Do's and Don'ts of Architecting the Right FPGA Solution for DSP Design

Alex Soohoo, Altera Corporation - September 13, 2005

Designing a flexible, programmable DSP system architecture is a daunting task. Considering evolving mobile standards to the newest video compression techniques, the latest algorithms are rapidly growing in complexity. For example, a customer that was previously satisfied with standard definition resolution MPEG-2 video compression might now demand that the next product support high definition resolution H.264, which requires more than an order of magnitude increase in system performance. At the same time, the pressure to increase system channel count is unrelenting as network capabilities continue to grow. Consequently when starting a new design, engineers must consider not just today's requirements, but understand that this system might also be soon called upon to address unforeseen challenges.

So what are the design options? Historically, the choices for building a high-performance DSP design for high-speed digital communications or real-time video processing were limited. A typical approach was to populate a board with as many DSP processors as possible (colloquially known as a DSP farm) and then hope that the software engineers would not write applications that outstripped the maximum processing capacity.

Such issues as high design complexity and total system power limited the scalability of this method. Further, this design methodology hinged on the assumption that DSP processor vendors could continue to increase clock speeds and reduce power consumption, which was never guaranteed. Now, however, thanks to remarkable improvements over the last few years in FPGA performance, and the incorporation of hard embedded multipliers in the devices, there are new architectural options that address the issues of performance, flexibility, and scalability.

An FPGA co-processing architecture can be an ideal approach to tackle these challenges. By intelligently partitioning a DSP algorithm between a DSP processor and an FPGA co-processor, a number of benefits can be realized including dramatically boosted performance and a reduction in total system costs. However, there are numerous issues to consider before heading down this path. Specific system requirements and preferences of the engineering team will play a large role in final architecture decision. Some of the do's and don'ts system designers should consider when designing a FPGA co-processor solution for a high performance DSP system follow (See Figure 1).
Figure 1. FPGA Co-Processing

DON'T
Don't assume that you can use the same approach to developing DSP algorithms on an FPGA that you use on a DSP processor. It is tempting to think that you can simply instantiate a soft DSP processor on the FPGA and create code in a similar manner as traditional DSP software development. This is a common misunderstanding. A completely different approach must be used.

To realize the benefits of FPGA co-processing, the datapath must be re-architected and implemented in a parallel manner, not in a serial, sequential DSP processor coding style. While a DSP processor and an FPGA both have embedded multipliers, FPGA-based designs can potentially execute a much greater number of multiply-accumulate (MAC) operations per cycle than traditional DSP processors. Evaluate your DSP system and the required algorithms and consider how they might be "parallelized." Careful architectural planning and development of the FPGA co-processor can provide an order of magnitude performance boost over DSP processor based designs.

DO
Understand which DSP design flow methodology will work best for your designers, especially for those unfamiliar with FPGA design flows. One of the first questions to ask is, "How does the algorithm group prefer to prototype the DSP system? Will the group develop in-house models written in the C language that are not based on any specific tool or environment?" If so, there is a great deal of flexibility when choosing a DSP design flow. The team can select a modular approach and create a hardware implementation for each block using a particular chosen method. This preference may determine the best starting point for the FPGA co-processor design.

Perhaps the team is more comfortable using a simulation environment to quickly model and simulate the algorithms that are specified for the project. This may be a welcome approach for a team with more DSP software implementation experience. Does the team have a background with an ASIC or FPGA design flow? If so, it is also possible to develop the DSP datapath by directly writing VHDL or Verilog and bypass the use of higher-level design abstraction tools. While potentially the most labor intensive and time consuming path to follow, the final design can then be optimized for size and performance. What about a C-to-gates methodology? A few EDA vendors have introduced C-entry tools specifically targeted for DSP applications that generate HDL code ready to be synthesized and incorporated into FPGA design software. All of these approaches can be incorporated into DSP design flows to implement an FPGA co-processor.
DO
Decide the how the DSP algorithms will be partitioned in a DSP processor/FPGA co-processor architecture. A straightforward and well-understood approach is to offload the most computationally intensive pieces of a DSP algorithm to an FPGA and let the DSP handle the control-flow oriented segments. This datapath, control path architecture, while simple to visualize, may not necessarily be optimal for a project. The popularity of soft embedded processors instantiated on an FPGA makes it possible to execute a large part, if not all, of the control path on the FPGA. In fact, multiple soft processors can be incorporated to provide a finer degree of granularity to the control flow. On the other hand, the existence of legacy DSP code might make the team hesitant to implement the entire datapath processing on the FPGA, especially when a number of man-years are invested in library development on a DSP processor platform. In this case, the team may decide to initially move smaller and/or new parts of the processing chain to the FPGA.

