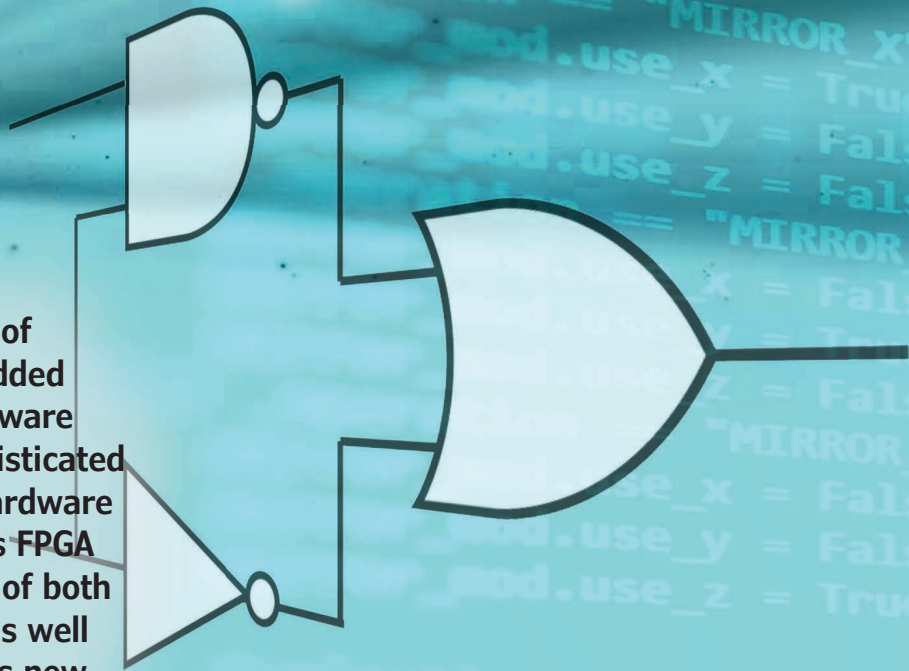


Managing FPGA Design Complexity

Easing IP Integration

Modern FPGAs can contain millions of logic gates and thousands of embedded DSP processors allowing FPGA hardware designers to create extremely sophisticated and complex application-specific hardware functions. Bob explores how today's FPGA technology has revamped the roles of both hardware and software engineers as well as how dealing with on-chip IP adds new layers of complexity.



By
Bob Sgandurra, Pentek

Over the past 35 years, there has been a constant progression of technologies for performing digital signal processing. Some of these have taken the form of processors dedicated to the task of efficiently executing complex math in parallel like the digital signal processors (DSPs) from Texas Instruments and Analog Devices and other specialized processors from various manufacturers. Another path has been to exploit the specialized processing engines inside more general-purpose processors from companies like Intel and Motorola (now NXP), or to repurpose highly-parallelized processors like graphics processing units (GPU) for DSP applications like radar. In each of these examples, it's been the software engineer's job to create programs or applications for a fixed-hardware architecture. This might be accomplished by programming on the "bare metal" and accessing internal registers and resources of the processor directly, or through the window of an operating system which manages the processor's resources.

FPGAs: A GAME CHANGER

This paradigm changed with the introduction of programmable logic devices and specifically with the advent of FPGAs. An FPGA's logic is a mesh of gates and interconnects that have no function until a logic design is loaded into the array connecting the gates to form circuits. Modern FPGAs can contain millions of logic gates and thousands of embedded DSP processors allowing FPGA hardware designers to create extremely sophisticated and complex application-specific hardware functions. And this is where the job of the software engineer takes a turn.

With fixed targets like Texas Instrument's DSPs or Intel's processors the software engineer writes programs for a static and well-defined hardware architecture. However, with FPGAs, the functions and even the interfaces into the hardware can vary. The FPGA functions are determined by what logic design the FPGA engineer uses to configure the FPGA and this can change with different iterations of the design.

