논문 2009-46TC-10-5

FPGA ORB를 고려한 ORB 연동 프로토콜 개발

(Development of Inter-ORB Protocol for FPGA ORB)

정 혜 경*, 배 명 남**, 이 인 환**, 이 용 석*

(Heakyung Jeong, Myungnam Bae, Inhwan Lee, and Yongseok Lee)

요 약

HAO는 로직 기반의 코바 컴포넌트 개발을 수용하기 위해 FPGA에 탑재되는 ORB엔진이다. 본 논문은 표준 코바와 HAO 간 연동을 제공하기 위한 과정으로, 표준 ORB간 연동 프로토콜인 GIOP의 구체화 방안에 대해 기술한다. 구체적으로, 시스템 보드의 하드웨어 자원을 직접 사용한다는 관점에서 HAO Core가 하드웨어 독립성을 갖도록 하여야 하며, 공유 자원에 대한 빠른 제어 반환을 고려한 최적화 방안을 포함해야 한다.

Abstract

HAO is a ORB engine to support the logic-based CORBA component developments in FPGA. In this papers, in order to give the interoperability between general CORBA and HAO, we propose newly the specialization of GIOP(General Inter-ORB Protocol) from consideration of FPGA. It compose of the two major capability. First, it can abstract the hardware structure from the various system board environments. Secondly, also it is possible to minimize the monopoly occupation for shared resource such as system bus, external memory.

Keywords : FPGA ORB, GIOP, SDR, SCA

Ⅰ. 서 론

고속 병행 처리와 재구성을 통해 하나의 플랫폼에 여러 무선체계(waveform)를 지원하려는 시도가 확대되면서 무선통신 시스템에 분산 구조를 채택하고 있다. 일례로, SCA(Software Communications Architecture)[1]는 JTRS JPEO에 의해 소프트웨어 수준으로 무선체계 간의 상호운용과 재구성을 보장하기 위해 제안되었으며, 다수개의 범용 프로세서(GPP, General Purpose Processor)와 FPGA(Field Programmable Gate Array), 그리고 DSP들로 이루어진 분산 플랫폼을 제공하기 위해 코바(Common Object Request Broker Architecture) 미들웨어를 기반으로 하고 있다. 이를 통해, 무선체계 개발자는 이종의 프로세서와 RTOS(Real-time Operating System) 등 소프트웨어 플랫폼과 다양한 네트워크 하드웨어와 독립적으로 개발할 수 있다는 장점이 있다.

그러나 범용 코바 미들웨어는 FPGA상에 존재하는 무선체계와의 연동을 위해 GPP에 추가의 소프트웨어 어댑터를 두거나 혹은 FPGA에 소프트 코어와 운영체제, 코바 미들웨어, 그리고 프로토콜 스택을 추가로 적재하여야 한다는 단점이 있다. 전자의 경우에는 FPGA 로직을 사용하면서도 소프트웨어 어댑터를 거치는 과정에 성능 개선 효과가 반감되며, 후자의 경우 고가의 FPGA 타입이 요구된다는 점에서 비용과 융통성면에서 비효율적이다[2].

최근에는 이러한 문제를 해결하기 위해, FPGA상에

* 정회원, 전북대학교 전자정보공학부
(Division of Electronics & Information Engineering, Chonbuk Nat'l Univ.)
** 정회원, 한국전자통신연구원 USN응용기술연구팀
(USN-based Application Technology Research Team, ETRI)
※ 본 논문은 국토해양부 지능형국토정보기술혁신사업(06국토정보C01)의 연구비지원에 의해 수행되었습니다.
접수일자: 2009년6월9일, 수정완료일: 2009년9월23일

소프트 코어나 운영체제 없이 직접 코바 미들웨어를 적용하려는 연구가 진행되고 있다[3~4].

본 논문에서는 이를 위해 필요한 GPP상의 표준 코바와 FPGA상의 ORB간 연동 프로토콜의 개발에 대해서 기술한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련연구로서, SCA 환경과 FPGA ORB에 대해 간단히 기술한다. Ⅲ장과 Ⅳ장에서는 GPP 상에 적재될 범용 코바 미들웨어와의 연동을 위해 시스템 버스를 사용한 메시지 송수신 기능과 GIOP 메시지의 효율적 처리에 대한 설계 및 개발 결과에 대해 기술한다. Ⅴ장에서는 결론 및 향후 과제에 대해 기술한다.

