DESIGN AND IMPLEMENTATION OF HIGH-SPEED DATA TRANSFER PROTOCOL IN CORBA ENVIROMENT

June Kim *(nzneer@dsplab.hanyang.ac.kr)
Seungheon Hyeon*(heon@dsplab.hanyang.ac.kr)
Seungwon Choi*(choi@dsplab.hanyang.ac.kr)
*HY-SDR Research Center, Hanyang University, Seoul, Korea

ABSTRACT

The heavy burden of CORBA(Common Object Request Broker Architecture) is one of the most serious hindrances in implementing the SDR(Software Defined Radio) System. As most GIOP(General Inter Object Request Broker Protocol) implementations are based on TCP/IP (Transmission Control Protocol/Internet Protocol) based IIOP (Internet Inter Object Request Broker Protocol), data exchange between CORBA components operating independently should pass through the TCP/IP layer. With such an inefficient operation, the waveform application required in SDR systems can hardly be provided in real-time. This paper presents a novel protocol that enables ORB(Object Request Broker) to exchange CORBA message through PCI bus.

1. INTRODUCTION

One of the major problems in implementing an SCA(Software Communication Architecture)-based SDR system is to have the CORBA ported to the signal processing components such as DSP(Digital Signal Processing) or FPGA(Field Programmable Gate Array). Since the DSP or FPGA is not equipped with an OS(Operating System), CORBA ORB which needs the OS-related services such as scheduling service, file system service, or memory management service cannot operate properly. In this situation, it is necessary for the ORB-ported GPP to employ an adapter for converting the CORBA message into a proper user-defined message in order to exchange data. However, the adapter causes a bottle neck and overload in GPP when the CORBA message is to be exchanged with any of non-CORBA elements because the message should pass through the GPP at every transfer as shown in Figure 1. Two procedures should be provided to solve this problem. Firstly, the ORB should be able to exchanges the CORBA message through the PCI bus. Secondly, the ORB should be ported to DSP or FPGA. In this paper, we mainly focus on the first procedure.
This paper consists of four sections as follows. In first section, we provide an overview of GIOP structure. In second section, the PCI bus protocol is explained. In third section, PCIIOP (PCI Inter Object Request Broker Protocol), a novel High-speed data transfer protocol, is proposed. The proposed protocol accelerates the data exchanging between CORBA components and guarantees an autonomy in SDR system design. In forth section, the protocol is implemented with TAO. TAO is the most famous open source CORBA implementation.

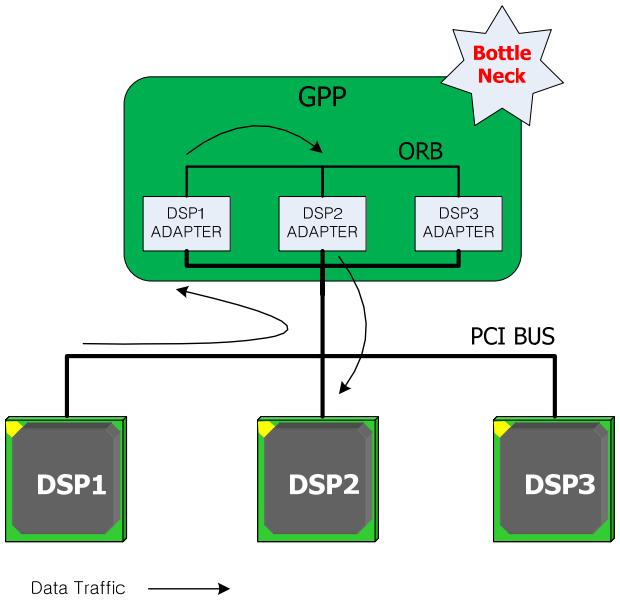

Figure 1- Data Exchange between DSP1 and DSP2

2. OVERVIEW OF GIOP

GIOP is a protocol for exchanging CORBA messages between ORB(Object Request Broker)'s as shown in Figure 2. After the client acquires a reference to CORBA object and accesses a method of the object, GIOP sends a CORBA message to ORB at which the object is located. The ORB, which receives the message via GIOP, forwards it to the object adapter, then, the target object executes the method. Result of the method returns in the other way around.