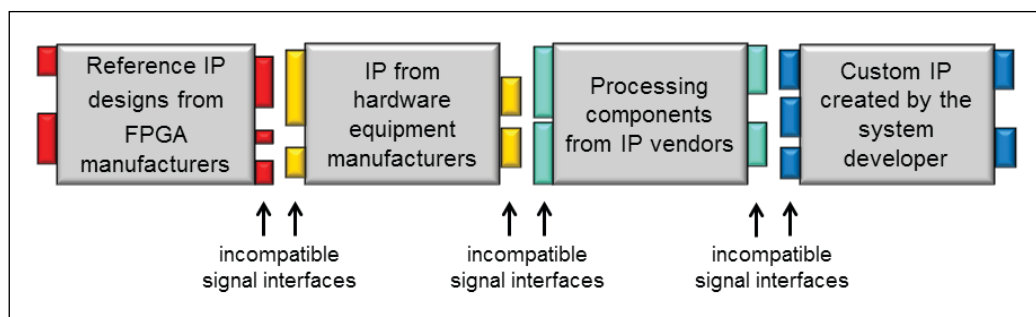


FIGURE 1

Incompatible signal interfaces are depicted here graphically.

In addition, FPGA logic design and the software to control it are intimately tied together. This relationship and the need to keep FPGA design changes and software changes in sync are a reality that must be managed in the design environment of sophisticated FPGA-based systems.

And let's not forget the FPGA design engineer. With the complexity of high performance FPGAs, the task of the FPGA designer has also become increasingly more demanding. The logic design—sometimes called Intellectual Property or IP—is typically created using either VHDL or Verilog hardware description languages. And while these languages are the cornerstone of FPGA design, new tools and design environments can improve design efficiencies, particularly when developing for very large FPGAs with millions of gates. And not only new tools, but some of the basic philosophy of how IP is defined is seeing a change.

THE AXI4 STANDARD

As the density of FPGA fabric becomes greater, the possibility of creating more and more sophisticated IP tends to increase. Often components of the IP design can come from multiple sources:

From the FPGA manufacturer: Much of the IP needed to create the overall dataflow through the FPGA fabric and control for specific FPGA interfaces is typically included in the libraries provide by the FPGA manufacturer. In addition, common peripheral resources like SDRAM are typically

supported by manufacturer provided tools to generate the required IP.

From equipment manufacturers: Often the FPGA is part of a development or deployable hardware solution manufactured by a company other than the FPGA manufacturer. In many of these systems additional hardware like analog-to-digital and digital-to-analog converters are part of the overall design and IP to control these components should be provided by the hardware manufacturer.

From an IP vendor: Specialized processing functions can be purchased as IP from companies and individuals who target specific applications. These are often delivered as encrypted cores or “black-boxes” where just the signals entering and leaving the processing block is exposed for the purchaser to connect to the rest of his IP.

Custom IP created by you: In most designs, the bulk of the IP is usually created by the engineer responsible for designing the system.

With IP coming from these different sources an immediate challenge is making sure the different IP components have similar signal interfaces enabling data to pass from one block to the next (**Figure 1**).

The FPGA manufacturers have addressed this by standardizing on a common signal interface specification. Borrowed from Arm processor technology, both Xilinx and Altera are using the Advanced eXtensible Interface (AXI). Now in its second generation (AXI4), is an open standard, on-chip interconnect specification for the connection of functional

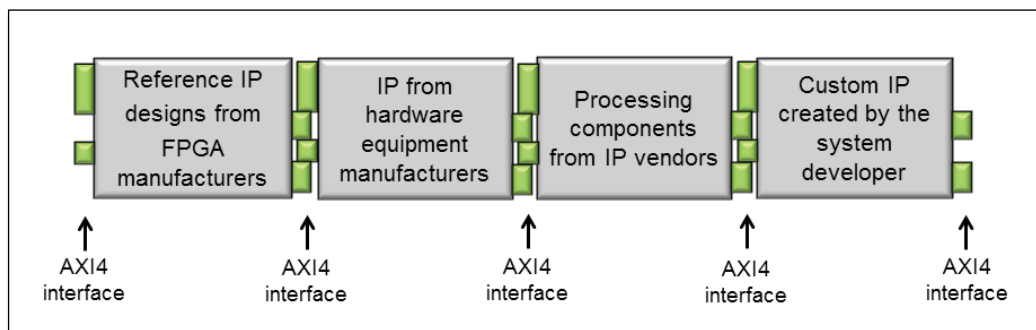


FIGURE 2

AXI4 provides a common interface definition.

