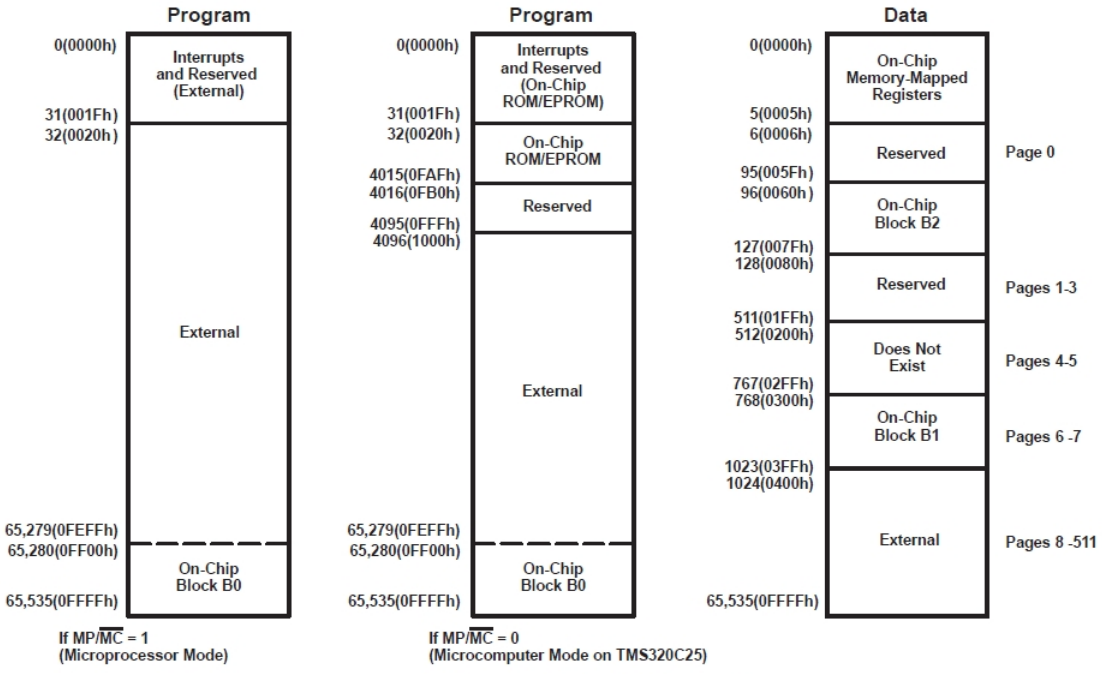
- Un registro temporal (Temporary Register - TR) de 16 bits, que almacena uno de los operandos del multiplicador.

- Un registro de producto (Product Register - PR) que almacena el resultado de la operación

Dentro del conjunto de operaciones incluido en el C25 se encuentran instrucciones de multiplicación/acumulación en un solo ciclo que permiten que ambos operandos sean procesados de manera simultanea. Los datos de estas operaciones pueden residir en cualquier lugar de la memoria interna o externa y pueden ser transferidos al multiplicador cada ciclo por medio de los *buses* de programa o datos.



(a) Memory Maps After a CNFD Instruction



(b) Memory Maps After a CNFP Instruction

Figura 5.2: Mapas de memoria para el TMS320C25.

5.2.3. Timer

El C25 provee de un temporizador de 16 bits asignado a la memoria para el control de operaciones. El registro del temporizador del chip (TIM) realiza un conteo regresivo que es continuamente actualizado por CLKOUT1 en el TMS320C25. Una señal de interrupción de temporizador (timer interrupt - TINT) es generada cada vez que el temporizador llega a cero. El timer se habilita con el valor indicado en el registro de periodo (period register - PRD) en el siguiente ciclo después de que alcanza el valor de cero de manera que las interrupciones pueden ser programadas para ocurrir a ciertos intervalos regulares cada $PRD + 1$ ciclos.

5.2.4. Memoria

El TMS320C25 provee de un total de 544 palabras de 16 bits de almacenamiento en RAM, divididos en tres bloques distintos (B0, B1, B2). De las 544 palabras, 288 de ellas (pertenecientes a los bloques B1 y B2) son asignadas a memoria de datos, las 256 palabras restantes (el bloque B0) pueden ser programadas como memoria de datos o programa. La memoria de datos de 544 palabras permite que el C25 maneje un arreglo de datos de 512 palabras (256 palabras si la memoria RAM del chip se utiliza para memoria de programa), dejando 32 localidades para almacenaje intermedio. Cuando se usa el bloque B0 como memoria de programa las instrucciones pueden ser descargadas desde una memoria externa a la memoria RAM interna y de ahí ser ejecutadas.