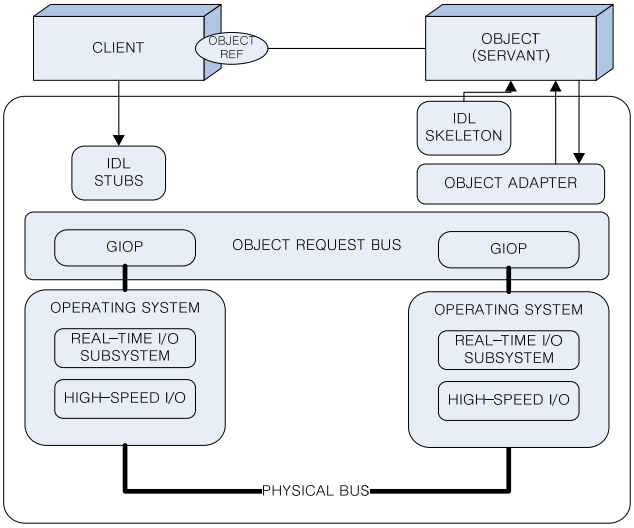

Figure 2- CORBA structure

When the CORBA message is transferred via GIOP, we need an I/O subsystem for physically transferring the data. Ethernet is the most well known I/O subsystem.
IIOP(Internet Inter Orb Protocol) is a protocol for transferring the CORBA messages using TCP/IP over

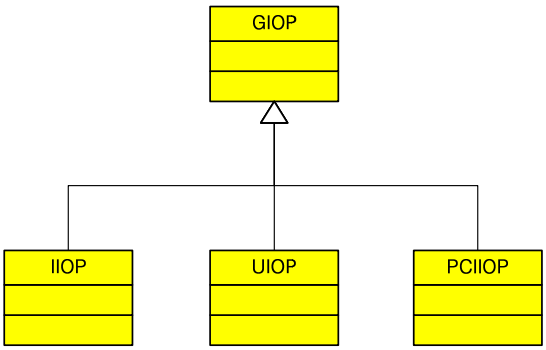

Figure 3- UML Diagram of GIOP

Ethernet, which means that IIOP is implemented in most CORBA implementations.
All CORBA message transfer protocols should inherit the interfaces of GIOP. For example, IIOP (Internet Inter ORB Protocol) and UIOP (Unix Inter ORB Protocol), PCIIOP (PCI Inter ORB Protocol) inherits all interfaces of GIOP and implements the interfaces according to their I/O subsystem. Figure 3 describes this relationship with UML(Unified Modeling Language). GIOP sends a header and body message for one request. The definition of the message header is described using IDL(Interface Definition Language) in Figure 4. The message header has a message body and message type. Reference to target object is acquired through IOR (Interoperable Object Reference) in

```
Module GIOP{
    struct Version{
        octet major;
        octet minor;
    };
    Enum MsgType_1_1{
        Request, Reply, CancelRequest, LocateRequest,
        LocateReply, CloseConnection, MessageError, Fragment
    };
    struct  MessageHeader_1_1{
        char            magic[4];
        Version         GIOP_version;
        octet           flags;
        octet           message_type;
        unsigned long   message_size;
    //…
};
```

Figure 4-Definition of GIOP Message Header

CORBA. GIOP should be equipped with IOR parser for acquiring a unique address of the target object in I/O subsystem.

3. PCI

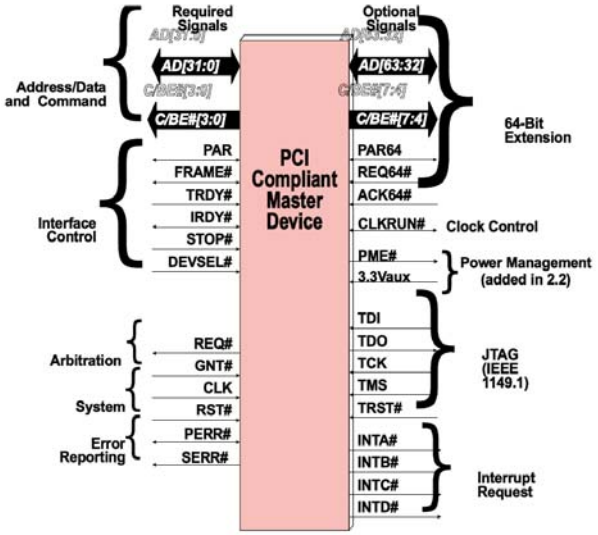

Figure 5- Description of PCI Signal

PCI is a very well known system bus in communication system. As the 32bit PCI bus that consists of 32bit data and address bus has been widely used, we only consider the 32bit PCI bus in this paper. PCI bus should include a host device which, as a bus arbitrator, assigns each device a PCI address. At least, one host device exists on PCI bus. Each PCI device is able to send an interrupt signal to the host device. INTA, INTB, INTC, INTD signals are this purpose pin in Figure 5 but there are no means sending interrupt signal form PCI device to PCI device in PCI specification.

Most PCI devices have ATU(Address Translation Unit). This unit translates automatically PCI address and internal address in order to make an internal address space for the external PCI device access. In order to exchange data over PCI, internal receiving buffer should be mapped to PCI bus

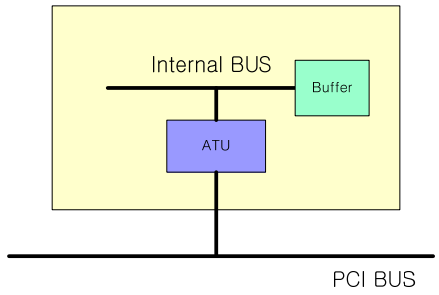

Figure 6- Receiving Buffer

using ATU like Figure 6. If the system registers, generating the internal interrupt, are mapped to the PCI address, the external PCI device can generate an interrupt signal.

