CORBA 컴포넌트를 지원하는 FPGA 설계

FPGA design for CORBA component

이 창 훈, 김 준, 현 승 헌, 정 재 호, 최 승 원*
(Changhoon Lee, Jun Kim, Seungheon Hyoen, Jaeho Chung, Seungwon Choi)

Abstract : The CORBA that supports FPGA has not been used generally and it is difficult to implement and to develop the CORBA for FPGA. In this paper we propose the way to design FPGA to support a CORBA component. For FPGA to support the CORBA component, embedded processor provided by FPGA and PCI based CORBA is utilized. The PCI based CORBA is for improving data transfer throughput. This paper will be organized as follows. In Chapter I, existing research trend and background are presented for why we propose design of FPGA that support the CORBA component. In Chapter II, FPGA design for supporting CORBA components is proposed and described in detail. In Chapter III, simple experiment is tested to confirm the proposed FPGA design. Finally session 4 is conclusion of this paper.

Keywords: FPGA, CORBA, PCI

I. 서론

CORBA (Common Object Request Broker Architecture)는 ORB(Object Request Broker) 부분에 대한 분산 객체지향 구조의 표준을 정의한 것으로써 분산객체간의 상호 운용을 위한 통신 미들웨어 역할을 하는 버스 구조로 정의할 수 있다. CORBA는 분산객체 소프트웨어의 기본 틀로써 분산 환경에서 응용 소프트웨어를 쉽게 개발할 수 있도록 지원하고 있으며, 미들웨어의 역할로써 서비스를 제공하는 부분과 제공받는 부분간에 투명한 정보 교환이 가능하도록 지원한다. CORBA는 재사용성, 상호운용성, 휴대성을 목적으로 하는 SDR system을 구현하기 위한 필수요소이다. CORBA를 구현하기 위해서는 운영체제가 필요하다. DSP나 GPP같은 processor는 자체적으로 제공하는 운영체제가 있어 약간의 소스 변경과 추가 만으로 CORBA를 구현할 수 있다. 그러나 FPGA의 경우는 내부에 탑재된 운영체제가 없다. 다른 시스템이나 소자입장에서는 FPGA는 메모리로서의 역할을 할 뿐, FPGA 자체는 주도적인 역할을 하지 못하기 때문에 다른 컨트롤러에 의해서 제어를 받아왔다. 이러한 이유로 현재까지 FPGA에서는 CORBA를 구현하고 사용하는데 어려움이 있었다. 본 논문에서는 FPGA에서 CORBA를 사용하기 위하여 FPGA내부에 CPU IP core를 사용하여 CPU를 탑재 하였다. 또한 DSP와 GPP등의 다른 프로세서와 통신을 하기 위하여 PCI 프로토콜로 통신하였다. 기존에는 다른 프로세서와 통신을 하기 위해 Ethernet을 사용하였으나 PCI 프로토콜을 사용하면 기존의 Ethernet을 사용하는 것 보다 좋은 성능의 throughput을 기대할 수 있다. FPGA에 CPU를 설계하기 위해 사용되는 모든 core는 공개용 소스를 사용하므로 비용을 절감할 수 있다.

* 책임저자(Corresponding Author)
논문접수 : 20xx. x. x., 채택확정 : 200x. x. xx.
최승원 : 한양대학교 전자통신컴퓨터 대학원
(choi@dsplab.hanyang..ac.kr,)
※ 본 연구는 대학 IT 연구센터 육성지원사업의 연구결과로써 HY-SDR연구센터의 연구비 지원으로 수행되었음..

