Using FPGAs as a Flexible PCI Interface solution

Jim McManus, PCI Applications Engineer, Xilinx Inc

Why do PCI in FPGAs? One of the key reasons is the flexibility only available in FPGAs. This flexibility can save significant time-to-market when designing a fully PCI compliant board, where the design can be verified and debugged with several different PC platforms before it is released for manufacturing. This technical session explores the System Level Integration possibilities available from FPGAs using a flexible PCI interface. Standard PCI Chipsets and Fixed FPGA PCI cores are rigid and unable to accommodate changes. The Modular FPGA approach gives users a choice of FIFOs, DMAs and control over transaction handling that is unavailable elsewhere.

Introduction

Interfacing your custom design to a PCI bus is a challenging proposition under almost all circumstances. However, the phenomenal popularity of PCI has created a market for a variety of solutions that allow you to interface to the PCI bus. FPGAs have become an increasingly popular solution since they allow the designer to have programmable control over the design without the restrictions of Standard PCI Chipsets and the delays of ASIC design. But not all FPGA PCI designs were created equal; there are some significant differences between the current FPGA PCI designs. It is important to select a PCI design that allows the user to take a very modular approach to doing PCI. Since the backend interface is not predetermined, this modular approach facilitates deterministic transaction handling (the ability to control the PCI transactions to match your backend design's data requirements). The modular approach also permits a high degree of System Level Integration, which allows the designer to approach the system-on-a-chip ideal.

Selecting the Right PCI Solution

When selecting a PCI solution, the following requirements should be considered. These requirements are summed up in Table 1.

Customizable FIFO, DMAs and User Design

Everyone's design has unique requirements on FIFOs, DMA and the backend design. The right PCI solution should not have a predetermined set of FIFOs and DMA. The designer must be free to use the type and number of channels of DMA that his design requires. Likewise, FIFO configuration and depth is not fixed. The user can interface his logic as needed to the backend of the PCI Interface.

Parameterizable PCI Features.

The PCI solution must be easy to configure for the designer's unique application. That way, the designer doesn't have to learn the actual implementation of the complex PCI core. The user should be able to set any of the BARs to Memory or I/O, and to any desired decode size. The Subsystem IDs must be settable either internally in the core, or from an external bus, if the design requires loading the external subsystem from off chip. If the new PCI V2.2 engineering change requests (ECRs) for power management are of interest, the PCI interface must include a hook to point to these linked list registers.

Time-to-Market

Time-to-market is becoming an increasingly important factor for hardware designers. The right PCI solution allows the designer to get his product to market faster than all other approaches. The solution should include a PCI prototyping system to promote fast time-to-market. The PCI interface must allow quick design cycles, without leaving the lab. Since an FPGA design can be recompiled in a short time, the designer can do several spins of his design in a single day. A PCI prototyping system allows immediate testing of a PCI design in hardware.

PCI V2.1 Compliance

PCI V2.1 compliance must be considered when choosing a PCI solution, as some PCI Interfaces have been shipped with compliance problems. For example, one FPGA solution actually used multiple pins per signal, which causes PCI loading issues. A non-compliant zero wait state implementation was also seen in some of these programmable solutions. It is very important to obtain a completely filled out PCI compliance checklist from a potential PCI solution vendor. It is important that the solution has been tested on a number of different PCI machines, as well as many different embedded systems to assure PCI problems do not crop up in the field. The PCI solution provider should also be a regular attendee of the PCI compliance workshops to insure their solutions continue to work on a wide variety of PCs. Be sure to check errata lists for the PCI solutions you examine.

Designer Flexibility and Predictability

The designer must be able to handle transactions as required by the backend design, without disturbing the most critical paths. The designer can set up his backend to handle transactions deterministically. Deterministic transaction handling means you can match the PCI Interface to your backend design, not the other way around. Fixed PCI solutions only allow transactions to proceed on their terms, and not yours.

Sustained Bandwidth

The longer the burst length, the closer your design will come to the 132 Mbyte/s maximum performance of the PCI bus. To achieve this long burst length, the design must continuously source or sink the data by means of a FIFO or RAM. The PCI solution must support long bursts if a high sustained bandwidth is important.

