DESIGNER'S NOTEBOOK

TMS320C25 Logical Shifts in Parallel with ALU Operations

Contributed by Keith Larson

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. Figure 1 shows an AR0 bit reverse added to itself (ARP=0). Figure 2 shows what is normally used in FFT bit reversals and other DSP algorithms (ARP != 0), with a "mirror" line drawn in for reference.

```
            LRLK      AR0,07191h
            LARP      AR0          ;
            MAR       *BR0+        ;   Note carries propogate 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)
```

Figure 1.

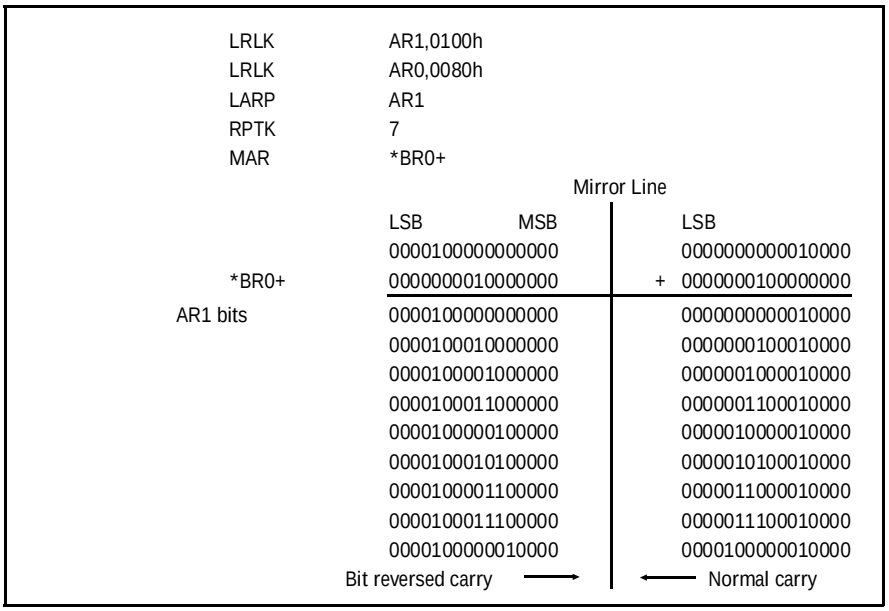

Figure 2.

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 ½ 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.