AXI4 (full)	For high-performance memory-mapped requirements. While providing high throughput, it does come with the cost of using more FPGA resources to implement. For many applications the combination of AXI4-Lite and AXI4-Stream, described below, can provide similar performance at the cost of less FPGA resources.
AXI4-Lite	For simple, lower-throughput memory-mapped communication. For example, read and write access to status and control registers.
AXI4-Stream	Provides an interface for high-speed data streaming.

TABLE 1

AXI4 is flexible enough to handle different signal requirements for different types of processing and data moving IP. AXI4 does this by providing three different types of interfaces.

blocks in system-on-chip designs. Now extended to all FPGA IP, it provides the common interface for IP from different sources to remain compatible, providing a level of plug and play functionality not previously possible (**Figure 2**).

In addition to providing IP compatibility from different sources, IP reuse is enabled when a block from a previous design can get reused in a new design. This is possible when the interface signals are the same for both the old and new designs. Overall productivity is also improved when developers need to learn only a single interface protocol for IP.

For AXI4 to be useful as a universal standard, it must be flexible enough to handle different signal requirements for different types of processing and data moving IP. AXI4 accomplishes this by providing three different types of interfaces as shown in **Table 1**.

To add further flexibility, FPGA manufacturer's like Xilinx provide cores for interconnecting AXI4 interfaces of different widths and speeds. The Xilinx AXI Interconnect IP core can accept and connect AXI4 interfaces with data widths of 32, 64, 128, 256, 512 or 1024 bits and with different synchronous or asynchronous clock rates.

BLOCK DIAGRAM DESIGN TOOLS

As IP designs become larger and more complex, the job of structuring and visualizing data flows and the hierarchy of the design becomes an increasing challenge. Both Xilinx and Altera have addressed this in recent versions of their Vivado and Quartus Prime tools. For this example, we'll look at Xilinx Vivado and the included IP Integrator tool (**Figure 3**). Building on the foundation of the common signal interface provided by AXI4, IP Integrator allows IP to be "packaged" into blocks that can be interconnected on a graphical design canvas.

Because AXI4 is the common interface used to create the signals entering and leaving the blocks, much of the wiring details can be abstracted leaving the interconnects to become "wires" that can be "drawn" between signal ports on each block. As described earlier, Xilinx's AXI Interconnect IP core handles buses of different widths and speeds, further simplifying the interconnection of blocks that might otherwise not be compatible.

A standard is only valuable when it's accepted and used. As mentioned earlier, both Altera and Xilinx support AXI4 with virtually all of Xilinx IP delivered in this format. A noticeable shift to AXI4 support can also be seen from IP suppliers and hardware manufacturers with FPGA based products. Pentek, as a designer and manufacturer of high-performance FPGA based data acquisition and processing products, has also embraced AXI4 and block diagram design. To edit the IP design of a



ABOUT THE AUTHOR

Robert Sgandurra serves as director of product management for Pentek's DSP, data acquisition, digital receiver and software products where he's responsible for product definition, educating and presenting technical seminars to systems engineers and Pentek's sales force on the latest product technologies. Prior to joining Pentek, his background included seven years in the medical electronics industry where he designed and managed projects for ultrasound imaging. Robert joined Pentek in 1994, working as an application engineer and system integrator.

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials

RESOURCES

Pentek | www.pentek.com

Pentek product, an FPGA engineer opens Pentek's Navigator FPGA Design Kit in Vivado. He or she then has immediate access to the product's entire FPGA design as a block diagram. Individual IP cores can be removed, modified, or replaced with custom IP to meet the application's processing requirements. Viewing the product's FPGA design as a block diagram enables the designer to see the products functions at a higher level and simplifies the design processes by working at the "interface" and not the "signal" level. If at any time a designer needs to work with the VHDL code directly, it is always accessible in a source window, as well as full on-line documentation of every Pentek IP core.

SYNCHRONIZING IP & SOFTWARE

Up to this point we've been looking mostly at FPGA IP and the challenges faced by IP designers. And while some processing done in FPGAs is fixed with no runtime interface that needs to be controlled or initiated for operation, much of the IP created for FPGAs looks like a piece of hardware with control and status registers. And just like a piece of hardware, software—running typically on a Windows or Linux based machine—is controlling the FPGA through an interface like PCIe or Ethernet. But as mentioned earlier, FPGAs and the very fluid nature of hardware designs created with FPGAs, creates a challenge for software engineers. During the development of a project or product, the IP and

the software to control it will often need to go through many iterations. From initial design to debugging and through feature changes and redesign, the jobs of the IP designer and software engineer are intimately tied together as changes in the IP require changes in the software. For a small project this can be the same person, but often—especially for large projects—there can be teams of IP and software engineers at work.