El TMS320C25 provee de tres direcciones separadas para memoria de programa, memoria de datos y datos de entrada/salida (I/O). La memoria interna esta asignada, ya sea dentro de los 64K palabras de memoria de datos o en el espacio de memoria de programa; dependiendo de la configuración de memoria, como se muestra en la figura 5.2. Las instrucciones CNFD (configurar bloque B0 como memoria de datos) y CNFP (configurar el bloque B0 como memoria de programa) permiten configuración dinámica de los mapas de memoria a través de software. Sin importar la configuración el usuario puede ejecutar programas desde la memoria externa de programa.

El TMS320C25 tiene seis registros que están asignados en el espacio de la memoria de datos: un registro de datos recibidos en el puerto serial, un registro de transmisión de datos en el puerto serial, registro temporizador, registro de periodo, registro de mascara de interrupción (interrupt mask register) y un registro de asignación de memoria global.

5.2.5. Interrupciones y Subrutinas

El TMS320C25 tiene tres interruptores (INT2-INT0) disponibles para que dispositivos externos interrumpan el procesador. Interrupciones internas son generadas por el puerto serial (RINT y XINT), el temporizador (TINT) y por instrucción de software (TRAP). Las interrupciones son priorizadas, *reset* (RS) tiene la más alta prioridad y el transmisor del puerto serial (XINT) la más baja prioridad. Un mecanismo de seguridad interno previene

que instrucciones multi-ciclo sean interrumpidas. Si una interrupción ocurre durante una instrucción multi-ciclo, la interrupción no es procesada hasta que la instrucción multi-ciclo es completada.

5.2.6. Interfaces Externas

El TMS320C25 soporta una amplia variedad de dispositivos externos para realizar comunicación con ellos. Un puerto serial *full-duplex* permite la comunicación con dispositivos externos como CODECS, convertidores A/D seriales y otros sistemas. Las señales de las interfaces son compatibles con codecs y otros dispositivos con un mínimo de hardware externo. Este puerto serial también puede ser usado para comunicación entre procesadores para aplicaciones multiprocesador.

El puerto serial tiene asignado dos registros en memoria: el registro de transmisión de datos (DXR) y el registro de recepción de datos (DDR). Ambos registros operan ya sea en modo de byte o modo de palabra de 16 bits, y pueden ser accesados de la misma manera que cualquier otra locación en la memoria de datos. Cada registro tiene un reloj externo, un pulso de sincronización de trama y sus registros asociados.

5.2.7. Conjunto de Instrucciones

Con un total de 131 instrucciones, el microprocesador TMS320C25 provee instrucciones tanto de propósito general como aquellas especialmente diseñadas para el cálculo intensivo.

Para un máximo desempeño, la siguiente instrucción es pre-cargada mientras la instrucción actual es ejecutada. Debido a que el mismo bus de datos es usado para comunicar datos/programa externo o espacio de entrada/salida, el número de ciclos puede variar dependiendo de donde sea pre-cargada la siguiente instrucción (de memoria de programa interna o externa). Un mejor desempeño se logra manteniendo los datos de programa en la memoria interna o en memoria externa de alta velocidad.

5.2.8. Modos de direccionamiento

El conjunto de instrucciones del TMS320C25 provee de tres modos de direccionamiento: directo, indirecto e inmediato.

Los modos de direccionamiento directo e indirecto pueden ser usados para accesar la memoria de datos. En modo directo, siete bits de la instrucción son concatenados con nueve bits del apuntador a la página de la memoria de datos para formar los 16 bits de la dirección de datos. En modo indirecto la memoria se accede a través de los registros auxiliares. En modo inmediato, los datos son basados en una porción de las palabras de instrucción.

En direccionamiento directo, la palabra de instrucción contiene los últimos siete bits de la

dirección de la memoria de datos. Este campo es concatenado con nueve bits del apuntador pagina de memoria de datos para formar la dirección completa de 16 bits. En este modo, la memoria contiene un total de 512 paginas, cada una conteniendo 128 palabras.