II. CORBA를 구현하기 위한 FPGA 설계

이 장에서는 CORBA 컴포넌트를 지원하는 FPGA를 어떻게 구성할 것인가에 대해 알아보고 그 세부 파트에 대한 설명을 한다. Figure 1은 CORBA 컴포넌트를 지원하기 위한 FPGA 디자인의 전체 다이어그램이다. CORBA 컴포넌트를 이용하기 위해 CPU IP core를 이용하여 FPGA내부에 RISC구조의 CPU를 구현하였다. Memory interface core는 CPU와 외부 FLASH, SDRAM과의 연결을 위해 사용한다. PCI IP core는 PCI를 이용한 외부 프로세서와의 통신을 위해 사용된다. Signal processing 컴포넌트는 CORBA 컴포넌트에 대응하여 실제 signal processing을 처리하는 컴포넌트이다. Signal processing 컴포넌트와 CPU는 memory interface를 이용하여 연동되며, CPU에 load된 CORBA 컴포넌트가 signal processing 컴포넌트의 제어를 담당하게 된다. 마지막으로 UART IP core는 CPU를 디버깅 할 목적으로 사용된다

a. Wishbone protocol

wishbone protocol은 core간의 통신을 하기 위한 프로토콜이다. 예전에 IP core간의 통신은 규약이 정해지지 않았으므로 IP core간의 통신을 할 때 어려운 점이 많았고 신뢰성이 보장되지 않았지만 WISHBONE 프로토콜을 사용함으로써 IP core간의 호환성을 보장해 주고 core 개발자들 사이에 재사용성을 강조하였다. 또한 이는 코어 개발자와 최종 사용자 사이에 소통을 쉽게 할 수 있게 한다. 따라서 본 논문에서는 core간의 통신 수단으로 wishbone 프로토콜을 사용한다.

b. Wishbone 버스

Wishbone 프로토콜로 통신을 하는 IP core들은 모두 Wishbone 버스에 [3] 연결되어 있으며 마스터와 슬레이브로 나누어져 있다. 버스의 점유권을 할당하는 wishbone 아비터에 의해 마스터에 버스 소유권이 주어지면 그 마스터는 wishbone 버스를 점유할 수 있게 된다.

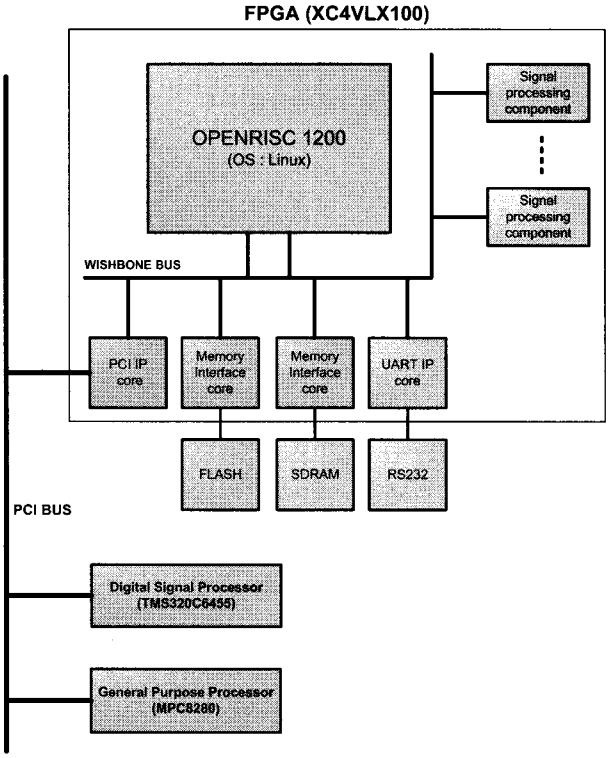


그림 1. CPRBA 컴포넌트 구현을 위한 FPGA 설계 블록도

Wishbone 버스의 연결 방식은 크게 두 가지로 나누어 지는데 shared bus방식과 crossbar switch방식이다. 논문에서는 shared bus 방식을 사용하였다. 이는 간단한 구조를 지향하기 위함이이고 crossbar switch방식 보다 적은 로직게이트와 라우팅 자원을 필요로 하기 때문이다. Shared bus 방식은 각각의 마스터가 동일한 우선순위를 가지고 버스를 차례대로 점유하는 방식이다. 논문에서 사용한 wishbone 버스의 구조는 그림2와 같다. Wishbone 슬레이브는 wishbone 버스에서 각각 사용할 수 있는 주소가 할당된다. Wishbone 버스가 wishbone 슬레이브를 제어하거나 사용하기 위해서는 해당 주소를 접근하여야 한다. Wishbone 마스터는 2개이고 모두 CPU IP core에서 나온다. Wishbone 슬레이브는 6개 이며 wishbone 마스터에 의해 제어받는다.

