EVALUATING FPGAS FOR COMMUNICATION INFRASTRUCTURE APPLICATIONS

Mel Tsai (Berkeley Design Technology, Inc., Berkeley, CA, USA; tsai@bdti.com)
Jeff Bier (Berkeley Design Technology, Inc., Berkeley, CA, USA; bier@bdti.com);
and Jennifer Eyre (Berkeley Design Technology, Inc., Berkeley, CA, USA; eyre@bdti.com);

ABSTRACT

Until recently, FPGAs were rarely used for signal processing. FPGAs were held back by factors such as limited capacity, design flows that were unfamiliar to DSP engineers, and a lack of DSP-related intellectual property libraries. In the last few years, though, a growing number of FPGAs and related products targeting DSP applications have begun to address these shortcomings. At the same time, the requirements of important emerging DSP applications like software-defined radios have begun to exceed the capabilities of traditional DSP processors, motivating system developers to consider alternatives.

In this paper we examine the key requirements of communications infrastructure applications targeted by FPGAs. We present a methodology for evaluating FPGAs for these applications using metrics such as capacity and cost/performance, and use this methodology to evaluate the latest DSP-enhanced FPGAs from Altera and Xilinx. We also compare FPGAs and their associated DSP-oriented development tools to products offered by DSP processor vendors, and assess the features, strengths, and weaknesses of key products in both categories. Finally, we discuss key technology trends pertinent to FPGAs and DSPs.

1. INTRODUCTION

Implementing the digital signal processing (DSP) tasks in communications applications typically requires chips with very strong number-crunching capabilities. At the same time, communications applications place stringent constraints on cost and power consumption. DSP tasks in telecom products have historically been implemented using DSP processors (often referred to as “DSPs”) or application-specific integrated circuits (ASICs). ASICs can achieve high levels of performance with hard-to-match cost and energy efficiency, but they require massive design efforts. DSPs, on the other hand, ease the development process, and provide adequate performance and reasonable efficiency for many applications.

Throughout most of their history, field-programmable gate arrays (FPGAs) have rarely been used to implement DSP tasks, for a number of reasons. Until fairly recently, FPGAs lacked the gate capacity to implement demanding DSP algorithms and did not have good tools support for implementing DSP tasks. They have also been perceived as being expensive and power hungry. All this may be changing, however, with the introduction of new DSP-oriented products from FPGA vendors like Altera and Xilinx.

2. DSP APPLICATION REQUIREMENTS

To help in understanding when it is appropriate to use an FPGA for the DSP tasks in a communications infrastructure application, it is useful to understand the unique demands DSP applications place on an implementation technology. (By “implementation technology,” we mean the type of device—e.g., microprocessor, FPGA, ASIC, etc.—that is used to implement the DSP functionality.) There are several common requirements that set DSP applications apart and make typical DSP applications particularly challenging to implement. These requirements drive the choice of implementation technology, and include:

Real-time processing requirements. Most DSP applications process signals in real time, placing strict constraints on the time available to process each digital data sample. In particular, communications infrastructure equipment must be capable of processing multiple independent channels simultaneously and in real time. Indeed, these requirements often exceed the performance capabilities of even the fastest DSP processors.

Diverse computational requirements. The sample rates (that is, the rate at which new samples arrive) in communications infrastructure applications can range up to the hundreds of MHz. Thus, these applications are very computationally intensive, performing repetitive arithmetic operations over vast amounts of data. This is distinctly different from typical computer software, for example, which tends to rely more heavily on decision-making (if-then-else) processing. A basic operation used in many DSP algorithms is the multiply-and-accumulate (MAC) operation. Hence, it is highly desirable for many DSP applications to be able to execute many MACs per second and this metric is sometimes used as a proxy for the DSP performance of a device. DSP applications also often make heavy use of bit-manipulation operations, memory accesses, and various arithmetic operations.

