Microprocessors or FPGAs?: Making the Right Choice

By: Steve Edwards, Curtiss-Wright Controls Embedded Computing

The “right choice” is not always either/or. The right choice can be to combine the best of both worlds by analyzing which strengths of FPGA and CPU best fit the different demands of the application.

There has been stunning growth in the size and performance of FPGAs in recent years thanks to a number of factors, including the aggressive adoption of finer chip geometries down to 28nm, higher levels of integration, the use of faster serial and communication links, specialized cores, enhanced logic and innovative designs from the major FPGA vendors. Meanwhile, the overall performance growth curve of traditional microprocessors has somewhat flattened due to power density hurdles, which have limited clock rates to around 1.5-2 GHz because the faster the new processors go the hotter they get. This power density barrier has been somewhat mitigated by the emergence of multicore processors, but as the number of cores increases these devices bring their own set of issues including how to make optimal use of their parallelism while operating systems and automatic parallelization tools lag far behind.

Improvements in FPGAs have driven a huge increase in their use in space, weight and power (SWaP) constrained embedded computing systems for military and aerospace applications. They are ideal for addressing many classes of military applications, such as Radar, SIGINT, image processing and signal processing where high-performance DSP and other vector or matrix processing is required. SWaP performance rather than cost is the key driver for this. Given this trend, system designers are more frequently asking the question, “Should I use FPGAs or microprocessors for my next embedded computing project?” And the answer today is, “it depends.”

FPGAs are an increasingly attractive solution path for demanding applications due to their ability to handle massively parallel processing. FPGAs tend to operate at relatively modest clock rates measured in a few hundreds of megahertz, but they can perform sometimes tens of thousands of calculations per clock cycle while operating in the low “tens of watts” range of power. Compared to FPGAs, microprocessors that operate in the same power range have significantly lower processing functionality. Typically, a similarly power rated microprocessor may run at 1-2 GHz clock rate, or roughly 4 or 5 times as fast as an FPGA, but it will be much more limited in how many operations it can perform per clock cycle, with a maximum typically in the range of four or eight operations per clock. This means that FPGAs can provide 50 to 100 times the performance per watt of power consumed than a microprocessor. This might seem to give FPGAs an unbeatable edge over microprocessors, but the advantage of FPGAs in these applications is, unfortunately, not that clear cut. Despite their apparent computational strengths, there are three key factors that determine the utility of FPGAs for a particular application. These factors are algorithm suitability, floating point vs. fixed point number representation, and general difficulties associated with developing FPGA software. The embedded system designer must carefully weigh the benefits of FPGAs against the real-world limits and the challenges of hosting their application on these devices.

The first question to consider is for which types of algorithms FPGAs are best suited? FPGAs work best on problems that can be described as “embarrassingly parallel,” that is, problems that can be easily and efficiently divided into many parallel, often repetitive, computational tasks. Many applications in the military and aerospace environment fall into this class of problem, including radar range and azimuth compression, beamforming and image processing. On the other hand, there are types of computational problems for which FPGAs are not ideal, such as target classification and moving target indication problems.

Problems like these, that are by nature unpredictable and dynamic, are much better performed on traditional microprocessors because they require a more dynamic type of parallelization that is unsuitable for FPGAs, whose strongpoint is repetitive operations. With the advent of multicore processors that will soon be offered with 16, 32, or more cores, the question of how parallel an algorithm is will become more and more important. With more cores microprocessors become increasingly less suitable to address dynamic parallelization problems because the challenge of distributing the problem between the various cores can become intractable.

A second key issue when considering a choice between FPGAs and microprocessors, is the fact that FPGAs are not particularly well suited to floating point calculations, which microprocessors address with well developed vector math engines (Intel’s AVX and Power Architecture’s Altivec). While FPGAs can perform these types of calculations, it requires an undue amount of logic to implement them, which then limits the calculation density of the FPGA and negates much its computational advantage and value. If an application requires high-precision floating point calculations, then it is probably not a good candidate for implementation on an FPGA. Even high-precision fixed-point calculations require a large quantity of logic cells to implement (Figure 1).