Here again, the FPGA manufacturers have risen to the challenge and their latest tool offerings include features to generate templates from the IP design that can be used as the framework of software for control and status of the FPGA functions. The challenge can become greater when modifying existing IP and software. As an example, all Pentek products are delivered with a full suite of IP based functions. A typical Pentek product is built around a high-performance FPGA surrounded by additional hardware include analog to digital and digital to analog converters, hardware circuitry for generating and synchronizing clocks, SDRAM or SRAM memory, specialized optical interfaces, a PCIe and Ethernet interface and so on. At some level, each of these hardware features is controlled by a piece of IP in the FPGA. Add to that IP based data processing functions like, DMA engines, data acquisition and waveform generator engines, data tagging and metadata creation, FIR filters, digital downconverters and so on.

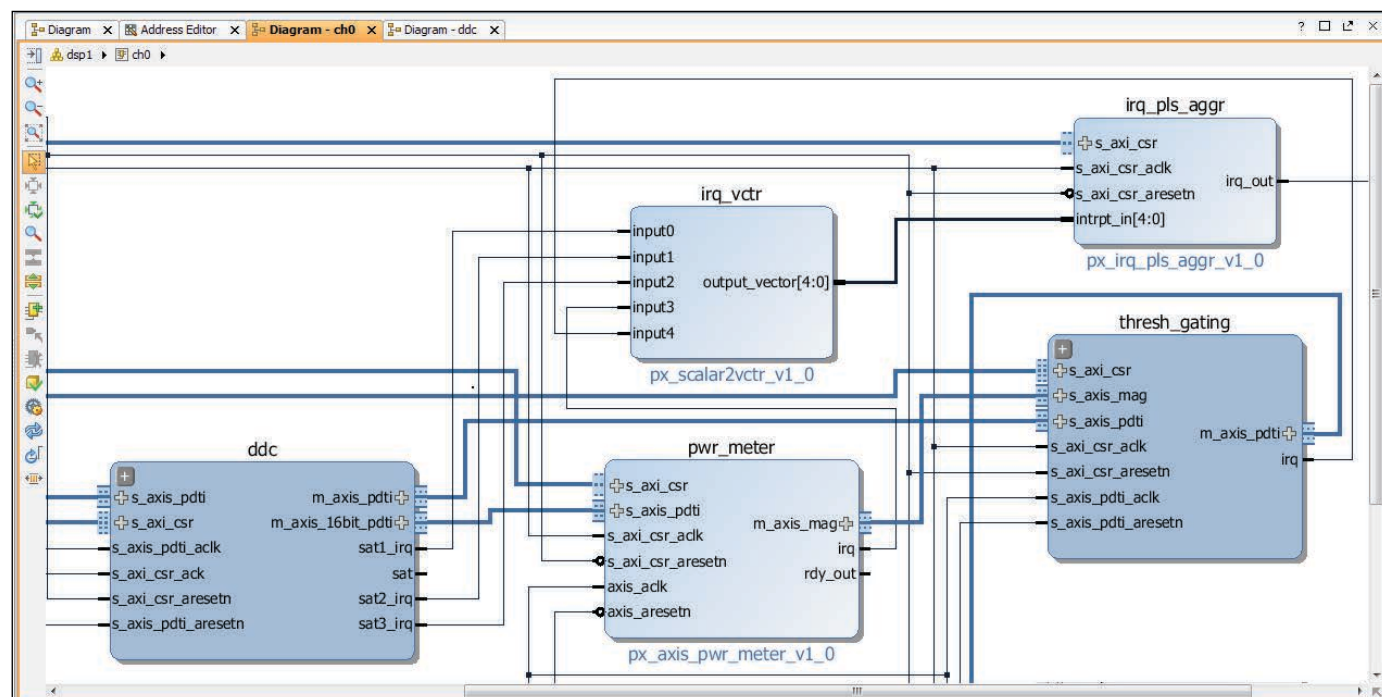
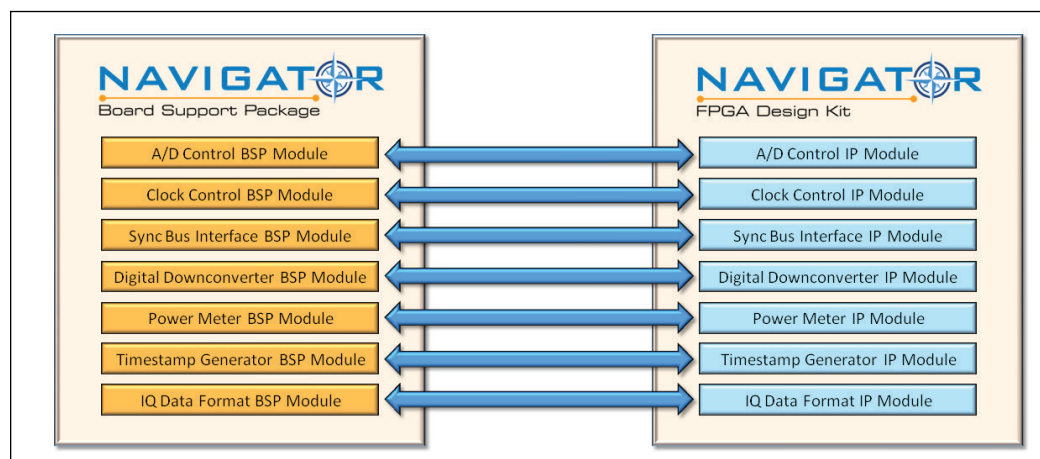


