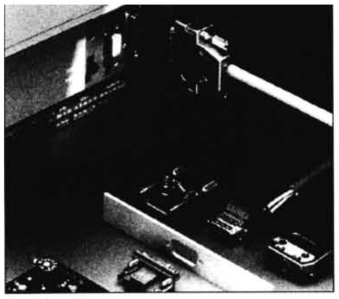

Phoenix Contact

Connectors and IC Sockets
PRODUCT FEATURE
Page 59

Switches and Circuit Breakers
NEW PRODUCT HIGHLIGHT
Page 102

Programmable ICs and Device Programmers
TECHNOLOGY UPDATE
Page 127

ep&t

SEPTEMBER 1996

ELECTRONIC PRODUCTS AND TECHNOLOGY

FEATURE PRODUCTS

- Clamp meter provides ac/dc measurements, true rms sensing
- Enclosure protects PCs in harsh environments
- Heat-shrinkable tubing operates to 185, 200C

FPGAs tackle multimedia, communications DSP jobs

by Steve Knapp, vertical applications manager, Xilinx Inc., San Jose, Calif.

During your last DSP design did you open your box of tricks to find you have too much information to process in an efficient performance and cost profile? DSP systems architects are becoming more frequently hindered by the lack of off-the-shelf performance obtainable from programmable DSP devices.

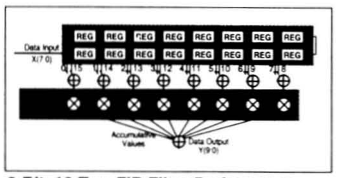
XC4000E family of FPGAs is typical of devices

Advances in multimedia and communications applications have evolved so rapidly that the performance required for these applications exceeds the processing capability that can be obtained from today's most advanced programmable DSP devices. This can lead to the assumption that using multiple DSP processors or a full-custom ASIC are the only possible solutions.

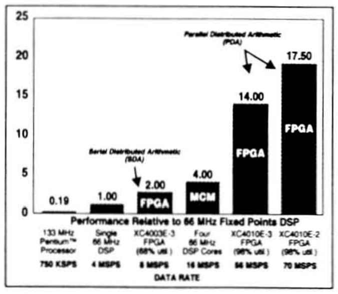
Quite often a field programmable gate array (FPGA) can offer the performance of many parallel DSP devices at a fraction of the cost, while still retaining the rapid prototyping and design flexibility not offered via ASIC design. The first assumption many engineers make is that FPGAs can't do multiply-intensive applications efficiently. Secondly, engineers may assume FPGAs are too slow and too small. These assumptions couldn't be further from the truth.

FPGAs are rapidly changing the way high-performance DSP-based systems are designed, with increased densities (greater than 50k gates), performance (70MHz system speeds) and architectural features (dedicated arithmetic support, distributed RAM, etc.). FPGA-based DSPs provide a well balanced compromise between the flexibility of a programmable DSP and the high performance of an ASIC or a multi-core DSP.

Many capabilities are built into FPGA architecture that greatly facilitate high-performance DSP design. FPGAs provide all the benefits and have all the features of a fixed point programmable DSP and more. Internal distributed synchronous RAM (a must for DSP applications) and plenty of registers (for delay lines) coupled with the lookup-table structures for efficient construction of w, x, y and z within the FPGA makes implementing parallel DSP structures a simple task.



8-Bit, 16-Tap, FIR Filter Performance Comparisons (external performance)



FPGAs can outperform traditional fixed-point DSP devices.

The performance advantage of FPGA-based DSP design comes from the ability to build parallel data flow structures internal to the device. Even when a function might be slower than that of a dedicated DSP, the ability to perform 10s to 1000s of the same function in parallel results in magnitudes of performance gain over a dedicated DSP. This is because a DSP device must process data sequentially.

DSP designers commonly run into problems when they use traditional DSP approaches to implement DSP algorithms in an FPGA. In high-performance design, the key is to craft the DSP algorithm into the unique architecture of the FPGA. Rather than using a forced approach, designers can use a vast supply of RAM or look-up-table (LUT) structures to their advantage.

If you needed a finite impulse response (FIR) filter or a circuit that adds the products of several data samples, for example, there are a couple of methods that could be used. One approach is to enter the design with either a schematic or HDL using any of the many electronic design automation (EDA) tools available, as well as a DSP library, if available. The problem here is that DSP design tools are typically optimized for ASIC or CPU-based DSP devices and don't work efficiently with multiplier-rich FPGA-based DSP interpretation.

