Comentario Técnico

Herramientas de Diseño y Aritmética de Punto Fijo

Por Ing. Manuel José Peirone
manuelpeirone@gmail.com
www.latinaudio.com.ar

Introducción

Estimado lector, el objetivo de este artículo es introducirlo en las técnicas de diseño de sistemas de procesamiento digitales empleando aritmética de punto fijo. El enfoque del mismo es sustancialmente práctico, dejando las cuestiones más teóricas a la abundante literatura disponible.

Aritméticas existentes

Existen dos tipos de representación numérica:

Figura 1. Diferencias entre punto fijo y punto flotante

En el caso de la aritmética de punto fijo, el punto binario se encuentra siempre en la misma posición, es decir, existirán *m bits* para la parte entera y *n bits* para la parte decimal. En algunos casos puede ocurrir que $m = 0$ (no existe parte entera) o bien, $n = 0$ (no existe parte decimal). En cambio, para la aritmética de punto flotante, la ubicación del punto binario puede variar (Figura 1).

Este artículo se centrará en la aritmética de punto fijo ya que es la más utilizada en sistemas embebidos, desde MCU de 8 bits (como el MCF51JM60 de Freescale) hasta DSP´s de 24 bits (como el DSP56371 de la misma compañía).

Aritmética de punto fijo

Muchas veces cuando se utilizan procesadores digitales no se tiene en cuenta el tipo de aritmética utilizada, dado que el programador puede estar trabajando en código C y es el compilador quien se encarga de codificar y asegurarse de que las operaciones matemáticas y aritméticas se ejecuten correctamente respetando la arquitectura del procesador.

En general, se sabe bien que un registro de 4 bits permite representar 16 combinaciones diferentes, pero pocas veces nos detenemos a pensar en **cómo interpretar qué representan en términos numéricos**. Analicemos esto con más detalle. Supongamos que un ADC de 4 bits entrega la siguiente muestra binaria:

$$x_Q = 0101_2 \quad \textbf{(1)}$$

Ahora hagamos la siguiente pregunta: **¿qué número decimal representa esta combinación?** La respuesta no es una sola, ni varias, sino *infinitas*. La interpretación dependerá de lo que el programador defina como *convención*. Por ejemplo, se podría proponer la siguiente interpretación:

$$x_Q = \left(2.b_3 + 0{,}5.b_2 + b_1 + 4.b_0\right)_{10} \quad \textbf{(2)}$$

Así, el número decimal representado por la muestra anterior será:

$$x_Q = \left(2.0 + 0{,}5.1 + 0 + 4.1\right)_{10} = 4{,}5_{10} \quad \textbf{(3)}$$

Pero lógicamente la cuestión no es tan liberal. En microcontroladores, DSP´s y microprocesadores comerciales esta convención no es determinada por el usuario sino por el **fabricante o por un estandar**.

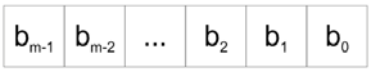

Figura 2. Conjunto de bits

Para un conjunto de bits como el de la Figura 2, existe una serie de interpretaciones posibles que son ampliamente utilizadas:

Tabla 1. Posibles interpretaciones de un conjunto de bits

Interpretación	Expresión	Rango
Entero sin signo	$x(n) = \sum_{i=0}^{m-1} b_i.2^i$	$0 \le x_Q \le 2^m - 1$
Entero con signo	$x(n) = -b_{m-1}.2^{m-1} + \sum_{i=0}^{m-2} b_i.2^i$	$-2^{m-1} \le x_Q \le 2^{m-1} - 1$
Fraccional sin signo en complemento a 2	$x(n) = \sum_{i=0}^{m-1} b_i.2^{-(m-i)}$	$0 \le x_Q \le 1 - 2^{-m}$
Fraccional con signo en complemento a 2	$x(n) = -b_{m-1} + \sum_{i=0}^{m-2} b_i.2^{-(m-i-1)}$	$-1 \le x_Q \le 1 - 2^{-(m-1)}$