Ⅱ. 관련 연구

JTRS JPEO는 무선통신 플랫폼에 FPGA상의 디지털 로직(하드웨어적으로 구현된)들을 수용하기 위한 여러 노력을 진행하고 있다. 초기 어댑터 방식[1]은 GPP에 어댑터를 별도로 두고, 이를 통해 FPGA상의 로직에 접근하도록 하였다. 이러한 방식의 단점은 어댑터가 FPGA의 하드웨어 자원(메모리 맵, 외부 인터럽트 등)을 직접 사용함으로 인해 결국 GPP상의 소프트웨어가 통신시스템의 하드웨어 구성에 의존성을 갖게 된다는 것이다. 이는 성능 저하 등의 문제 뿐 만 아니라 소프트웨어적인 재구성을 궁극적인 목적으로 하는 SCA의 목적에도 부합하지 않는다는 문제가 있다. 이에 따라, John Huie[5]등은 GPP상의 어댑터 대신에 FPGA상의 소프트 코어(예, MicroBlaze)에 운영체제와 코바 미들웨어, 그리고 이더넷 프로토콜의 포팅을 통해 로직과 연동하는 방식도 고려하였다. 이를 통해 GPP상의 코바 객체에 대한 완벽한 상호운용성을 제공할 수 있지만, FPGA에서 이들은 매우 큰 오버헤드라는 점에서 문제 해결이 아니고, 또한 이 오버헤드로 인한 비용과 규모 측면의 부담으로 크게 선호되지 않고 있다. 이에 따라, 보다 근본적으로 추가의 오버헤드를 제외하고 GPP 뿐만 아니라 FPGA까지 포괄한 무선체계간의 상호운용성 제공을 위해, FPGA에 운영체제나 이더넷의 지원 없이 코바 미들웨어를 직접 제공하는 방식이 새로이 대두되었다[3, 6].

이러한 맥락에서, FPGA ORB인 HAO[4]는 FPGA에서 빠른 분산 무선체계 개발을 목적으로 사용하기 위해, 로직 수준으로 개발한 ORB이다. 현재, HAO는 Xilinx FPGA에 IP core 형태로 제공되고 있다. 이때, 추가로 FPGA에 적재되는 HAO는 다른 GPP 혹은 다른 FPGA의 ORB와 상호 연동하기 위한 ORB 연동 프로토콜이 필요하다.

Ⅲ. ORB간 연동 프로토콜 설계

이 장에서는 코바 기반의 SCA 미들웨어 구조와 이에 따른 ORB 연동 방식, 그리고 이를 위한 소프트웨어 및 하드웨어적 환경과 주요 설계 이슈에 대해 기술한다.

1. 코바 기반 SCA 미들웨어 구조

소프트웨어적으로 재구성 가능한 무선체계는 적용될 플랫폼의 하드웨어 구조에 의존적이지 않아야 하며, 다양한 프로세서로 구성되는 분산 환경에서 운영되어야 한다. 이에 따라, SDR(Software Defined Radio)에서는 코바기반의 SCA미들웨어 구조를 제안하였다. SCA는 GPP상에 운영체제와 코바를 적재하고, 무선체계의 관리를 위한 SCA CF(Core Framework)와 무선체계내 개개 컴포넌트에 대한 바인딩 정보를 포함하는 도메인 프로파일 파서(DPP, Domain Profile Parser)로 구성된다. 무선체계 개발자는 이를 기반으로 서비스와 어플리케이션을 탑재하게 된다.

이에 따라, 현재 구축된 SCA 미들웨어에서 GPP에는 고속 통신용 코바인 UniORB[7]를 기반으로 SCA CF와 DPP를 구축하였으며, 아울러 FPGA에는 로직 컴포넌트의 지원을 위해 FPGA ORB인 HAO를 포함하도록 구성되었다.

OMG(Object Management Group)[8]는 이들 ORB간의 상호 연동을 위해, GIOP(General Inter-ORB Protocol)을 권고하고 있다. 따라서, UniORB나 다른 코

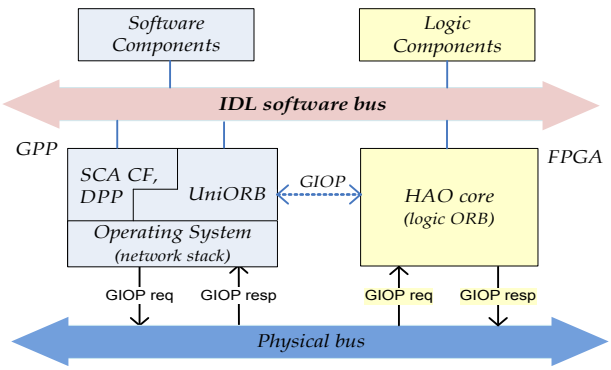

그림 1. 코바기반 SCA 미들웨어 구조
Fig. 1. SCA Middleware Architecture using CORBA.

바가 HAO와의 연동이 가능하도록 하기 위해 GIOP를 수용하여야 한다. GIOP의 대표적인 구현인 IIOP (Internet Inter-ORB Protocol)는 네트워크상의 원격 ORB에 대한 물리적인 접근을 제공한다. 그러나, SCA 기반 무선통신 시스템내 프로세서들(GPPs, FPGAs)은 단일 시스템 보드내에서 내외부 버스를 통해 물리적으로 연결된다. 특히, HAO의 적용과 같이 운영체제나 네트워크 프로토콜 오버헤드를 포함하지 않는 보드 형상일 경우, HAO는 이의 지원없이 외부와의 GIOP 메시지 송수신과 해석을 직접 담당하여야 한다.

2. SCA 환경에서 ORB 연동 프로토콜

이 장에서는 SCA와 HAO의 특징에 따라, ORB간 연동에 추가로 고려되어야 할 사항에 대해 기술한다.

