TMS320 DSP
DESIGNER'S NOTEBOOK

TMS320C25 Logical Shifts in Parallel with ALU Operations

APPLICATION BRIEF: SPRA207

Keith Larson
Digital Signal Processing Products
Semiconductor Group

Texas Instruments
January 1993

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

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

Tables

TMS320C25 Logical Shifts in Parallel with ALU Operations

Abstract

With an easy trick, a logical right or left shift can be accomplished in parallel with another instruction without disturbing the accumulator, multiplier, or any other part of the ALU. This process is explained with examples to illustrate the details. Specific code commands to implement this process are given.

Design Problem

Is there a way to perform a logical shift in parallel with the ALU's normal operations?

Solution

With an easy trick, a logical right or left shift can be accomplished in parallel with another instruction without disturbing the accumulator, multiplier, or any other part of the ALU.

The trick involves thinking differently about how to use the Auxiliary Register Arithmetic Unit (ARAU). The ARAU is capable of incrementing, decrementing, and index register modification, as well as the following two important features.

First, to double the value of a number, add it to itself. The ARAU can have the current ARP=0 such that a *0+ modification will add AR0 to itself. In code:

```
LRLK  AR0,Value     ; load a value into AR0
LARP  AR0           ; point the current ARP to AR0
MAR   *0+           ; add AR0 to itself (logical left shift!)
```

Second, consider how bit-reversed carry addition is performed in the ARAU.

The logic of the ARAU is designed to propagate the carries from any half adder to the right rather than to the left as in normal addition. One way to remember how bit-reversed carry addition works is to think about looking at the inputs and outputs through a mirror, reversing the order. This causes the LSBs to switch with the MSBs, which is another way to think about bit-reversed carry addition. Table 1 shows an AR0 bit reverse added to itself (ARP=0). Table 2 shows what is normally used in FFT bit reversals and other DSP algorithms (ARP != 0), with a "mirror" line drawn in for reference.

Table 1. AR0 Bit Reverse Added To Itself

```
                    LRK     AR0,07191h
                    LARK    AR0         ;
                    MAR     *BR0+       ;          Note: carries propagate right

          C  C  C                 C  C        1
 0   1    1  1  0   0  0  1  1  0  0  1  0  0  0  1  <-- AR0
+0   1    1  1  0   0  0  1  1  0  0  1  0  0  0  1  <-- AR0
-------------------------------------------------------------------------------
 0   0    1  1  1   0  0  0  1  1  0  0  1  0  0  0  <-- New AR0
     C>   C> C>             C> C>       C>          C>     (last carry is lost)
```

Table 2. As Normally Used in FFT Bit Reversals and Other DSP Algorithms

LRLK	AR1,0100h		
LRLK	AR0,0080h		
LARP	AR1		
RPTK	7		
MAR	*BR0+		
		Mirror Line	
	LSB	MSB	LSB
	0000100000000000		0000000000010000
*BR0+	0000000010000000		+ 0000000100000000
AR1 bits	0000100000000000		0000000000010000
	0000100010000000		0000000100010000
	0000100001000000		0000001000010000
	0000100011000000		0000001100010000
	0000100000100000		0000010000010000
	0000100010100000		0000010100010000
	0000100001100000		0000011000010000
	0000100011100000		0000011100010000
	0000100000010000		0000100000010000
	Bit reversed carry --->		<---Normal carry

This trick is useful as a logical shifter that does not use the accumulator in any way. It is also helpful for performing a decimation in frequency FFT. In this case the DFT block size decreases by 1⁄2 for every stage of the FFT. When completed, the DFT block size will be two and the address offset one. By using a 'BANZ Not_done,*BR0+', a good deal of code is eliminated in a tightly looped, and reasonably efficient FFT. The value of AR0 can at the same time be used to access a bit-reversed twiddle table lookup. The same lookup table will work for any size FFT smaller than the overall size of the table permits.

The code for this FFT, written as a complete spectrum analyzer setup for the 'C2x SWDS and AIB2, is available on the TMS320 BBS (713-274-2323). This same code also works with the 'C26. The file to download is C2X_ANAL.EXE, a self-extracting PKZIP file. Also available on the BBS is code to perform successive approximation routines. A 32-bit integer square-root routine can be found in the file BFLTLIB.EXE.