FIGURE 3

Shown here is a design consisting of IP blocks connected in Xilinx's IP Integrator.

FIGURE 4


Navigator provides a one-to-one relationship between IP and BSP modules.



To provide a complete product, all of the IP based functions need software libraries provided to control the IP. While some product users will be able to satisfy their system requirements with only the suite of IP functions provided, most will need to modify the supplied IP or create custom processing for their application. With each change in IP, a software change is most likely required. Here Pentek has taken a very specific approach to help keep IP and software changes synchronized. The company's Navigator Board Support Package (BSP) is the complimentary tool to the Navigator FPGA Design Kit. Designed to work together, every IP module function is matched to a complimentary BSP module (**Figure 4**). As a change is made to an IP module, the matching BSP function can be easily found and the required change can be made in the software.

IP PLUMBING WORK

The IP library also includes modules that are not part of the shipped hardware product but may be used by the IP designer as needed. The DMA engines found in the library are an example. Consider an example where built-in board functions stream data from an analog to digital converter, through some default processing and out through the PCIe interface where it can be sent to a computer for recording. The user in that example may also need to split off the data to feed some custom processing or analysis function. The Navigator IP library includes a number of DMA engines for streaming data. The IP designer can "draw" this block into the board design, connecting it between the existing data streaming path and his custom processing IP block. The Navigator Board Support Package includes BSP modules for controlling these DMA IP modules that can be turned on as needed. While the IP designer still needs to create a software function to control his or her custom processing IP, the Navigator tools provide much of the "plumbing" to enable the designers custom IP.

As each new generation of FPGA grows in processing power and logic density, the trend is for IP designs to grow larger and more complex to exploit the increasing hardware capabilities. While this constant migration towards higher density hardware and IP designs can deliver advantages in overall size, cost and power, it often complicates the IP developer and software engineer's job by requiring larger and more complex IP and software designs. FPGA manufactures as well as IP vendors and FPGA based product manufactures like Pentek recognize this trend. The industry wide adaptation of an interface like AXI4 can make the process of IP design and reuse more efficient, and individual innovations from manufactures in the FPGA space can help manage the very complex process of IP based design. 

When it comes to robotics, the future is now!

Advanced Control Robotics simplifies the theory and best practices of advanced robot technologies, making it ideal reading for beginners and experts alike.

With this book, you'll learn about:

- Communication
- Technologies
- Control Robotics
- Embedded Technology
- Programming Language
- Visual Debugging... and more

ADVANCED CONTROL ROBOTICS
HANNO SANDER

Get it today, cc-webshop.com