WP394 (v1.0) July 14, 2011

Advancing the Use of FPGA Co-Processors through Platforms and High-Level Design Flows

By: Tom Hill

FPGAs can be used to accelerate the performance and extend the capabilities of embedded processors, often with less cost and power than alternative multi-processor solutions. The main barrier to adoption of FPGAs for new users has been ease of use and design flow, which is now being addressed with new co-processing development platforms combined with high-level design methodologies.

Introduction

Virtually all applications that perform signal or video processing seek levels of performance beyond what a traditional single-core processor or DSP can deliver. A variety of devices are now available to address this performance gap, including application-specific ICs (ASSPs), multicore DSPs, Graphics Processing Units (GPUs), and FPGA co-processing platforms that feature DSP-optimized programmable hardware.

ASSPs typically combine ARM-based processing on a single device with market-specific hard accelerators and peripherals. Today's leading ASSPs, while providing a highly optimized hardware platform, take up to two years and $20M–$50M to develop, which limits their availability to large markets supported by mature standards. While ASSPs provide a compelling hardware platform for these applications, their dependency on hard accelerator and interface blocks limits their flexibility to address off-target markets.

Some ASSP devices attempt to address this flexibility issue by including DSP or media processor cores in their devices. Developing custom accelerators on processor cores has the benefit of maintaining a software-centric design flow, but performance gains are limited, and the additional processor cores do nothing to address the need for custom interfaces or peripherals.

Multicore processors are another option often considered to address the need for better system performance. Multicore DSPs include up to 128 independent 16 x 16 multipliers and GPUs that offer up to 447 independent processing engines. The challenge facing the use of these devices is to develop software that can effectively take advantage of this processing capacity. New multicore programming guides and programming languages, such as OpenCL, have emerged to enable software to be structured with inherent parallelism, but this is a painstaking manual coding effort.

A computer science model called Amdahl's Law predicts the possible performance gains versus the number of parallel processors as a function of the percentage of code that can be parallelized. Figure 1 plots this relationship.

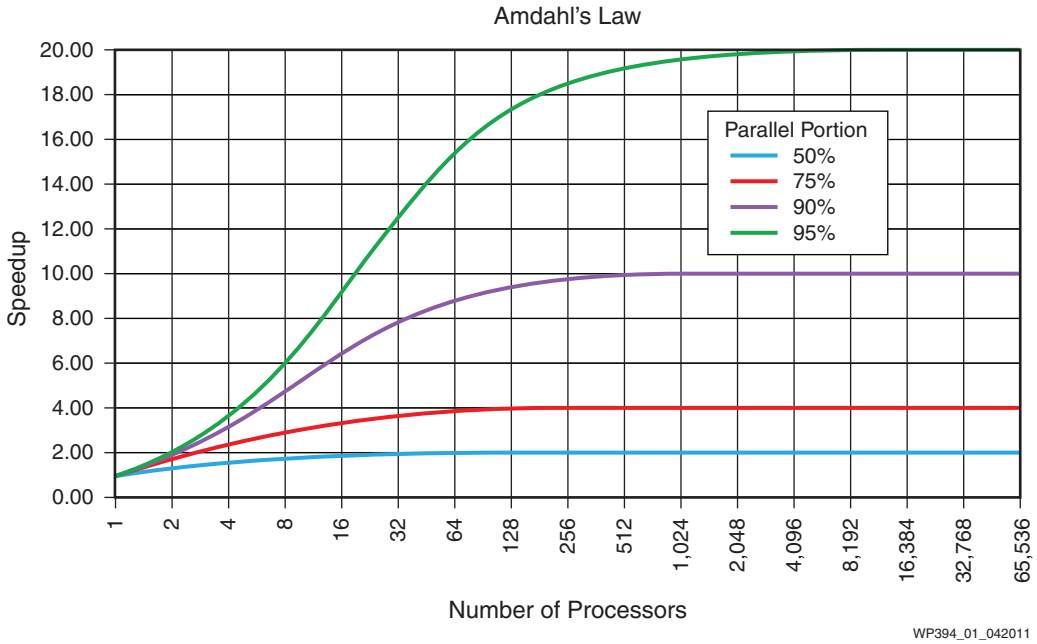

Figure 1: **Amdahl's Law**

Even in a best case scenario where 95% of the code can be made parallel, the performance gains are far from linear to the number of available processors and drops significantly with even a slight decrease in code parallelism. For example, for code that is 95% parallel, 512 processors only delivers a 19X performance improvement.This gain drops to 4X for code that is 75% parallel. And similar to applications, processors, and multicore DSPs, GPUs do not address the need for custom interfaces and peripherals.

FPGAs offer a truly unique reconfigurable hardware platform than can be used to create both custom accelerators and custom interfaces to a design's exact requirements. FPGA co-processors can be used to accelerate and extend the functionality of ASSPs or processors in a fraction of the time—with no NRE costs.

The primary challenge to using FPGAs as co-processors becomes the design flow. FPGA co-processors, traditionally developed by hardware designers, are often deemed too difficult to use by processor and system designers. This, however, is changing due to recent improvements in the quality and availability of co-processing development platforms and high-level design methodologies.