Remember, flexibility is a key benefit of this architectural approach. It may be reasonable that for a first FPGA-based design, a conservative approach is taken and only a small portion of the processing is implemented on the FPGA with the rest executed on DSP processors in the system. For the next generation design, shift more of this processing to the FPGA and boost system performance without having to redesign the current board architecture. Providing this kind of extensibility will require careful planning.

DO
Evaluate whether to "make or buy" key DSP intellectual property in the design. Is the target DSP design composed of standard DSP blocks, or will most of it be a completely proprietary effort? More than likely, the final design will use a combination of classic textbook IP cores and the team's own custom logic. The best design option will depend on project requirements which may include cost considerations, future design reuse, or time-to-market.

Using off-the-shelf cores may be a less expensive, faster option compared with building a block from scratch, assuming they are well supported and have the correct feature set. The next question is whether you can identify a provider to meet the design requirements. Certainly a large 3rd party IP network exists around DSP processors to fulfill this need. A similar ecosystem has exists around FPGAs in the last few years to accommodate the large number of FPGA-based DSP designs. The most common blocks such as FIR filters, fast Fourier transforms (FFTs), and forward error correction (FEC) cores are readily supplied and are successfully deployed. Even more exotic or specialized IP such as an H.264 video codec are available from IP vendors as packaged FPGA cores. Finally, make sure that the seller provides complete documentation, performance benchmarks, verification test benches, and a well-staffed support organization to address any issues.

DO
Determine how FPGA co-processor system integration will be performed. Once the processing partition is decided upon, how will the two halves be integrated? Specifically, what will be the primary hardware interface between the DSP and FPGA? The peripheral feature set of the DSP will likely determine which choices are available. More than likely there will be multiple links between DSP processors and FPGAs in the system. Will they be low-speed serial connections for control or high-speed parallel connections to shuttle data between the devices?

Depending on the processing partition between the devices, the interface with the appropriate throughput will need to be selected. Perhaps the FPGA will be called upon to create an ad hoc bridge for proprietary audio or video data buses in the system. FPGAs can be used to increase the capabilities of the DSP processor by providing peripheral and memory expansion. This can be especially useful when trying to adapt a design to meet emerging industry standards not previously envisioned by DSP processor vendors.
Now that a preferred hardware interface is selected, does the FPGA design flow incorporate a seamless method to integrate the interface into the design? While it is possible to create a custom block to perform the function, there are comprehensive system integration tools that can perform the potentially tedious task of connecting it all together. This software typically includes libraries of peripheral components to address a wide range of connectivity options. Secondly, will this design tool generate an application-programming interface (API) or a memory-mapped header file that can be incorporated into the DSP software integrated development environment? Don’t underestimate the value of this step. The integration of hardware-accelerated algorithms into the DSP software architecture is critical to extracting the benefits of the FPGA co-processor architecture.

DO"T
Don’t be constrained by the requirements of an initial design. Now that your first FPGA co-processing architecture is created, you are ready to exploit the benefits of this flexible, scalable platform. If the system feature set needs to be enhanced or the system bill of materials (BOM) cost reduced, there are several options that do not involve redesigning the current board. FPGA vendors typically offer pin-compatible devices across a range of densities to allow vertical migration. To reduce manufacturing costs, investigate using a smaller FPGA (design permitting). Alternatively, move more of the functionality from DSP processors into the FPGA and reduce the total number of components without changing the current board layout. To add performance to the platform, use a higher density FPGA and build a more powerful design with greater capabilities. This approach will allow you to maximize design reuse and shorten your next generation product’s time-to-market. Just make sure that your original design is made as modular as possible to enable this option.

About the Author
Alex Soohoo, DSP marketing manager, joined Altera in 2004 as a marketing manager for digital signal processing products. Prior to joining Altera, Alex held management positions at IDT, PMC-Sierra, and LSI Logic (formerly C-Cube Microsystems). He earned a BS in EECS from UC Berkeley and a MS in electrical and computer engineering from UC Davis. He can be reached at asoohoo@altera.com