가. GIOP 특징

SCA에서 GIOP는 UniORB 뿐만 아니라 범용 코바와의 연동을 위해, 반드시 지켜져야 하는 규약이며 다음과 같은 전제를 갖는다. 첫째, GIOP는 객체식별자에 의해 식별된 두 코바 객체(컴포넌트)간에 유니캐스트(unicast) 연결 구조를 갖는 것으로 정의한다. 전송 방식에 대한 다양한 이슈와 특성들이 있지만, 현재 SCA 환경에서는 유니캐스트만을 전제하고 있다. 둘째, GIOP 메시지내 구성되는 데이터의 형식으로 CDR(Common Data Representation)을 사용한다. 비록, CDR이 갖는 데이터 정렬 방식의 효율이 떨어지지만, 향후 범용 코바와의 호환성을 유지하기 위해 수용하도록 한다. 셋째, GIOP의 메시지에서 객체식별자와 오퍼레이션 명칭은 메시지의 상당 부분을 차지한다. FPGA의 특성상 IDL 컴파일 단계에서 HAO내 로직 컴포넌트의 물리적인 컴포넌트의 개수 등이 미리 결정되므로, 객체식별자와 오퍼레이션 명칭 등에 쓰이던 문자열 기반 식별자 대신에 최소 크기 할당 방식 등을 통해 HAO 처리를 보다 최적화하도록 한다.

나. SCA 하드웨어 환경 독립성 보장

SCA는 상용 COTS의 사용을 적극 권고하고 있기 때문에, 하위 하드웨어 환경에 대한 투명성을 보장하여야 한다. 범용 코바의 경우, 운영체제를 통해 하드웨어 구조에 대한 독립성을 보장받을 수 있지만, HAO의 경우 FPGA의 외부 IO를 직접 제어해야 하므로 하드웨어 독립성을 보장받기 위한 추가의 고려가 필요하다.

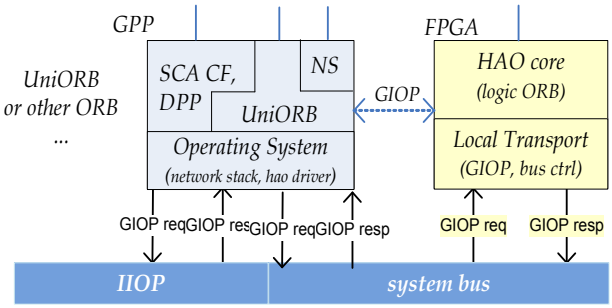

그림 2. 코바기반 SCA 미들웨어 구조
Fig. 2. SCA Middleware Architecture using CORBA.

즉, HAO는 무선체계 개발자의 컴포넌트가 하드웨어 구조에 영향을 받지 않도록, 하드웨어 의존적인 부분을 이들로부터 분리해야 할 필요가 있으며, 이를 위한 별도의 블록으로 local transport를 포함하도록 하여야 한다.

또한, UniORB는 ORB간 연동을 위해 두 개의 GIOP 구체화 수단을 가져야 한다. 하나는 자신 혹은 다른 상용 코바와의 연동을 위한 IIOP(Internet Inter-ORB Protocol)을 가지며, 동시에 HAO와의 연동을 위해 시스템 버스를 활용하여 GIOP 메시지를 송수신하는 수단도 제공하여야 한다. 즉, UniORB 역시 기존 방식과 호환하여 HAO와의 연동에 대한 투명성 보장 방안을 제공하여야 한다.

3. 주요 설계 이슈

이 절에서는 ORB간 연동 과정에서 해결하고자 하는 주요이슈에 대해서 기술한다.

가. 범용 코바의 GIOP 지원 확장

무선통신 시스템에서 최적의 성능을 보장하기 위해, UniORB(혹은 범용 코바)는 HAO와 효율적인 연동 방식을 제공하여야 한다. 범용 코바에서 ORB간 상호 연동은 IIOP와 같은 네트워크 프로토콜 기반 방식을 활용한다. 코바의 능력에 따라, 단일 프로세서내 여러 ORB간에 IIOP 대신에 공유메모리(shared memory)를 활용하는 방안 혹은 다중 코어내 ORB간에 전용(dedicated) 메모리를 사용하는 방안 등의 여러 방식이 가능하다. 이러한 방식은 선정된 코바가 제공하는 기능에 따른다.

HAO의 경우 이더넷 코어의 탑재로 인한 오버헤드를 줄일 필요가 있다. 이로 인한 FPGA의 비용 및 제한을 최소화할 수 있기 때문이다. 또한, 하드웨어 자원에 대한 직접적인 제어를 통해 상당한 성능 향상도 가능하다. 따라서, 범용 코바와 HAO는 커널 영역에서 외부

메모리를 직접 사용하는 방식을 통해 성능 저하없이 연동하는 방식을 적용한다. 이를 위해, IIOP와 시스템 버스를 사용한 추가 GIOP 구체화 수단을 제공한다.

나. 하드웨어 독립성 보장

HAO는 적용되는 통신 시스템의 하드웨어 구조와 밀접한 관계가 있다. 하지만, 이러한 하드웨어에 대한 의존성은 HAO의 적용과 이식에 많은 문제가 있을 수 있다. 따라서, HAO는 두 개의 계층으로 분리되며, 하위의 local transport 영역은 보드 규격에 따라 조정되어야할 외부 IO 제어와 GIOP 메시지에 대한 해석을 수행하며, HAO Core는 이외의 ORB 기능만을 담당하도록 구조화하여야 한다. 또한, local transport와 HAO Core간의 클록과 처리 동기화를 분리하기 위해 듀얼포트 블록 메모리를 통해 연동되도록 한다.

