Implementing PCI Express Designs using FPGAs

Abhijit Athavale, Xilinx - June 07, 2006

As the industry transitions from shared, arbitrated, bus-based system interconnect architectures like PCI to high-performance, serial, point-to-point architectures like PCI Express, designers are looking for implementations that allow them to save costs without affecting performance. They must choose from several alternatives - such as FPGAs and several flavors of ASICs and ASSPs - based on system requirements such as performance, volume, cost, etc. While each approach has its merits, FPGAs offer a total cost of ownership advantage as compared to ASICs or ASSPs for most embedded applications. With a short time-to-market and no up-front NRE, programmable logic eliminates the excessive costs related to the supply chain such as inventory and costs related to multiple suppliers and multiple product qualification cycles. Plus, the risk reduction benefits associated with FPGAs, their continuously improving performance, and their ability to offer instant gratification to the designer cannot be overlooked.

Cell-based ASICs/ASSPs are still the way to go when performance or functionality requirements are extreme and volume requirements are in the millions per year. When implementing a root complex, for example, chances are that an ASSP solution is already available. Of course, if the plan is to build a custom root complex that implements a superset or subset of the specification, then an FPGA would be ideal. When building a switch with multiple ports and very high bandwidth requirements, an existing ASSP or ASIC implementation may be available. If the implementation will be a PCI Express endpoint - like most embedded applications - FPGAs will most-likely be the ideal choice.

Today's high-end 90nm FPGAs such as the Virtex-4 family offer performance that matches those of ASICs/ASSPs for x1, x4 or x8 lane PCI Express endpoint implementations, while the Spartan-3 generation FPGAs offer very low cost x1 PCI Express implementations for mid-high volume markets. Before selecting a technology for implementing a PCI Express design, designers must carefully look at several factors such as logic performance, gate count, link efficiency, compliance testing, and ease of implementation. This article will review the considerations for building PCI Express design in the latest 90nm FPGAs.

**Single-chip or two-chip solution?**
The designer must first consider how much bandwidth their design will need. The PCI Express specification defines lane widths from an x1 implementation that delivers up to 2 Gbps (gigabits-per-second) aggregate bandwidth, all the way up to an x32 realization that provides up to 64 Gbps. FPGAs are ideally suited for widths from one to eight lanes and designs can be implemented using two different approaches. The first is to use a single chip FPGA solution (**Fig 1**) such as the Xilinx Virtex-4 FX with an integrated PHY.
1. Single-chip FPGA-based solution

Alternatively, the second approach is to use a two-chip solution (Fig 2) in which a low-cost FPGA such as a Spartan-3 is connected to a standalone PCI Express PHY such as Philips PX1011A over a PIPE (Physical Interface for PCI Express) interface.

2. Two-chip FPGA-based solution

In both implementations, the PCI Express logical and transaction layers will be implemented using FPGA fabric resources; however, the single-chip solution allows more performance while the two-chip solution is more cost effective. Both the internal and/or external PHYs are designed to be electrically and physically compliant to the PCI Express specification and to take the risk out of using high-speed serial interfaces. When using an ASIC-based approach, integration of the analog high-speed serial I/O with the digital portion of the design is always most challenging and can produce multiple design revisions. Having said this, if the design will require 16 or more lanes of PCI Express – a graphics port, for example – then a cell based ASIC or an ASSP is probably a better solution.

Resource usage

It is very important to choose the right FPGA device in the right package and speed grade when implementing PCI Express. As the logical and transaction layers will be implemented using the programmable fabric, designers must pay special attention to the number of block RAMs, look-up
Another factor to consider is the width of the data path. If the choice is to implement a x1 lane implementation with a 32-bit data path, the number of FPGA fabric resources can be 25-30% less compared to a 64-bit data path implementation. For example, a x1 lane endpoint in a Virtex-4 FPGA with a 32-bit data path requires 6300 lookup tables (LUT) and 5100 flip flops (FF), while a 64-bit implementation will need 7800 LUT and 7000 FF resources. The user interface for the first design runs at 62.5 MHz, while the second design runs at 31.25 MHz. While not much of a concern for x1 implementations, 32-bit data path designs will require faster speed grade selection if using x4 lanes. The x8 lane implementations in FPGAs will also need a faster speed grade as it would be necessary to use a 64-bit data path with the user interface frequency of 250 MHz.

The choice of package directly impacts the placement of regular pins and serial transceivers on board. This is really critical if implementing a two-chip solution over PIPE. PIPE is a source-synchronous interface that requires a minimum of ~32 pins on the package per lane; an x4 lane PIPE implementation will require ~140 pins just to implement the interface. Plus, PIPE data is clocked either at 250 MHz or 125 MHz - depending on whether it is a single data rate (SDR) or dual data rate (DDR) implementation - thereby making the choice of package even more important. If the device/package/speed grade combination is too tight a fit, the implementation may work but board routability can become a major concern.