Existen ocho registros auxiliares (AR0-AR7) que hacen posible el direccionamiento indirecto. Para seleccionar un registro auxiliar en específico, el apuntador del registro auxiliar (Auxiliary Register Pointer - ARP) es cargado con un valor entre 0 y 7 para AR0 o AR7 respectivamente.

5.3. Implementación del algoritmo LMS

Aprovechando las ventajas que ofrecen el conjunto de instrucciones del TMS320C25, el desarrollo de un filtro de Control Activo del Ruido con algoritmo LMS es simplificado usando funciones tales como MPYA, MPYS o ZARL. Notamos en la ecuación (3.27) del algoritmo LMS, que el factor $2\mu e[k]$ es un valor constante que puede ser calculado una sola vez y almacenado en una variable en el registro T. De este modo, el cálculo se vuelve una instrucción de multiplicación/acumulación y redondeo. En este caso el bloque B1 contendrá el vector de datos y el bloque B0 los coeficientes del filtro. El código, en el archivo **adap.asm**, se muestra a continuación:

```
1  **********************************
2  * Filtro adaptivo, algoritmo LMS  *
3  *    Junio de 1996                *
4  **********************************
5  *
6           AORG >0000
7  RESET    B    INIT
8  *
9           AORG >0020
10 *
11 * INICIALIZACION DEL MICROCONTROLADOR
12 *
13 ONE      EQU  >0060
14 BETA     EQU  >0061
15 ERR      EQU  >0062
16 ERRF     EQU  >0063
17 YN       EQU  >0064
18 XN       EQU  >0065
19 DN       EQU  >0066
20 ORDER    EQU  10
21 *
22 FRSTAP   EQU  >0300
23 LASTAP   EQU  768+ORDER
24 *
25 COEFFP   EQU  >FF00
26 COEFFD   EQU  >0200
```

```
*
*************************
* Inicializacion        *
*************************
INIT    SOVM              ; Trabaja en saturacion
        LDPK 0            ; Trabaja con bandera zero
        LACK 1            ; Carga variable ONE con un bit para redondeo
        SACL ONE          ; a la variable ONE
        ZAC               ; A CERO el registro ACC y las variables:
        SACL YN           ; YN = 0
        SACL BETA         ; BETA = 0
        SACL ERR          ; ERR = 0
        SACL ERRF         ; ERRF = 0
        SACL DN           ; DN = 0
        LARP AR4          ; Habilita registro AR4
        LRLK AR4,>61      ; La direccion  >61 de B2 a la AR4
        BLKP MU2,*        ; 4CCC=0.5833 a variable MU2
        LRLK AR4,LASTAP   ; 768+10=778 al registro AR4
        RPTK ORDER-1      ; Se repite la instrucción 10 veces
        IN   *-,PA1       ; <PA1>--> 778,777,...,768 Memoria B1
        RPTK ORDER-2      ; Se repite la que sigue 9 veces
        IN   DN,PA2       ; <PA2>--> DN
        IN   DN,PA2       ; <PA2>--> DN
        LRLK AR4,512+ORDER ; 512+10=522--> AR4
        RPTK ORDER-1      ; Se repite la instruccion 10 veces
        SACL *-           ; Limpia bloque B0
*************************
* Realiza filtro FIR    *
*************************
CICLO   CNFP              ; B0 como Memoria de programa
        MPYK 0            ; 0-->PR
        LAC  ONE,14       ; En ACC se carga el bit de redondeo
        LARP AR3          ; Se habilita registro auxiliar AR3
        LRLK AR3,LASTAP   ; 778--> AR3
FIR     RPTK ORDER-1      ; La instruccion que sigue se repite 10 veces
        MACD COEFFP,*-    ;
        CNFD              ; B0 como la memoria de datos
        APAC              ; Se suma <PR>+<ACC>-->ACC
        SACH YN,1         ; El resultado --> YN: salida del filtro
        OUT  YN,PA5
        NEG               ; <-ACC>-->ACC
        ADD  DN,15        ; <DN-ACC>-->ACC
        SACH ERR,1        ; <ACC>-->ERR
        OUT  ERR,PA3      ; <ERR>--> PA3  El error al registro PA3
*************************
*  ALGORITMO LMS        *
*************************
        LT   ERR          ; <ERR>--> TR
        MPY  BETA         ; <ERR*BETA> --> PR
        PAC               ; <ERR*BETA> --> ACC
        ADD  ONE,14       ; REDONDEA EL RESULTADO
        SACH ERRF,1       ; <ERROR(i)*BETA> --> ERRF(i)
```

```
79
80          MAR  *+              ;
81          IN   XN,PA1          ; <PA1>—>XN Actualiza el vector X1(n+1)
82          LAC  XN              ; <XN>—>ACC
83          SACL *               ; <ACC>—>AR4
84
85          LARK AR1,ORDER-1     ; 9-->AR1
86          LRLK AR2,COEFFD      ; <0200>-->AR2
87          LRLK AR3,LASTAP+1    ; LASTAP+1= contiene el vector X1(n)
88                               ; 769-->AR3
89          LT   ERRF            ; <ERRF>—>TR
90          MPY  *-,AR2          ;
91          SPM  1               ; 1--> al registro PM
92
93  ******************************
94  * Actualiza los coeficientes *
95  *   W(n+1)=W(n)+2Mue(n)X1(n) *
96  ******************************
97
98  ADAPT   ZALR *,AR3           ; <W19>—>ACCH, 0-->ACCL AR3 Activo
99          MPYA *-,AR2          ; <X19>*<ERRF>—>PR, <W18>, AR2 Activo
100         SACH *+,0,AR1        ; <ACCH>—>AR2, <X18>
101         BANZ ADAPT,*-,AR2    ; Salto a ADAPT si Wn no es cero
102
103         CALL NUEVO           ; Se llama subrutina NUEVO
104         B    CICLO           ; Salto a etiqueta CICLO
105 NUEVO   IN   DN,PA2          ; Nueva muestra <PA2>—>DN
106         SPM  0               ; 0-->PM sin corrimiento
107         RET                  ; Regresar a CICLO
108
109 ***************************
110 *    Valor de 2*Mu        *
111 ***************************
112 MU2     DATA >4CCC           ; La constante MU2
113 *
```