한편, HAO가 수신해야 할 GIOP 메시지에 대한 버퍼링이 필요하다. IIOP의 프로토콜 스택내 버퍼 대신에 보드상의 메모리 혹은 FPGA내 블록메모리를 사용할 수 있다. 성능 관점에서 FPGA내 메모리 활용은 여러 외부 요인으로 인해 바람직하지 않다. 또한, IO를 수행하지 않는 GPP의 처리와 HAO의 메시지 처리를 병행처리할 수 없기 때문에도 외부 메모리 활용이 바람직하다. 그러나, 하드웨어 특성(GPP대비 FPGA의 시스템 클록 비율, 시스템 버스의 공유 레벨 등)은 목적에 따라 다양화 될 수 있다. 따라서, 최적의 방안은 GPP와 FPGA간의 무선체계의 적용 범위와 역할이 먼저 고려되어야 할 것이다.

본 논문에서 범용 코바와 HAO간 메시지 버퍼는 보드상의 메모리로 가정한다.

다. 메시지 처리 성능

수신된 GIOP 메시지 처리와 관련하여, 일반적으로 GIOP 메시지는 부가적인 오버헤드를 많이 포함하고 있다. 예를 들어, 일반적으로는 사용되지 않는 데이터 필드들이 존재하며, 데이터 구성시에도 데이터 정렬(4byte alignment), 패딩 방비 요인 등이 있다. 이들은 HAO 내부 로직 처리에서 객체식별자와 오퍼레이션 명칭과 같은 큰 오버헤드는 아니지만, HAO Core내 로직의 단순화를 위해 이에 대한 최적화가 수행되어야 한다. 한편, 객체식별자 처리와 같이 HAO와 관련하여 외부 지원(예, IDL 컴파일러)을 통해 표준의 변경없이 가능한 개선을 적극 적용하여야 한다.

Ⅳ. 구현 및 결과

이 장에서는 SCA 환경에서 GPP상의 범용 코바와 HAO간의 연동 방식을 구현한 내용과 그 결과를 기술한다.

1. IOP를 위한 하드웨어 인터페이스

적용된 플랫폼은 GPP와 FPGA를 함께 탑재하고 있는 상용 보드를 사용하였다. GPP는 PXA272 프로세서[8](520MHz)이며 Linux(Kernel v2.6.x)를 사용한다.

FPGA는 Virtex-4의 보급형 모델인 XC4VLX60[9]을 사용하며 탑재된 HAO는 시스템 버스를 통해 UniORB (혹은 다른 코바)와 연결된다. 메시지의 송수신을 위한 메시지 버퍼는 보드상의 SRAM을 사용한다. 버퍼는 환형큐로 유지되며, 송신 버퍼와 수신 버퍼는 2MB의 크기로 별도로 구성된다.

무선체계에서 대부분의 GPP가 마스터가 되므로, 보드의 시스템 버스에 대한 제어는 PXA272의 alternative bus master interface를 사용한다.

HAO가 송신할 메시지를 가진 경우, 인터럽트(fpga_int)를 통해 송신 메시지가 존재함을 알리고, UniORB로 부터 HAO에 대한 버스 접근이 허용(MER[0]=1)되면 MBREQ/MBGNT 처리를 통해 버스

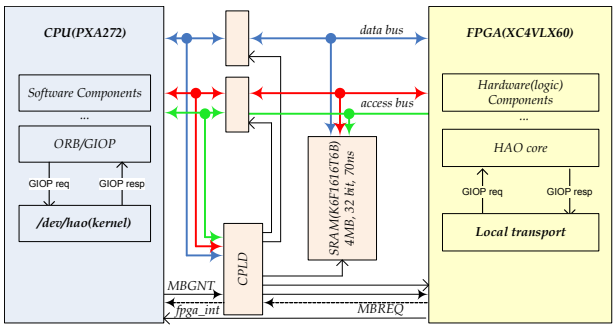

그림 3. 시스템 보드의 블록 다이어그램
Fig. 3. Block diagram of system board.

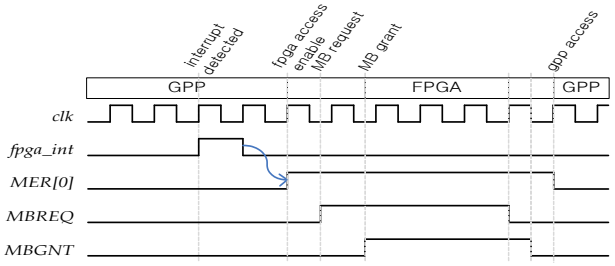

그림 4. 제어 시그널을 통한 IO 제어
Fig. 4. IO control with control signal.

제어를 얻는다. 이후, 송신 메시지를 SRAM에 저장하고 완료 후 버스 제어를 반환한다.

반대로, UniORB가 HAO에게 송신할 경우, 송신 메시지를 SRAM에 저장 완료 후 MER[0]를 활성화함에 따라 HAO에 의한 버스 제어 획득 절차를 진행하도록 요구한다. 버스 제어권을 획득한 HAO는 내부 처리 상태에 따라, SRAM에 대한 시도를 수행할 지 판단한다.

