

FPGA memory controllers improve DSP performance

By Richard M. Mathews

Signal processors spend a significant portion of time and resources moving data, shuffling it in preparation for manipulation. This inefficiency can be significantly reduced for downstream DSP processors by using a large, multi-ported memory buffer tightly integrated with a user-programmable FPGA logic block and a corner-turning Direct Memory Access (DMA) engine. This allows DSP and other processors to spend a higher percentage of time and resources on intelligent data manipulation, reducing overhead and system complexity. This article examines design issues and technology advances that can increase efficiency and optimize performance.

FPGAs are being increasingly used in DSP applications. They are especially effective at performing many kinds of repetitive operations and typically are combined with general-purpose processors for control operations and for data processing operations that require more complex decision making.

A common I/O path bottleneck problem exists in many DSP systems when moving data from node to node. Another problem occurs when a significant fraction of processing power is utilized for merely shuffling data in preparation for processing instead of using CPU resources for actual processing.

These data movement problems can be significantly alleviated by using a multi-ported memory controller that includes FPGA processing capabilities. Advantages include:

- Multiple I/O ports allow the controller to efficiently receive and transmit data on independent paths using the most practical protocol on each interlink.
- User Programmable Logic (UPL) within the controller can provide the processing power in an FPGA to offload the DSP processors.
- Large memory resources provide data buffering and temporary storage for intermediate results. Memory requirements for DSP processors can be reduced, and memory can be allocated to those processors in a manner that allows more efficient use of CPU cycles.
- Striding DMA that provides efficient corner turning to further offload processors.

FPGA-based System-on-Chip (SoC) products are starting to appear on the market and include features such as the multi-ported controller shown in Figure 1.

User programmable logic and buffer memory

Many of the data transformations performed in DSP applications are highly repetitive and are efficiently implemented in FPGAs. Fast Fourier Transforms (FFTs), convolutions, FIR filters, and

IQ demodulation are among the processing often implemented in an FPGA. The parallel processing enabled by FPGA hardware implementations allows these operations to be performed much faster than using traditional processors, and the general-purpose or DSP processors can be offloaded to perform work that is not as well-suited to a hardware implementation – such as tasks that require decisions to be made based on the data. Fewer processors are thus needed in the system to complete all operations.

The large memory on a multi-ported controller can be used in many ways. Input or output data can be rate buffered. An input device may provide bursts of data, while real-time processing proceeds most efficiently at a steady pace. The buffered input data can be sent to processing units just in time for processing. Similarly, output buffering may be needed if the output device is not ready immediately as it comes from the real-time processing system.

The memory also may be needed between stages of processing. If the UPL is programmed to perform several independent kinds of processing such as FFTs or convolutions in several dimensions, large amounts of data may need to be stored between the stages. High bandwidth paths within the multi-ported controller allow the UPL to access the data several times without ever having to send the data over the external interlinks. The intermediate data can be stored in the buffer memory until enough accumulates to begin work on the next stage. Data sizes are application dependent and can vary significantly, from tens of megabytes to several gigabytes. The buffer memory also plays an important role in corner-turning operations.

Many FPGAs such as the Xilinx Virtex-II Pro and the Virtex-4 FX include embedded processors. A multi-ported controller based on these FPGAs may use these processors for additional data manipulation. The processors can also be used for control operations such as programming DMA and processing exceptions. This can offload the other processors in the system to help them concentrate on data manipulation and improve real-time performance.

Corner-turning DMA

A common problem in DSP applications is that data must be processed in multiple dimensions. In a 2-D image, for example, processing must first be performed on each row and then on each column of pixels. To accomplish this, the pixel matrix must be transposed. This is called *corner turning*.

On a multiprocessor system, distributed corner turning is needed to accomplish the transpose across all of the processors. Each row processor is assigned a subset of the rows. Each column processor is assigned a subset of the columns. When a row processor completes a row, it breaks up the row into pieces according to