En la parte de inicialización de variables se observa que el vector de coeficientes (bloque B0) se inicializa con ceros y el vector de datos (bloque B1) se inicializa con $\{x[1], x[2], \cdots, x[L]\}$, repitiendo L veces la instrucción IN con L el orden del filtro ($L = 20$ en este caso).

Enseguida se realiza el filtrado FIR, el valor de la salida del filtro se resta a la señal de referencia; la cual se obtiene del puerto PA2, para así generar la señal de error y enviarla al puerto de salida PA3. La constante BETA contiene el valor de 2μ de manera que al multiplicar por el error se obtiene $2\mu e[k]$ y se procede a almacenar esta contante en el registro T.

Finalmente, se actualizan los coeficientes del filtro conforme la ecuación (3.27), donde el último elemento se calcula como.

$$w[n+1] = w[n] + BETA * x[n] \tag{5.1}$$

La instrucción MPYA realiza la multiplicación y almacena el resultado en el registro P, además de que en la misma instrucción almacena el producto previo $i-1$.

En general, ruido por redondeo ocurre después de cada multiplicación. Sin embargo, el TMS320C25 tiene un multiplicador de 16×16 y un acumulador de 32 bits, por lo que no existe redondeo durante la suma de los productos. Todas la multiplicaciones se presentan con precisión completa y el redondeo ocurre después de la multiplicación. Como los coeficientes del filtro adaptivo son almacenados en la memoria RAM, el orden máximo del filtro transversal en esta implementación puede ser 256.

5.4. Consideraciones de implementación

Las estructuras y algoritmos descritos en capítulos anteriores fueron derivados usando como base la aritmética de precisión infinita. Al momento de implementar estas estructuras y algoritmos en un DSP que maneja números enteros de punto fijos, existe una limitación en la precisión de estos filtros, debido al hecho de que el DSP trabaja con un número finito de bits. Por lo tanto, el diseño debe de ser cuidadoso para reducir los efectos de la longitud de palabra finita. En general estos efectos son: cuantización de las señales de entrada, redondeo en las operaciones aritméticas, limitaciones del rango dinámico y cuantización de los coeficientes del filtro. Estos efectos pueden causar desviaciones del diseño original o causar ruido a la salida del filtro.

