TMS320 DSP
DESIGNER'S NOTEBOOK

Circular Buffer in Second Generation DSPs

APPLICATION BRIEF: SPRA203

Contributed by Randy Restle
Digital Signal Processing Products
Semiconductor Group

Texas Instruments
December 1992

TEXAS INSTRUMENTS

TRADEMARKS

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Circular Buffer in Second Generation DSPs

Abstract

Third and fourth generation DSPs (TMS320C3x/4x) include a circular buffer addressing mode. The second generation DSPs (TMS320C2x) do not. The integer TMS320 devices make the use of a circular buffer unnecessary because they can perform data movement simultaneously with arithmetic processing, with no penalty to code size or execution time. This is very efficient because it circumvents the overhead of maintaining a buffer pointer. However, some applications still benefit from circular buffers.

The TMS320C25 can manipulate circular buffer pointers without penalty to code size or execution time. This document discusses how to implement this, and describes how this techniques works for a TMS320C25.

Design Problem

Third and fourth generation DSPs (TMS320C3x/4x) include a circular buffer addressing mode. The second generation DSPs (TMS320C2x) do not. The integer TMS320 devices make the use of a circular buffer unnecessary because they can perform data movement simultaneously with arithmetic processing, with no penalty to code size or execution time. This is very efficient because it circumvents the overhead of maintaining a buffer pointer. However, some applications still benefit from circular buffers. An example is a decimation filter because multiple data values must be skipped. In this case, it is usually more efficient to add an offset to a pointer rather than perform multiple data movements.

Solution

The TMS320C25 can manipulate circular buffer pointers without penalty to code size or execution time. This is done by using its integral bit-reversed addressing capability, normally used in FFT solutions. In this mode, carries from each bit of the addition of AR0 and the current auxiliary register are propagated to the right instead of the left. The carry from the rightmost bit is ignored, effectively performing a modulo N addition, where N is the size of the buffer. N must be restricted to a power of 2.

Figure 1 shows the order in which data values will be stored and their corresponding binary addresses for a buffer of size 8. AR0 must be loaded with the size of the buffer divided by 2, and the coefficients that are to be multiplied by the circular-buffered values must be stored in a corresponding bit-reversed fashion. Traversing the data in bit-reversed order does operate on every data point - just not in linear order. For buffers where order is not important, but efficiency is important, this method works well.

Figure 1. Data order for bit-reversed circular buffer