The Design of PCI Bus Interface

Haruyasu Hayasaka
Graduate school of Engineering
Tokai University

Hiroaki Haramiishi
Graduate school of Engineering
Tokai University

Naohiko Shimizu
School of Information Technology and
Electoronics, Tokai University

Kitakaname 1117, Hiratsuka, Kanagawa, 259-1292
Tel: +81-463-58-1221(ex.4084)
Fax: +81-463-58-8320
e-mail: {2aepm061,1aepm041,nshimizu}@keyaki.cc.u-tokai.ac.jp

Abstract— These days, the PCI bus is the standard bus which not only x86 architecture but also other architecture is equipped with. However, the details are difficult to understand. We designed the general-purpose PCI bus interface for developing PCI devices, and implemented it. In this paper, the feature of design is reported.

I. Introduction

A PCI bus interface(PCI I/F) that defined by PCI SIG.(Special Interest Group)[**?**] is access interface. Moreover, as compared with other bus standards, it is the interface which was packed briefly and which is easy to understand. However, considering the position of the developer of peripheral equipment, there are complicated portions which must be careful of besides the portion about the data-transfer which is the fundamental function of a bus. It becomes a burden that a designer treats an original interface like direct PCI from the device of a user design.

Then, we aimed at becoming the software IP for making easy verification and trial production of a device core which used PCI by designed PCI I/F. Moreover, we used SFL, which is the hardware description language of the PARTHENON[**?**] system which NTT developed, for design.

II. Feature of designed PCI I/F

Our PCI I/F was designed as PCI2.1 conformity. A block diagram of designed PCI I/F is shown in Fig. **??**. The feature of designed PCI I/F from the following is described.

A. Target Device

In the PCI bus, a master is called Initiator and a slave is called Target. We designed only the Target this time.

A.1 Burst Transfer Support

The PCI bus is set up so that transfer performance's may become high at the burst transfer which transmits two or more

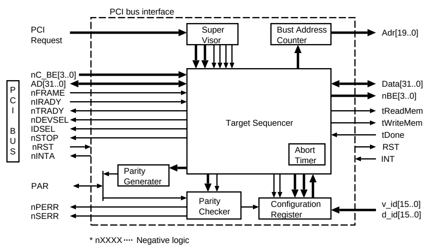


Fig. 1. Block diagram

data continuously. Therefore, it designed so that a burst transfer might be designed.

A.2 Other Functions

- **1MB of 32-bit memory space**
- **High speed adress decode**
- **Parity error derection support**
- **System error response support**
- **One interrupt**
- **Vender ID and Device ID are programmable**

B. Improvement in the Speed and a Miniaturization

As for the clock signal of PCI I/F, the highest frequency is specified as 33MHz. It is not necessary to be 33MHz or more, however to be fully quick for a device core is demanded. Therefore, address decoding was performed on speculation for the improvement of a path which is critical path in designed PCI I/F.

B.1 Improvement of Configuration Registers

The improvement in the speed and the miniaturization of configuration registers used as the bottleneck of the performance (a speed of operation, scale) of PCI I/F were considered.

There are 256 bytes of configuration registers in PCI I/F. Among these, address 0x00-0x3F are called configuration space header, and all PCI I/F must implement it. Address

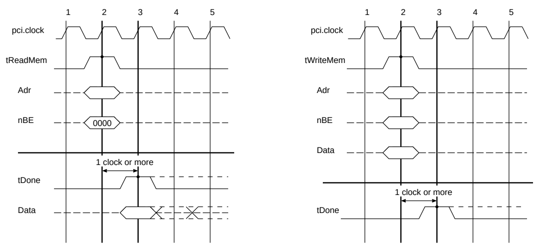
0x40-0xFF are the domain which it is called a device peculiar register domain and an PCI I/F designer can use freely. Then, we decided to implement only in the place which must actually assign a storage cell among this register space in the design of PCI I/F. Consequently, decoding size was able to decrease by miniaturization, and the decoding time was able to lower.

B.2 Improvement of Burst Transfer

In the burst transfer, it accelerated by calculating the following address on speculation.

C. Interface and Data Transfer Cycle

We designed the simple interface for accessing PCI bus. It is shown in Fig.**??** And we specified simple data transfer rule. The example of read/write cycle is shown in Fig. **??**. Memory read(write) cycle is started by ”tReadMem(tWriteMem)”, and ended by ”tDone”.