c. PCI IP core

DSP, GPP와 같은 외부 프로세서와 연동하기 위하여 PCI IP core를 [4] 사용하여 PCI 통신을 하였다. 이는 널리 사용되는 방식으로 고속성을 위해 만들어진 32bit 규격의 프로토콜이다. 또한 어드레스 버스와 데이터 버스가 멀티플렉스로 활용될 수 있기 때문에 확장성에 용의 하다. 전통적으로 CORBA를 구현하는데 TCP/IP기반의 통신을 하였다. CORBA 미들웨어를 구현하기 위해서는 Ethernet뿐 아니라 PCI, RapidIO등의 방식으로 통신을 할 수 있다. 논문에서 PCI 기반의 통신을 사용한 이유는 첫째로 일반적으로 임베디드 환경에서 내부 버스로 이용하는 것은 Ethernet보다 PCI 기반이다. 둘째로, Ethernet 기반의 통신은 최대속도가 100Mb/s이지만 PCI기반의 통신은 최대속도가 133MB/s로 속도가 훨씬 빠르며 throughput이 더 좋다.

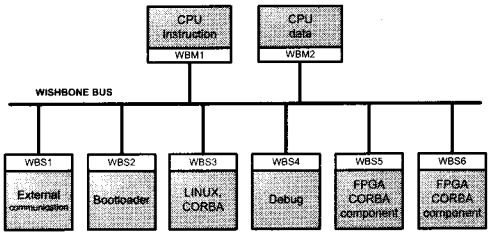


그림 2. Wishbone 버스 구조

d. CPU IP core

CPU IP core의 [5] 탑재는 FPGA에 CORBA를 사용하기 위해 가장 핵심적인 부분이다. CPU IP core 탑재로 FPGA는 외부 프로세서에서 보았을 때 더 이상 메모리가 아니고 하나의 OS를 가진 컴퓨터의 역할을 하게 된다. FPGA에 CPU가 탑재되면 FPGA는 프로세서로서의 역할을 하며 주도적으로 다른 프로세서에 명령을 지시할 수 있게 된다. 논문에서 사용한 CPU IP core는 openrisc1200 이다. Openrisc12000은 Harvard architecture방식을 사용한 CPU이다. 또한 RISC (reduced instruction set computer) 방식의 CPU를 채택하였다. RISC는 컴퓨터의 실행속도를 높이기 위해 복잡한 처리는 소프트웨어에게 맡기는 방법을 채택하여, 명령세트를 축소 설계한 컴퓨터이며 하드웨어의 부하를 줄일 수 있다. 일단 CPU가 탑재된 후 CPU IP core의 Harvard architecture에 입각하여 instruction wishbone bus와 data wishbone 버스를 통해 Wishbone 슬레이브에 연결된 컴포넌트에 명령을 지시하고 각 컴포넌트에 해당하는 기능을 사용할 수 있게 한다. 논문에서 OS는 Linux를 선택하였고, Linux를 기반으로 CORBA 컴포넌트를 탑재 하였다. 사용하는 CORBA 컴포넌트는 FFT CORBA 컴포넌트 와 IFFT CORBA 컴포넌트이다. 자세한 응용은 다음 챕터에서 설명할 것이다.

e. Memory interface

Memory interface는 wishbone 슬레이브 중 하나로써, wishbone 프로토콜을 이용하여 SDRAM과 FLASH을 제어하게 해주는 역할을 한다. Memory interface core는 2개를 사용하며 각각 SDRAM과 FLASH에 연결된다. SDRAM과 FLASH은 CPU에서 Linux를 구동 시키는 역할을 한다. Linux를 구동하기 위해 FLASH에 Bootloader 바이너리 파일을 저장한다. Bootloader는 전원을 인가한 후 CPU가 작동할 수 있도록 해주는 기능을 한다. 또한 Bootloader의 내용이 끝난 다음주소부터는 Linux와 사용하려는 CORBA 컴포넌트의 내용이 바이너리로 압축된 형태로 저장된다. 이는 FLASH의 용량이 적고 느리기 때문에 Linux를 OS로서 사용하고 CORBA를 구현하기 위한 최소의 내용만이 저장하기 위함이다. 여기서 사용된 FLASH의 용량은 100KByte 정도이다. FLASH에 의해서 CPU가 구동되면 Linux와 CORBA 컴포넌트의 압축된 내용이 CPU에 의해 해제되며 이 내용이 SDRAM에 저장된다. 이로서 FPGA에서 CPU가 구동되고 Linux를 OS로 사용할 수 있는 기반이 마련되며, 사용하려는 CORBA 컴포넌트가 Linux위에 구현될 수 있다.

