# A High Speed DMA Transaction Method for PCI Express Devices

Bo Li, Yu Peng, Da-Tong Liu, and Xi-Yuan Peng

***Abstract*—A novel PCI Express (peripheral component interconnection express) direct memory access (DMA) transaction method using bridge chip PEX 8311 is proposed. Furthermore, a new method on optimizing PCI Express DMA transaction through improving both bus-efficiency and DMA-efficiency is presented. A finite state machine (FSM) responding for data and address cycles on PCI Express bus is introduced, and a continuous data burst is realized, which greatly promote bus-efficiency. In software design, a driver framework based on Windows driver model (WDM) and three DMA optimizing options for the proposed PCI Express interface are presented to improve DMA-efficiency. Experiments show that both read and write hardware transaction speed in this paper exceed PCI theoretical maximum speed (133 MBytes/s).**

***Index Terms*—Bus-efficiency, DMA-efficiency, direct memory access (DMA), PCI Express.**

## 1. Introduction

As the most important deputy of the third generation I/O bus (3GIO), PCI Express (peripheral component interconnect express) was firstly proposed by Intel in 1997. It is now widely deployed and becoming mainstream of computer I/O bus in the future. PCI Express has four main advantages:

a) PCI Express x1 link has one lane, and this lane contains a pair of separated sending and receiving channel, which doubles the communication bandwidth;

b) PCI Express uses 2.5 G embedded clock and 8 b/10 b encoding mechanism, which produces a 0.5 GBytes/s bidirectional communication bandwidth for x1 link and 16 GBytes/s bidirectional bandwidth for x32 link.

c) PCI Express devices adopt point to point communication and enjoy a monopolized high bandwidth.

d) Software layer of PCI Express is completely compatible with PCI.

PCI Express specification supports two data transaction modes namely single cycle and burst cycle. In this paper the burst mode is deployed and optimized to improve bus-efficiency. Data transacted by PCI Express goes through two parts: computer memory and PCI Express bus. Data transaction using computer memory is called memory-based transaction. Memory-based PCI Express communication can be categorized into normal memory access and direct memory access (DMA). Normal memory access is controlled by CPU and may cost too much CPU resource when used in continuously transactions. DMA works independently of CPU by accessing memory directly. Our method combines bus burst and DMA together, and proposes optimization for both of these two factors.

In this paper, we define data transactions from computer memory to peripheral device as DMA write, and define transactions from peripheral device to computer memory as DMA read. The remainder is organized as follows: related work is briefly introduced in Section 2. Section 3 introduces key technique for a successful PCI Express 2.5 Gbps interface design. A bus-efficiency optimization method is introduced in Section 4 and DMA-efficiency optimization is given in Section 5. Section 6 describes a novel speed-testing method and detailed evaluation. Section 7 concludes the paper.

## 2. Related Work

Because of its architecture merits, PCI Express is becoming more and more popular in high speed cases[1]-[3]. However, very few words about PCI Express transaction method can be found in [4]. Reference [5] introduces a PCI Express data acquisition card, but there is no detailed data transaction method. A successful PCI DMA transaction method is given in [6]. In this paper we extend the method to PCI Express devices together with several new optimization options. Reference [7] describes DMA optimizing options for PEX 8311 but device driver method is not given. PCI driver development is discussed in [8]. In this paper we make an extension of Windows driver model (WDM) driver from PCI to PCI- Express devices, and more details are given in [9] and [10].

Manuscript presented at 2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis, April 28-29, 2009; revised November 4, 2009.

B. Li, Y. Peng, D.-T. Liu, and X.-Y. Peng are with School of Electronic Engineering, Harbin Institute of Technology, Harbin, 150001, China (e-mail: lmsmb@hit.edu.cn, pengyu@hit.edu.cn, ldatong@163.com, and pxy@hit.edu.cn).

B. Li currently is also with Department of Civil and Structural Engineering and Department of Computing of Hong Kong Polytechnic University as a visiting scholar.

