FPGA DESIGN TO SUPPORT A CORBA COMPONENT

Changhoon Lee (HY-SDR Research Center, Hanyang Univ., Seoul, Korea; daljoon@dsplab.hanyang.ac.kr) ; June Kim (HY-SDR Research Center, Hanyang Univ., Seoul, Korea; nzneer@dsplab.hanyang.ac.kr) , Seungheon Hyeon HY-SDR Research Center, Hanyang Univ., Seoul, Korea; hsheon@dsplab.hanyang.ac.kr); and Seungwon Choi(HY-SDR Research Center, Hanyang Univ., Seoul, Korea; choi@dsplab.hanyang.ac.kr)

ABSTRACT

The Common Object Request Broker Architecture (CORBA) standard that supports the Field Programmable Gate Array (FPGA) is not generally used because it is difficult to implement and to develop using Hardware Description Language (HDL). In this paper, we propose a way to design an FPGA to support a CORBA. For FPGA to support the CORBA, an embedded microprocessor implemented in FPGA, a peripheral component interconnect (PCI)-based CORBA software, is used. The PCI-based CORBA improves data transfer throughput.

1. INTRODUCTION

CORBA is a standard structure to create, distribute, and manage distributed processing objects in a network. CORBA enables programs which are in different places in a network and are developed by various venders to communicate through an interface broker. CORBA is an essential part of the Software Defined Radio (SDR) system that supports reusability, portability, and interoperability of the waveform application.

To implement a CORBA middleware, CORBA and its operating system (OS) are required. Systems like Digital Signal Processor (DSP) and General Purpose Processor (GPP) have their own operating systems. A few modifications and additions of source code enable the microprocessors to implement CORBA

FPGA, however, has no embedded operating system. For this reason, it has been difficult to implement and to use CORBA middleware. In this paper, to implement CORBA middleware in FPGA, a microprocessor is implemented by using the open IP core on FPGA and PCI-based CORBA middleware is loaded on the microprocessor. The traditional way to communicate with other system in CORBA is via Ethernet but it requires many software stacks and doesn't guarantee real-time requirement of wireless communication system. Because we can expect high performance throughput and real-time response by using the PCI bus, FPGA uses a PCI bus to communicate with other system and PCI-based CORBA were favored in our system. Also, all IP cores are designed using open source and open core which allows the reduction of development costs.

This paper is organized as follows. In Section 1, existing research trends and background are presented for the proposed FPGA design to support the CORBA component. In Section 2, an FPGA design for supporting CORBA components is proposed and described in detail. In Section 3, a simple experimental test is performed to confirm the proposed FPGA design.

2. FPGA DESIGN FOR CORBA

This section explains the FPGA design to support a CORBA component with a detailed description of each part. Figure 1 is a diagram of the FPGA design. To use CORBA components, we implement a reduced instruction set (RISC) microprocessor on FPGA using the CPU IP core. We also use the memory interface core to control external FLASH and SDRAM. The PCI IP core is used to communicate with other processors via the PCI bus. The signal processing component is the component which performs signal processing corresponding with the CORBA component. The signal processing component and CPU exchange the data and control signal with each other via the memory interface. The CORBA component is loaded into the CPU to control the signal processing component. In other words, the CORBA component is the proxy component of the signal processing component. Finally UART, which is supported by the UART IP core, is used for monitoring the CPU.

2.1. Wishbone protocol

The wishbone protocol is used to communicate between cores. The wishbone system-on-chip (SOC) interconnection architecture for portable IP Cores is a flexible design methodology for use with various IP cores. In the past, protocols between IP cores have been problematic due to the absence of a standard and poor reliability.