5.4.1. Limitaciones en el rango dinámico

Como se mencionó en la sección 3.3, el filtro transversal más usado junto con el algoritmo LMS se especifíca con las siguientes equaciones:

$$y[n] = \sum_{i=0}^{N-1} w_i[n]x[n-1] \tag{5.2}$$

y

$$w_i[n] = w_i[n] + u * e[n] * x[n-1] \quad , \; para\, i = 0, 1, \cdots, N-1 \tag{5.3}$$

Donde $x[n-1]$ es la secuencia de entrada y $w_i[n]$ los coeficientes del filtro.

Si la secuencia de entrada y los coeficientes del filtro están apropiadamente normalizados de modo que sus valores estén entre 1 y -1 usando el formato Q15, no aparece error en la suma. Sin embargo, la suma de dos números puede ser mayor a 1; esto se le conoce como *overflow*. El TMS320C25 provee de 4 maneras de manejar esta situación.

- Modo "Overflow" (aritmética de saturación)
- Ramificaciones de las condiciones de sobresaturación.

- Desplazamiento a la derecha del Registro de Producto
- Desplazamiento a la derecha del acumulador

Una técnica para evitar la probabilidad de error es el escalamiento, esto es, no permitir que la magnitud de los nodos del filtro adaptivo sobrepasen |1|. La condición es:

$$x_{max} < 1/\sum_{i=0}^{N-1} |w_i[n]| \tag{5.4}$$

Donde x_{max} denota el valor máximo absoluto de la entrada. Usando desplazamiento a la derecha, el C25 puede implementar el escalamiento para evitar sobre-saturación en la ecuación (5.2). Al colocar los bits de PM en el registro de estado ST1 a 11 usando la instrucción SPM o LST1, la salida de registro P es desplazada 6 bits a la derecha. Esto permite hasta 128 acumulaciones sin la posibilidad de sobre-saturación. La instrucción SFR también permite desplazar un bit a la derecha cuando existe la posibilidad de sobre-saturación.

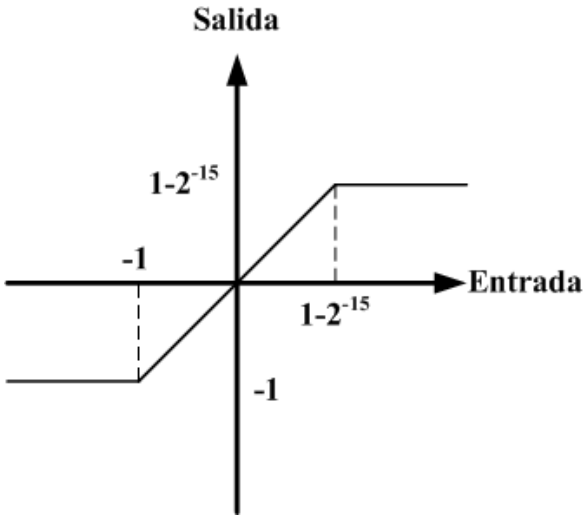


Figura 5.3: Aritmética de Saturación

Otra forma efectiva de prevenir overflow en la ecuación (5.2) es usar aritmética de saturación. Como se ilustra en la figura 5.3, si el resultado de una suma resulta en sobre-saturación, la salida se reduce al valor máximo. Si la aritmética de saturación se usa, los valores de la entrada serán mayores que el límite superior dado en la ecuación (5.4). La aritmética de saturación es controlada en el C25 por medio del bit OVM en el registro de estado ST0 y puede ser controlado con las instrucciones SOVM (set overflow mode), ROVM (reset overflow mode) y LST (load status register).

5.4.2. Errores de precisión finita

El TMS320C25 tiene un procesador de punto fijo de 16 bits. Cada muestra de datos es representada por un número fraccional que usa 15 bits de magnitud y un bit de signo. El intervalo de cuantización:

$$\delta = 2^{-b} \tag{5.5}$$

Con $b = 15$, donde los números son cuantizados en incrementos de δ. Los productos de

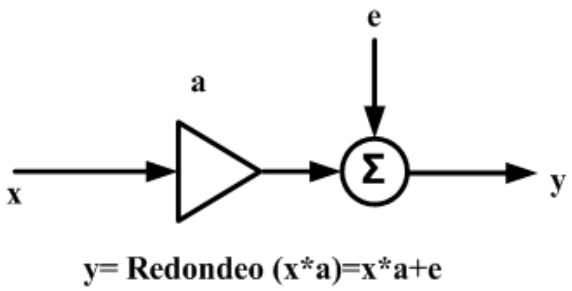


Figura 5.4: Modelo de Ruido por Redondeo de Punto Fijo

las multiplicaciones de datos por los coeficientes dentro del filtro deben de ser redondeados o truncados para poder ser almacenados en la memoria o los registros de CPU. Como se muestra en la figura 5.4, el error de redondeo puede ser modelado como ruido añadido al filtro por cada operación de redondeo. Este ruido blanco tiene una distribución uniforme sobre el intervalo de cuantización y para el redondeo

$$-1/2\delta < e \leq 1/2\delta \tag{5.6}$$

y

$$\delta_e^2 = (1/12)\delta^2 \tag{5.7}$$

Donde δ_e^2 es la variancia del ruido blanco.