To achieve this high sustained bandwidth, variable FIFO depth and flexibility in the FIFO arrangement must be supported. Since the emptying or filling of a single, single-port FIFO takes almost as long as the transaction, flexibility in arranging the FIFOs is required. Ideally, the FIFOs should be de-coupled from the PCI interface, allowing the designer freedom to implement the FIFOs as needed.

Several possible FIFO arrangements must be supported:

- Dual FIFOs - One for read transactions and one for write transactions.
- Dual-port FIFOs - That can be filled and emptied at the same time.
- Ping-Pong FIFOs – Fill one while the other is emptying.
- Deep FIFOs – To the required depth.
- External FIFOs – If the depth is too large to implement in the device, it should support SRAM and SDRAM speeds so a very large FIFO could be implemented off chip in lower cost RAM chips.

PCI Solution Requirements	Standard PCI Chips	ASIC	Fixed PCI Core	Modular PCI Core
Customizable FIFO, DMA, and User Design	Poor	Good	Poor	Good
Parameterizable PCI features	Fair	Good	Fair	Good
Time-to-market	Fair	Poor	Good	Good
PCI V2.1 Compliance	Check errata list	Check errata list	Check errata list	Good
Design Flexibility	Poor	Depends	Fair	Good
Design Predictability	Good	Depends	Fair	Good
Sustained Bandwidth	Fair	Good	Poor	Good

Table 1: Summary of PCI solution Requirements

Comparison of Available Solutions

Available PCI solutions have evolved into four distinct types today:

- Standard PCI Chipsets
- Soft coded, technology independent ASIC cores
- Fixed (shrink-wrapped) FPGA cores
- Modular FPGA cores with an open backend interface

Each of these has its advantages and disadvantages; we will explore each in detail.

Standard PCI Chipsets

The first type, Standard PCI Chipsets such as those made by AMCC, PLX, Tundra, V3, and others. These have been used in a variety of applications, since they were the first to this market.

These devices are often a good fit when targeting the specific application that the chipset was designed for, provided that no customization is required. Specific applications for these devices are interfaces to certain embedded processors such as the Intel i960, and the AMD Am29k family. Prior to the advent of FPGA-based PCI solutions, this was the only solution available to designers who did not need ASIC volumes, so these devices have often been used in lower volume non-specific applications. See Figure 1 and Figure 2 for an example of using these devices in both Specific and Custom Applications.

Benefits:

1. These devices have no NREs other than the manufacturer's design kit.
2. Easy to use in the application it is intended for.

When used in a custom application, Standard PCI Chipsets have several drawbacks:

1. They exhibit low performance due to the mismatch in the interface. FIFO depth can not be varied, though some may make provisions for off chip FIFOs. DMA cannot be varied; e. g. if you need more channels of DMA than these devices provide, your backend design must work around these limitations.
2. Difficult to interface to custom applications. These devices are not easily used in custom applications since the backend interface targets a specific processor and lacks programmability (other than the ability to modify PCI configuration space parameters). This forces the use of external glue logic to connect to the user's design.
3. Poor integration. When used in a custom application, a PLD is usually required to implement the glue logic. This results in a two-chip solution. See figure 2 for an example of this, using these devices in a custom application.
4. Cost can be an issue, especially when a PLD is required to mate the non-specific application to the Standard PCI Chipset. In some simple cases, only a CPLD is needed; however, adding a more complex FPGA to the mix may almost double the cost of the solution.
5. Deterministic design is not possible in a non-specific application. If you need to handle transactions in a specific manner, this will not be possible, since the transaction handling is predetermined and cannot be modified.
6. PCI Compliance can be a problem in these devices. If your design is impacted by one the of errata in these devices, your only recourse may be to await the next revision of the device.