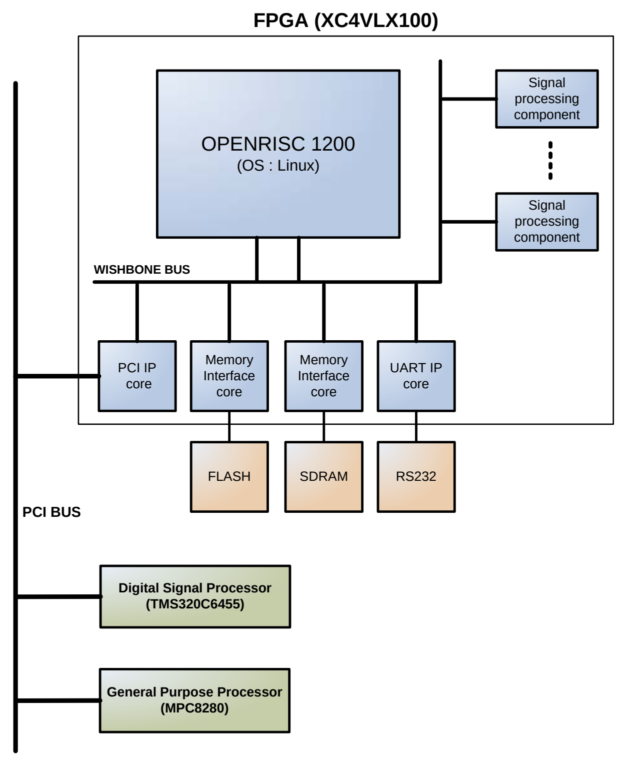

Figure 1. Block diagram of the FPGA design for CORBA.

By using the wishbone protocol, we can enforce compatibility between IP cores and design reusability. The wishbone protocol makes it easy to consider the needs of both a core developer and an end user.

2.2. Wishbone BUS

IP cores that communicate using the wishbone protocol are connected to the wishbone bus [3] and are divided into two parts: wishbone master and wishbone slave. Once a wishbone master gets a grant signal by a wishbone arbiter, the wishbone master takes possession of the wishbone bus. There are two methods of wishbone interconnect: shared bus interconnect and crossbar switch interconnect. In this paper, the shared bus interconnect is used for the wishbone bus because the compact design structure of the shared bus interconnect requires fewer logic gates and routing resources compared to a crossbar switch interconnect. In a shared bus interconnect, each master has the same priority to take possession of the wishbone bus in turn. Figure 2 shows the structure of the wishbone bus using the shared bus interconnect. The wishbone bus assigns an address to each wishbone slave. When a wishbone master controls a wishbone slave, the wishbone master has to access the assigned address. There are two wishbone masters that originate from the CPU IP core: the instruction wishbone master and the data wishbone master. There are six wishbone slaves controlled by wishbone masters.

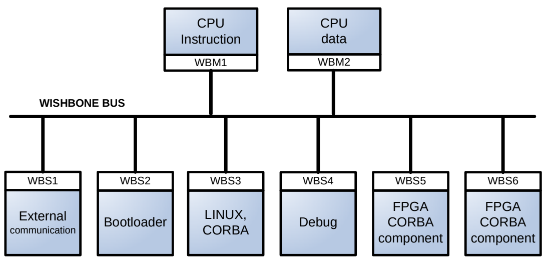

Figure 2. The structure of the wishbone bus.

2.3. PCI IP core

The PCI IP core [4] is used to interface between FPGA and other system like DSP and GPP. PCI is used because it has been developed to support a fast 32-bit or 64-bit standard interface. Furthermore, it is useful for scalability because the PCI bus is used for multiplex structure between the address bus and the data bus [1],[2]. To implement CORBA, TCP/IP communication was used traditionally. Not only Ethernet communication but also PCI communication, RapidIO communication, and etc. can be used to implementation. In the paper, we chose PCI communication for following reasons. In an embed environment, using PCI communication is more general than using Ethernet communication. The maximum speed of the PCI communication is 133MB/s but the maximum speed of Ethernet communication is 100Mb/s. The throughput of PCI communication is better