f. UART IP core

UART IP core는 [6] wishbone 슬레이브 중 하나로써, 외부와 씨리얼 통신을 가능하게 해준다. UART IP core를 통하여 외부 UART 소자인 RS232 트랜시버와 연결한다. UART 통신은 FPGA에서 CORBA를 구현하는데 직접적인 관련은 없지만 CPU내부 환경이나 필요한 정보를 얻을 수 있는 기능을 하게 해준다. 결국 FPGA에서 CORBA를 구현하는데 있어서 필요한 정보를 모니터링 하고, 그 정보를 이용해 디버깅 할 수 있게 해준다. 디버깅은 FPGA를 설계하는데 필수적인 요소이므로, UART 체계가 아니더라도 외부에서 FPGA를 모니터 할 수 있는 디버깅 컴포넌트를 첨가하는 것을 추천한다.

g. CORBA구현을 위한 컴포넌트

앞에서 설명한 항목이 CORBA를 지원하는 FPGA 디자인의 필수 요소라고 한다면, 나머지는 개발자가 필요한 core나 컴포넌트로 구현하도록 디자인 하여야 한다. 본 논문에서는 FFT와 IFFT 컴포넌트를 wishbone 슬레이브로 추가 하였다. FFT 또는 IFFT 컴포넌트를 추가하기 위하여 2.e에서 제시한 memory interface를 추가하여 한쪽 port는 wishbone 프로토콜을 연결하고 다른 포트에는 FPGA내의 FFT/IFFT를 수행하기 위한 메모리를 연결하여 FFT/IFFT core를 연결한다. FFT/IFFT 컴포넌트 구조는 그림3에 나타난다. 한가지 유의할 점은 추가할 core나 컴포넌트는 wishbone interface가 가능하도록 설계하여야 한다는 것이다. 또한 FFT와 IFFT를 가능하게 하기 위해 Linux에 FFT와 IFFT CORBA 컴포넌트가 탑재되어 있어야 한다. 이는 2.e에서 설명하였듯이 FLASH에 각 CORBA 컴포넌트에 해당하는 내용을 압축하여 바이너리로 저장함으로써 구현할 수 있다. 설계자에 따라서 GPIO나 CTC 등 원하는 core나 컴포넌트를 wishbone standard에 맞추어 설계하고 wishbone slave에 추가한다. 그리고 이에 대응하는 CORBA 컴포넌트를 FLASH에 압축하여 바이너리형태로 저장하여 설계자가 FPGA에서 사용하려는 기능을 수행 할 수 있다.

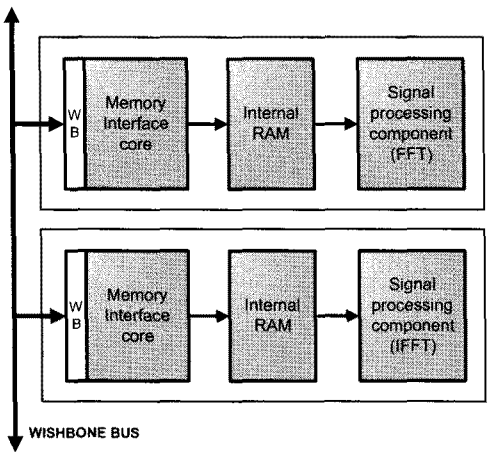


그림 3. CORBA 구현을 위한 컴포넌트

III. 검증

3장에서는 2장에서 설명했던 세부내용을 토대로 간단한 실험을 통해 데이터가 어떻게 FPGA로 입력되고 출력되는지 알아보고, 어떻게 프로세스가 수행되는지 과정을 알아 본다. 다음 그림4는 실험을 위해 FPGA 설계를 구현한 보드이다. 이 보드는 SDR system을 개발하고 구현하기 위해 제작되었다. 논문에서 수행하였던 실험은 GPP에서 데이터를 FPGA로 보내고 이 data를 FFT하고 다시 IFFT한 후 그 데이터를 GPP로 받아들여 원래 데이터와 일치하는지 알아보는 것이다.