A less complicated approach would be a 16-tap FIR filter. The designer could build a structure that would add the products of 16-data samples, each multiplied by a corresponding coefficient and added together. The brute force approach would consist of 16-multipliers and an adder tree of 15-adders. Another approach might be to use a multiplier and an accumulator to build a time shared circuit similar to a CPU-based structure.

The biggest issue with either of these two approaches is the multiplier blocks. Many designers will question how to build a multiplier in an FPGA. If a designer is using HDL for design entry, the synthesis-optimization tools construct a quasi ASIC-optimized gate-level solution which isn't efficient in an FPGA.

And, with schematic entry, the designer would have to start from scratch and create a multiplier or use the library block optimized for a multiplier. There is a remote chance that a designer might find an FIR filter block which is again optimized for gate level implementation and not an FPGA.

The brute force design approach might offer a faster data throughput, but there is a downside: a significantly larger throughput than the CPU-based structure. To optimize the design for an FPGA, the designer must build an efficient multiplier for speed and/or size.

Using an FPGA-based DSP solution can provide designers with a scaleable methodology for DSP design implementation. The key design technique is through the use of distributed arithmetic. The scalability of bit-serial and bit-parallel distributed arithmetic allows the designer to optimize the design for performance and density.

Bit-serial distributed arithmetic is an implementation technique which processes parallel data flow structure bit-sequentially. This process allows multiple functions to be performed simultaneously in a small amount of silicon by sharing resources.

In the example of the 16-tap FIR filter, all of the 16-data samples are multiplied in parallel in a bit-serial process. Bit-parallel distributed arithmetic is a similar technique to that of bit-serial with multiple bits being process in parallel. This enables performance scalability, where the overall performance of the design implementation will scale proportionately to the required resources.

If the designer has carefully selected the bit-level of distributed arithmetic, the DSP function is implemented in an optimized FPGA-based DSP solution based on performance requirements. At the higher performance levels, 30-70 million samples per second (MSps), the FPGA can be partitioned to perform all operations in parallel to minimize the number of clock cycles required to perform a function. While at the lower performance levels, 1-10MSps, the FPGA can use bit-sequential operations allowing more efficient resource sharing.

There are distinct cost and performance benefits for an FPGA-based DSP solution. In terms of performance, theoretically, most DSPs can perform a multiply and accumulate (MAC) function every clock cycle. For example, a 66MHz DSP 16-tap FIR filter would give a theoretical maximum sample rate of 66 MHz/16=4.125MSps. In contrast, with FPGA-based DSP, the designer could use bit-serial distributed arithmetic (at 15ns per bit at 66MHz) where all 16-taps are processed in parallel. For 8-bit data this equates to 15x8=120ns or 8.33MSps, twice the sample rate of the 66MHz programmable DSP device.

For even greater performance, the distributed arithmetic technique can be scaled from one to n-bits, for n-bit data, inclusive. As the number of bits processed are scaled, the performance increases from 8 to 66MSps. The full-parallel distributed arithmetic implementation can operate at 16 times the data rate compared to a 66MHz programmable DSP device.

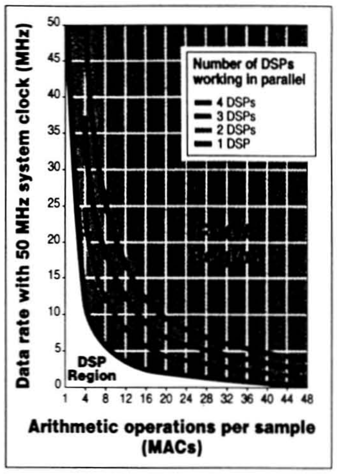


Chart shows in which situations FPGAs are suitable for DSP implementation.

If a designer used multiple or a multi-core DSP, the performance would attempt to track linearly with the number of processors. But even a multi-core DSP doesn't compare with the solid performance achieved when using an FPGA-based DSP design.

While FPGA-based DSPs often provide superior solutions, it should be noted that some designs have functions that are better handled by a DSP, for example, floating point arithmetic operations.

Often, the DSP device can be enhanced by using an FPGA to accelerate functions that bottleneck the data flow. Additionally, using an FPGA-based DSP coprocessor can increase overall throughput by more than 10 times that of a standalone or even multiple DSP devices.

The real benefit of an FPGA-based DSP is achieving higher performance without having to resort to an ASIC and its high NRE, long prototyping lead-times or inflexibility.

Designers have been using FPGA-based DSP as a competitive advantage for performance, price and flexibility in their applications for several years. If your bag of tricks doesn't include an FPGA-based DSP, you may need a new bag.

For more information on Xilinx FPGA devices, circle Reader Service Number 424.