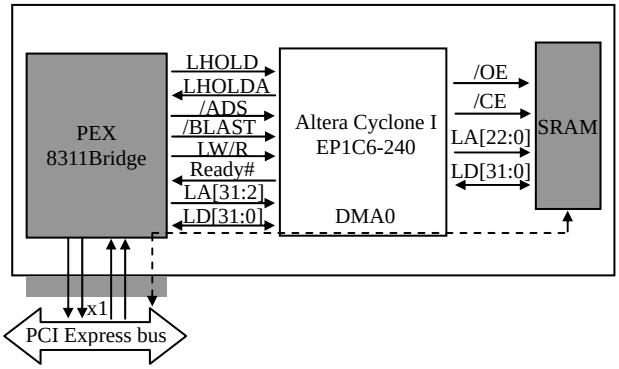


Fig. 1. Framework of PCI Express x1 interface.

## 3. PCI Express Interface Design

PCI Express transaction method in this paper is based on a successful 2.5 Gbps hardware interface design. The interface is implemented by applying a PCI Express bridge chip PEX 8311. PEX 8311 is master mode I/O bus accelerator and is fully compatible with PCI Express specification 1.0. As a bridge chip, PEX8311 has two ports, one is PCIE-bus and the other is LOCAL-bus. LOCAL-bus works with 66 MHz clock, 32 bit bus width and a bandwidth of 266 MBytes/s. PEX 8311 has two DMA channels: DMA0 and DMA1, both of which have a 256 Bytes bidirectional buffer FIFO (first in first out). Method in this paper deploys DMA0 and DMA operations proceeds in form of data blocks. Fig. 1 outlines our PCI Express x1 interface framework.

As shown in Fig. 1, on LOCAL-bus, we deploy 2 MBytes SRAM (static random access memory) with 10 ns the fastest access time as data buffer. The FPGA (field programmable gata array) is Altera Cyclone I series with the largest delay of 7 ns. Hence, both SRAM and FPGA work properly when LOCAL-bus is running at a speed of 66 MHz (15 ns per cycle).

PCI Express interface working at 2.5 GHz frequency means single clock period lasts only 400 ps, which takes challenges for PCB hardware design. We have successfully finished the 2.5 GHz PCB design by restricting to following rules:

1) PCI Express PCB board can be a 4 layers or 6 layers architecture using FR4 material and 62 mil (1.5748 mm) thickness is preferred.

2) PCI Express PCB must have 100 Ω differential impedance, of which a 10% precision should be guaranteed.

3) Length difference of wires in a high-speed differential pair should be guaranteed less than 5 mil (0.127 mm).

4) PCB footprint of differential AC coupling capacitors should be guaranteed to be 0603 or 0402. Capacitance value should be 75 nF to 500 nF. In our design 100 nF works properly.

5) 2.5 GHz high speed differential pair should lie on either top or bottom surface to achieve the best speed.

6) PCB routing for high speed differential pair should be guaranteed a 5 mil (0.127 mm) width and the distance between wires in a high-speed pair should be guaranteed 7 mil (0.1778 mm).

7) The number of via holes on high speed differential pair should be controlled within 2 and be placed symmetrically.

8) High speed differential pairs must not go across different power planes.

9) To meet rule 2) and 3), both power reference plane and ground plane under the golden-edge should be removed.

Following above rules we successfully finished a high speed design, and the PCB layout and hardware fabrication result are as in Fig. 2, where high-speed pairs are routed as serpentine to keep in-pair length difference within 5 mil (0.127 mm).

Layout in Fig. 2 is realized by Cadence Allegro 15.5. Allegro can reduce many difficulties particularly to meet rules 3), 6), 7), and 9). For details, please refer to [9].

## 4. Bus-Efficiency Optimization

As mentioned above, PCI Express has two kinds of data cycle. One is single cycle while the other is burst cycle. PEX-8311 supports both single and burst cycles. Burst cycle approximately transfer 32 bit data per clock during burst, thus draw a higher bus-efficiency than single cycle. In this paper we not only realize PCI Express burst but also present furthermore improvement for burst transactions.