2. UniORB의 제약과 확장

UniORB는 IIOP 대신에 이러한 절차를 진행하는 GIOP 구체화 수단이 필요하다. 이 절에서는 UniORB에서 로직 컴포넌트에 대한 바인딩과 메시지 처리 효율을 높이기 위한 추가의 고려를 제공한다.

가. 정적 IOR 생성과 추출

로직 컴포넌트는 FPGA의 특성상 합성과 P&R과정에서 개수와 위치가 물리적으로 결정된다. 따라서, 모든 로직 컴포넌트의 IOR은 정적으로 결정될 수 있으며, IOR은 IDL 컴파일러에 의해 컴파일 시점에 생성하도록 한다. 따라서, HAO에서 동적인 IOR 처리에 대한 부담과 오버헤드를 크게 경감할 수 있다.

SCA 미들웨어 환경에서 IOR의 검색은 네이밍 서비스(NS, Naming Service)를 통해 획득하도록 권고하고 있지만, 강제 사항은 아니다. 하지만 위의 고려에 따라, UniORB(혹은 다른 코바)는 FPGA상의 로직 컴포넌트에 대한 IOR 취득시 반드시 네이밍 서비스를 통해 얻도록 하여야 한다.

나. IOR 해석과정에 IOP_TAG_HAO를 추가

네이밍 서비스로부터 얻은 로직 컴포넌트의 IOR은 코바 표준 IOR과 달리 일부 변경을 갖는다. 가장 큰 차이는 첫째 새로운 IOP 태그를 추가한 것이며, 둘째 객체식별자는 정적으로 할당됨에 따라 최소화된 크기를 갖도록 구성되었다는 점이다.

다음 표 1은 이를 고려한 IOR 구성을 보인다.

IOR에서 목적 FPGA를 식별하기 위해 Host를, 해당 FPGA내 특정 HAO를 식별하기 위해 Port를 사용한다. object_key는 로직 컴포넌트를 식별하기 위해 사용되며, 이때, HAO에 바인드된 로직 컴포넌트의 개수와 계층 구조 등을 고려하여 최소 크기로 구성된다. 이후, 이의 활용을 통해 GIOP 메시지 처리의 부분적인 성능 개선을 얻을 수 있다.

표 1. HAO에서 IOR 구성
Table 1. IOR definition in HAO.

01 000000	Host endian
0d000000	Str_len(13)
49444c3a4563686f3a312e3000 000000	IDL:Echo:1.0
01000000	Seq_len(1)
0F000000	Tag(F=IOP_TAG_HAO)
18000000	Mislen(24)
01 000000	Data endian
01000000	Strlen(1)
10 000000	Host(=Id0)
0300 0000	Port(=3)
04000000	Len(4)
f2000000	Object_key(=f2)

UniORB가 네이밍 서비스로부터 얻은 IOR의 태그로 IOP_TAG_HAO를 인식하면, 다음 방식을 통해 HAO와의 연동을 시도하게 된다.

다. 하드웨어를 직접 활용하는 IOP 방식

HAO는 이더넷 혹은 소프트 코아의 탑재를 요구하지 않는다. HAO와 연동만을 고려할 때, 이들은 큰 비용적 부담이 요구되며 성능상의 이점도 감쇄한다. 따라서, HAO에 대한 UniORB의 IOP는 IIOP와 같은 부가의 프로토콜 스택(예, TCP 혹은 UDP on IP)을 거치지 않고 하드웨어 자원의 직접적인 제어를 통해 GIOP 메시지의 송수신하며, 그 결과 UniORB에서도 이에 대응하여 해당 하드웨어 자원을 통한 송수신 장치 드라이버가 필요하다.

정의된 HAO 연동용 장치 드라이버는 IOR상의 host(=Id0)에 의해 식별되는 FPGA에 대한 접근과 해당 FPGA내 HAO별로 송수신 버퍼에 접근을 위한 레지스터 맵을 정의하며, 추가로 다른 FPGA와 구분된 GPP로의 인터럽트를 추가로 갖도록 정의된다(하드웨어 형상에 따라 변경가능).

HAO 장치 드라이버를 통한 IOP 연동 방식에서는

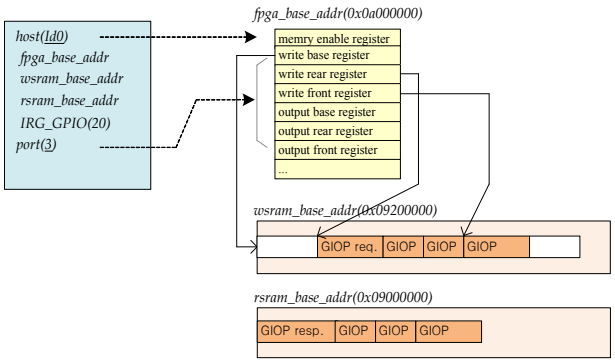

그림 5. 레지스터 맵과 인터럽트 정의
Fig. 5. Register map and interrupt handler in device driver.