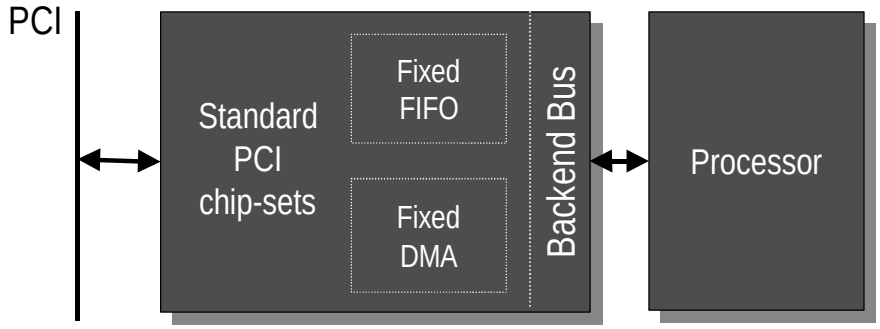

Figure 1: Standard PCI Chipset Interface used in a Specific Application

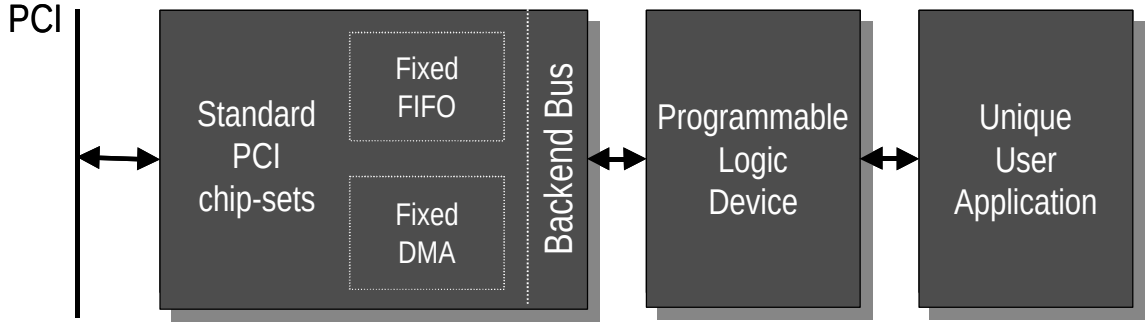

Figure 2: Standard PCI Chipset Interface used in a Custom Application

Soft-Coded, Technology Independent ASIC Cores

The second type, soft coded ASIC Cores are frequently used by designers who require an ASIC solution. This approach is useful in high volume applications, or where design requirements exceed available FPGA sizes. Some of the vendors of this technology are SAND, Virtual Chips, and some of the ASIC manufacturers.

Benefits:

1. ASIC Cores are generally customizable and allow low cost if used in very high volume
2. Good integration can be achieved, provided the ASIC works as planned; bugs in the ASIC could require the addition of a PLD to fix problems.
3. Deterministic design may be possible, if the core vendor supports modification of the core.

There are several drawbacks to the ASIC PCI Cores:

1. Usually requires very high volumes to justify high NRE and licensing cost and design risk.
2. The opportunity must be worthwhile to the ASIC house; cherrypicking design opportunities is a fact of life with some ASIC houses.
3. High NRE on the ASIC and high license cost on the PCI core per design. Some core vendors may charge a royalty per device.
4. High design risk, as problems in the ASIC will require a respin, or a PLD. The latter option reduces the design's integration.
5. ASICs can have long turnaround times. For designs under intense time to market pressure, this may not be the best choice.
6. Attempting to use a core designed for an ASIC in an FPGA is difficult, if not impossible, as the core is not optimized for a given FPGA configuration. Synthesis tools and FPGA place and route tools have not yet reached the degree of sophistication required to perform this feat.

Fixed (Shrink-wrapped) FPGA PCI Cores

Fixed or shrink-wrapped FPGA PCI Cores were an early method of implementing PCI in FPGAs. By integrating the PCI interface in an FPGA, a higher integration than Standard PCI Chipsets is possible (a single chip interface), but requires the user to live with any inherent limitations of the design as it is difficult to modify the design without significant engineering effort. Currently the two major suppliers of this type of PCI core are Altera and Lucent. These designs have a single, single-ported FIFO and a simple DMA.