FPGA Device Architecture

FPGAs are reconfigurable devices that can be programmed and reprogrammed as needed. They include of an array of logic blocks, memory blocks, digital signal processing (DSP) blocks, and hard peripherals, such as PCIe® or Gigabit Ethernet. See Figure 2.

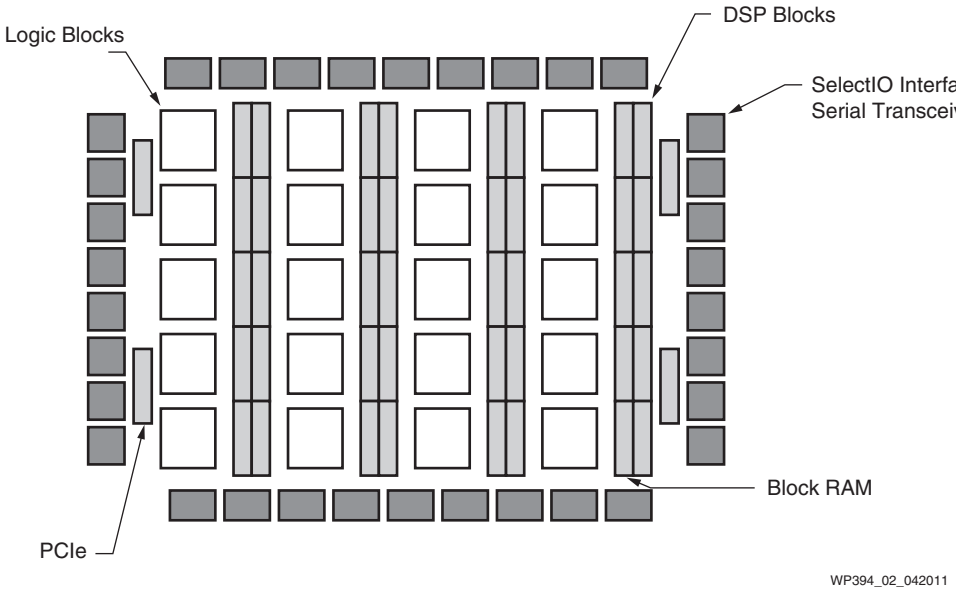

Figure 2: **FPGA Architecture**

This architecture enables the following co-processing functionality:

- Off-chip I/O for over 100 I/O standards
- Hardware accelerators that leverage parallelism to meet performance
- Custom system peripherals

Table 1 lists the available resources for the Spartan®-6 family.

Table 1: **Spartan-6 FPGA Resources**

Resource	Spartan-6 FPGAs
Logic Cells	3.8K–147K
Clock Frequency	390 MHz
DSP Blocks	8–180
GMACs	60
Maximum Transceivers	8
Transceiver Performance	3.2 Gb/s
Memory (Block RAM)	4,824 Kb

Accelerated Processing through the DSP Block

FPGAs, such as Spartan-6 devices, include dedicated DSP logic, called DSP48 slices, that can be dynamically configured to perform a variety of DSP operations on a clock-by-clock basis.

DSP performance is achieved through dedicated DSP blocks called DSP48A1 in Spartan-6 devices. The Spartan-6 family has up to 180 DSP blocks and can deliver up to 60 GMACs of DSP peak performance. Figure 3 shows a block diagram of the DSP48A1 slice.

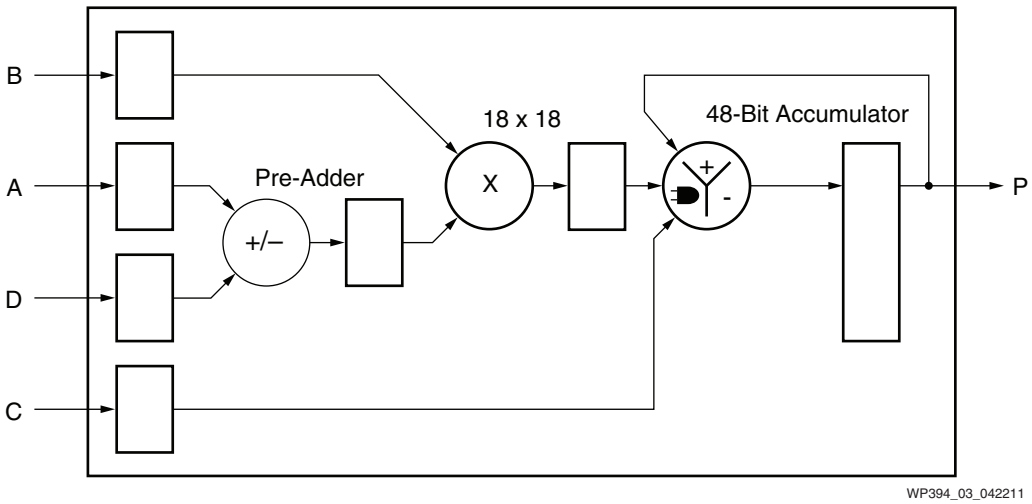

Figure 3: **Spartan-6 FPGA DSP48A1 Slice Architecture**

DSP48A1 slice benefits include:

- Hard pre-adder for efficient polyphase and half-band filters
- Up to 18 x 18 multiply in a single slice
- 48 bits of accumulation
- Cascadeable to support wide multiplies and floating-point operations

The Hardware Parallelism Advantage

When creating accelerators using programmable logic, the performance advantage comes from leveraging hardware parallelism. Hardware parallelism refers to designs that use multiple hardware resources to perform a calculation, such as a vector multiply, in fewer clock cycles. For example, performing a 58 element convolution operation on a processor running sequentially at 1.2 GHz requires 58 clocks, resulting in a throughput of 20 MSPS. An FPGA, on the other hand, can use hardware parallelism to complete the entire convolution operation in a single clock. For Spartan-6 devices, which can run at 390 MHz, a maximum throughput of 390 MSPS can be achieved for the same operation, resulting in a 19X performance improvement. See Figure 4.

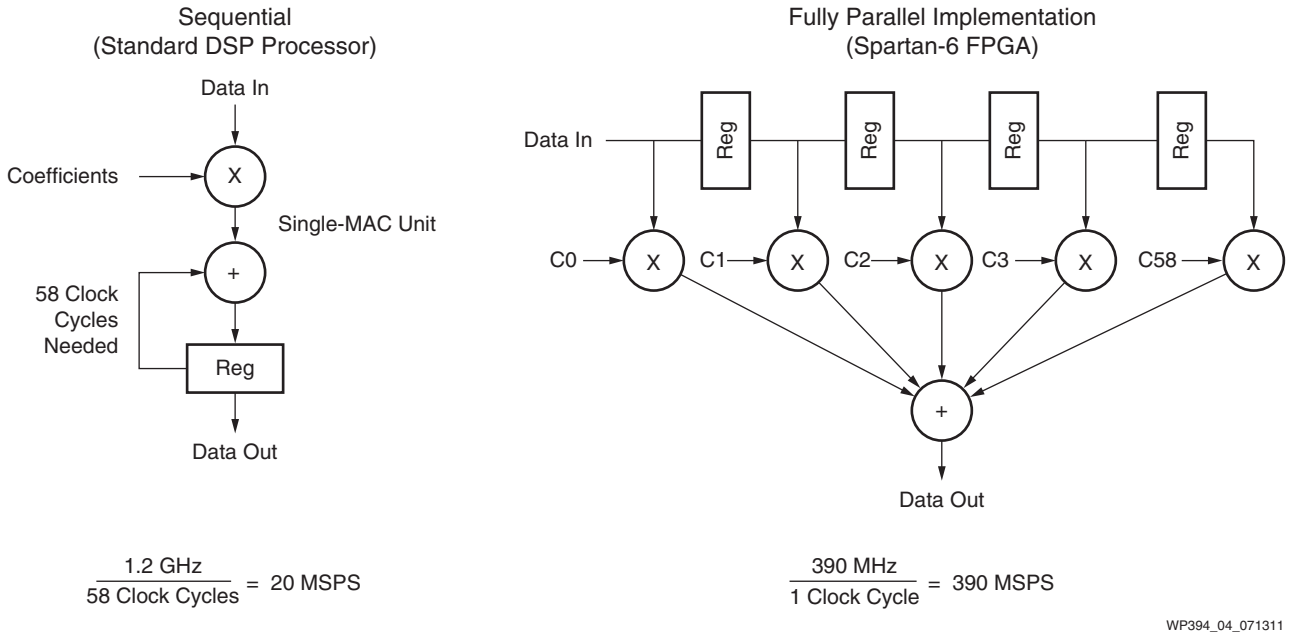

Figure 4: **The Hardware Parallelism Advantage**

Xilinx conducted a benchmarking exercise using a 32 x 32 floating-point matrix multiply operation to evaluate the performance acceleration of an FPGA co-processor. This benchmark takes into account the DMA transfer latency required to move a frame of data between the DSP processor and the FPGA. The results are summarized in Table 2.

Table 2: **Multiply Operation Benchmarking Results**

Platform	Clock Speed	Floating Point Performance	Improvement
TI C64x DSP Core	420 MHz	603 µs	N/A
Spartan-6 LX45T FPGA Co-processor	390 MHz	47 µs	12X

The benchmarking results highlight the advantage of developing accelerator blocks using custom hardware versus a processor.

High Bandwidth Memory

Spartan-6 FPGAs have up to four dedicated memory controller blocks (MCBs) that target a single-chip DRAM and support access rates of up to 800 Mb/s. Spartan-6 devices provide:

- DDR, DDR2, DDR3, and LPDDR support
- Data rates up to 800 Mb/s (12.8 Gb/s peak bandwidth)
- Internal 32-, 64-, or 128-bit data interface—simple and reliable interface to the MCB
- Multiport bus structure with independent FIFO to reduce design timing issues
- Predictable timing for memory interface designs
- Software wizard to guide through the entire process

FPGA I/O Support

Processor-based hardware platforms, such as ASSPs, are pre-configured with market-specific interfaces but offer little flexibility to address non-supported system I/O requirements. FPGAs, such as Spartan-6 devices, can be used to expand the I/O options using SelectIO™ technology and GTP serial transceivers.

