

The future of high-performance COTS signal processing

Hybrid FPGA/DSP architecture: the optimal solution

By Jeffrey Milrod

Much energy, hype, and column space has been focused on pitting DSP and FPGA-based signal processing solutions against one another. But FPGAs and DSPs are fundamentally different technologies with different strengths and weaknesses and neither one of them can claim outright superiority today. It has become increasingly clear that the best approach is a combination of the two technologies, leveraging the strengths of both.

Until recently the basic architectural approach to signal processing COTS products was to put as many DSP chips on a given board format as possible. In the last few years, the same basic approach has been applied with FPGAs in place of programmable DSPs. This approach is simply brute force: use more DSPs or use bigger FPGAs. Subsequently, each new COTS signal processing board generation has become more complex, more expensive (relative to other boards), and more power hungry.

Despite this, application requirements are still outpacing technology improvements – clearly the old paradigms aren't working and a different approach must be developed. Given that both technologies have their place in a signal processing design, this "different approach" needs to be a new hybrid architecture that uses the correct DSP and FPGA technology to solve the correct problem.

Embracing differences

The FPGA vs. DSP debate has been an endless source of discussion in recent times. It may be that FPGAs ultimately obsolete DSPs for high-end applications, or it could be that DSP vendors will provide new price, performance, or power advantages that FPGAs can't compete with. However, we can't predict the future, and in this case it is pointless to try to do so since the two technologies are fundamentally different, each with their own strengths and weaknesses.

It is generally understood that FPGAs are better for high data rates, parallelism, processing densities, and I/O flexibility, while DSPs have advantages in power consumption, ease-of-development, and floating point. FPGAs tend to be better at solving straight forward, well defined, high-speed problems while DSPs are better to implement complicated algorithms that may evolve, involve decision making, or can benefit from the use of floating point.

Most real-world embedded signal processing applications have components that exhibit both of these natures. Therefore, rather than competing these technologies against one another, it is usually better to constructively embrace these differences by leveraging the strengths of both technologies while mitigating their respective weaknesses.

While this has been done for a while at a system level by putting a DSP board in one slot and an FPGA board in another, this loosely coupled approach is *still fundamentally an inelegant brute force technique*, and often induces longer latencies and lowers bandwidths while increasing costs and power consumption.

A better approach to achieve synergy is to embrace the differences by combining both FPGA and DSP compute elements onto one 'hybrid architecture' signal processing card.

Hybrid architecture issues

The concept is simple enough – put DSPs and FPGA together on the same COTS board. However, for this to be truly synergistic and effective, the COTS vendor must carefully navigate a sea of tradeoffs. Major issues including I/O interfacing, interprocessor communication, memory configuration, host interface, control, and an FPGA framework must all be carefully thought out, along with unifying software that makes it all accessible to the user.

I/O interfacing

Given the previously mentioned generally understood advantages of FPGAs, it clearly makes architectural sense to place them close to the signal I/O. In fact, many COTS boards already have FPGAs on them to facilitate I/O reconfigurability. By simply placing the FPGA at the point of data ingress/egress, virtually any data transfer mechanism and/or protocols can be supported, including such diverse methods as PCI, PCI Express, USB, GigE, and Serial RapidIO. This is especially advantageous on newer board formats like VITA 41, VITA 46, and AMC, which support high speed serial, or SerDes communication ports that can implement several different protocols. The FPGA can be reconfigured to allow the board to support a different dot spec and network.

An additional architectural advantage of placing the FPGA at the point of I/O is to facilitate the type of high-speed front-end data reduction, pre-processing problems that FPGAs are so good at solving.

Hybrid architecture optimization issues	
Interface	Host interface
Interprocessor communication (IPC)	Control
Memory configuration	FPGA framework

Inter-Processor Communications (IPC)

Aside from the choice of DSPs and FPGAs themselves, the hybrid architectural issue that most directly impacts signal processing performance is the communication between these compute elements. Obviously, these interfaces must be tightly coupled, low latency, and deterministic. Less obvious is the fact that these IPC should also be much faster in the aggregate than the board's I/O bandwidth. Since the basic idea of hybrid architecture is that the proper type of compute element can be used at the proper place in a given application, the data usually will need to move between several different compute elements – often several times – before it's fully processed.

As shown in Figure 1, data enters the board via the FPGA I/O interface and is optionally pre-processed. Sometimes, as in the case of a down converter or pulse compression algorithm, the pre-processing will reduce the data rates, but a good architecture cannot assume this as many other common pre-processing algorithms, including filters, decoders, and FFTs, don't significantly change the data rate or could even increase it. Therefore, the minimal data bandwidth case is one where the data