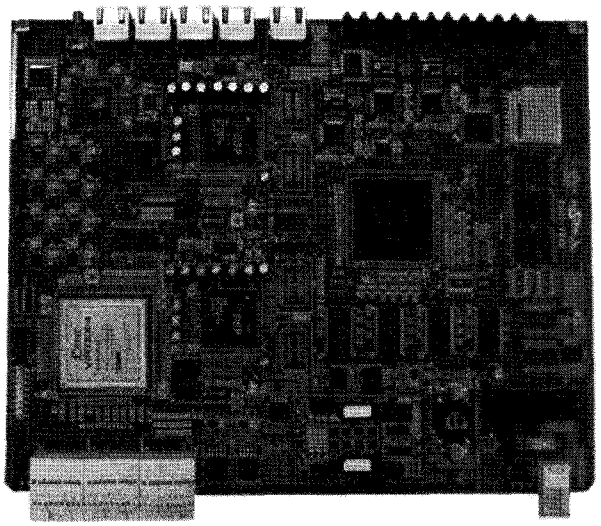


그림 4. CORBA 컴포넌트 구현을 위한 보드

그림1의 구조에 따라 FPGA 와 다른 프로세서는 PCI 버스를 통하여 연결되어 있다. 연결되는 프로세서는 개발자의 요구에 따라 DSP가 될 수도 있고 GPP가 될 수도 있으며, CORBA가 탑재된 또 다른 FPGA가 될 수도 있다. 논문에서 GPP인 MPC8280을 PCI를 통하여 FPGA와 연결한다. MPC8280을 통하여 FPGA에서 FFT와 IFFT를 수행하는 과정을 알아 본다. FPGA는 이미 CPU를 통해 FLASH에 저장된 Linux, FFT CORBA 컴포넌트, IFFT CORBA 컴포넌트를 압축한 바이너리 파일을 RAM에 압축이 풀린 형태로 저장하며, 이는 FPGA가 Linux기반의 프로세서가 되었음을 의미한다. 또한 FFT와 IFFT CORBA 컴포넌트가 Linux위에 탑재 되었음을 의미한다. MPC8280에서 FPGA로 FFT를 수행하기 위한 GIOP 메시지를 보내면, FPGA는 PCI BUS를 통해 GIOP 메시지를 받고 PCI IP core와 wishbone 버스를 통해 CPU에 GIOP 메시지를 전송한다. CPU에서는 GIOP 메시지를 해석하고 GIOP 메시지에서 보낸 데이터와 FFT CORBA 컴포넌트를 이용하여 데이터를 FFT 컴포넌트에 전달해 FFT를 수행하고 출력데이터를 CPU로 받아들이게 된다.

이 받아 들인 데이터는 미리 받은 GIOP 메시지에서 받은 IFFT 명령을 수행하기 위해 IFFT CORBA 컴포넌트를 사용하고 데이터를 IFFT 컴포넌트로 보내고 IFFT를 수행한 후 다시 CPU로 데이터를 받아 들이게 된다. 최종적인 데이터는 FPGA에서 CPU를 통해 GIOP 메시지형태로 만들어지며 PCI IP core를 거쳐 PCI 버스를 통해 MPC8280으로 보내지게 된다.