Numeric fidelity. Different DSP applications require varying levels of numeric fidelity. For example, signal fidelity in an OFDM (orthogonal frequency division multiplexing) receiver will be limited by communication channel noise and analog to digital converters; the receiver implementation may therefore tolerate relatively low numeric fidelity. Numeric fidelity in a DSP algorithm implementation is affected by the precision and dynamic range of the data, which are in turn a function of the format of the data word (its length in bits and whether it is a fixed-point or floating-point format). 16-bit data word widths (and occasionally 24 or 32 bits) are commonly used in communications applications implemented using DSPs.

High memory bandwidth. DSP applications process vast amounts of data at a high rate. The ability to bring large amounts of streaming data on and off the chip, and keep on-chip computational resources fed with data, is essential to meeting computational demands and real-time constraints.

Low cost. Hardware cost is a key consideration in most applications. However, other costs may become the overriding concern in a communications infrastructure application. For example, in low-volume applications, development costs can be more important than bill of materials costs. When evaluating chip costs in communications infrastructure applications, cost per channel can be more important than simple chip cost.

Low energy consumption. Communications infrastructure equipment may be sensitive to energy consumption due to heat dissipation and power supply design considerations. Like cost, energy efficiency is often evaluated on a per-channel basis rather than per chip in these applications.

Reprogrammability. New and emerging communications applications often are based on evolving standards and other changing requirements, and equipment developers may have little advance knowledge about how their product may need to adapt to new standards and changing requirements in the future. This lack of predictability is at odds with time-to-market pressures; developers often must design a product before the standards and requirements are fully fixed. These conflicting objectives motivate use of a technology that allows field upgrades so that unanticipated future demands can be met after product delivery. In addition to the challenge of adapting to evolving standards, the high complexity of many communications infrastructure systems (such as third-generation cellular base stations) makes it nearly impossible to identify all of the bugs and implement all of the desired functionality before the product reaches the market. Use of a reprogrammable technology allows the developer to fix bugs in the field and add features after the product has been released.

The relative importance of each of the above requirements varies depending upon the specifics of the application. Thus, the designer must weigh each of these (and possibly other) requirements carefully before choosing an implementation technology.

3. FPGA TECHNOLOGY OVERVIEW

An FPGA is a chip composed of an array of configurable logic blocks (also called logic cells), programmable interconnect resources, I/O blocks, and sometimes embedded specialized fixed-function blocks. The basic structure of an FPGA device is shown in Figure 1. Each logic block can be configured, or programmed, to perform one of a variety of simple functions, such as computing the logical AND of two inputs. Configuration information that is read by the device when it is powered up specifies the configurations of each of the logic blocks and the connections between them, and thus specifies the functionality that is implemented by the FPGA. Hence,

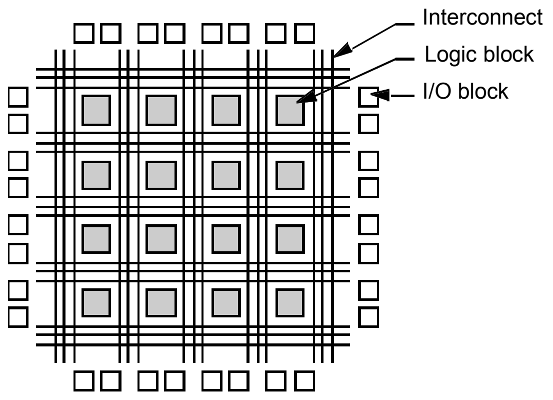

Figure 1. Simplified FPGA structure.

the FPGA's logic blocks can be used as building blocks to implement any kind of functionality desired, from low-complexity state machines to complete microprocessors. However, FPGAs are limited; the clock speed at which an FPGA executes is determined in part by the functionality it is implementing (that is, by its configuration), and each FPGA provides a finite set of logic blocks and interconnect resources.