En general el ruido de redondeo ocurre despues de cada multiplicación. Sin embargo, el TMS320C25 tiene un acumulador de precisión completa, esto es, un multiplicador de 16×16 con un acumulador de 32 bits, por lo que no hay redondeo cuando se implementa una serie de multilpicaciones y sumas como en la ecuación (5.2). El redondeo se aplica cuando el resultado es almacenado en memoria $y[n]$, de modo que solo una fuente de ruido este presente en cualquier operación.

5.5. Simulación

Para probar el código desarrollado necesitamos usar el simulador *SIM25.EXE*, pero para poder hacerlo debemos de compilar primero el programa. Para esto se usa el programa *XAS25.EXE* el cual, al ser ejecutado, genera dos archivos: **adap.lst** y **adap.mpo**, que son

archivos auxiliares conteniendo las enlaces a las librerías necesarias y el el programa compilado en lenguaje de máquina.

Para probar el funcionamiento del programa desarrollado, el TMS320C25 deberá de filtrar un tono senoidal contaminado por otro tono de diferente frecuencia. El primer tono o señal primaria es una señal senoidal de 200 Hz de frecuencia; el segundo tono o señal contaminante, será una señal cosenoidal con 60 Hz de frecuencia.

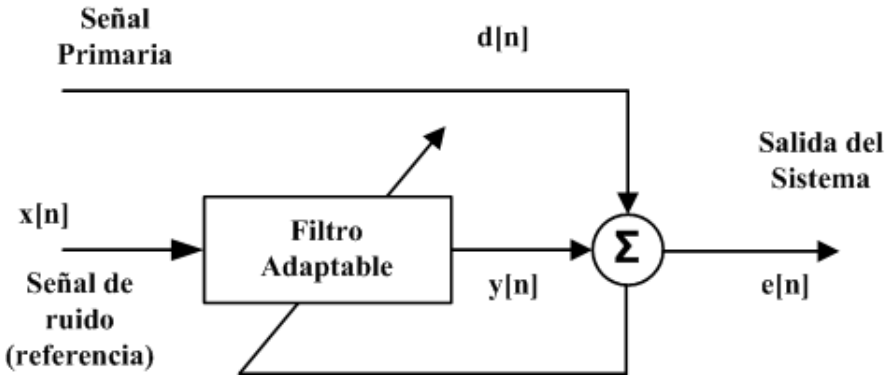


Figura 5.5: Esquema de cancelación de ruido usado en el Programa.

Para lograr el objetivo, usamos el esquema básico de cancelación del ruido (mostrado en la figura 5.5) monocanal. Donde se usan tienen las dos señales de entrada. La señal senoidal de 200 Hz ($d[n]$) es la señal primaria o de datos, la cual está contaminada con la señal de ruido, misma señal que es usada como señal de referencia. Entonces tenemos la señal de datos:

$$d[n] = s[n] + A_o \cos(\omega_o n + \phi_o) \tag{5.8}$$

Y la señal de referencia

$$x[n] = A \cos(\omega_0) \tag{5.9}$$

Una representación gráfica de ambas señales puede ser observada en las figuras 5.6 y 5.7. Después de ejecutar y dejar el filtro adaptivo por 5000 iteraciones en el SIM25 obtenemos de salida las señales mostradas en las figuras 5.8 y 5.9.

A la salida del filtro adaptivo con algoritmo LMS obtenemos la señal recuperada, observamos que contiene algunos errores, sobre todo al principio de la secuencia, esto es evidencia del filtro adaptando sus coeficientes para poder eliminar la señal no deseada.

También a la salida del filtro adaptivo obtenemos la señal de error del proceso, observamos que el error disminuye conforme el tiempo de procesamiento aumenta, de acuerdo con lo ya establecido en la teoría del algoritmo LMS.