The key features of the SelectIO interface are:

- Up to 1,080 Mb/s data transfer rate per differential I/O
- Selectable output drive, up to 24mA per pin
- 3.3V to 1.2V I/O standards and protocols
- Low-cost HSTL and SSTL memory interfaces
- Hot-swap compliance
- Adjustable I/O slew rates to improve signal integrity

GTP transceivers extend the I/O capabilities. They are highly configurable and tightly integrated with the programmable logic resources of the FPGA, supporting a wide variety of applications. GTP transceivers also implement serial protocols at the lowest price. The GTP transceivers in Spartan-6 LXT FPGAs include:

- Up to 3.2 Gb/s performance
- High-speed interfaces: Serial ATA, Aurora, 1G Ethernet PCI Express, OBSAI, CPRI, EPON, GPON, DisplayPort, and XAUI
- Low power consumption: < 150 mW (typical) at 3.2 Gb/s

The I/O capabilities of Xilinx FPGAs provide the flexibility to support over 100 common interface standards. A subset of these standards is shown in Figure 5.

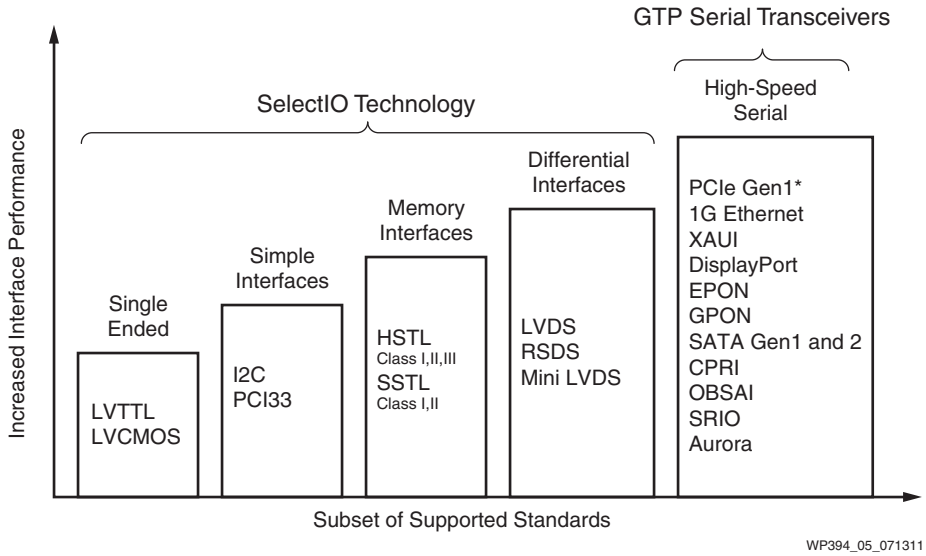

Figure 5: **FPGA Interface Flexibility**

The Spartan-6 FPGA family also includes a dedicated memory controller block (MCB). The MCB is an embedded block multiport memory controller that greatly simplifies the task of interfacing Spartan-6 devices to the most popular memory standards.

The key features of the MCB are:

- DDR, DDR2, DDR3, and LPDDR (Mobile DDR) memory standards support
- Up to 800 Mb/s (400 MHz double data rate) performance
- Up to four MCB cores in a single Spartan-6 device

System Partitioning

When creating an FPGA co-processing development system, maximum flexibility is achieved when the FPGA surrounds the processor. This allows the programmable logic to be configured as needed for interfaces and maximum flexibility, place accelerators to pre or post-process streaming data, or accelerate instructions as a memory mapped co-processor. See Figure 6.

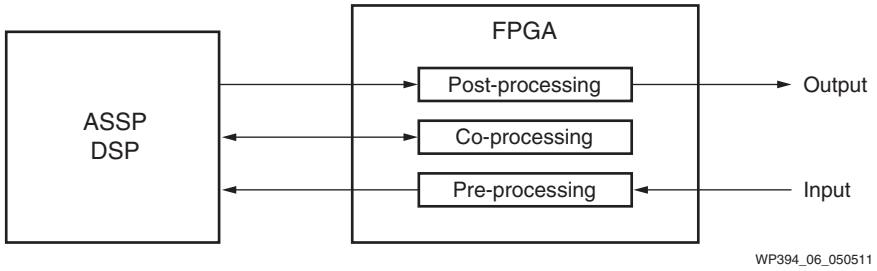

Figure 6: **Co-Processing Platform System Architecture**

In a dual-core processing system, an ARM-based processor is typically used to perform system execution control, which includes running the operating system, hosting the user interface, and interfacing to the network and storage devices. A

second processor can then be used to create soft accelerators that offload performance-critical operations from the ARM processor. This might include video CODECs or floating-point math. Designers often attempt to run the entire algorithm in software on this second processor to determine performance bottlenecks. These bottlenecks are then good candidates to offload to an FPGA co-processor. The FPGA can also be used for custom interfaces and peripherals that are not supported by the processor. System architectures that use typical dual-core processing enable developers to address a wide variety of system performance and interface requirements using an FPGA. See Figure 7.