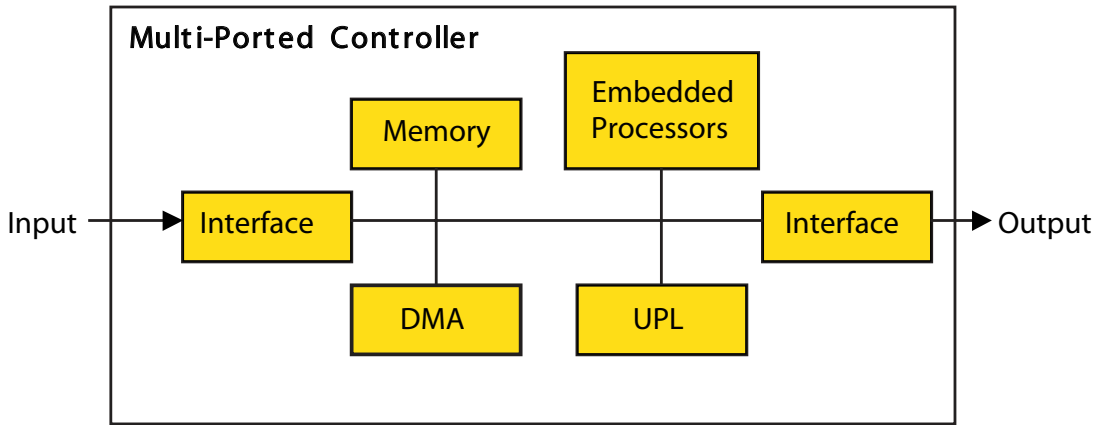


Figure 1

which column processor will work on each piece. It must then transmit each of these pieces to all of the column processors. The column processors receiving a piece of a row thus get just one element of each column. They must wait to accumulate enough rows before they can begin column processing.

This puts a large memory requirement on each column processor to buffer so many rows. Because each column processor must transpose the matrix it receives, many CPU cycles are also spent on corner turning. Figure 2 shows distributed corner turning, where data is distributed from the output of each row processor to every column processor by a chain of L DMA block transfers.

By adding a striding DMA that performs matrix transposition to the multi-ported controller, processors can be relieved of this duty.

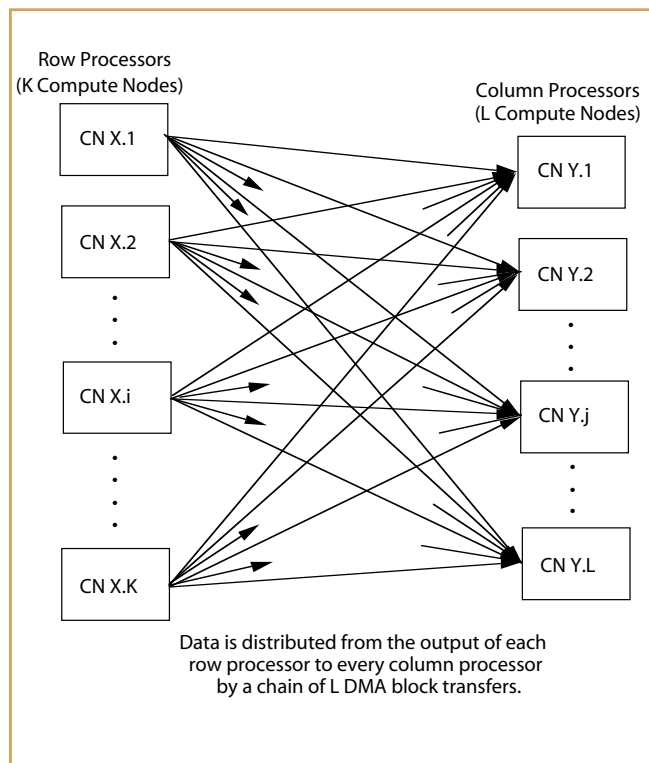


Figure 2

Since the processors can better concentrate on data transformations, fewer processors are needed to accomplish all the work. The savings on processors offset the cost of the multi-ported controller.

In such an application, the first stage of processing is performed by the FPGA, which works on rows of data received from the input device. The controller uses its large buffer memory to save the results of transformations on many rows as can be seen in the striding DMA in Figure 3. After completing the calculations on each single row of data, internal DMA built into the controller transfers the results of the calculation into buffer memory. These rows of results are collected to form a matrix in buffer memory. When enough rows have accumulated to allow efficient column processing, each column is transmitted to one of many DSP processors. The column consists of just one element, or a small number of elements, in each row.

The elements of the rows are stored consecutively in buffer memory. The consecutive elements of columns are separated in memory by the corresponding elements of all of the other columns, so they are not stored consecutively in memory. The DMA to the DSP processors must “stride” past all of these other columns. The striding DMA effectively performs a scatter-gather operation, collecting the nonconsecutive elements from the multi-ported controller’s buffer memory and transferring the elements into consecutive memory locations on the DSP processor.

Any interlink can support this striding DMA. Since the data is sent to the DSP processors in order of consecutive memory locations, it does not matter to the DSP processor or the interlink that the DMA engine in the multi-ported controller fetched the data from nonconsecutive locations in its own memory. This process implies, however, that the striding DMA must reside on the multi-ported controller.