Generalmente los DSP´s emplean la **aritmética fraccional con signo en complemento a 2** para el procesamiento de señales, mientras que para indexado y punteros a memoria se utiliza la **aritmética entera**. En cambio, los microcontroladores usualmente operan internamente con aritméticas enteras para ambos casos. Sin embargo, esto no implica que, por ejemplo, en código C no podamos definir una variable flotante y manipularla como tal. Pero debemos tener cuidado ya que, si el procesador no está diseñado para esta aritmética, cada operación demorará mucho más en ejecutarse que aquella para la que sí fue diseñado.

Implementar por hardware una aritmética de punto fijo conlleva una mayor simplicidad lógica, lo cual se traduce directamente en menores costos ya que además ocupa **menor superficie de silicio** respecto a una unidad de punto flotante.

Entre las desventajas más significativas se encuentra el hecho de que los errores de truncamiento con esta arquitectura pueden ser significativos, degradando la calidad de los algoritmos implementados (truncar implica *cortar* la palabra binaria en un punto determinado y descartar los bits que no se necesitan). Para lograr en punto fijo la misma precisión que se logra con punto flotante se necesitaría una cantidad muy grande de bits. De hecho, recurriendo al standard *ANSI/IEEE Std. 754-1985* se encuentra que el menor valor representable en punto flotante de 32 bits es $\pm 1{,}2.10^{-38}$. Para lograr la misma precisión en punto fijo se requerirían ni más ni menos que:

$$2^{-n} = 1{,}2.10^{-38} \Rightarrow n = -\log_2\left(1{,}2.10^{-38}\right) \quad \textbf{(4)}$$

$$¡\, n = 125{,}97\ bits \cong 126\ bits\,! \quad \textbf{(5)}$$

casi el **cuádruplo** de bits que para la misma precisión en aritmética flotante.

Multiplicación y suma

Se supone el siguiente valor binario. Debajo se ven dos interpretaciones distintas:

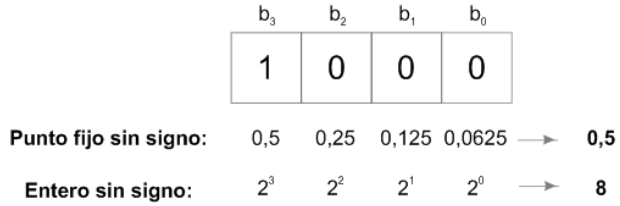

Figura 3. Valor binario arbitrario

Ahora se realiza la multiplicación de este valor por si mismo. Se obtendrán dos resultados distintos, según cómo se interpreten los operandos:

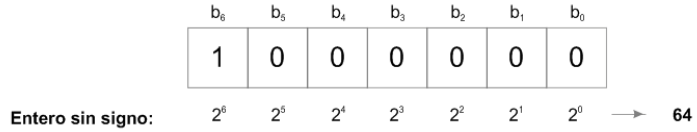

Figura 4. Resultado de la multiplicación considerando operandos enteros

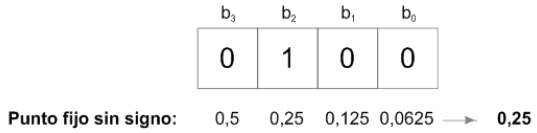

Figura 5. Resultado de la multiplicación considerando operandos en punto fijo

Hay que recalcar que una multiplicación produce resultados de (2.m - 1) bits de longitud, donde m es la longitud de los operandos (suponiendo que es la misma para ambos).