### 4.1 Novel Design of Burst Finite State Machine (FSM)

In PCI Express DMA transaction, DMAC (DMA Controller) is the controller of both LOCAL-bus and PCIE-bus. In our method, DMAC in PEX 8311 generates data and address cycles according PCIE specification 1.0. It is necessary to build both data and address responding mechanism and respond data/address cycles precisely. Our mechanism is realized based on a novel FSM built in FPGA using Verilog HDL (Verilog hardware description language). FSM flow is shown in Fig. 3.

FSM in Fig. 3 is based on timing corporation mechanism of /ADS signal (address available) and /BLAST signal (data ready). One important property of this

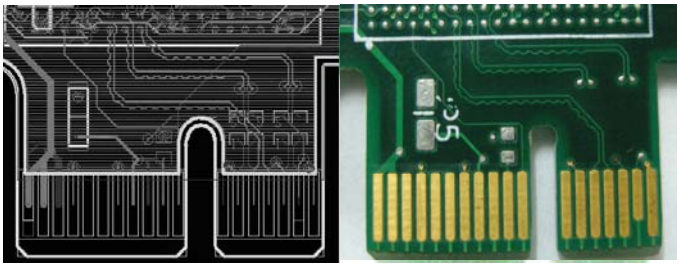

Fig. 2. PCB layout and fabrication result of PCI Express x1 interface.

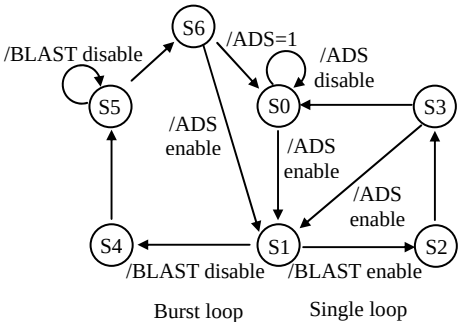


Fig. 3. Working flow of the responding FSM.

FSM is that it can precisely respond to both single cycles and burst cycles, which greatly improve the response flexibility. As shown in Fig. 3, burst responding cycle in the left part has one more looping step (S5) than the right part. Number of continuous looping steps on S5 corresponds to the length of one burst operation. Experiments show that FSM in Fig. 3 meets setup time and hold time requirement of PCI Express bus operation, and responds to data and address cycles reliably.

### 4.2 Continuous Burst Optimization

In our method, PEX 8311 works as a bridge between LOCAL-bus and PCIE-bus, and supports asynchronous burst between these two buses. To enable a burst, we firstly need to enable the burst-enable-bit in DMA0 configuration register, then enable the bterm bit.

BTERM# is a flag signal for PEX 8311. When BTERM# is enabled (set low), a burst cycle is terminated immediately. Through data cycles monitoring, it is surprisingly found that continuous bursts are always interrupted at certain burst length. When keeping BTERM# disable (set high), a much better burst with less interruptions can be achieved. When BTERM# is keeping high, whether a burst will stop depends on if the 256 Bytes receiving FIFO is full (read) or the sending FIFO is empty (write). In our PCI Express x1 card, the 256 Bytes buffer FIFO of PEX8311 is deep enough to compensate speed difference between LOCAL-bus and PCIE-bus. Thus buffer FIFO is seldom read empty or written full. Based on our FSM design, we realize continuous burst controlled by buffer FIFO, and higher bus-efficiency is achieved. For performance evaluation, please refer to Section 7.

## 5. DMA-Efficiency Optimization

Obviously, if a good bus-efficiency is achieved, and data-continuity becomes better, transaction speed will be improved. The purpose of DMA-efficiency optimization is to promote data-continuity. DMA-efficiency optimization in our method can be divided into the three steps as below.

### 5.1 Single DMA Volume Optimization

Single DMA volume (SDV) is the upper bound of data size transacted in one DMA operation. Data volume larger than this size should be decomposed and transacted in more than one DMA. Data decomposition means more DMA setup delay and more software delay, which decrease data transaction efficiency. Hence increasing SDV is an optimization option.

