Implementation of PCI-Express on FPGA Board

Poornima.P¹, Trupti.Kumbhar² and C.V.Ravishankar³
¹ Sambhram Institute Of Technology/Electronics and communication, Bangalore, India
Email: poonisuma@gmail.com

Abstract - PCI Express is rapidly establishing itself as the successor to PCI, which provides higher performance, increased flexibility, and scalability for next-generation systems. FPGA offers a flexible solution for PCI Express compared with ASIC. Using IP Core and designing a DMA module in FPGA, this paper implemented a high-performance PCI Express system. IP Core handles the protocols defined by PCI Express specification. DMA module accelerates the transmission between FPGA and PC memory, sufficiently exploiting PCI Express bus bandwidth. This article proposes and achieves a PCI Express system having DMA transfer ability and emphatically describes the design of the DMA engine in system-logic, in addition to the corresponding Windows WDM driver and GUI application. The result of performance test is presented finally.

Keywords - PCI Express, Virtex-5, DMA, WDM.

I. INTRODUCTION

As data acquisition rates increase with advances in silicon technologies, larger amounts of data must be transferred to the PC for processing. These transfers are handled by the data bus connecting the device to PC memory. When the PCI bus was first introduced in the early 1990s, it had a unifying effect on the plethora of I/O buses available on PCs at that time, such as VESA local bus, EISA, ISA, and Micro Channel. The PCI bus brought plenty of advantages over previous bus implementations, like processor independence, buffered isolation, bus mastering, and true plug-and-play operation. Although PCI has enjoyed great success, it now faces a series of challenges, including bandwidth limitations, host pin-count limitations, lack of real-time data transfer services and so on. His derivatives such as PCI-X and AGP temporarily revolve the requirement of bandwidth, but the intrinsic problem hasn’t been settled down. The bandwidth of the PCI bus and its derivatives can be significantly less than the theoretical bandwidth due to protocol overhead and bus topology. To satisfy the growing appetite for bandwidth, a new bus technology called PCI Express was introduced. Now, PCI Express is replacing the current I/O bus technologies PCI, PCI-X, AGP. PCI Express offers a serial architecture that alleviates some of the limitations of parallel bus architectures by using clock data recovery (CDR) and differential signaling. Using CDR as opposed to source synchronous clocking lowers pin count, enables superior frequency scalability, and makes data synchronization easier. PCI Express employs dual-simplex point-to-point links to overcome the limitations of a shared bus. The links use high-speed serial transceivers with embedded clock and data differential signals operating at 2.5 Gbits/s with industry standard 8b/10b encoding. A link can consist of a single (x1) lane providing peak bandwidth of 500 Mbytes/s (2 directions x 2.5 Gbits/s x 8/10 encoding).
II. ARCHITECTURE OF PCI EXPRESS

A. Layers

Compared to legacy PCI, the PCI Express protocol is considerably more complex, with three layers—the transaction, data link and physical layers, as shown in figure 1. The Transaction Layer is the upper layer of the PCI Express architecture, and its primary function is to accept, buffer, and disseminate transaction layer packets (TLPs). TLPs communicate information through the use of memory, IO, configuration, and message transactions. The Data Link Layer acts as an intermediate stage between the Transaction Layer and the Physical Layer. Its primary responsibility is to provide a reliable mechanism for the exchange of TLPs between two components on a link. The Physical Layer interfaces the Data Link Layer with signaling technology for link data interchange and is responsible for framing, de-framing, scrambling, descrambling, 8b/10b encoding and decoding of TLPs and DLLPs. Layered protocols have been used for years in data communications. They permit isolation between different functional areas in the protocol and allow updating or upgrading of one or more layers, often without requiring changes in the other layers. For example, new transaction types might be included in newer revisions of a protocol specification that don’t affect lower layers, or the physical media might be changed with no major effects on higher layers. PCI Express uses packets to communicate information between a transmitting component and a receiving component. Figure 2 gives the diagram of two PCI Express devices communicating.

From the transmitting side of a PCI Express transaction, transaction layer packets (TLPs) are formed at the transaction layer with protocol information (transaction type, recipient address, transfer size, etc.) inserted in header fields. These are then passed down to the data link layer to calculate and append a 32-bit cyclic redundancy check (CRC-32) and a sequence ID for link level data integrity management (packet acknowledgement and retry mechanisms).
The physical layer then frames the packet with start and end of packet bytes before lane striping (multi-lane links only), byte scrambling to reduce electromagnetic emissions (EMI), 8b/10b encoding to ensure sufficient transitions for clock and data recovery, and serialization of the 10-bit symbols before transmission across the link to the receiving device. The packet then traverses up the protocol stack at the receiving device until data is extracted and passed to the device core.

III. TOPOLOGY

The topology of PCI Express illustrated in figure 3 consists of Root Complex, Switch, and Endpoint. A Root Complex denotes the root of an I/O hierarchy that connects the CPU/memory subsystem to the I/O. Each port is connected to an endpoint device or a switch which forms a sub-hierarchy. A Switch provides fan-out capability and enables a series of connectors for add-in high-performance I/O. It can be thought of as consisting of two or more logical PCI-to-PCI bridges, each bridge associated with a switch port. One port of a switch pointing in the direction of the root complex is an upstream port. All other ports pointing away from the root complex are downstream ports. An Endpoint is an I/O device connected to the PCI Express, such as graphic, network and storage controller attached to the PCI Express. Endpoints are different from switches that are requesters or completers of PCI Express transactions.