All in stride

Figure 4 further illustrates this striding operation. The strides are local to the DMA master, so this is called *local striding*. The transferred segment size is the same on both the local system, the multi-ported controller, and the remote system. The multi-ported controller skips over the local stride during its access to its local memory. The multi-ported controller may transmit data to the DSP processors or receive data from them.

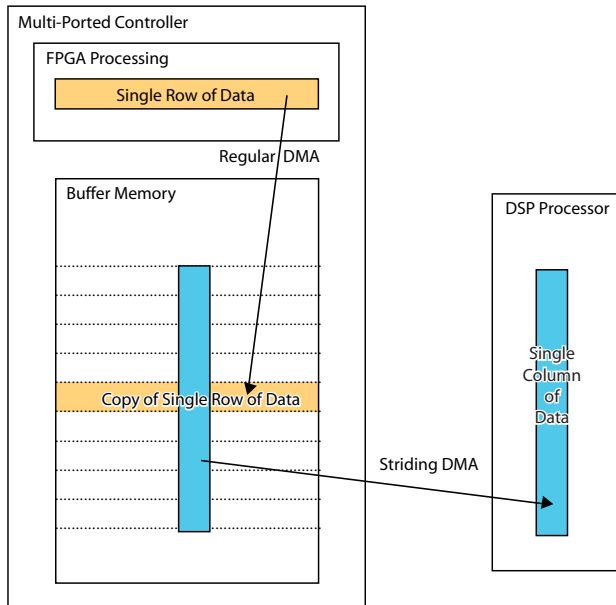


Figure 3

Figure 5 illustrates remote striding. The multi-ported controller is now shown on the right, but it is still acting as the DMA master. It must use separate transactions to access each segment on the remote system. Once again, it can transmit data or receive data. While not as efficient as local striding since separate transactions must be used for each segment, this can be useful if the remote system is not capable of mastering its own striding DMA.

Because the multi-ported controller buffers the data until a sufficient number of rows has been accumulated, the memory requirement of the column processors is greatly reduced. In some cases without a multi-ported controller, the column processors are memory-limited. Compromises have to be made to reduce the size of the rows or columns in order to keep the matrix size within the limits imposed by the system.

By using a multi-ported controller, larger rows or columns become practical because the column processors never have to deal with more than a few columns at a time. Since some operations, such as FFTs, require fewer instructions per pixel when performed on larger columns, the CPU time needed to process the data can be reduced. The result is that fewer processors, each with less memory, will be able to complete the job.

Real-world controller

While FPGA processing can reduce the need for traditional DSP processors, a multi-ported controller tightly integrated with a UPL can reduce that requirement even further. Having a multi-ported controller reduces the memory requirements for other processors, and the remaining memory can be used for longer columns. This allows for more efficient processing. Throughput is improved by using the most efficient protocol on each side of the multi-ported controller, without sacrificing efficiency to protocol translation, which eats CPU cycles.

Based on the Xilinx V-4 series of FPGAs, the CoSine SoC from Micro Memory combines UPL processing capabilities, a large multi-ported memory controller, and a striding DMA in a fully preconfigured solution (Figure 6). With this integrated functionality, CoSine provides a proven design and powerful platform for building DSP systems. †