(a) Read Cycle (b) Write Cycle

Fig. 2. Data Transfer Cycle

III. IMPLEMENT

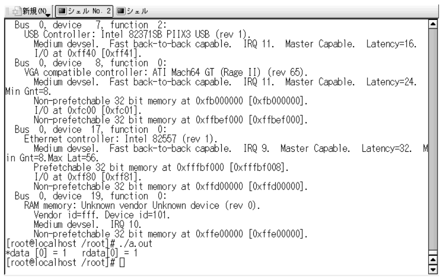
We compiled by MAX+plus II of ALTERA[**?**] and implemented designed PCI I/F in FPGA. we used GPCI-10K30 of GRAPHIN[**?**]. The compiled result assigned to GPCI-10K30 with memory(EAB) and the other device are shown in Table I. Since we did not have other PCI I/F, we were not able to compare with other sample.

TABLE I
COMPILED RESULT OF DESIGNED PCI I/F

Device	Max Frequency[MHz]	Total Logic Cell
EPF10K30AQC240-3[α]	28.90	508
EPF10k30EFC256-1	57.14	497
EPF10K100ARC240-1	41.66	497
EP1K50FC256-1	85.74	510
EP20K100BC356-1V	88.86	506

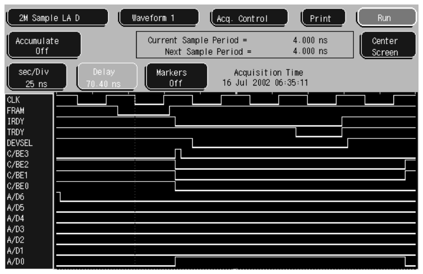
[α]It is assigned to CPCI-10K30

We wrote device driver of Linux for designed PCI I/F and it was run. The result of ”cat /proc/pci”(set VenderID=0x0fff and DeviceID=0x0101) and the result of running program using device driver are shown in Fig.**??**. The information of designed PCI I/F is displayed as ”Bus 0, device 19, function 0:” in Fig.**??**. A program writes “1” to the address 0xffe00000 of a PCI device core, and reads it. The result of a logic analyzer when running that program is shown in Fig.**??**. It is shown “memory write command (C/BE[3..0]=0111)” when runing that program.



```
  Bus  0, device   7, function  2:
    USB Controller: Intel 82371SB PIIX3 USB (rev 1).
      Medium devsel.  Fast back-to-back capable.  IRQ 11.  Master Capable.  Latency=16.
      I/O at 0xff40 [0xff41].
  Bus  0, device   8, function  0:
    VGA compatible controller: ATI Mach64 GT (Rage II) (rev 65).
      Medium devsel.  Fast back-to-back capable.  IRQ 11.  Master Capable.  Latency=24.
Min Gnt=8.
      Non-prefetchable 32 bit memory at 0xfb000000 [0xfb000000].
      I/O at 0xfc00 [0xfc01].
      Non-prefetchable 32 bit memory at 0xffbef000 [0xffbef000].
  Bus  0, device  17, function  0:
    Ethernet controller: Intel 82557 (rev 1).
      Medium devsel.  Fast back-to-back capable.  IRQ 9.  Master Capable.  Latency=32.  M
in Gnt=8.Max Lat=56.
      Prefetchable 32 bit memory at 0xfffbf000 [0xfffbf008].
      I/O at 0xff80 [0xff81].
      Non-prefetchable 32 bit memory at 0xffd00000 [0xffd00000].
  Bus  0, device  19, function  0:
    RAM memory: Unknown vendor Unknown device (rev 0).
      Vendor id=fff. Device id=101.
      Medium devsel.  IRQ 10.
      Non-prefetchable 32 bit memory at 0xffe00000 [0xffe00000].
[root@localhost /root]# ./a.out
*data [0] = 1   rdata[0] = 1
[root@localhost /root]# 
```

Fig. 3. Result of “cat /proc/pci”



```
Accumulate Off
Current Sample Period =   4.000 ns
   Next Sample Period =   4.000 ns
sec/Div 25 ns    Delay 70.40 ns    Markers Off
Acquisition Time
16 Jul 2002 06:35:11
```

Fig. 4. Result of logic analyzer

IV. SUMMARY

We designed PCI I/F as software IP, and implemented it. It was able to have speed sufficient as IP of a PCI I/F. We want to design a PCI device using this.

REFERENCES

[1] PCI SIG., http://www.pcisig.org

[2] PARTHENON, http://www.kecl.ntt.co.jp/parthenon/

[3] “OpenDesine No.7,” CQ Publishing Co. Ltd., 1995.

[4] “TECH I Vol.3,” CQ Publishing Co. Ltd., 2000.

[5] Alessndro Rubini, “ LINUX DEVICE DRIVER,” O’REILLY, 1998.

[6] ALTERA Co., http://www.altera.com/

[7] GRAPHIN CO.,LTD., http://www.g-in.co.jp/