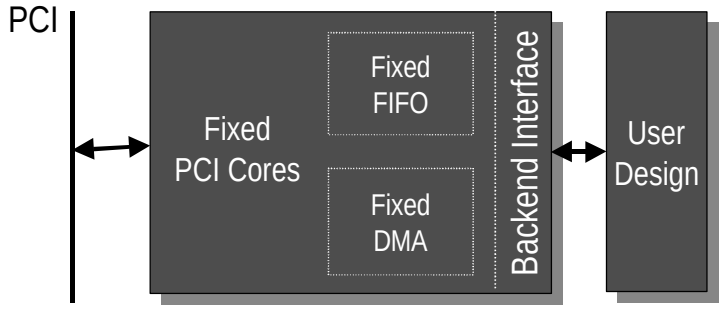

Figure 3: Fixed FPGA PCI Cores

Benefits:

1. Easy to use for applications that do not require modifications of the core.

There are a number of drawbacks to this approach:

1. Low performance in some applications. If the core only supports a single FIFO of a single port type, the core will not allow filling the FIFO and doing PCI bus transactions at the same time. This has the effect of limiting sustained PCI burst bandwidth to < 50 Mbytes/sec. In certain cases, these cores are not capable of bursting, which further restricts bandwidth to < 20 Mbytes/sec.
2. Deterministic Design is not possible with these cores since they do not lend themselves to major modifications. Since the entire core is one design, making modifications inside the core to change the transaction ordering/processing may cause the core to fail.
3. These is an NRE associated with these cores.
4. Fixed FIFOs and DMA. These designs use a fixed FIFO and DMA arrangement. If your FIFO or DMA requirements do not match the interface exactly, then your design must work around these limitations.

The shrink-wrapped approach will only work for you if your design requirements are minimal enough to fit in the performance and interface envelope provided.

Modular FPGA PCI Cores with an Open Backend Interface

The final type is the Modular FPGA PCI Cores with open backend interface. An open backend interface allows the inclusion of FIFOs and DMA as the design requirements specify. Modular PCI cores can be fitted to a wide variety of applications, and allow higher sustained bandwidth. The only current vendor of this type of core is Xilinx.

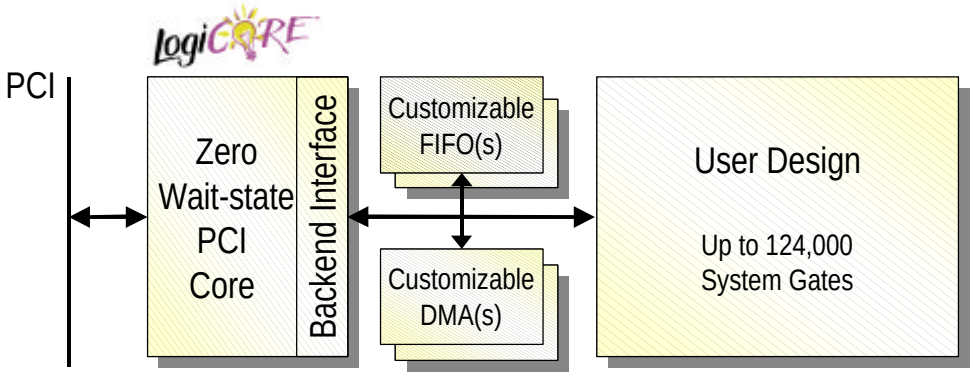

Figure 3: Modular PCI Interface Block Diagram

Benefits:

1. High Performance. With this type of PCI interface, the designer can design to the required performance, up to the total available bandwidth. Since the FIFO depth can be specified, the designer can achieve the required performance for his application.
2. Good integration can be achieved with this interface. It is offered for a variety of different sized FPGAs, which allows the designer to integrate most if not all of his design into the FPGA.
3. Deterministic design is a key feature. This back end interface allows the designer to control how PCI transactions are handled. Depending on the dataflow requirements, the designer can develop a data path that matches his needs. For example, some designs may only need a single DMA channel, while others may need multiple channels. Some designs may need to provide initial latency while others may have fetched the data and can burst without initial latency.
4. Low cost. Not a feature traditionally associated with FPGAs, Modular FPGA PCI cores are available for some FPGAs that cost less than some Standard PCI Chipsets.