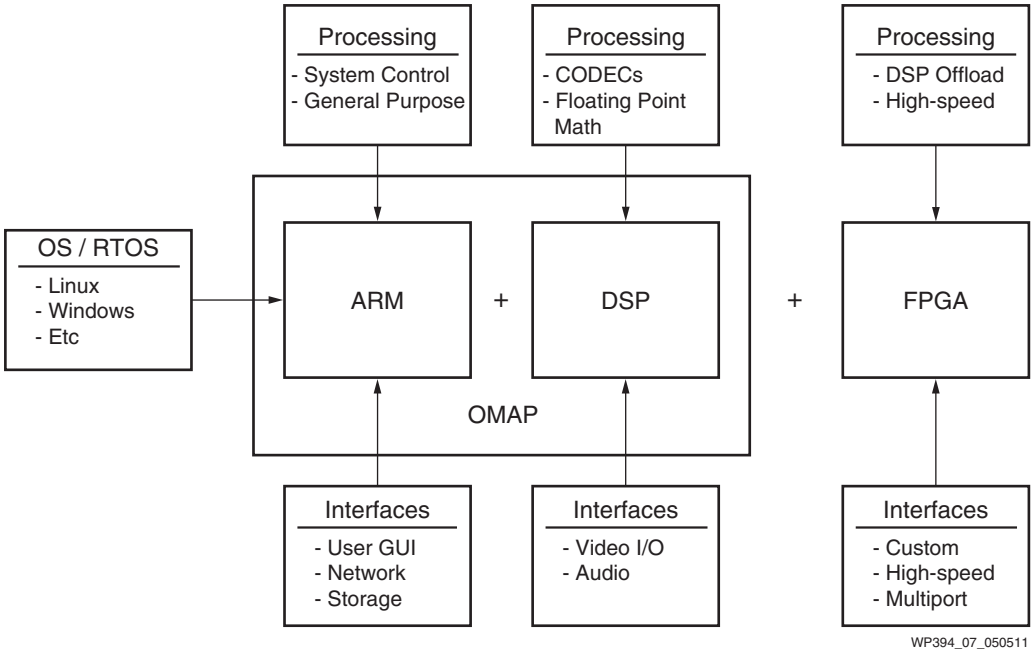

Figure 7: **Hardware/Software System Partitioning**

Design Challenges

The development of a co-processing system that includes system execution control running on an ARM processor and the hardware/software partitioning between a DSP and an FPGA is a daunting challenge that requires three distinct skill sets not typically possessed by a single individual. A typical co-processing design flow is shown in Figure 8.