Scatter/gather DMA (S/GDMA) and common buffer DMA (CBDMA) are two existing DMA types, whose main difference is the way to use computer memory. S/GDMA deploys discrete memory while occupying valuable mapping registers. Mapping register is limited resource in computer, thus to increase SDV in S/GDMA is hard. CBDMA does not need mapping register as it uses continuous physical memory. However, to allocate continuous physical memory is hard and improper operations will lead to computer break down. After many experiments, we finally get a reliable CBDMA solution.

Physical memory in CBDMA is called common buffer. In the proposed method, common buffer is allocated in kernel mode by means of PCI Express WDM driver. The driver development tool is DriverStudio 2.7 from Numega Corporation. DriverStudio fully encapsulates Microsoft DDK (driver development kit) and provides friendly function interfaces. Windows 2000 abstracts DMA hardware operation into DMA adapter object so as to manage DMA resources. DriverStudio provide three specified DMA classes for DMA operation: KDmaAdapter aims at building DMA adapter object, KDmaTransfer controls DMA transaction progress, and KCommonDmaBufferr manages common buffer allocation. These three classes work in device driver, their work principle is shown in Fig. 4.

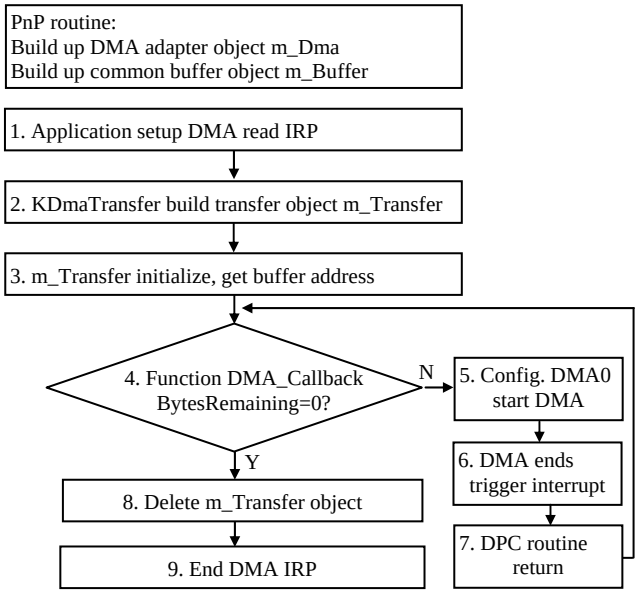


Fig. 4. Driver working flow of CBDMA read.

As shown in Fig. 4, DMA adapter object and common buffer object are built in PnP (plug and play) progress. In the DMA read progress, user application firstly sets up a DMA read IRP (I/O request packet). Then an instance of KDmaTransfer class called m_Transfer is generated, which controls the DMA transaction. As mentioned in Section 3, our DMA is based on block transaction. In Step 4, DMA_Callback is responsible for judging whether remaining data volume is zero, if zero the 4→5→6→7→4 loop will be stopped, otherwise it loops on.

Our method successfully allocates a 2 MBytes common buffer on an IBM desktop, which means a DMA volume smaller than 2 MBytes can be finished within one DMA operation. This method greatly promotes the DMA-efficiency. The speed testing computer has a 2.4 G CPU, one piece of 1 G 667 MHz DDRII memory and a Windows 2000 sp4 operating system. Experiments show that if a larger buffer is allocated, the IBM desktop may not work properly. In fact, in the performance evaluation, a 2 MBytes common buffer is already large enough to achieve a very good transaction performance. SDV optimization is introduced by an example of DMA read, but it also can be applied into DMA write.

### 5.2 PEX8311 DMA Write Prefetch Optimization

Inside PEX 8311, DMAC controls DMA operations. In DMA write, data blocks flow from computer memory to peripheral devices. However, data in computer memory is not always ready for transactions, thus there are unexpected delays.