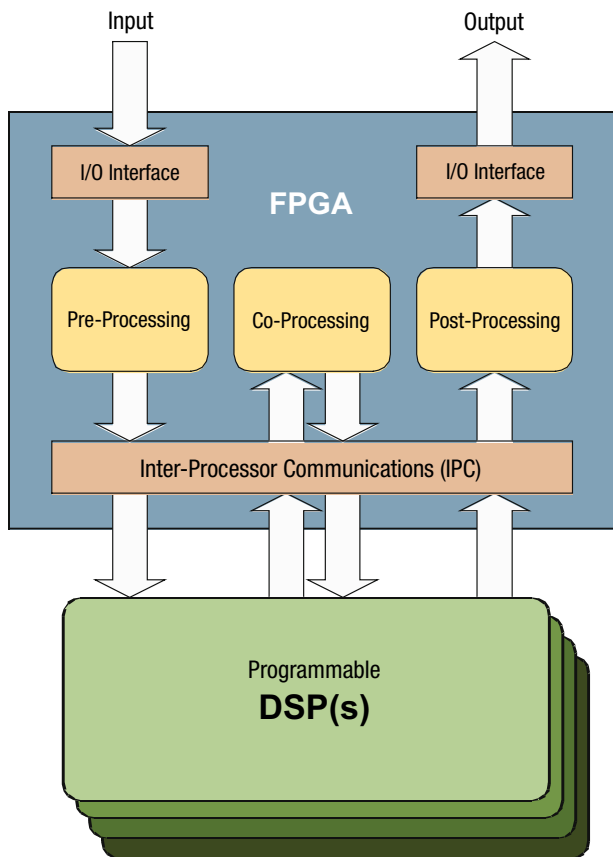


Figure 1

is sent on to a DSP for additional processing, and then maybe back to the FPGA only for post-processing and output. In this case the IPC data rate only needs to be equal to the I/O data rate. However, in the more general case when the FPGA provides co-processing resources for the DSP, or multiple FPGAs and/or DSPs are required for compute elements, then the IPC data rate requirements can be many times the I/O data rates. For example, if we have a 4x SerDes I/O at 3.125 GHz to the FPGA, this translates to ~1 GBps of external I/O data bandwidth. To support this external I/O, at least 2 GBps of non-blocking data bandwidth is needed between the FPGA and the DSP, with 4 GBps preferable.

Memory configuration

Not all signal processing applications require bulk memory, but most do. The optimal type, configuration, and even the location of the bulk memory depends on the specific application requirements. Like the I/O interfacing, this is where the inherent flexibility of the FPGA can be leveraged to implement a more versatile COTS board. Connecting modular bulk memory to the FPGA allows many types and configurations of memory to be supported by changing the memory module and reprogramming the FPGA. For example, a 64-bit data bus could be used to support a single 64-bit wide memory bank or reconfigured to support two independent 32-bit banks, possibly even of different types if desired.

Host interface and control

The best processing board in the world is of little value if it can't readily be integrated in to a system. In the COTS world this means standard mechanisms for host interfaces and control, virtually all of which can be support using PCI and/or GigE. The architectural issue then becomes how best to connect these standard interfaces to the COTS board's hybrid compute resources.

One way this can be done is via a *master* board resource that can then command and control the other compute

“It may be that FPGAs ultimately obsolete DSPs for high-end applications, or it could be that DSP vendors will provide new price, performance, or power advantages that FPGAs can't compete with.”

resources via existing IPC; while this approach has the benefit of simplicity, it has downsides including increased control latencies and reducing the effective bandwidth and determinism of the IPC. Another approach could be to implement the standard interfaces on each of the hybrid compute resources directly, but is an ineffective use of resources, would severely limit the selection of DSPs, and doesn't facilitate command and control between the compute elements on the board.

A better way to implement host interface and control on a COTS hybrid signal processing board is by using a bridge from the standard interfaces to a separate command and control bus to each of the FPGAs and DSPs. Since this path is orthogonal to the data paths, the interface to it is often referred to as a control plane (as opposed to the data plane). Having an independent control plane allows the host to directly access and control each resource without impacting data bandwidths. Additionally, the command and control bus can provide a path for the host to directly access the bulk memory on the module, as well as a supplementary method of IPC.

The generic COTS hybrid signal processing architecture shown in Figure 2 shows one way that the previously discussed architectural issues could be handled.

FPGA framework

With infinite possibilities, programming an FPGA can be daunting. This is exacerbated when the FPGA needs to implement