![Fig. 3. Topology of PCI Express system](image)

IV. COMPATIBILITY

Software compatibility is significant for PCI Express. So it uses the same load/store I/O architecture as PCI and PCI-X. This similarity makes PCI Express fully compatible with the PCI software model. Because of this compatibility, current operating systems that are compatible with the PCI software model can boot and run on PCI Express systems without any change to device drivers or the operating system. However, to take advantage of the new, advanced features of PCI Express, software modification will be necessary. PCI Express extends the 256-byte Configuration Space of PCI to 4096 bytes while maintaining compatibility with existing PCI enumeration and configuration software as shown in figure 5. PCI Express accomplishes this by dividing the PCI Express Configuration Space into two regions: the PCI-compatible region that is the first 256 bytes and the extended region the remaining 3840 bytes. The extended region is useful for complex devices that require large numbers of registers to control and monitor the device. What should be noticed is that the operating system you current use has to support accessing extended region besides the device.

V. FPGA-BASED IMPLEMENTATION

A. Selection

Many companies provide PCI Express solutions, because PCI Express bus has lots of prominent advantages. These solutions can be classified into FPGA-based and ASIC-based according to implementation. The typical solution based on ASIC utilizes PEX8311 provided by PLX. PEX8311 is a bridge chip which can translates protocol between PCI Express bus and local bus, meanwhile offers a
DMA engine. The typical solution based on FPGA implements PCI Express using IP core. The IP core plays the role of the bridge chip. The two kinds of solutions are both verified by the PCI-SIG, whose validity and compliance are unquestioned. But FPGA gives designers the ability to create a design that exactly matches their requirements and more flexibility. FPGA is the most likely idea choice. There are two alternatives: one-chip solution and two-chip solution, when adopted FPGA. The one-chip solution utilizes a high-performance FPGA such as Virtex-5 to perform PCI Express protocol, physical interface and LVDS transmission. The two-chip solution connects a low-cost FPGA such as a Spartan-3 to a standalone PCI Express PHY such as Philips PX1011A over a PIPE (Physical Interface for PCI Express) interface to coordinately perform that job. The single-chip solution allows more performance while the two-chip solution is more cost effective. In order to obtain better performance and more functions, Xilinx Virtex-5 FPGA is selected for PCI Express.

B. Design

The aim of this article is to propose and achieve a high-performance PCI Express system which can transfer data between device and PC swiftly. The system is divided into two parts, hardware and software as shown in figure 4. The hardware subsystem adopts Xilinx’s ML505 Virtex-5 FPGA development as the plug board and programs a system-logic residing in FPGA to satisfy system requirement with verilog. The software subsystem consists of application and driver. The two subsystems communicate through PCI Express bus. The logic residing in FPGA have to complete two tasks, communication with PCI Express bus and data transfer from device to PC. The logic is divided into IP core and system-logic as shown in figure 5. IP core is responsible for converting data protocol between local bus and PCI Express bus, while system-logic is responsible for transmitting data to PC.

Xilinx presents an IP core named Endpoint Block Plus for Virtex5, which consists of transaction layer module, data link layer module, physical layer module and configuration management module. The former three modules tackle the respective protocols defined in PCI Express specification. The physical layer module is connected to RocketIO transceiver which is built in FPGA and transmits LVDS signals. The transaction layer module is visible to system-logic, giving several TRN interface signals to exchange data. The configuration management module manages all three layer modules, implements configuration space, power management, interrupt generation, and also offers CFG interface signals to control PCI Express bus for system-logic.

System-logic is indispensable to hardware subsystem, because IP core can’t work separately. System-logic is designed to be composed of transmit engine, receive engine, endpoint memory, DMA engine for demand functions. Transmit engine implements the transmission of TLPs. Receive engine implements the reception of TLPs. Endpoint memory stores the data acquired outside FPGA. DMA engine initiates DMA transfer and manipulates its process. The four modules finish their own work and collaborate with each other to finish an integral job.

Transmit engine is a state processor in nature, complying with PCI Express specification and timing requirement of IP core. Transmit engine has three states, reset state in which it doesn’t packs 8 bytes into TLP header and turn into TLP start state until gets TLP transmission request, TLP start state in which it packs 4 bytes into TLP header and handles the TLP according with the type, payload states in which it packs 4 bytes
data into TLP circular. Figure 6 illustrates the states transition of transmit engine. Receive engine is similar with transmit engine, having four states owing to requiring handshake with transmit engine when tackles specific tasks. In reset state receive engine parses TLP header and turns into TLP start state after receiving TLP, in TLP start state it parses the rest TLP header and handles the TLP according with the type, in wait state it waits for response until other module satisfy its request, in payload state it parses the TLP circular until TLP frame gets end. Figure 7 illustrates the states transition of receive engine.