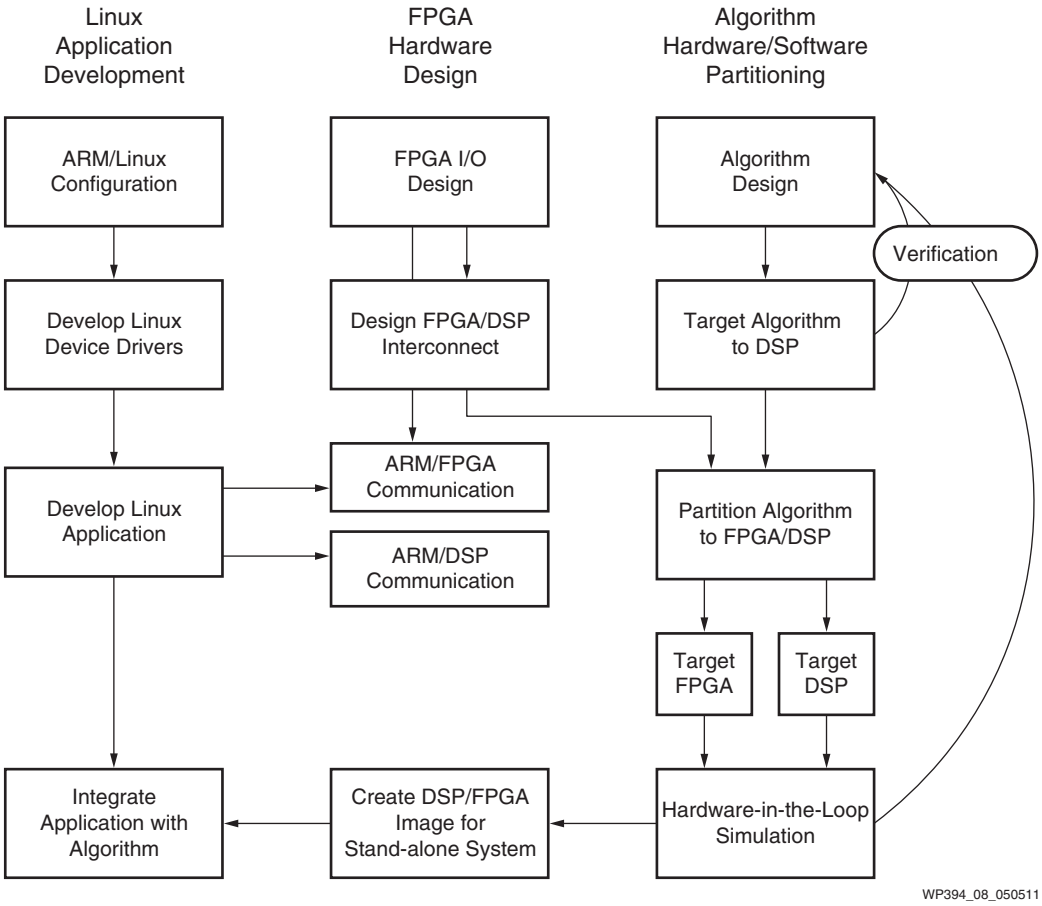

Figure 8: **Co-Processing System Development Flow**

Adding the FPGA to the system requires both FPGA hardware design skills to develop the interface logic as well as system algorithm hardware/software partitioning skills. FPGAs have long been used as co-processors by hardware designers with the appropriate backgrounds. The challenge now is to enable system engineers or DSP software developers, who do not have hardware design backgrounds, to take advantage of programmable logic.

Co-Processing Targeted Design Platforms

Designers with a variety of technical backgrounds can use the leading Targeted Design Platforms to adeptly evaluate the benefits of an FPGA in their system, which can greatly reduce development schedules and improve time to market. Recently, Xilinx and Avnet released two new Targeted Design Platforms for co-processing development. These include the Spartan-6 FPGA/TI OMAP-L1x Co-Processing Development Kit and the Nano-ITX/Spartan-6 FPGA Development Kit.

The Spartan-6 FPGA/TI OMAP-L1x Co-Processing Development Kit integrates the key strengths of high-performance FPGAs, system control processing using an ARM processor, and digital signal processing, all within a single environment. The kit includes: Avnet Co-Processing Development Board, TI OMAP-L138, Spartan-6 LX45T FPGA, one FMC Expansion Slot, hardware and software development tools, a reference design, and tutorials.

The Nano-ITX/Spartan-6 FPGA Development Kit connects an Intel Atom E640 based motherboard with a Spartan-6 LX75T FPGA based mezzanine card. The combined small-form-factor system enables custom peripheral and co-processing development for medical, industrial, military, and many other applications. This Targeted Design Platform is also accompanied by hardware and software development tools, a reference design, and tutorials.

A closer look at the reference design for the OMAP co-processing kit highlights how a co-processing platform accelerates development. This reference design provides reusable design infrastructure for the universal parallel port (uPP) and I2R interfaces. It also includes an FFT design example implemented on both the DSP and FPGA that can be easily extended and modified to end-user requirements. See Figure 9.

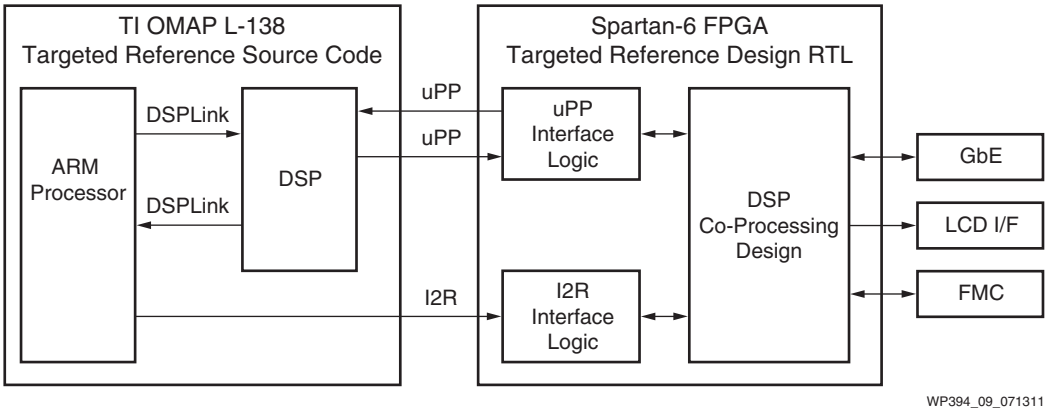

Figure 9: **Spartan-6 FPGA/TI OMAP-L1x Co-Processing Development Kit**

One of the greatest design challenges is often the development of the interfaces between the different processing domains, which requires design proficiency in both domains. In this reference design, all communications interfaces between the three processing blocks have been pre-developed, including a DSPLink interface between the ARM processor and the DSP, a uPP between the FPGA and the DSP, and an I2R interface between the ARM processor and the FPGA. Design tutorials provided in the kit give the user the necessary information to extend the design to specific requirements, eliminating the need for designer proficiency in all domains.

To take advantage of the specific architecture features of each device, the user must implement hardware/software partitioning, which requires both system understanding and multi-domain development skills. This can be a time-intensive task and, depending on the data transfer latencies, it can be difficult to predict the exact performance gains of the system prior to implementation. Xilinx offers the TI OMAP/Spartan-6 FPGA Co-Processing Development Kit. The kit includes a rich set of design tutorials that take users step by step through the hardware/software partitioning processes and offers a repeatable methodology that is usable to both software and hardware developers.