PXA272에서 제공하는 alternative bus master interface(MBREQ/MBGNT)와 인터럽트(FPGA_int)를 통해 시스템 버스를 제어하도록 정의된다(그림 4 참조). 또한, 해당 FPGA내 존재하는 HAO가 자신의 개별적인 GIOP 메시지 버퍼에 접근하기 위한 6개의 추가 레지스터를 갖는다. WBR(Write base register)는 UniORB에서 HAO로의 메시지 전달을 위한 버퍼의 시작 주소를 가지며, WRR(write rear register)와 WFR(write front register)를 통해 환형큐로 유지된다.

3. HAO에서 연동 프로토콜 구체화

UniORB의 장치 드라이버와 동등한 레벨로 보드상의 SRAM을 통해 메시지 송수신을 제어하기 위해, HAO는 local transport를 제공하고 있다. 이 절에서는 local transport의 구현과 특징에 대해 기술한다.

가. HAO와 연동

Local transport는 HAO Core가 하드웨어 보드와의 독립성을 갖도록 구조화되었다. 이를 위해, 하드웨어 구조(버스 크기, IO 시그널, 외부 메모리 동작 클록 등)를 파라미터화 한 형상 파일에 따라 합성되며 이에 따라 재구성되도록 정의되었다. 또한 local transport는 xilinx의 coregen에 의해 생성된 HAO Core와 비동기 블록 메모리(Asynchronous FIFO)를 통해 연동된다. 그 결과, FPGA 내외부 하드웨어 구성의 다양성으로부터 HAO Core의 독립성을 보장하도록 할 수 있다.

한편, 그림과 같은 외부 SRAM의 활용 대신에 FPGA 내 블록 메모리를 직접 활용하는 방식도 가능하다. 이러한 형태의 하드웨어 구성 및 인터페이스의 변경은 전적으로 local transport에 의해 추상화되며, HAO Core에게는 FIFO를 통한 GIOP 송수신 인터페이스만을 유지한다.

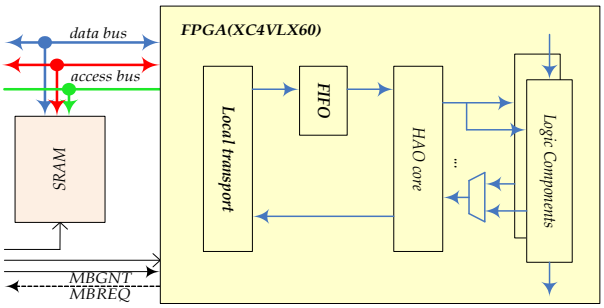

그림 6. HAO와 연동 구조
Fig. 6. Interworking with HAO.

나. GIOP 송수신 기능

Local transport의 가장 중요한 역할은 하드웨어 구성과 독립적으로 GIOP 메시지를 송수신하는 것이다.

이를 위한, local transport의 구조는 다음과 같이 구성되며, SRAM으로부터 메시지를 수신하여 FPGA내 블록 메모리로 전달하는 RD_CON 블록, HAO Core로부터 받은 송신 메시지를 SRAM에 전달하기 위한 WR_CON 블록, 그리고, 하드웨어 보드의 자원 제어와 FPGA내 유지되는 레지스터 맵을 관리하는 LT_CON을 포함한다.

LT_CON은 다수 레지스터들을 통해 시스템 버스의 상태(MER)와 환형 큐 형태로 유지되는 SRAM내 메시지 버퍼 주소를 관리한다. 아울러, HAO Core로부터 메시지 전달 관련된 시그널들을 참조하여 local transport의 처리를 결정한다.

그림 8은 FPGA가 송신 메시지를 전달하는 과정을 보이는 상태 다이어그램이다.

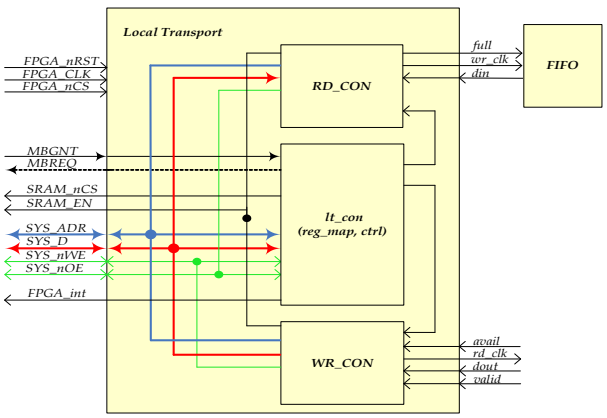

그림 7. LT의 블록 다이어그램
Fig. 7. Block diagram of Local Transport.

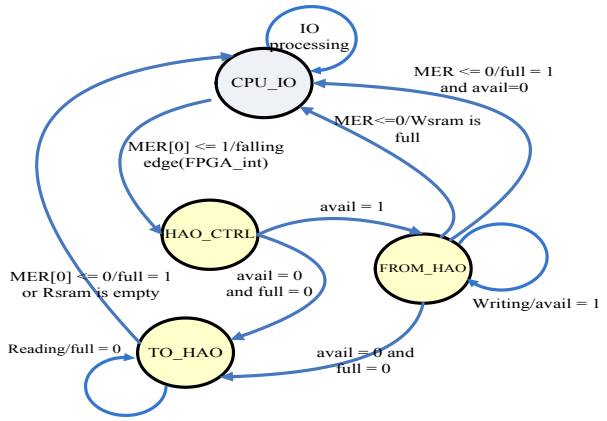

그림 8. LT에서 메시지 IO의 상태 다이어그램
Fig. 8. State diagram of Message IO in Local Transport.