2.4. CPU IP core

The embedded CPU IP core [5] is an essential requirement of CORBA for FPGA. Once the CPU IP core is embedded in FPGA, the FPGA acts like a microprocessor which is able to load an operating system such as Linux, ucLinux. This allows FPGA to execute the CORBA middleware. In this paper, openrisc 1200 is used as the CPU IP core. Openrisc1200 is a CPU using the Harvard architecture which physically separates storage and signal pathways for instructions and data therefore the instruction wishbone bus and data wishbone bus send the desired commands to components connected to the wishbone slave bus. The components can be used by wishbone masters. Openrisc1200 is a type of RISC that uses complex software processing to enhance computer speed so hardware loads can be reduced. In this paper Linux is adopted as an operating system and CORBA components are embedded onto Linux. The CORBA components in this paper are FFT and IFFT. Detailed applications are explained in the next section.

2.5. Memory interface

Memory interface is one of the wishbone slaves. It controls and uses SDRAM and FLASH through the wishbone protocol. Two memory interface cores are used and connected to SDRAM and FLASH. SDRAM and FLASH run Linux in the CPU. To run the Linux environment, the FLASH saves a binary file to use as a boot loader. The boot loader enables the CPU after the system is powered up. The FLASH also saves compressed Linux and CORBA components as binary image. Because FLASH has little storage and is slow, low-complexity Linux is the optimal operating system for implementing CORBA. In this paper we use 16MB of FLASH capacity. After the CPU is driven by FLASH, compressed Linux and CORBA components are decompressed by the CPU and the decompressed data is then saved into SDRAM. As a result, the CPU is driven by the FPGA.

2.6. UART IP core

The UART IP core [6] is the one of the wishbone slaves that enables communication with an external monitoring program through a serial port. Through the UART IP core, the RS-232 transceiver is connected to a serial port. The UART interface is not involved in implementing CORBA for FPGA directly. However, UART IP provides the internal CPU status with external program such as HyperTerminal™. To implement CORBA for FPGA, we can monitor necessary information through the UART IP core. By using this information, we can debug the FPGA design. Since debugging is essential to implementing an FPGA Design, a UART IP core is recommended.

2.7. Other components for CORBA

Previous subsections discussed essential parts required to implement FPGA design for CORBA. This subsection details other components that are needed for CORBA. This paper adds FFT and IFFT components as wishbone slaves. To connect FFT/IFFT IP core with CPU, an memory interface core is added. One side port of the memory interface connects to the wishbone bus and the other port connects to the FFT/IFFT core. Figure 3 shows the structure of the FFT/IFFT IP core. To enable the core to be controlled by CORBA component, CORBA component should be loaded in the CPU. As stated above, CORBA component is compressed and saved to FLASH in binary. The developer can design IP cores like GPIO, CTC, etc. according to the wishbone standard. Then the developer should add IP core to wishbone slaves and write the proxy CORBA component using preferred language like C or C++ to connect the IP core with CORBA software bus.

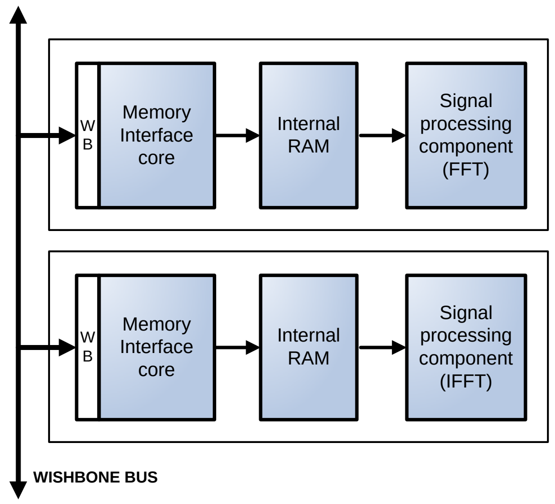

Figure 3. Other components (FFT/IFFT) for CORBA.