High-Level Hardware/Software Co-Design

Hardware/software algorithm portioning can be greatly enhanced using a high-level design methodology where a common modeling environment can be used for both hardware and software development. The two most prominent modeling environments in use today are C/C++ and MATLAB®/Simulink® from The MathWorks. Both environments are widely used to develop algorithms and also support implementation paths to processors and FPGAs. Recent advancements in the usability and quality of these high-level design flows make them a viable option for hardware/software partitioning.

Model-Based Design Using MATLAB and Simulink

The MathWorks promotes the concept of "Model-Based Design," where algorithms can be developed abstractly and then used as "executable specifications" in a correct-by-construction design flow. See Figure 10.

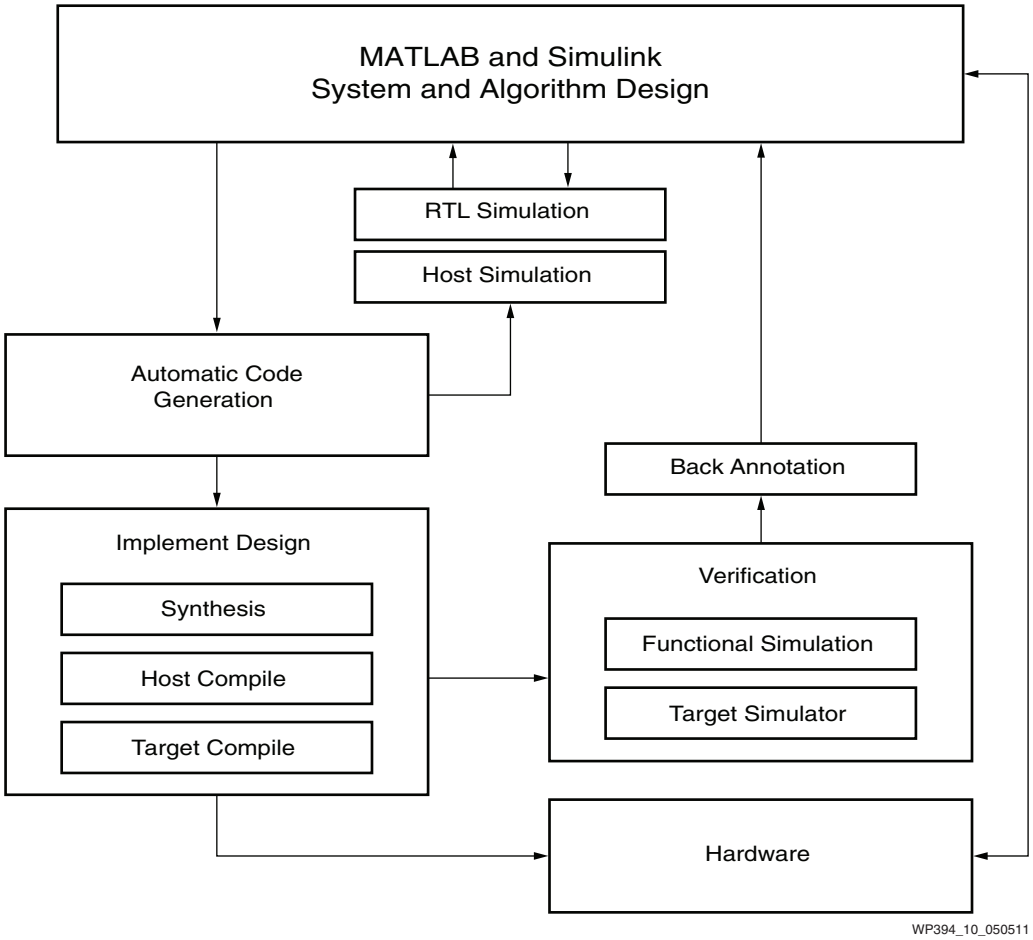

Figure 10: **Model-Based Design Flow Using Simulink**

The Simulink environment is well suited for performing graphically assisted hardware/software partitioning. Once partitioned, the model can be further modified to reflect the desired hardware or software architecture. Automatic code generation creates C and RTL code for programming the processor and FPGA respectively. To ensure that no errors are introduced during implementation, hardware-in-the-loop

verification flows validate both the processor and FPGA against the original Simulink system model.

Simulink alone does not fully abstract the user from platform implementation details. The MathWorks and Avnet have jointly developed a platform support package for the OMAP co-processing kit that does provide sufficient abstraction to specify the dataflow of the system. For example, the Harris Metric corner detection algorithm can be partitioned between an FPGA and a DSP, and the DSP operations "Find Local Maxima" and "Draw Markers" can be modeled using two blocks from the Simulink Imaging and Video Blockset as shown in Figure 11.