Por ejemplo, en el DSP56805 de Freescale el ancho de las variables de punto fijo es de 16 bits y lógicamente están expresadas en **complemento a 2 con signo**. Su rango es:

$$-1 \leq x_{Q_{DSP56805}} \leq 1 - 2^{-15} \quad \textbf{(6)}$$

donde:

$$x_{Q_{DSP56805}} = -b_{15} + b_{14}.2^{-1} + b_{13}.2^{-2} + \ldots + b_1.2^{-14} + b_0.2^{-15} \quad \textbf{(7)}$$

Multiplicación y adición de números en punto fijo. Para la multiplicación de dos números en punto fijo cuyo rango sea el especificado por la ecuación 6, el resultado *siempre* será menor que 1. Sin embargo, para la adición de dos números en punto fijo, se debe cuidar que el resultado se mantenga entre -1 y 1 para evitar desbordamientos. Por ejemplo, sumar 0,5 + 0,7 produce un resultado mayor que 1. Por lo tanto se deberá realizar un escalamiento previo de la siguiente manera:

$$\frac{0,5}{2} + \frac{0,7}{2} = 0,6$$

Así, se consigue que el resultado se mantenga dentro los límites.

Aritméticas presentes en Pro Tools

Quienes estén familiarizados con el mundo de la grabación sabrán que Pro Tools es una DAW (digital audio workstation o estación de trabajo para audio digital) considerada como el estandar de la industria. Es ampliamente utilizada para grabar, mezclar y hasta masterizar, ya sea en estudio o en vivo. Pero lo que pasa desapercibido para muchos es su arquitectura interna. Hagamos un breve repaso por las distintas versiones de este programa.

Mientras que la version económica de ProTools (LE) posee un bus que opera internamente en 32 bits de punto flotante, en las versiones TDM y HD (con arquitectura de mixer de 48 bits de punto fijo) el bus de plugins es de 24 bits de punto fijo. Esta característica de diseño llevó inicialmente a muchos reconocidos Ingenieros de Mezcla y de Mastering a discutir largamente acerca de la conveniencia de utilizar uno u otro sistema. Veamos las verdaderas conclusiones de esta polémica:

Si nuestro sistema de edición digital permite mezclas en 24 bits, entonces debe trabajar internamente con más de 24 bits de capacidad de procesamiento simultáneo, de lo contrario no se preserva la integridad de nuestro audio. Básicamente, cualquier proceso que se aplique al audio original debería arrojar un resultado en 32 o 48 bits como mínimo, pero con un "piso" de ruido no mayor a 24 bits. De esta forma, cuando se hace el dithering final para volver a los 24 bits originales, la pérdida es realmente despreciable. Esta primera conclusión deja afuera la opción de trabajar, en audio profesional, con una arquitectura interna de solamente 24 bits de punto fijo. La arquitectura de punto flotante no permite el uso de dithering entre las distintas etapas de procesamiento, debido a que el exponente e cambia permanentemente, alterando la significación de la mantisa. En este sentido, la opción de 48 bits de punto fijo resulta superior dado que se le puede aplicar dithering. Si bien es cierto que en una arquitectura flotante los errores producidos siempre estarán en proporción con la magnitud de la señal procesada (y por lo tanto pueden ser difíciles o aún imposibles de detectar), la opción de dithering es una garantía de que estos errores van a ser eliminados.

Por otro lado, la verdadera forma de aprovechar las características de un sistema con arquitectura interna de 48 bits pero donde el bus de plugins es de 24 bits es utilizando únicamente plugins de doble precisión (es decir, que toman el audio sample de 24 bits y lo elevan a 48 para operar internamente y así obtener un resultado de mayor precisión). Estos plugins deberán también tener capacidad de realizar dithering a 24 bits para que este proceso de DSP de alta precisión no se pierda en el truncado o redondeo de la información que no puede quedar contenida en los 24 bits de salida. En un sistema como ProTools TDM, el uso de plug-ins de doble precisión con opción de dithering marca una diferencia con respecto al uso de plug-ins de simple precisión.