MPC8280에서는 받은 GIOP 메시지를 해석하여 보내진 데이터와 일치 하는지 확인한다

IV. 결론

논문에서는 CORBA를 위한 FPGA 설계방법을 제안하였다. 다른 프로세서와의 통신으로 속도와 좋은 성능 throughput을 위하여 PCI통신 방식을 채택하였으며, FPGA내부에 CPU를 탑재하여 그 위에 운영체제를 구동시키게 하였으며, 개발자가 원하는 CORBA 컴포넌트를 탑재된 운영체제로 실행시킬 수 있었다. 사용된 소스들은 모두 opencore로 라이센스가 필요하지 않다는 장점이 있다. 이는 개발 비용을 감소하고 개발자들 사이에 호환성 및 재사용성이 가능한 환경을 제공해 줄 수 있다. CORBA를 사용하지 않는다면 우리는 FPGA를 메모리로만 사용하거나 특수한 기능만을 위해서만 사용할 것이다. 그러나 FPGA에 CORBA 컴포넌트를 탑재하고 그에 해당하는 컴포넌트를 wishbone 버스에 연결함으로써, 외부에서 FPGA를 CPU처럼 사용하여, 정의된 컴포넌트를 제어하고 사용할 수 있게 된다. 제안한 FPGA 설계를 통해 SDR system을 구현하는데 FPGA 개발 solution을 갖게 될 것이다.

참고문헌

[1] "PCI system Architecture" Tom Shanley and Addison – Wesley , June 1999

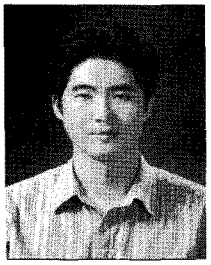
[2] "explanation of PCI BUS and design interface card" International Technology Information Institute

[3] "specification for the WISHBONE System-On-Chip(SOC) Interconnection Architecture for Portable IP Cores Revision: B.1" Silicore Corporation January 8 , 2001

[4] "Openrisc 1200 IP core Specification" Damjan Lampret , September 6 , 2006

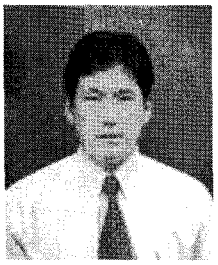
[5] "UART IP core specification" Jacob Gorban , August 11 , 2002

[6] "GPIO IP core specification" Damjan Lampret , December 17 , 2003



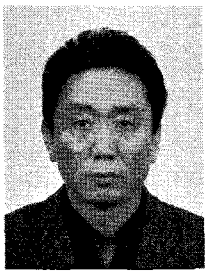
이 창 훈

2007년 한양대학교 전자통신컴퓨터공학부 졸업
2007년~현재 한양대학교 전자통신컴퓨터대학원 석사과정 재학중.
관심분야는 FPGA, SDR , Smart Antenna.



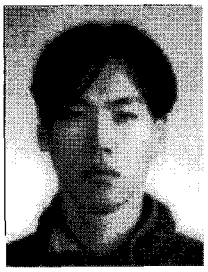
현 승 헌

2002년 한양대학교 전자통신컴퓨터공학과 졸업.
2004년 한양대학교 전자통신컴퓨터공학과 석사.
2004년~현재 한양대학교 전자통신컴퓨터공학과 박사과정 재학 중.
관심분야는 SDR, 스마트안테나



최 승 원

1980년 한양대학교 전자공학과(공학사).
1982년 서울대학교 전자공학과(공학석사).
1985년 Syracuse University(공학석사).
1988년 Syracuse University(공학박사).
1992년~현재 한양대학교 전자통신컴퓨터공학부 교수. 관심분야는 Smart antenna, SDR, MIMO



김 준

2006년 중앙대학교 전자공학부 졸업
2008년 한양대학교 전자통신컴퓨터대학원 석사
2008년~현재 한양대학교 전자통신컴퓨터공학과 박사과정 재학 .
관심분야는 Smart antenna, SDR.

정 재 호

1994년 중앙대학교 전자공학과 졸업
1994년 1월~2001년 4월 (주) 데이콤
2001년 University of Minnesota at Twin Cities 전기 및 컴퓨터공학과 석사 졸업
2001년 8월~2005년 7월 삼성전자 디지털미디어연구소 책임연구원
2005년 8월~현재 KT 인프라연구소 책임연구원
2008년 2월~현재 한양대학교 전자통신컴퓨터대학원 박사과정 재학중. 관심분야는 다중 안테나 기술 (MIMO, Beamforming), 통신시스템 설계, WiBro 및 3G/4G 기술.