Drawbacks to this approach:

1. These is an NRE associated with this core.

Additional Requirements on an FPGA Solution for PCI

A useful FPGA PCI solution facilitates SLI (System level integration) solution – coupling PCI with a user design on a single chip. This higher level of integration allows reduced chip count and smaller overall board space. If a series of FPGAs are supported with PCI cores, then your design risk will be lowered if the design grows out of one device; a large one can be used. Availability in a variety of device/packages is another useful feature.

Another important aspect when choosing a PCI solution for FPGAs is knowing where the PCI core is being developed. For example, each new family of FPGAs requires testing and modification of the design to assure PCI compliance. If a FPGA vendor is not doing the PCI core development in-house, there may be a problem in assuring that the designs are PCI compliant. Another issue to consider is that the rapid pace of IC development, combined with a lack of PCI focus may allow devices to be developed which are not fully PCI complaint. An in-house PCI development team that works with the IC designers to assure full PCI compliance is an important requirement.

The level of design expertise should be similar to that required for using a Standard PCI Chipset. A FPGA PCI Interface shouldn't require the designer to have complete PCI knowledge. When using a good PCI core, the designer can concentrate on his design and leave the complexities of PCI to the FPGA PCI core. Again, this ideal core should have a detailed User's Guide, and example backend designs. The User's Guide should detail step by step how to build the back end interface, how to handle target and initiator transactions, termination conditions and bursting.

The backend should be cleanly and clearly isolated from the PCI core, so modification of the backend does not affect the critical paths in the PCI interface; this will facilitate design reuse. The FPGA PCI Interface should be easily moved into the next design with little added work and without re-synthesis of the PCI core. Once the initial design is done, new versions of the design could then be produced at a very low time cost. PCI critical path timing should be guaranteed to assure that setup and hold time issues do not become an issue. The FPGA timing analysis tools should monitor critical design paths. The FPGA Core tools should automatically check the timing of the critical paths in the FPGA PCI core to determine if timing was met.

For volume production, the ideal FPGA PCI interface should have an ASIC conversion path that will maintain PCI compliance. This will allow a designer to further reduce your costs, plus deliver the product to market while an ASIC is being produced.

Design methodology accommodating varying design formats is desirable. The core should be available in both VHDL and Verilog, with support for various synthesis and simulation tools. VHDL and Verilog simulation models and testbenches are an added plus to help in determining the functionality of your design.

Case Studies

Following are case studies where the Modular FPGA PCI Interface approach were chosen. These are actual designs in which the implementation was done with a Xilinx FPGA.

The Flash Boot Card

In this example the customer designed a Flash ROM Boot card. The purpose of this design was to allow booting a PC from flash memory, which is not supported in current Intel PCI chipsets. The goal was to produce a reference design that PC OEMs could copy to achieve this function, and to support use of Pentiums in embedded systems, using Intel PCI chipsets. This design had several design requirements, as shown in Table 2.

Flash Boot Card Design Requirements	PCI Chipset	ASIC	Fixed PCI Core	Modular PCI Core
Expansion ROM Base Address Register		✓		Flexible
2.1 PCI Compliant Interface	✓	✓	Lucent Only	✓
Fast Design Cycle	✓		✓	✓
Low Cost Solution /Reference Design			✓	✓
ASIC Conversion Path		✓		✓
Deterministic Design		✓		✓

Table 2: Flash Boot Card Design Requirements vs. Available PCI Solutions

It became quickly apparent that only the Flexible FPGA PCI Interface could support these requirements. The customer chose the Xilinx PCI design that had all of these features except the Expansion ROM BAR. The version of the core available at the time had 2 BARs. The customer only required a single BAR, so the second BAR was changed into an Expansion ROM BAR. This was facilitated by the fact that the Xilinx BARs are modules within the PCI core. This modification took about three days of the designer's time to make the change and test the design.