El proceso de dithering correctamente aplicado solamente sacrifica rango dinámico (por el pasaje de 48 bits a 24) pero no la linealidad de la señal (lo cual es fundamental). El nivel de error introducido por el dithering en esta etapa está en el orden de -144 dBFS, lo cual es realmente insignificante. En cambio el error producido por el truncado de la señal sin dithering aplicado está en el orden de -100 dBFS (considerablemente mayor).

- Un mismo algoritmo de DSP puede requerir diferentes profundidades de bits en las distintas etapas de su realización. Por ejemplo, un EQ se implementa como un filtro recursivo, donde el feedback juega un papel fundamental. Si el filtro tiene una frecuencia de corte suficientemente baja, la cantidad de feedback generada puede ser muy alta, lo cual amplifica enormemente el error de cuantización original, aumentándolo en dos y hasta tres órdenes de magnitud. Nuevamente, un procesador capaz de trabajar internamente con mayor cantidad de bits asegura una relación Señal a Ruido (SNR) suficiente para amortiguar incluso estas condiciones extremas de uso del DSP.

- Un EQ que trabaja en 32 bits de punto flotante a lo largo de todas sus etapas tendrá una performance de ruido considerablemente peor que un EQ que opera internamente con 48 bits de punto fijo y posee un bus de interconexión de 24 bits

- A su vez, un EQ que opera en 48 bits de punto fijo a lo largo de todas sus etapas será sólo un poco mejor que el que opera con bus de 24 bits, ya que el verdadero responsable del error de cuantización es el alto feedback producido en baja frecuencia.

- Por último, si estimamos el ruido de cuantización producido por la interconexión de plugins llegamos a la conclusión de que cada vez que se duplica el número de cuantizaciones, el umbral de ruido aumenta hasta 3 dB. Si suponemos que cada track de una mezcla tiene un máximo de 8 plug-ins insertados, tenemos que el ruido se incrementa unos 9 dB. Sumando estos 9 dB al umbral de ruido de nuestro sistema de 24 bits (-144 dB) obtenemos un ruido de cuantización total de apenas -135 dB (considerablemente menor que el ruido producido por el propio convertidor), es decir que el encadenamiento de plugins no provoca serias degradaciones de calidad de audio, aún trabajando sobre un bus de 24 bits.

- Los filtros digitales, especialmente cuando son de alto Q, tienen respuestas muy variables y con alto grado de error, aún trabajando con 64 bits de punto flotante. En estos casos extremos, la performance de 24 bits punto fijo o 32 bits punto flotante es claramente insuficiente. La solución de 48 bits punto fijo aparece como la más precisa, a la vez que es económicamente realizable. En este caso, la asignación de 8 bits de "extra headroom" (bits 40 a 47) y 8 bits de guarda (bits 0 a bit 7), dejando los 32 bits centrales para el muestreo de la señal dan al sistema suficiente precisión para responder ante condiciones exigentes de filtrado, con la capacidad de sumar hasta 256 canales de audio sin overflow, preservando así un resultado de 24 bits consistente aún después de varias etapas de procesamiento.

Bibliografía

Steven W. Smith, 2003. The Scientist and Engineer's Guide to Digital Signal Processing. www.dspguide.com.

B. A. Shenoi, 2006. Introduction to digital signal processing and digital design. Editorial Wiley.

DSP56800 Familiy manual, 16-bit digital signal controllers. DSP56800FM. Rev. 3.1, 11/2005.

DSP56800 User manual, 16-bit digital signal controllers. DSP56F801-7UM. Rev. 8, 13/2007.

CodeWarrior Development Studio for Freescale 56800/E Digital Signal Controllers: DSP56F80x/DSP56F82x Family Targeting Manual. Rev. November 6, 2007.

N. de R: El lector puede descargar el artículo completo del sitio web de **Edudevices** **www.edudevices.com.ar** en la sección ***"Comentarios Técnicos"*** dentro de la solapa ***"Artículos".***