Local transport는 HAO Core로부터 메시지 송신 요구에 따라, 인터럽트를 발생시키며 버스 허용 레지스터(MER)를 통해 시스템 버스 사용을 인가한다. HAO는 송신에 우선순위가 높으며 SRAM으로 전달완료 후, HAO Core가 수신 가능할 경우 SRAM으로부터 수신 버퍼(FIFO)로 수신 메시지를 전달한다. 만일, 수신 버퍼가 full(=1)이 되거나 UniORB로부터 송신 메시지가 더 이상 없는 경우, 시스템 버스 접근을 종료한다.

다. GIOP 메시지 변환 기능

Local transport는 ORB간 표준 프로토콜을 지원한다. 현재, 외부 코바로부터 SRAM 상에 전달되는 메시지는 GIOP v1.1을 요구한다. 이는 UniORB와의 연동뿐만 아니라 범용 코바와 HAO간의 연동을 보장하여야 하기 때문이다. 하지만, local transport가 수신한 GIOP 메시지를 HAO Core에 전달할 때, 불필요한 부분을 제외하고 자체적인 최적화를 수행한다.

GIOP v1.1에 따른 메시지는 다음과 같다.

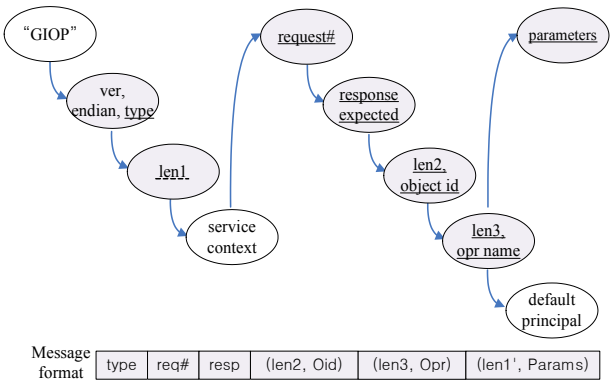

그림 9. LT에서 메시지 변환
Fig. 9. Message Change in Local Transport.

실제 환경에서 사용되지 않는 GIOP 필드들(service context, default principal)은 제외되며 부분적인 데이터 패딩(type, response expected), 그리고 데이터 바이트 순서의 변환(endian)이 수행된다. 파라미터의 경우 4 바이트 데이터 정렬에 따라 많은 패딩을 포함하게 되지만, 로직 컴포넌트의 역할이 복잡한 데이터 처리 보다는 고속 병행 처리에 있기 때문에 실제 환경에서는 이러한 복잡성이 낮으므로 성능에 큰 차이가 없으며, 오히려 앞에서 설명한 GIOP에서 사용하지 않는 필드의 제외와 객체식별자와 오퍼레이션 이름에 대한 최적화가 보다 큰 개선 요소이다.

4. 성능 결과

본 연구에 앞서, UniORB의 모든 Testbench를 컴파일하고 얻어진 이들 결과물들을 HAO와 함께 합성/P&R 과정을 거친 후 실제 FPGA에서 성능을 평가한 바 있다. 그 결과, HAO와 개발자 로직인 로직 컴포넌트를 포함해서 평균 2,900여 로직 셀로 구성되고 있으며, 커널 레벨(/dev/hao)에서 소프트웨어 ORB인 UniORB의 성능과 비교를 통해 평균 30배 정도의 성능 개선을 제공하고 있다[4].

GIOP 메시지 수신 처리 성능은 표준 GIOP V1.1에 준하며, 메모리 접근 시간(T_{mem})과 FIFO에 쓰기 시간(T_{FIFO}), 그리고 레지스터 접근 시간(T_{reg})이 고려된다.

ORB에 따라 가변성은 있지만, 24byte 식별자를 사용하며, 짧은 오퍼레이션 이름(길이 <4)과 한 워드 파라미터를 사용하는 경우에 대해(인터럽트 발생후, MER 설정, 그리고 시스템 버스 제어 요청/승인은 공통이므로 이에 대한 고려는 제외함), 헤더(3 wd)와 바디(14 wd) 단위가 되며 $(3+14)\times(T_{mem}+T_{FIFO})$이다. 반면에, 최소 객체식별자를 사용하고 불필요한 헤더를 제외하는 경우, $(3+14)\times T_{mem}+8\times T_{FIFO}$의 결과를 얻을 수 있다.

여기에서, $T_{mem}=T_{FIFO}$일 경우, 26%의 개선효과를 가진다. $T_{mem}<T_{FIFO}$일 경우, 메시지의 압축을 통해 개선된 만큼 선형적으로 개선효과가 증가하는 반면, $T_{mem}>T_{FIFO}$일 경우에는 상대적으로 FIFO의 저속처리로 메시지 압축으로 인한 개선 효과를 줄어들게 된다.