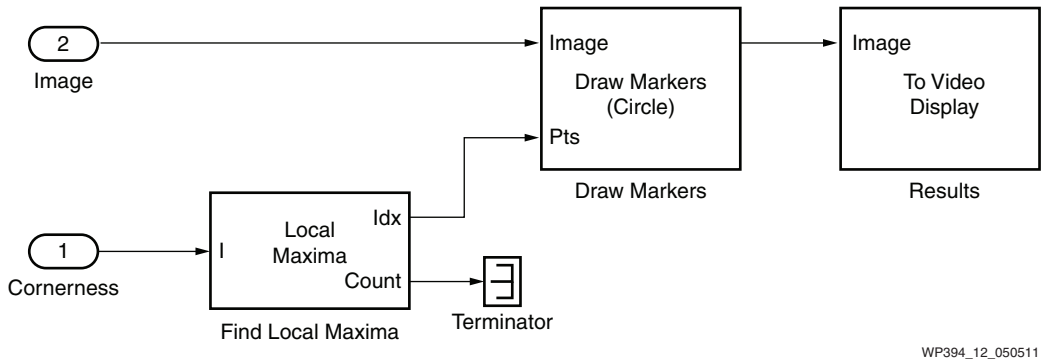

Figure 11: **Executable Specification in Simulink**

In a real world system, the video data is often passed to the DSP from the FPGA, using an interface such as the uPP interface from Texas Instruments. The Avnet Spartan-6 FPGA TI OMAP co-processing platform support package provides these interfaces as blocks as shown in Figure 12.

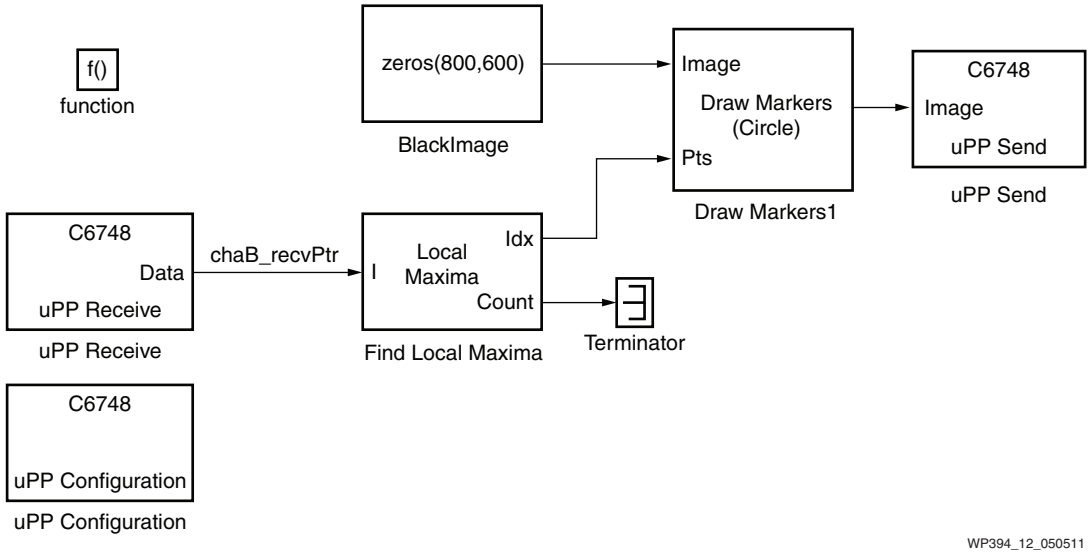

Figure 12: **Design Changes for Code Generation**

The input port has been replaced with the uPP receive block and the video view has been replaced with the uPP Send block. It is the combination of the high-level design methodology and the targeted design platform that delivers sufficient abstraction to allow a single developer to quickly evaluate and to effectively partition a design between hardware/software.

Hardware/Software Design Using C/C++

Significant advances have been made in C-Synthesis for targeting FPGAs, in terms of both quality of results and ease of use. Today's leading C-Synthesis products, such as AutoESL™ high-level synthesis tool, combine natural coding styles with quality of results on par with RTL designers. For developers who model using C/C++, having the ability to partition at the C-level allows for a natural transition between domains. The AutoESL tool has consistently delivered results on par with what a typical RTL designer would produce. This represents a major advancement in the viability of high-level design flows using C/C++. Although more work needs to be done to abstract more of the FPGA design details from the user, C-Synthesis tools now offer sufficient abstraction to allow a DSP software developer to generate RTL.

Conclusion

FPGAs are the perfect complement to the legacy code, design skills, and ecosystem support of ASSPs and DSP-based hardware platforms when additional device flexibility is required. Rather than migrate to a different device, performance bottlenecks and custom I/O requirements can be addressed with the flexibility of programmable hardware in a low-power and low-cost device. Spartan-6 FPGAs and the associated co-processing kits offered by Xilinx enable the designer to:

- Add high-performance hardware accelerators and custom I/O to ASSP devices for low cost and low power
- Apply a high-level design methodology that uses either MATLAB/Simulink or C/C++ and can be used to perform hardware/software partitioning without hardware design experience

For more information, go to:

- Spartan-6 FPGAs:

  http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/index.htm
- AutoESL Synthesis Tool:

  http://www.xilinx.com/tools/autoesl.htm
- Avnet TI OMAP/Spartan-6 FPGA Co-Processing Kit:

  http://www.xilinx.com/products/boards-and-kits/AES-S6COP-LX45T.htm
- Avnet Spartan-6 FPGA/Intel Atom Development Kit:

  http://www.xilinx.com/products/boards-and-kits/AES-S6NITX-LX75T.htm

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
07/14/11	1.0	Initial Xilinx release.

Notice of Disclaimer