3. EXPERIMENTAL TEST

In this section, based on the detailed block in section 2, we experimentally evaluate the performance of our FPGA design. We know how data is transferred to FPGA and how the process is performed. Figure 4 shows the board for FPGA design. The board was produced to develop and implement an SDR system. An experimental test was performed as follows. GPP transfers data to FPGA and FPGA perform FFT to the data. FPGA uses IFFT for the data FFT sent. Finally the data processed by IFFT is sent to GPP. We compare the data returned with the original data.

In Figure 1, FPGA is connected to other processors through the PCI bus. In this paper, FPGA is connected to GPP(MPC8280) through the PCI bus. We confirm procedures for FFT and IFFT. We compress Linux, FFT CORBA, and IFFT CORBA component and then the compressed data is saved in FLASH with boot loader data. Compressed data is uncompressed by the CPU and SDRAM saves uncompressed image of Linux, FFT CORBA component, and IFFT CORBA component i.e. FPGA becomes a microprocessor and FFT and IFFT CORBA components are embedded in FPGA. MPC8280 sends FPGA a General Inter-ORB Protocol (GIOP) message to perform the FFT process.

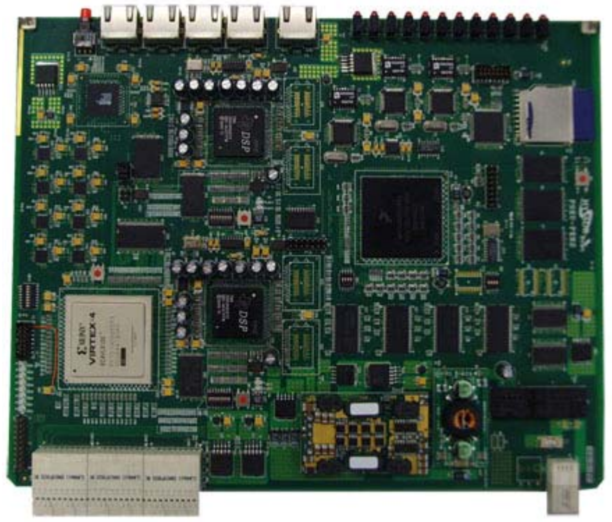

Figure 4. Board to implement FPGA design for CORBA.

FPGA receives the GIOP message from the PCI bus and then sends it to the CPU through the PCI IP core and wishbone bus. CPU decodes the GIOP message. FFT CORBA component transfers the data from the GIOP message to the FFT IP core and makes the FFT IP core perform FFT on the data. The CPU receives the performed FFT data and the data is sent to IFFT CORBA component. The IFFT IP core performs IFFT on the data. The CPU receives the IFFT data and encodes a GIOP message by using the IFFT data. The GIOP message is transferred to MPC8280 through the wishbone bus, PCI IP core, and PCI bus. MPC8280 decodes the GIOP message and compares the data from the GIOP message with the original data.

4. CONCLUSION

In this paper, we proposed an FPGA design for CORBA. We chose a PCI interface for fast communication between processors. The CPU was built in FPGA so that we could execute another operating system with the desired CORBA middleware. All sources of IP core we used are open source and therefore don’t require licenses. This allows developers to reduce development costs. If we embed CORBA in FPGA and connect a related signal processing IP core using a wishbone bus, we can control and use the IP core anywhere in the system. In conclusion, using the proposed FPGA design, we have provided a solution to implement an SDR system.

5. REFERENCES

[1] “PCI system Architecture” Tom Shanley and Addison – Wesley , June 1999
[2] “explanation of PCI BUS and design interface card” International Technology Information Institute
[3] “specification for the WISHBONE System-On-Chip(SOC) Interconnection Architecture for Portable IP Cores Revision: B.1” Silicore Corporation January 8 , 2001
[4] “PCI IP core specification” Miha Dolenc and Tadej Markovic , July 16 , 2004
[5] “Openrisc 1200 IP core Specification” Damjan Lampret , September 6 , 2006
[6] “UART IP core specification” Jacob Gorban , August 11 , 2002