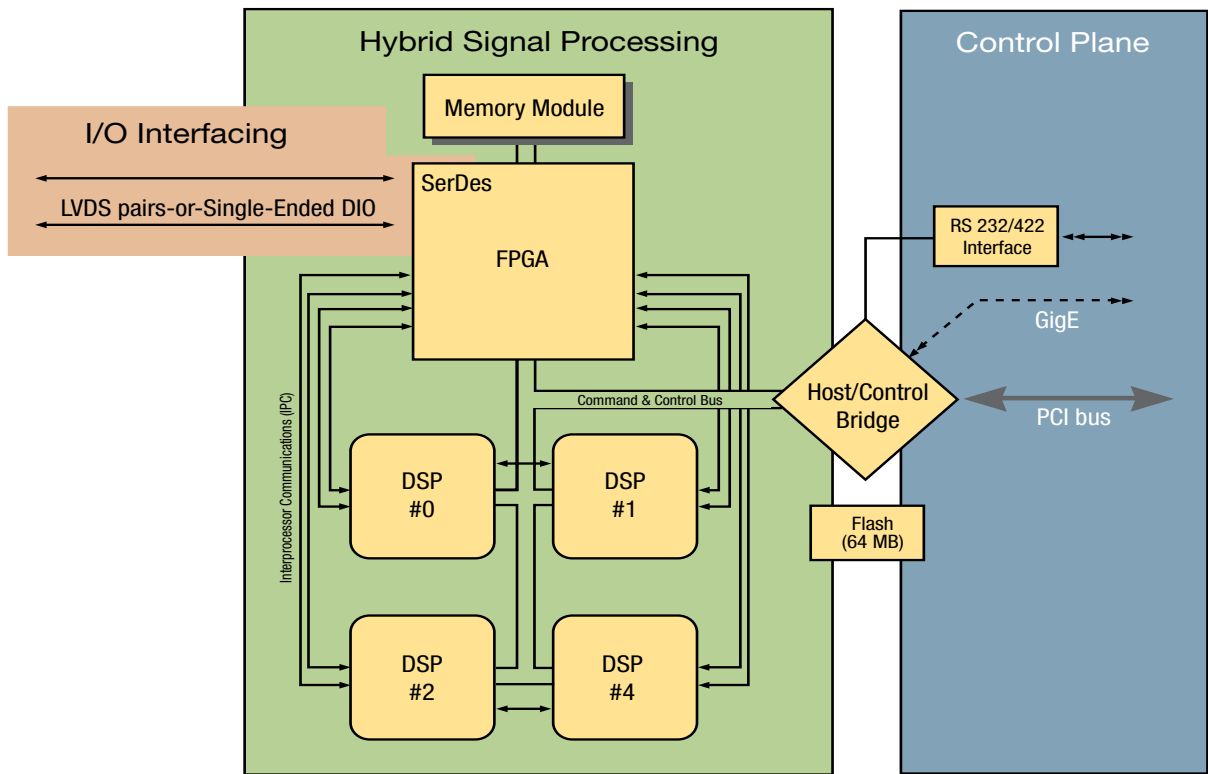


Figure 2

high-performance signal processing and is on a COTS board, with the pin-outs and external interfaces fixed. To reduce these problems, IP modules must, at a minimum, be provided for the board-level hardware interfaces including I/O, memory, IPC, as well as command and control.

Ideally, a complete FPGA framework will also be provided that supports software controlled data routing between the FPGA IP modules, along with documented IP data interfaces that allow the user to develop processing modules that can be inserted into these data flows. In this case, the conceptual data flows shown in Figure 1 can be readily implemented simply by dropping the required pre/co/post-processing blocks (shown in yellow) into the software configurable data flows – at run-time the command and control bus can configure the data flows required for a given application between I/O interfaces, IP processing modules, DSPs, and board-level resources.

Software

The more complex the hardware, the greater the need for low-level software for host interfacing, debug, and run-time command and control. A hybrid architecture has to be more complex than one that is homogeneous; therefore, the need for unifying software is great. For example, even performing a simple board reset now has added complexity: do you want to reboot the DSPs, reconfigure the FPGA, or both?

The challenge to the COTS vendor is to provide software that supports the additional flexibility and complexities – without making it complex to use. Detailed debugging may very well require specialized resources for the different technologies, but the configuration, control, and data of all compute resources must be unified into a single host driver interface library. If an aforementioned FPGA framework is implemented, software must be provided that configures and controls the data routing. Furthermore, the DSPs must have software support for interacting with the FPGA for such things as data transfer, interrupts, coordination, and synchronization; this can be provided via example code, libraries, or better yet, a messaging interface or operating system.

A new solution for traditional COTS

The traditional brute force method of architecting COTS signal processing boards by cramming either more DSPs or more FPGAs in to a slot forces COTS users to choose one set of technical strengths and weaknesses or the other, and has failed to keep up with real-world application demands. Rather than arguing the relative merits of one of these signal processing technologies versus the other while forcing their customers to perform an EXCLUSIVE OR decision, COTS vendors should be offering their customers an AND solution that mitigates the weaknesses and risks of using either

technology alone. This is all do-able, and in fact has been done, but it's not easy. While the resulting hybrid signal processing architectures may be initially more difficult to implement, optimize, and make useable, this is exactly the type of heavy-lifting, added value that COTS vendors are supposed to be providing to the COTS users community. DSP-FPGA.com

Jeffrey Milrod,

President and CEO of BittWare, Inc., joined BittWare in 1998. He gained extensive design experience at NASA and business experience at Booz, Allen & Hamilton, then merged his technical expertise with his business savvy by starting Ixthos in 1991, which he ran until it was acquired by DY4 Systems in 1997. Jeff holds a BA in Physics from University of Maryland and an MSEE degree from The Johns Hopkins University.



BittWare, Inc.

31B South Main St.
Concord, NH 03301
Tel: 603-226-0404

Email: jeff_milrod@bittware.com

Website: www.bittware.com