TMS320 DSP
Number 38
DESIGNER'S NOTEBOOK
TEXAS INSTRUMENTS

Mastering the 'C4x DMA

Contributed by Rosemarie Piedra

Design Problems

What are the basic differences between the 'C3x and 'C4x DMA?
What to do if the DMA is slower than expected or never finishes?
How to program the 'C4x DMA?
Examples?

Solution

The 'C4x DMA is one of the most powerful DMAs available in the market. It gives features not available in traditional DSPs.

Basic Differences Between 'C3x and 'C4x DMAs

'C3x and 'C4x DMAs are functionally similar. The 'C4x adds the following features:

1. More DMA channels (1 for the 'C3x vs. 6/12 for the 'C4x).
2. The 'C4x DMA is faster: the 'C3x DMA requires one cycle of internal register setup time when the DMA is reading from external memory. The 'C4x DMA doesn't require this extra cycle.
3. The 'C4x DMA has more features than the 'C3x DMA, such as autoinitialization mode, bit-reversed addressing, and split mode.
4. The 'C4x DMA allows you to control the priority between CPU and DMA. The 'C3x DMA has always lower priority than the CPU.
5. The 'C4x DMA interrupts are totally independent of the CPU interrupts. The 'C4x DMA doesn't require an instruction fetch boundary to acknowledge the interrupt. The 'C3x DMA detects DMA interrupts in fetch boundaries.

What to do if the DMA is Slower Than Expected or Never Finishes?

The maximum sustained data transfer rate of the 'C40 DMA is one word every two cycles (50 MB/s for a 50-MHz 'C4x), provided a 0-wait-state memory and a DMA with higher priority over the CPU (or if there are no conflicts). Memory (0-wait-state) with no conflicts is 50 MB/s. If DMA reads and writes to external memory are interleaved, the maximum sustained rate is 33 MB/s (one word transfered every three cycles: one for read and two for write). Arbitration between DMA channels does not impose any overhead cycles.

The following factors may slow down or even stop the DMA:

1. Contention with the CPU: Even though DMA has its own internal buses, CPU/DMA memory access conflicts may exist. You can avoid that by allocating DMA src and destination addresses in buses that the CPU is not using at that time. The 'C4x offers two external buses and a dual-access on-chip RAM. Study carefully the 'C4x block-diagram (Figure 2-1 in the 'C4x Users Guide, 1993) to discover possible contentions.
   A double-buffering scheme with two data buffers in the system (one for CPU processing and one for DMA transfer) being switched between CPU and DMA may help in some applications. An example of this can be found in "Parallel 2-D FFT Implementation with TMS320C4x FFTs" (SPRA031). If contention with the CPU cannot be avoided, select the DMA priority (bits 0,1 in DMA control register) more convenient to your application.
2. Src and/or destination addreses are not ready.
   This may be caused by:
   - a non-zero-wait-state memory. Remember, after reset the default value for external wait states is 7, therefore you have to set the global and local memory-control registers to your specific settings.
   - an interrupt not being received if using DMA read and/or write synchronization. For example, reading (writing) with sync mode from/to the comm ports can only take place when there is data in the input FIFO (or when there is space in the output FIFO).
3. The DMA data transfer rate is slower in the sync transfer mode because it takes two cycles to reset the request from the interrupt. Therefore the maximum transfer rate in the sync mode is one word every four cycles. However, these two extra cycles can be absorbed if multiple DMA are running at the same time and most of the time the effect is neglictible. Refer to section 9.11.2 of the 'C4x Users Guide, 1993.

If none of two first factors explain why a DMA transfer never finishes, take a look at the DMA registers values. Wrong values in any of the nine DMA registers may indicate a programming error. Take a special look at the DMA control register, start and status bits to detect if the DMA has been halted.

Programming the 'C4x DMA?

The DMA is a memory-mapped peripheral. Therefore you can program it from C as well as from Assembly in a very easy way. The following examples are provided in this application note:

*** Unified Mode DMA ***

1. Example 1: Unified-mode DMA transfers data between comm ports using read sync.
2. Example 2: Unified-mode DMA uses autoinitialization (method 1) to transfer two data blocks.
3. Example 3: Unified-mode DMA uses autoinitialization (method 2) to transfer two data blocks.

*** Split Mode DMA ***

4. Example 4: Split-mode auxiliary DMA transfers data between comm ports using read sync.
5. Example 5: Split-mode auxiliary and primary channel send/receive data to and from comm ports.

6. Example 6: Split-mode DMA autoinitializes both auxiliary and primary channels (auxiliary transfers one block and primary transfers two blocks).

You can compile those examples by typing:

```
goex example1
```

(this invokes a batch file that runs the compiler and linker). Source code and batch files can be downloded from the BBS (filename: C4xdmaex.exe).

You can find examples of DMA programming in Assembly language in the 'C4x Users Guide (Chapter 12). Also, you can use a C-callable Assembly routine to achieve the same result. Refer to set_dma.asm routine in [1] for source code. Here is an example of how it can be invoked (use register for parameter passing to reduce instruction cycles):

```
set_dma(DMAADDR,CTRLREG,SRC,SRC_IDX,COUNTER,DST,DST_IDX,LINK_PTR);
```