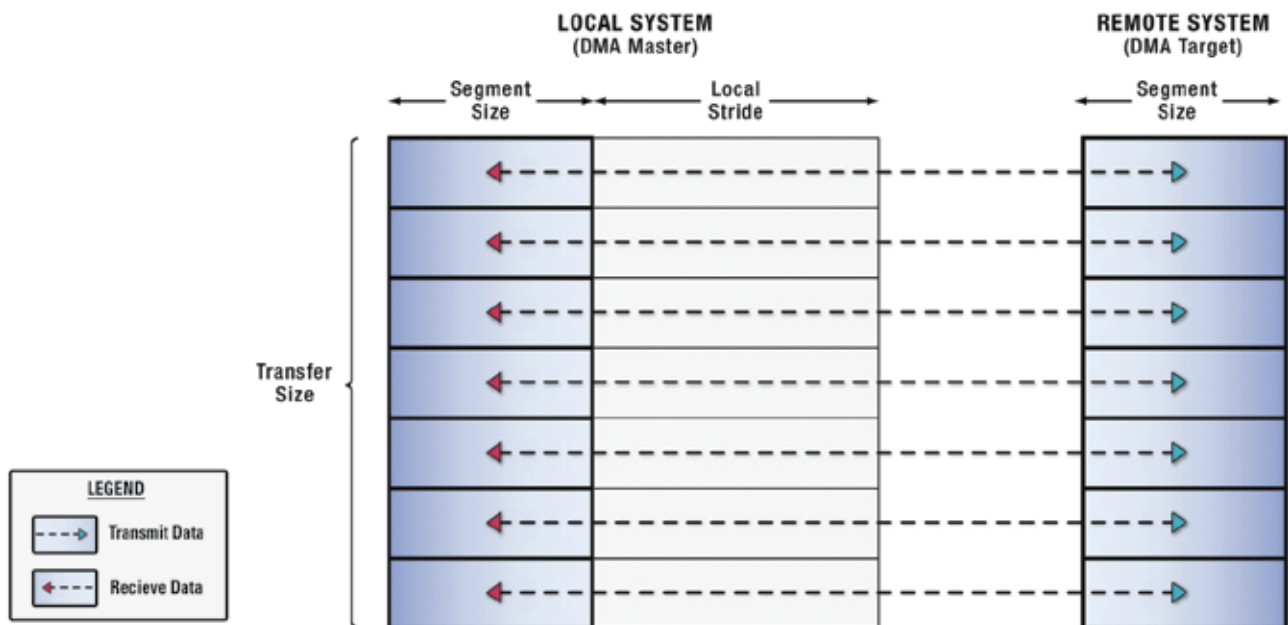


Figure 4

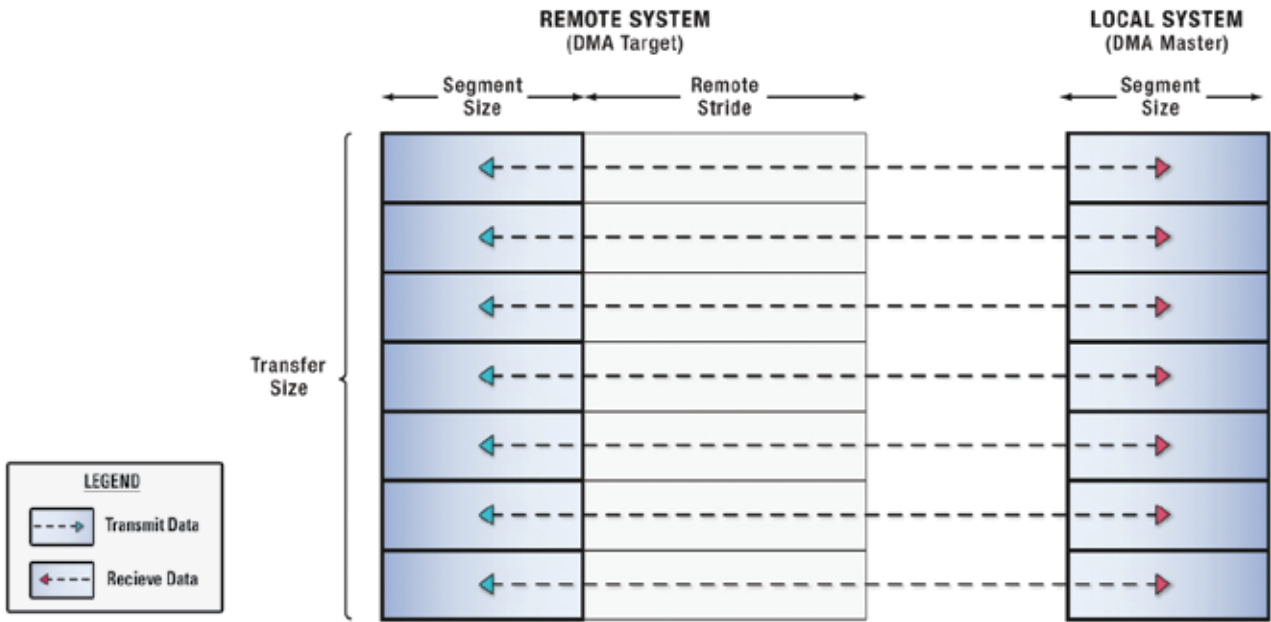


Figure 5

Richard Mathews has been at Micro Memory for the past three years, where he currently leads Software Engineering. During that time, he has played key roles in both hardware and software system architecture. Prior to joining the company, Richard spent nine years at Sun Microsystems where he was a leading member in the Solaris port to x86, developed a course for writing Solaris device drivers, and worked with Tier I OEMs that utilized the ChorusOS. After studying physics at Caltech, Richard joined Locus Computing Corp. where he specialized in clustering mechanisms, memory management, load balancing, and remote file replication.



For more information, contact:
Micro Memory, LLC
 9540 Vassar Ave.
 Chatsworth, CA 91311
 Tel: 818-998-0070
 Fax: 818-998-4459
 Email: sales@micromemory.com
 Website: www.micromemory.com



Figure 6