The customer achieved his design requirements, and was able to offer this as a reference design for those needing to boot a Pentium from flash memory.

Host Bridging Capability

An embedded system customer needed to implement the ability to configure the local PCI bus. Their design requirements are listed in Table 2.

Host Bridging Design Requirements	PCI Chipset	ASIC	Fixed PCI Core	Modular PCI Core
Basic Host Bridging	✓	✓		Flexible
3.3 V PCI		✓	Lucent Only	✓
VHDL Design Methodology		✓		✓
Field Upgradable			✓	✓
ASIC Conversion Path		✓		✓
Deterministic Design		✓		✓

Table 3: Host Bridging Design Requirements vs. Available PCI Solutions

The customer chose Xilinx's modular PCI solution. The primary modification needed to support the customer's design was the being able to set the Master Enable bit in the Command/Status Register. One gate was added to the design to give the user direct control over this bit. After this bit is set, configuring other agents on the bus is relatively simple, since the Xilinx PCI Master already supported issuing configuration cycles.

Configuring the host could be done in one of two ways. The first method was direct writes to the PCI registers from inside the design. This method would have required considerable reworking of the core and retesting. The second method was to issue a bus cycle with the Host as the target. Since the master and slave state machines run independently in the Xilinx PCI Interface, the slave state machine is capable of responding to the configuration transaction without interference to the master state machine. The Xilinx PCI core was able to handle bus cycles issued by itself as the master, with it's target as the other agent on the bus. No additional logic (other than the one gate) was needed to support the setting of the internal registers. More details on this are available in the Host Bridge Operation chapter of the Xilinx PCI User's Guide.

Per Data and Address Phase Selectable Parity

This customer wanted even or odd selectable parity on a per address and data phase basis. Only two of the four approaches had a chance at doing this: the Modular FPGA PCI core and soft coded ASIC cores. This design didn't have the volumes to warrant an ASIC, and the customer needed a rapid design cycle plus the ability to do field upgrades, which ruled out using an ASIC core. The customer chose the Xilinx Modular FPGA PCI core that was easily modified to support selectable parity.

Selectable Parity Design Requirements	PCI Chipset	ASIC	Fixed PCI Core	Modular PCI Core
Even or Odd Parity, Selectable		✓		Flexible
Fast Design Cycle	✓		✓	✓
Moderate Volumes	✓		✓	✓
Deterministic Design		✓		✓

Table 4: Selectable Parity Design Requirements

Custom PCI2PCI Bridge design

A customer needed a PCI2PCI bridge with custom logic between the two PCI interfaces. While this is not one of the usual functions (the Modular FPGA PCI Interface only supports type zero configuration space), the customer chose Xilinx, modified the core to support this and built this custom PCI2PCI bridge.

PCI2PCI Bridge Design Requirements	DEC Bridge	ASIC	Fixed PCI Core	Modular PCI Core
PCI2PCI Bridge	✓	✓		Flexible
Custom logic between the PCI interfaces		✓		✓
Fast Design Cycle	✓		✓	✓
ASIC Conversion Path		✓		✓

Table 5: PCI2PCI Bridge Design Requirements

Conclusion

When choosing the right PCI solution, the Modular FPGA PCI core has distinct advantages over other types of PCI interfaces. Standard PCI Chipsets are useful in specific applications. However, for custom applications, the Modular FPGA PCI Interface is the design of choice. ASIC cores have some advantages, such as larger possible design sizes, but for most designs, the Modular FPGA PCI core holds the advantage. Fixed or shrink-wrapped PCI cores hold few advantages over Standard PCI Chipsets and soft ASIC cores but should generally not be chosen over the Modular FPGA PCI interface.

Biography:

Jim McManus is the PCI Applications Engineer in the CORE Solutions Group at Xilinx. Jim has 3 years of PCI experience in applications engineering for PCI designs in FPGAs. He has been involved with the Xilinx LogiCORE PCI design since it's inception. Jim holds a BSEE from Cal Poly Pomona.