A key advantage of FPGAs is their flexibility; an FPGA can be configured to match the requirements of an application. This advantage does not mean that FPGAs are appropriate for every application, however. This is because, in general, there is a tradeoff between a device's flexibility and its efficiency. FPGAs are extremely flexible, but because they are reconfigurable and not optimized for a specific task, they are typically not as efficient (in terms of speed, cost, power consumption, and/or die size) as devices that are based on fixed-function hardware, such as ASICs. In this respect, the FPGA is a jack of all trades, but master of none. Another tradeoff exists between device performance and application development effort. While FPGAs can achieve much higher performance than programmable DSPs or general-purpose processors (GPPs), developing DSP applications for FPGAs is generally far more challenging.

4. FPGAS FOR DSP

FPGAs were not originally designed with the needs of DSP applications in mind, but a number of significant advances in FPGA technology have improved the suitability of FPGAs for DSP. These advances include:

Increased Capacity. In the past, FPGAs simply did not have the capacity (in terms of the number of logic cells) needed to implement challenging DSP algorithms. This was a fundamental limitation that prevented their use in DSP applications for which they would otherwise be well suited. Today's FPGAs, however, have capacity far in excess of that available in FPGAs of even a few years ago and are now able to accommodate complex DSP functionality. For some applications, increased capacity also yields increased application performance.

Increased Speed. Historically, FPGAs were not able to execute DSP tasks fast enough to meet the real-time constraints of many DSP applications. One reason for this is that FPGA chips did not operate at particularly high clock frequencies (for example, relative to microprocessors). Recent improvements in the process technologies used to fabricate FPGAs have increased their operating frequencies, helping to boost their performance. Clock speed is only one factor in application performance, however. Equally important is the amount of work the chip can execute in each clock cycle. FPGAs have inherently highly parallel architectures; that is, they can execute many operations in parallel. Thus, in applications that can benefit from parallel processing (such as multi-channel applications), the new high-capacity FPGAs (even without DSP-oriented enhancements) can perform much more work per clock cycle than most processors.

Increased memory bandwidth. Older FPGAs did not have sufficient memory bandwidth to meet the needs of demanding DSP applications. Recent FPGAs address this problem by including numerous hardwired memory blocks (embedded within the logic array). The new high-capacity FPGAs have far higher memory bandwidth than DSP processors. This is an important advantage, because many DSP applications are heavily data-intensive.

Better DSP-oriented tools. Until recently, FPGA development tools provided little specialized support for development of DSP applications. For example, the tools did not explicitly support implementation of common DSP algorithms, such as filters. In addition, the tools were not integrated with other tools commonly used to develop DSP algorithms, such as MATLAB. As a result, an engineer who needed to implement a DSP algorithm with an FPGA would typically first design and simulate the algorithm using software such as MATLAB, then manually create an HDL (hardware description language) description of the algorithm for implementation with an FPGA—a time-consuming, error-prone process. Recent improvements in FPGA tools have significantly improved their usability for DSP applications.

Increasing availability of IP libraries. Intellectual property (IP) modules can significantly accelerate design cycles by offering carefully designed and tested functional blocks. Until recently, there were few DSP-oriented IP modules available for FPGAs, forcing engineers to develop nearly every function from scratch. This process was particularly painful because of the lack of DSP-oriented development tools. In comparison, the supply of optimized algorithm implementations for DSP processors from DSP manufacturers and from third-party developers has been much richer. This is another area in which FPGAs have recently made significant strides, with both of the major FPGA vendors (Altera and Xilinx) now offering libraries of DSP-oriented IP modules.

Architectural enhancements for DSP. Altera's recently announced Stratix FPGA family and Xilinx's Virtex-II family both offer significant DSP-oriented architectural enhancements. For example, both products offer hard-wired on-chip multipliers (embedded throughout the reconfigurable logic array) intended to accelerate the multiply-accumulate (MAC) and similar operations common in DSP algorithms. By including some hardwired processing elements, FPGAs can improve their energy efficiency and cost/performance while offering outstanding DSP performance.