Endpoint memory is an abstract conception, indicating the memory storing data residing in FPGA. There are two types memory in endpoint memory, FIFO and RAM. FIFO is used to store data form front-end system and RAM is used to store data from PC, because of their storage characteristics. DMA is a technique
used for efficient transfer of data to and from host CPU system memory. During DMA transfer, CPU is not used which means it can perform other operations. There are two types of DMA mode: system DMA and bus master DMA. PCI Express bus introduces bus master DMA, which means a DMA engine is implemented within a PCI Express endpoint device that plays a role of bus master. A DMA engine is a pith of system-logic, which takes charge of controlling bus and moving data to (memory write) or requesting data from (memory read) system memory. Because transmit engine and receive engine have completed single TLP processing which are both connected to TRN interface, the DMA engine is composed of some registers, control signals, and an interrupt module in nature as shown in figure 8, whose main task is to manage that two engines to implement DMA transfer.

<table>
<thead>
<tr>
<th>Image</th>
<th>Diagram of interrupt module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 8.</td>
<td>The diagram of interrupt module</td>
</tr>
</tbody>
</table>

When the DMA engine initiates a DMA write operation, some operational information should be setup by a software application firstly, then it generates a memory write TLP using data in endpoint memory and sends that TLP using transmit engine. When the DMA engine initiates a DMA read operation, there is a different. The DMA engine sends a memory read TLP request using transmit engine, then it accepts data to endpoint memory using receive engine. There are several registers in DMA engine involved in this process. They are Controller Register which is responsible for resetting, starting, setting direction of one DMA transfer; Start Address Register which is responsible for recording the physical memory start address; TLP Size Register which is responsible for recording each TLP size; TLP Count Register which is responsible for recording the sum of TLPs and Status Register which is responsible for displaying the situation of this DMA transfer. An interrupt is generated once the number of TLPs sent or accepted matches the expected value.

<table>
<thead>
<tr>
<th>Image</th>
<th>Diagram of state transition module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 9.</td>
<td>The state transition diagram of interrupt module</td>
</tr>
</tbody>
</table>

DMA engine generates an interrupt to notify PC using interrupt module. The interrupt module is also a state processor in nature. It has four states, reset state in which it doesn’t request a generation of interrupt from IP core and turn into ACK state until a DMA transfer has finished, ACK state in which it waits for an acknowledge from IP core, wait state in which it waits until PC responds or a timer runs out, clear state in which it request an clearance of interrupt from IP core. Figure 9 illustrates the states transition of interrupt module.

A typical sequence of register operation is given next. First step is to resetting DMA engine. Second step is to specify the start address, TLP size and TLP count. Third step is to start DMA operation. Fourth step is to wait for finish of DMA operation. Fifth step is to inform a computer of the finish by generating interrupt. The driver establishes a link between hardware and software. In most modern operation system, a software application is running in user mode in which the software application can’t operate hardware directly. It only can call driver interface sending a request, when intends to access hardware resources. The actual operation accessing hardware resources is accomplished by routines in driver which is running in kernel mode. Some tools such as DDK, DriverStudio, WinDriver aim for easing driver development. Driverstudio
encapsulates DDK into a C++ framework, which increases the speed of development at the same time guarantees the code efficient. Some key functions in that framework which accomplish primary tasks are required to introduce. Above all, driver serializes the read or write request (IRP) sent by a software application in StartIO function. DMAReady function is called in StartIO function for configuring DMA operational information, and then calling StartDMA function. StartDMA function writes the appropriate register to inform FPGA of starting DMA operation. ISR clears the appropriate bit declaring the interrupt has been serviced and schedules a DPC. All succeeding things needed to do after interrupt are executed in DPC. At last, DMAReady is called again to determine which to do: complete or continue.

VI. CONCLUSION

As PCI Express becomes the standard interconnect for next-generation embedded applications, system designers address the challenges associated with more complex physical, data link and transaction layers, as well as the challenges associated with the higher performance requirements. FPGA can provide fully integrated PCI Express solution and DMA engine can obtain higher bus throughput and lower CPU utilization. This article introduced the key technology in PCI Express specification and provided a whole fundamental design to refer.

ACKNOWLEDGMENT

We the authors of this paper would like to grateful acknowledge with thanks to SAIT, Management Bangalore, for providing us the infrastructure and giving us an opportunity to carry out the work. We are very grateful to our Principal SAIT Bangalore, for his kind cooperation and moral support through out this work. We express our gratitude to HOD ECE department SAIT, Bangalore for his constant encouragement valuable timely advise.

AUTHORS PROFILE

Mrs. Poornima. P
Currently working as a Lecturer in Electronics and communication department at Sambhram Institute of Technology, Bangalore.
Specialization: VLSI DESIGN & EMBEDDED SYSTEM.

Mrs. Truti. Kumbhar
Currently working as a Lecturer in Electronics and communication department at Sambhram Institute of Technology, Bangalore.
Specialization: Electronics & Telecommunication.

Prof. C. V. Ravishankar
Currently working as a HOD in Electronics and communication department at Sambhram Institute of Technology, Bangalore.
Specialization: Power Electronics.

REFERENCES