PEX 8311 provides a prefetch mechanism to minimize delays mentioned above.The prefetch mechanism is to prefetch data and get prepared block transactions. Inside PEX8311, prefetch volume is controlled by PCI control register. 000b means none prefetch, and the maximum value 111b means a 4 kBytes prefetch operation. However, prefetch volume should not exceed total DMA volume. Otherwise time would be wasted for picking useful data from the over-transacted data. In the proposed method, a justify mechanism is programmed in the WDM driver. Before each transaction, total DMA volume is firstly obtained and smaller prefetch size is computed and programmed into control register at step 5 of Fig. 4. Thus an optimization is realized.

### 5.3 DMA Write Loop-Volume Optimization

Inside PEX 8311, a DMA write is divided into smaller loops. Each loop has an upper bound controlled by PCI Express device control register. Default value is 010b meaning maximum 512Bytes in each loop.

In this proposed method, the control register is set to the maximum configuration 111b, thus a maximum 4 kBytes inside DMA write is feasible. Intuitively, the number of data decomposition inside PEX8311 is reduced and data continuity is promoted.

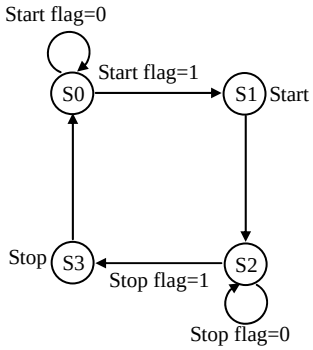


Fig. 5. Working flow of FSM counter.

## 6. Performance Evaluation

Performance of PCI Express DMA transaction depends on computer ability and application environment. Thus transaction performance can only be statistically evaluated. In this paper what we are caring about is the DMA hardware transaction speed. A novel FSM-based speed testing method is proposed. Basic principle is to build a hardware counting FSM in Verilog HDL. The FSM is driven by the LOCAL-bus 66 MHz clock. Working flow of FSM counter is shown in Fig. 5.

S1/S3 is the start/stop status while S2 is the counting step. FSM counter is started at step 5 of Fig. 4. In our test, the last 32 bit of the transacted data is set as the stop trigger of the FSM counter. Whenever the stop trigger shows up on physical data bus, the stop flag is set 1 and the counter is stopped. When we know communication size (volume) and clock cycles it costs (Counter_value), transaction speed (Speed) can be gotten as:

$$\text{Speed} = \frac{\text{Volume}}{(\text{Counter\_value} \times 15 \times 10^{-9})} \text{ (MBytes/s)} \quad (1)$$

Following testing method above, Fig. 6 gives DMA write and DMA read speed statistics. There are 4 groups in Fig. 6, of which from top to bottom are optimized DMA WRITE, optimized DMA read, normal DMA read and normal DMA write. In Fig. 6, each group contains five testing results. We can see that both optimized DMA write and DMA read perform much better than the normal DMA. Both DMA write and DMA read perform better along with the increment of single DMA volume. Best performance of DMA write (166 MBytes/s) and DMA read (136 MBytes/s) in proposed method are both better than PCI theoretically maximum speed (133 MBytes/s). Intuitively, maybe more optimizing options for DMA write in our method lead to the performance difference between DMA write and DMA read. Further possibility research is left for future work.

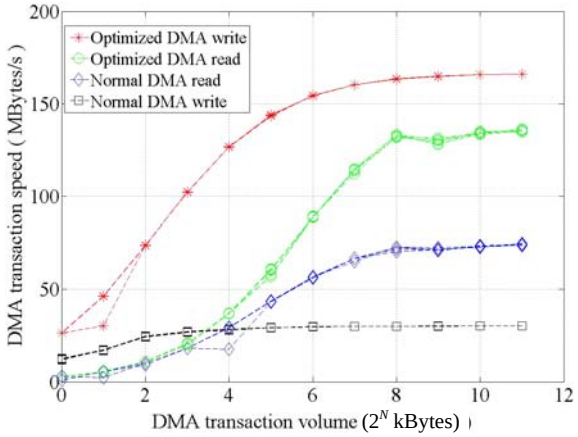


Fig. 6. Speed statistics of DMA transactions.