5. EVALUATING FPGA PERFORMANCE

The computational requirements of today's communications applications often exceed the performance available from even the fastest DSP processors. This makes the new breed of DSP-enhanced FPGAs a potentially attractive solution for certain applications. A key challenge for system designers, though, is understanding where it is appropriate to use these new devices. Unfortunately, designers have been stymied by the lack of a reliable way to evaluate the DSP performance of FPGAs or to compare their performance to that of DSP processors. Clearly, there is a need for DSP-oriented benchmarks that will enable engineers to make these comparisons.

5.1. FPGA Benchmarking Methodology

Good benchmarking requires careful selection of benchmarks and a well-developed methodology. BDTI's well-established benchmarking of processors for DSP applications uses a suite of common DSP algorithms, such as finite impulse response (FIR) filters, optimized in assembly language on each processor. A processor's results on each benchmark can be thought of as "basis vectors" that can be combined to estimate performance in an application.

Although the computation requirements of a typical DSP application are dominated by a handful of algorithms, individual algorithm kernels are not suitable as benchmarks for high-capacity FPGAs, for several reasons [1]. With a DSP processor, application developers aggressively optimize each of the key performance-hungry algorithms for speed. When a particular algorithm is running, it has exclusive use of all of the processor's execution units. With an FPGA, in contrast, designers have the flexibility to trade off parallelism (and hence performance) against resource (logic blocks, multipliers, etc.) utilization. Thus, unlike on a DSP, it makes little sense for a single algorithm to consume all of an FPGA's resources because no resources would remain for the rest of the application. Instead, the designer must optimize the application as a whole, allocating the available hardware among each of the constituent algorithms. These observations lead us to conclude that a benchmark for FPGAs must look more like a *complete application* and less like a single algorithm kernel.

5.2. Performance Metrics

Although they are powerful, the latest DSP-enhanced FGPAs are also expensive. The least expensive DSP-enhanced FPGAs such as those in the Xilinx's Virtex-II [2] and Altera's Stratix [3] families are priced in the hundreds of dollars, and the most expensive family members cost thousands of dollars per chip. Such prices render these chips unsuitable for highly cost-constrained products like cable TV set-top boxes or DSL modems. But in communications infrastructure equipment, a chip is not automatically disqualified due to high cost—especially if a single chip can handle the processing for many communications channels. Thus, benchmark results should be reported in terms of the number of channels that can be supported on a single chip and the associated cost per channel (based on the chip cost). These results can be used to compare an FPGA's performance to that of DSPs.

5.3. Benchmark Development

BDTI recently developed a new communications-oriented benchmark. Rather than using a single algorithm as a benchmark, The BDTI Communications Benchmark^{TM} models a single-channel OFDM receiver, as shown in Figure 2. OFDM is a complex technique that is finding increasing use in a variety of high-speed data communications applications. Thus, this benchmark is

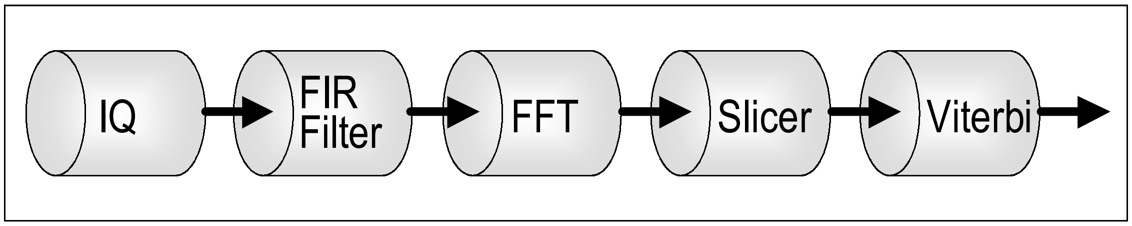

Figure 2. BDTI's Communications Benchmark^{TM}.