The third key factor in determining whether an FPGA or a microprocessor is better suited for a project is the degree of difficulty that confronts a system designer tasked with implementing the application and the talent and resources available for the work. The challenges of designing with traditional microprocessors are well established and long familiar. On the other hand, while FPGA development tools have improved dramatically over the last few years, it still takes specialized talent to develop code for an FPGA. And even with expert talent, FPGA development often takes much longer than an equivalent development task for a microprocessor using a high-level language like C or C++. This is partly due to the time and tedium demands of the iterative nature of FPGA code development and the associated long synthesis/simulation/execution design cycle. System developers need to weigh the computational or SWaP density benefits that an FPGA brings to an application against the cost of development.

In addition to these three main issues, there are other less critical considerations, such as which sensor interfaces are required by an application, which may also play into the FPGA vs. microprocessor decision. For example, if custom or legacy interfaces such as serial front panel data port (SFPDP) are needed for the application and which may not be supported on a modern processor, it might be preferable to use an FPGA. FPGAs have an inherent flexibility that enables them to be tailored to connect directly to a sensor stream as in the above case of legacy or custom interfaces. Microprocessors are limited to the interfaces provided on-chip, such as PCI Express, Serial RapidIO, or Gigabit Ethernet. FPGAs, on the other hand, often provide high-speed SERDES links that can be configured as a wide range of standard interfaces. In addition, the large number of discrete I/O pins on an FPGA can often be used to implement standard or custom parallel bus interfaces. When this I/O flexibility is combined with directly attached memories, SRAM for speed and random-access, SDRAM for memory depth, etc., FPGAs can be a powerful tool for front-end processing and a much better choice than a microprocessor.

Once these factors have been weighed, it is clear that both FPGAs and microprocessors have clear, competitive advantages for different applications. Given all this, it seems that a system designer might have difficulty deciding whether to pick FPGAs or microprocessors for their application. The good news is that the designer can, in fact, have both. In the past, systems tended to be homogeneous—that is, composed of one type of processing element. Today, systems designers have a wide range of products to choose from and can often mix-and-match computing components to get just the right mix of computing and I/O to meet their application, SWaP and ruggedization requirements. They can choose from a mixture of board types based on a common communications fabric like Serial RapidIO or PCI Express.

Another option, which provides the best of both worlds, is a hybrid FPGA/microprocessor-based board like Curtiss-Wright Controls's CHAMP-FX3. This rugged OpenVPX 6U VPX board features dual Xilinx Virtex-6 FPGAs and an AltiVec-enabled dual-core Freescale Power Architecture MPC8640D processor (Figure 2). It provides dense FPGA resources combined with general-purpose processing, I/O flexibility and support for multiprocessing applications to speed and simplify the integration of advanced digital signal and image processing into embedded systems designed for demanding Radar Processing, Signal Intelligence (SIGINT), ISR, Image Processing, or Electronic Warfare applications.

Using a hybrid FPGA/microprocessor board, designers can customize their system, providing FPGA components where they make sense with more general-purpose microprocessors or DSPs for the portions of an application for which they are best suited, all provided by standards-based components. This is an especially powerful approach when there is communications middleware and validated software drivers available to provide all of the common data movement and synchronization functions.

Today's embedded computing system designer has a wide range of computational and I/O components to choose from in a number of standards-based form factors. Suppliers are offering more choices with greater flexibility to mix components to solve any particular problem. In the near future, the market will likely see more and more product type fragmentation, with a myriad of different mixes of microprocessors and FPGAs offered on both the component and the board level. FPGAs will likely feature many features of traditional microprocessors and vice versa. While more options will surely make the decision process more complex, the wider range of choices in FPGAs and microprocessors promises to empower innovative designers, enabling them to solve more difficult problems than they could solve before.

Curtiss-Wright Controls Embedded Computing
Ashburn, VA.
(613) 254-5112.
[www.cwcembedded.com].