## 7. Conclusions

In this paper, a PCI Express device solution is presented. And a method to optimize PCI Express DMA transaction is proposed for the first time from two aspects: bus-efficiency and DMA-efficiency. Experiments show that, our DMA transaction method for PCI Express devices can reach a maximum 166 MBytes/s write speed and a maximum 136 MBytes/s read speed, both exceeds PCI maximum speed (133 MBytes/s). The method proposed has been successfully applied into a high speed PCI Express communication card. Notably, DMA method in this paper depends on continuous physical memory, which means it may be worthy of trying scatter/gather DMA, when continuous physical memory is much too precious.

## References

[1] F. Peng, "Integrating PCI Express into PXI and VXI for high performance systems," in *Proc. of IEEE Autotestcon*, Baltimore, MD, USA, Aug. 2007, pp.405-409.

[2] M. Ravindran, "Cabled PCI Express–a standard high-speed instrument interconnect," in *Proc. of IEEE Autotestcon*, Baltimore, MD, USA, Aug. 2007, pp. 410-417.

[3] *PCI Express Base Specification*, 1.0a ed., PCI SIG, 2003.

[4] H. Frock, M. Geruso, and M. Wetzel, "A survey of high speed bus technologies for data movement in ATE systems," in *Proc. of IEEE Autotestcon*, Anaheim, CA, Sep. 2006, pp. 655-657.

[5] L.-G. Zhou, H.-M. Liang, D.-D. Xie, and Z.-K. Wang, "Design and implementation of data transfer card based on PCI Express bus," *Electronic Measurement Technology*, vol. 30, no. 11, pp. 28-32, 2007 (in Chinese).

[6] B. Li, D.-T. Liu, and Y. Peng, "A high speed DMA transaction method for PCI devices," *Journal of Electronic Measurement and Instrument*, vol. 22, pp. 705-716, Aug. 2008 (in Chinese).

[7] *Design Note: Optimizing PEX8311 PCI Express-to-Local Bus DMA Performance*, PLX Technology Inc., Sunnyvale, CA, 2007.

[8] A.-H. Wu, *Windows 2000/XP WDM Devices Driver Development*, Beijing: Publishing House of Electronics Industry, 2005, ch. 5.

[9] B. Li, "Research on high performance computer I/O bus technology," M.S. dissertation, Dept. Instrumentation Science and Technology, Harbin Institute of Technology, Harbin, China, 2008.

[10] Y. Peng, B. Li, D.-T. Liu, and X.-Y. Peng, "A high speed DMA transaction method for PCI express devices," in *Proc. of IEEE Circuits and Systems International Conference on Testing and Diagnosis*, Chengdu, Apr., 2009, pp.1-4.

**Bo Li** was born in Shandong Province, China, in 1984. He received the B.S. and M.S. degrees from the Harbin Institute of Technology (HIT) in 2006 and 2008, respectively. He is currently a visiting scholar with Department of Civil and Structural Engineering and Department of Computing of Hong Kong Polytechnic University. His research interests include high performance computer bus and sensor networks.

**Yu Peng** was born in Shanxi Province, China, in 1973. He received the B.S., M.S., and Ph.D. degrees from HIT, in 1997, 1999, and 2005, respectively. He is currently a professor with the Department of Instrumentation Science and Technology, HIT. His research interests include automatic test technology, data mining, network instrumentation, and wireless sensor network.

**Da-Tong Liu** was born in Heilongjiang Province, China, in 1982. He received the B.S. and M.S. degree from HIT, in 2003 and 2005, respectively. He is currently a Ph.D. candidate with the Department of Instrumentation Science and Technology, HIT. His research interests include automatic test and simulation technology, fault diagnosis and prognosis, and time series analysis.

**Xi-Yuan Peng** was born in Inner Mongolia Province, China, in 1961. He received the B.S., M.S., and Ph.D degrees from HIT, in 1984, 1987, and 1992, respectively. He is currently a professor with the Department of Instrumentation Science and Technology, HIT. His research fields include automatic test theory and technology, advanced fault diagnosis technology and application.