La estructura y algoritmo usados probaron ser adecuadas para recuperar la señal de información eliminando la señal contaminante. El TMS320C25 tiene la ventaja de poder ejecutar en tiempo real el código y filtrar las señales de entrada de manera inmediata para poder recuperar la señal de información.

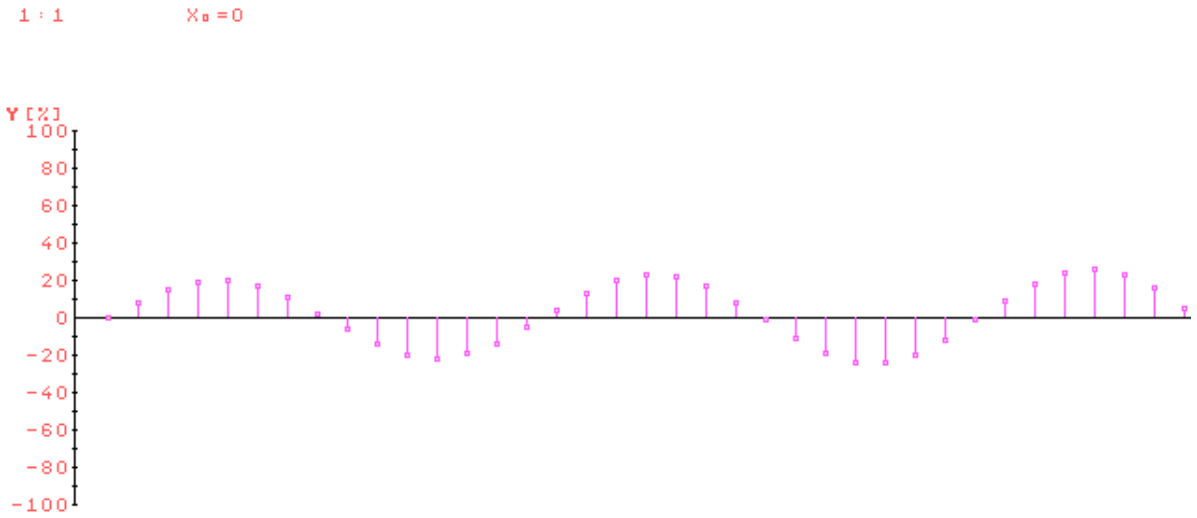


Figura 5.6: Señal de primaria.

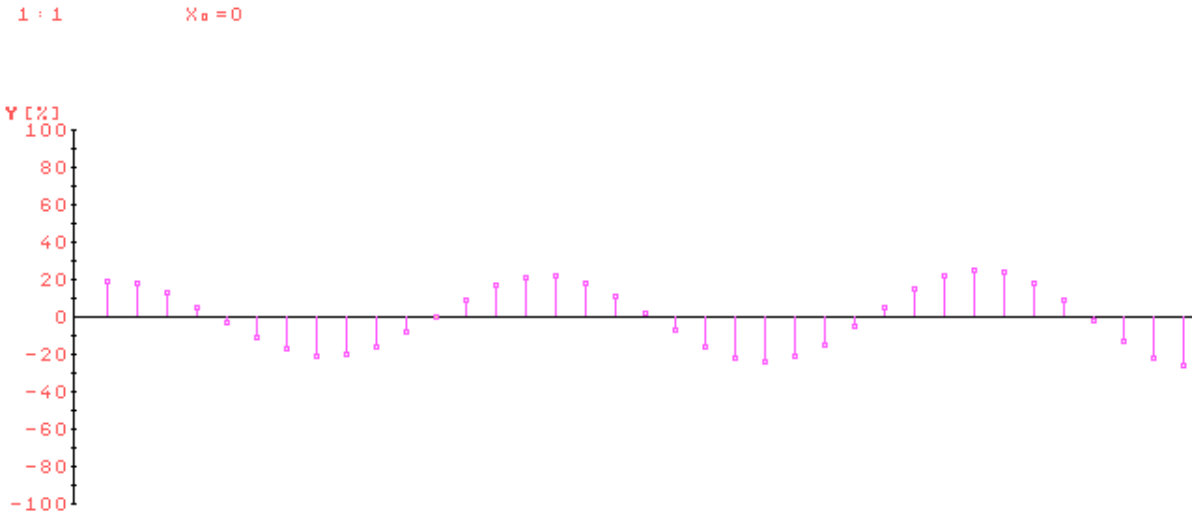


Figura 5.7: Señal de ruido.