The benchmark is a simplified single-channel receiver, shown in the block diagram above. The IQ block performs demodulation into in-phase (I) and quadrature (Q) signals; the Slicer block maps fast Fourier transform (FFT) outputs to points in a QAM (quadrature amplitude modulation) constellation. [4]

designed to be representative of the kinds of processing found in communications equipment for applications such as DSL, cable modems, and fixed wireless systems. The benchmark includes blocks for demodulation, filtering, time-frequency domain transformation, and channel decoding. Input and output data formats and sample rates, along with other implementation details, are specified as part of the benchmark definition. Benchmark implementers are tasked with implementing as many channels of the receiver as they can fit onto a single chip.

6. RESULTS AND CONCLUSION

BDTI invited Altera and Xilinx to implement the BDTI Communications Benchmark on their DSP-enhanced FPGAs. BDTI also invited Motorola and Texas Instruments to implement the benchmarks on their high-end DSPs, which target communications infrastructure equipment. Altera and Motorola took up our challenge, and each delivered a highly optimized implementation of the benchmark.

The FPGAs excelled in this benchmark. A typical member of Altera's Stratix family of FPGAs is projected to handle dozens of communications benchmark channels. In contrast, a high-end DSP could not support even a single a single channel [5]. With DSPs falling short of the needs of today's most demanding applications, such performance levels can make FPGAs an attractive solution.

While some high-capacity FPGAs carry staggering, multi-thousand-dollar price tags, both Altera and Xilinx offer DSP-enhanced FPGAs with prices in the low hundreds of dollars, which puts them in the same price range as many high-end DSPs. A huge performance advantage combined with comparable prices leads to a huge advantage in terms of cost/performance for DSP-enhanced FPGAs in many applications.

Our initial benchmarking work suggests that the new DSP-enhanced FPGAs can indeed achieve impressive performance in certain types of DSP applications. But our experience with these new devices, and discussions with users, indicate that factors other than performance are often decisive in decisions regarding use of an FPGA. For example, one key challenge facing DSP applications developers using FPGAs is the relative complexity of the design process and lack of DSP-specific features in the development tools, compared to what is available for the best-supported DSPs.

Clearly, as with most technology-selection choices, the decision of whether to use an FPGA for a DSP application requires a sophisticated, multidimensional evaluation—one that depends on a large number of specifics of the target application. Thus, although benchmark results are important, there are many "soft" considerations that are of equal importance when choosing between an FPGA and a DSP processor.

One of these considerations is the availability of relevant staff expertise. For example, most DSP application developers are not familiar with the design flow for FPGAs. Implementing even a simple FIR filter on an FPGA requires a totally different design process (and mind-set) than implementing the same function on a DSP processor. Altera and Xilinx offer tools and libraries to help simplify the process, but there will be a formidable learning curve for engineers who are primarily accustomed to working with processors. In addition, the time required to develop an optimized implementation of even a relatively modest DSP function for an FPGA can be dramatically longer than that required to write an optimized version for a DSP. For example, one source told BDTI that it can take six man-months to develop an optimal Fast Fourier Transform (FFT) implementation for an FPGA, compared to our own experience of approximately one week of development for a high-end DSP. Altera's and Xilinx's libraries of functions help address this issue—but often, the function that is needed is not exactly what is in the library, or is not in the library at all.

7. REFERENCES

[1] J. Bier, "Evaluating Performance: FPGAs vs. DSPs," *EDN*, Reed Business Information, Highlands Ranch, Colorado, October 3, 2002 (forthcoming).

[2] Xilinx Inc., *Xilinx Virtex-II Series FPGAs*, http://www.xilinx.com/publications/matrix/virtex_color.pdf, Xilinx Inc., San Jose, California, 2001.

[3] Altera Corporation, *Stratix: New Levels of Integration*, http://www.altera.com/literature/br/br_stx.pdf, Altera Corporation, San Jose, California, March 2002.

[4] J. Eyre, "FPGA/DSP Blend Tackles Telecom Apps," *EE Times*, CMP Media LLC, Manhasset, New York, pp. 49-50, July 1, 2002.

[5] G. Beloev et al., *Focus Report: FPGAs for DSP*, Berkeley Design Technology, Inc., Berkeley, CA, 2002 (forthcoming).