시험 로직 컴포넌트가 시스템 버스에 대한 제어/획득에 요구되는 시그널링 시간을 포함하여 local transport, HAO Core를 거쳐 처리하고, 역으로 응답 메시지를 송신하는 과정에서 local transport는 전체 처리 시간의 8%정도를 차지한다는 관점에서 단일 메시지 처리 관점에서는 큰 성능개선 요인은 아닐 수 있지만, local transport가 전체 시스템의 공용자원인 시스템 버스에 대한 제어를 가진 체 동작한다는 관점에서 26%의 개선

표 2. Local transport의 개선
Table 2. Improvement of local transport. (단위: ns)

Tfifo \ T_{mem}	40			80		
20	1,020	840	82%	1,700	1,520	89%
40	1,360	1,000	74%	2,040	1,680	82%
80	2,040	1,320	65%	2,720	2,000	74%
160	3,400	1,960	58%	4,080	2,640	65%

은 시스템 버스에 대한 빠른 반환으로 전체적인 시스템 성능 개선에 영향을 미치는 주요한 요소이다.

V. 결론 및 향후 과제

HAO는 SCA 환경에서 코바 기반의 로직 컴포넌트를 수용하기 위해 FPGA에 탑재되는 ORB엔진이다.

본 논문에서 HAO를 고려한 ORB간 연동을 제공하기 위한 방안에 대해 기술하였다. 구체적으로, UniORB에서 보드상의 SRAM에 접근하기 위한 GIOP 구체화 방안을 제시하였고, 또한 HAO의 local transport의 기능과 역할에 대해서 기술하였다.

특히, local transport는 HAO Core에 하드웨어 보드 구성의 변경으로부터 추상화되도록 구성되었으며, 불필요한 GIOP 메시지에 대한 압축 및 패딩 제거에 대한 고려를 포함하고 있다. 이를 통해, 하드웨어의 공용 자원인 시스템 버스에 대한 빠른 반환이 가능하다. 향후, HAO Core의 연동 구조 개선과 함께, 이에 따른 전체 시스템 측면의 개선 효과에 대한 평가와 분석을 진행할 예정이다.

한편, 범용 ORB는 내부 구조를 변경할 수 없다. 따라서, 범용 ORB가 HAO와 연동될 수 있도록 하기 위해, 범용 ORB는 일종의 네트워크 어댑터인 가상 디바이스에게 IIOP를 통해 패킷을 전달하고, 이를 local transport에 전달하는 기능을 추가로 확장하고 있으며, 아울러 local transport가 다양한 하드웨어 구조에 쉽게 적용할 수 있도록 IP Core화 하는 작업이 추가로 진행하고 있다.

참 고 문 헌

[1] Joint Tactical Radio Systems, “Software Communications Architecture Specification V2.2.” Nov. 2002.
[2] Mark Goosman, “Changing Horses in Midstream: Partial Reconfiguration for FPGA Designs,” COTS Journal, May 2006.
[3] Joe Jacob, “CORBA for FPGA: The Missing Link For SCA Radios,” COTS Journal, Vol. 9, No. 1, pp. 30-33, Jan. 2007.
[4] 배명남, 이병복, 박애순, “FPGA에서 SCA 컴포넌트 개발을 지원하는 하드웨어 ORB”, 한국통신학회논문지, 제34권, 제3호(무선통신), pp. 185-196, 2009.
[5] John Huie, et. al., “Synthesizing FPGA Cores for Software Defined Radio,” SDR Forum, Nov. 2003.
[6] S. Aslam-Mir, “ICO: Integrity Circuit ORB,” PrismTech white paper, 2006.
[7] 장종현, 이동길, 한치문, “개방형 통신 시스템을 위한 고성능, 고신뢰성 CORBA 플랫폼에 관한 연구”, 대한전자공학회논문지, 제41권 TC편, 제2호, pp. 19-29, 2004.
[8] Object Management Group, “The Common Object Request Broker Architecture: Core Specification Revision 3.0.” Dec. 2002.
[9] Intel PXA27x Processor Family Developer's Manual, 2004.
[10] Xilinx, Virtex-4 FPGA User Guide(UG070) V2.5, June 2008.

저 자 소 개

정 혜 경(정회원)
1990년 전북대학교 전산통계학과 학사 졸업
1992년 전북대학교 전산통계학과 석사 졸업
2009년 현재 전북대학교 전자정보공학부 컴퓨터전공 박사과정
<주관심분야 : 자연어처리, 음성/신호 처리, 통신 미들웨어>

이 인 환(정회원)
한국전자통신연구원 USN응용기술연구팀 책임연구원
대한전자공학회 논문지
제46권 TC편 제6호 참조

배 명 남(정회원)
한국전자통신연구원 USN응용기술연구팀 책임연구원
대한전자공학회 논문지
제46권 TC편 제6호 참조

이 용 석(정회원)
1977년 서울대학교 전자공학과 학사 졸업
1979년 한국과학기술원 전산학과 석사 졸업
1995년 일본 국립도쿠시마대학교 지능정보 박사 졸업
1979년 한국표준연구소 선임연구원
2009년 전북대학교 정보전산원장
1983년~현재 전북대학교 전자정보공학부 교수
<주관심분야 : 자연어처리, 정보검색, 음성처리, 데이터 색인>