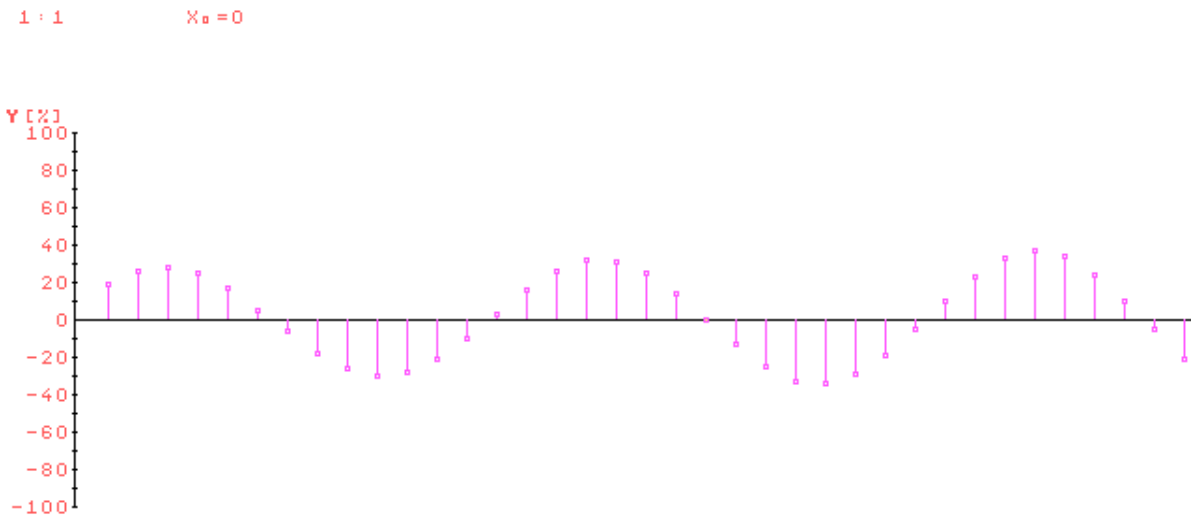


Figura 5.8: Señal de salida (señal recuperada).

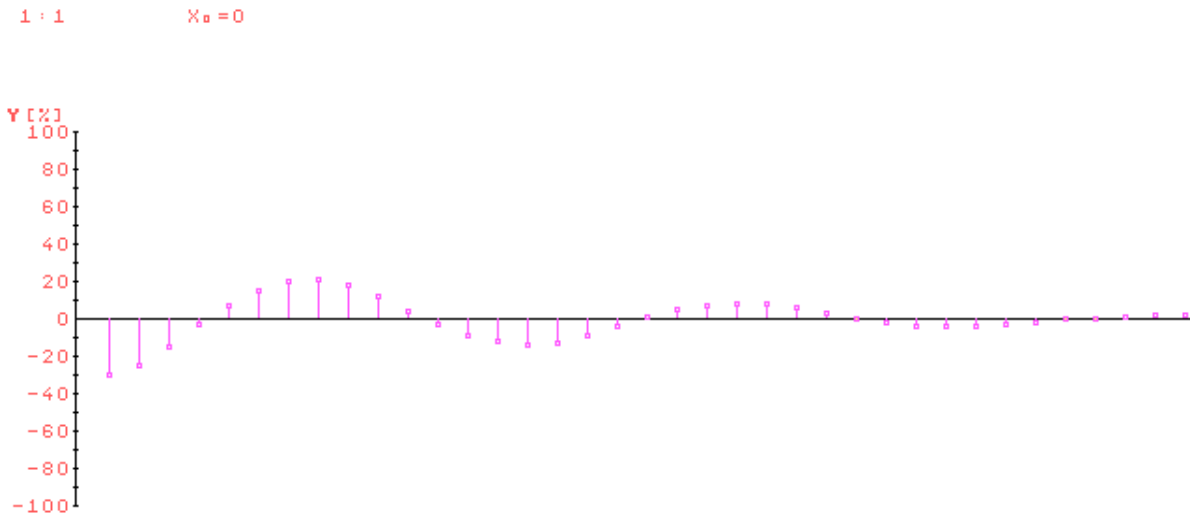


Figura 5.9: Señal de error.