**Design considerations**

The designer must also look at factors such as link latency, link efficiency, and performance while implementing PCI Express. These factors can also affect the way the user application is designed.

**Latency:** While the latency of a PCI Express controller will not have a huge impact on the overall system latency, it does affect the performance of the interface. Using a narrower data path will always be better from a latency point of view. The PCI Express latency is the number of cycles it takes to transmit a packet and receive that packet across the physical, logical, and transaction layers. A typical x8 lane PCI Express endpoint will have a latency of 20 - 25 cycles. At 250 MHz, that translates into 80 - 100 ns. If the interface is implemented with a 128-bit data path to make timing easier, i.e. at 125 MHz, the latency doubles to 160 - 200 ns. The latest 90 nm FPGAs like the Virtex-4 FX can implement a 64-bit data path at 250 MHz, a performance that is comparable to ASICs.

**Link Efficiency:** Link efficiency is a function of latency, user application design, payload size, and overhead. Typically, link efficiency increases with payload size up to a point. Normally, a payload of 256 bytes can get you a theoretical efficiency of 93% (256 payload bytes + 12 header bytes + 8 framing bytes). While PCI Express allows packet sizes up to 4K bytes, most systems will not see improved performance with payload sizes larger than 256 or 512 bytes maximum. An x4 or x8 PCI Express implementation in the latest 90nm FPGAs will give a link efficiency of 88 - 89% (*Fig 3*) due to link protocol overhead (ACK/NAK, re-transmitted packets) and Flow Control Protocol (credit reporting).
3. PCI Express link efficiency

Designers have better control on link efficiency when using FPGAs for implementation as this allows them to choose the receive buffer size that corresponds well with the end point implementation. If both link partners do not implement the data path in a similar way, the internal latencies on both link partners will be different. For example, if link partner one uses the 64-bit, 250 MHz implementation with a latency of 80 ns; while link partner two uses a 128-bit, 125 MHz implementation with a latency of 160 ns; then the combined latency for the link will be 240 ns. Now, if link partner one's receive buffer was designed for a latency of 160 ns expecting that the link partner would also be a 64-bit, 250 MHz implementation, then the link efficiency would go down. In case of an ASIC implementation, it would be impossible to change the size of the receive buffer meaning that loss of efficiency is real and permanent.

User application design will also have an impact on the link efficiency. The user application must be designed so that it drains the receive buffer of the PCI Express interface regularly and also that it keeps the transmit buffer full all the time. If the user application does not use packets received right away – or does not respond to transmit requests immediately - the overall link efficiency will be affected regardless of the performance of the interface.

Another thing to consider is whether a DMA controller is needed. If designing with an Intel processor, for example, it is more than likely that a DMA will need to be implemented (since Intel processors cannot perform bursts longer than 1 DWORD) translating into poor link utilization and efficiency. Most embedded CPUs can transmit longer than 1 DWORD bursts, so the link efficiency for such designs can be effectively managed with a good FIFO design.

Other considerations

Other factors such as availability of commercially available IP, development boards, and compliance requirements are important considerations when starting a design. If ready-to-go IP and boards are accessible, the design and test time can be cut significantly. If working with a commercially-available, proven, compliant PCI Express interface, then designers can focus on the most value added part of the design – the user application. Most FPGA vendors offer PCI Express IP Cores – either internally developed or through partners. Any of these can be used in conjunction with the development boards offered by these vendors.

If the entire solution has been validated at a PCI SIG (www.pcisig.com) PCI Express compliance workshop (“plug fest”), then it is pretty much guaranteed that your PCI Express portion of the design will always work. When a solution has been validated at a plug fest, it means that the vendor has invested significant time and resources in PCI Express to make sure that their solution interoperates with other industry standard products and also passes all PCI Express compliance tests. This also means that the vendor is serious about supporting PCI Express and has lot of expertise using the technology. Choosing the right IP and board can significantly reduce time to market.
Summary
Today’s 90 nm FPGAs offer high performance and cost effective platforms to implement PCI Express solutions. At the high end of the spectrum, FPGAs such as Virtex-4 can offer performance similar to ASICs for x1 – x8 lane implementations, while at the price-sensitive end, Spartan-3 FPGAs can offer cost that is similar or lower to ASSPs for x1 implementations. These factors combined with the inherent programmable logic advantages of flexibility, reprogrammability and risk reduction make FPGAs the most ideal platforms for PCI Express.

References

Abhijit Athavale is a Senior Marketing Manager of Connectivity Solutions at Xilinx. Prior to joining Xilinx in 1995, he was an R&D engineer designing communications products at Meltron. Abhijit holds a Bachelor’s Degree in Electrical Engineering from University of Pune in India and Master’s Degree in Electrical Engineering from Texas A&M University. He is an accomplished speaker and author of several published papers. Abhijit can be reached at abhijit.athavale@xilinx.com