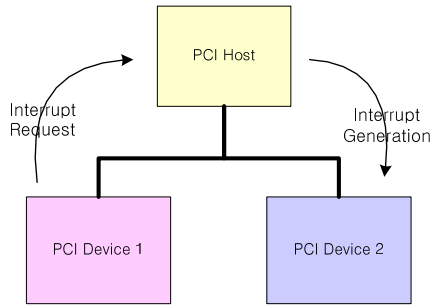

Figure 7- Interrupt sending

4. DESIGN OF PCIIOP

In order to send and receive GIOP messages over PCI, a unique address of the CORBA object existing on PCI bus should first be encoded and decoded into IOR. For example, IIOP encodes and decodes IP address and port number such as 192.168.1.1:8080 into IOR. PCIIOP defines an address format as [PCI address : length]. PCI address denotes the PCI bus address residing the message receiving buffer. Length is the address range of the buffer. The buffer should reside the PCI bus using ATU(Address Translation Unit) mapping. GIOP message sender decodes the IOR and determines the PCI address of the receiving buffer for the target object.

Receiver of the GIOP message should receive an interrupt signal since the message has been transmitted. But sending interrupt signal to the PCI device does not exist as discussed previously. Solution to this problem is to register an

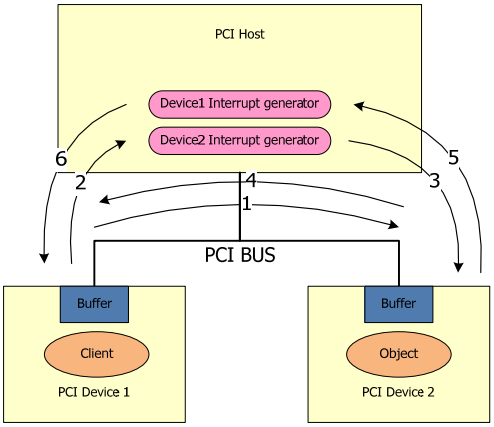

Figure 8- PCIIOP message exchanging sequence

interrupt generator to host device for each PCI device. Host device generates an interrupt signal from each interrupt generator when the PCI device requests the host interrupt generation of the specified PCI device. Figure 7 describes this solution. If PCI device is equipped with MSI(Message Signaled Interrupt), data transfer performance may be enhanced to some extent because it sends the interrupt signal to other PCI device directly. All these procedures are described in Figure 8.

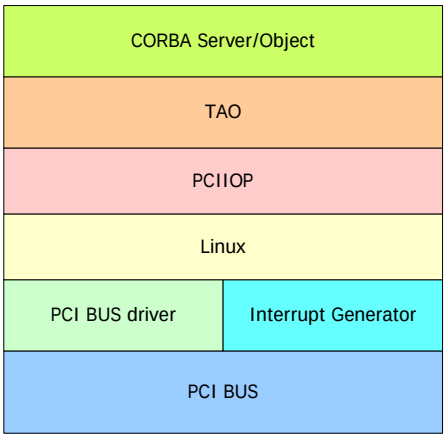

Figure 9-Functional Layers of PCIIOP based CORBA

5. IMPLEMENTATIO OF PCIIOP

PCIIOP is implemented on Linux using TAO. TAO supports the framework for GIOP implementation and rapid development of a new transfer protocol.

CORBA application needs Linux driver in order to access the PCI bus. The driver was written in C language and is similar to /dev/kmem driver. When the application reads and writes the address of the driver, PCI address is read and written on the PCI bus.

Interrupt generation driver was written in PCI host machine. When interrupt signal is detected from each PCI device, the driver invoke the interrupt handler, which acquires a device number of the target device and executes the interrupt generator.

6. CONCLUSIONS

PCIIOP was designed and implemented so that all CORBA-based PE(Processing Element) can exchange GIOP messages with one another. This protocol accelerates data exchange between CORBA components. Later, PCIIOP will be implemented on ORB in C language and All PE such as DSP, FPGA will be able to exchange GIOP message with one another.

7. REFERENCES

[1] Carlos O'Ryan, Fred Kuhns, Douglas C. Schmidt, Ossama Othman, and Jeff Parsons, "The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing Middleware" *IFIP/ACM Middleware 2000 Conference,*April 3-7,2000.

[2] Michi Henning, Steve Vinoski, *Advanced CORBA Programming with C++*, Addison-wesley, USA, 1999.

[3] Tom Shanley, Don Anderson, *APCI System Architecture*, Addison-wesley, USA, 1999.

[4] Remedy IT, *TAO Programmer Guide*

[5] TAO home, http://www.cse.wustl.edu/~schmidt/TAO.html