시뮬링크를 활용한 SCA 웨이브폼의 자동 생성 방법론

김선희, 심효택 ○, 설진호, 맹승렬
카이스트 전산학과
{sheekim, htsim, jhseol, maeng}@camars.kaist.ac.kr

Auto-Generation Methodology of SCA Waveforms by Using Simulink

Sunhee Kim○, Hyotaek Shim, Jinho Seol, and Seungryoul Maeng
Computer Science Department
KAIST

요 약

SCA(Software Communications Architecture)는 SDR(Software Defined Radio)를 위한 표준 플랫폼으로, 어떤 플랫폼에서도 SCA 표준을 이용하여 구현된 시스템이라면 SCA 규격을 지켜 작성된 소프트웨어 모듈을 실행할 수 있도록 하고 있다. SDR은 기존의 하드웨어로 구현하였던 무선 통신 시스템을 모두 소프트웨어로 구현하고자 하는 기술이지만, 임베디드 시스템의 경우에는 프로세서의 성능이 현저히 떨어지기 때문에 실시간 신호 처리를 보장할 수가 없다는 문제점이 있다. 따라서, 무선통신의 성능을 보장하기 위해서는 범용 프로세서와 함께 DSP나 FPGA와 같은 특화된 하드웨어의 사용이 필요하게 되었다. 이러한 경우에는 웨이브폼 어플리케이션 작성을 위해서 하드웨어와 소프트웨어의 파티셔닝도 고려해야 한다. 본 논문에서는 SCA 플랫폼에서의 웨이브폼 어플리케이션을 빠르게 생성하는 방법을 제안하여 다양한 하드웨어를 사용하는 플랫폼에서의 최종 웨이브폼 어플리케이션을 위한 설계 공간 탐색(Design space exploration)을 도와, 내장형 시스템에서도 효율적으로 실행 가능한 웨이브폼 어플리케이션을 개발할 수 있도록 한다.

1. 서 론

SCA(Software Communications Architecture)는 SDR(Software Defined Radio) 단말을 위한 소프트웨어 플랫폼 공개 구조(open architecture)로, 미 국방성의 JTRS(Joint Tactical Radio System) 프로그램 하에서 개발된 후, SDR 포럼에서 표준으로 채택되었다. SDR은 무선 이동 통신 시스템에서 RF 영역을 포함한 대부분의 신호 처리를 소프트웨어 기능 블록으로 구현함으로써, 하드웨어의 교체 없이 소프트웨어의 업그레이드나 재구성만을 통하여 다중 무선 접속 규격 및 서비스 등을 실현하는 기술이다. SDR은 하드웨어로 구현된 라디오와 달리, 사용 가능한 시스템에 맞도록 웨이브폼 어플리케이션을 바꿀 수 있고, 표준의 변화에 적응하기 쉬우며 새로운 서비스를 추가하는 등의 재구성 능력(reconfigurability)이 뛰어나다. 또한, 제3의 개발자의 컴포넌트 개발과 상용 소프트웨어(COTS, Commercial-off-the-shelf)의 사용을 촉진 시키며, 일반적인 데스크톱 컴퓨터에서도 다양한 응용 프로그램을 사용할 수 있는 등 비용 절감 효과도 크다. SCA는 이러한 SDR을 위한 플랫폼이므로, 라디오의 신호 처리 기능 블록 등의유연성(flexibility)과 이기종 라디오 플랫폼에서 실행 가능한 운용성(interoperability)을 증가시키고, SDR을 지원하기 위한 비용을 줄이면서도 새로운 기술을 접목시키거나 기존 기술의 업그레이드가 용이하도록 하는데 주안점을 두고 있다.

SCA는 이러한 목표에 부합하고자 여러 가지 표준을 채택하고 있다. POSIX(Portable Operating System Interface)를 만족하는 운영체제를 사용하게 함으로써 코드의 이식성(Portability)을 보장하고, 분산 객체 모델의 산업 표준인 CORBA(Common Object Request Broker Architecture)를 미들웨어(Middleware)로 채택하여 상호 운용(interoperate) 가능한 시스템을 구성하고 있다. CORBA는 이기종의 하드웨어와 소프트웨어의 통합 환경을 제공하므로, 플랫폼 독립성이나 컴포넌트의 재사용성이 뛰어나고 웨이브폼 어플리케이션의 업그레이드와 관리가 용이하게 된다. 이들 외에도, CORBA Naming Service나 OMG Event Service, XML (eXtensible Markup Language) 등이 사용된다[1].

"본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (IITA-2008-C1090-0801-0045)

현재 SCA는 Canada CRC, PrismTech, Virginia Polytech 등의 기관에서 공개 또는 상용 프로그램으로 만들어지고 있다. 하지만 이것들은 AP(Access Point)등의 서버급 머신을 타겟으로 구현된 것으로, 아직 내장형 시스템을 위한 SCA는 상용화 되지 못하고 있다. 2.0GHz에서 3.0GHz 정도의 서버급 머신의 범용 프로세서(GPP, General Purpose Processer)의 처리속도에 비해서 내장형 시스템에서 주로 사용되는 GPP의 처리속도는 400MHz 정도가 보통이다. 이렇게 낮은 성능의 프로세서로 인하여, 내장형 시스템에서는 높은 대역(High bandwidth)의 웨이브폼 어플리케이션을 실시간으로 처리하기 힘들다.

이러한 문제점 때문에 많은 업체와 기관들로부터 부가적인 하드웨어 사용의 필요성이 논의되었다. 기존의 범용 프로세서 위주의 SCA 스팩에 DSP와 FPGA의 사용 방법을 제안하는 논문이 나왔고[2], SCA 표준에서도 하드웨어 사용에 대한 명세를 추가하고 있다[3,4]. 명세에서는 FPGA와 DSP와 같은 특화된 하드웨어 프로세서들(SHPs, Specialized Hardware Processors)을 어떻게 사용할 것인지에 대한 내용이 기술되어 있지만, 세부적인 구현 방법에 대해서는 논의하고 있지 않다. 구현 방법으로는 SCA 시스템과 하드웨어 간의 어답터(Adapter)를 두는 방법과 어답터 사용의 이식성을 향상시키기 위한 하드웨어 추상 계층(Hardware Abstraction Layer)을 사용하는 방법, 그리고 추상 계층의 인터페이스를 표준화 시키는 방법과 하드웨어 적으로 구현된 CORBA를 사용하는 방법 등이 사용될 수 있다[5].

하지만, 범용 프로세서 이외에 FPGA와 같은 특화된 하드웨어를 함께 사용하는 플랫폼을 위한 웨이브폼 어플리케이션을 개발하는 것은 범용 프로세서만 사용하는 플랫폼에서 보다 시간과 비용이 많이 든다. 우선, 하드웨어에서 실행할 컴포넌트를 개발하는 것은 전문가가 아니면 어려운 일이다. 그리고, 해당 컴포넌트를 하드웨어로 구현할 것인지 소프트웨어로 구현할 것인지에 대한 파티셔닝이 이루어 지기 전에 모두 개발해 본다는 것은 더 큰 시간과 비용을 필요로 한다. 따라서 본 논문에서는 대표적인 상용 모델링 툴인 시뮬링크(Simulink®)[6]의 코드 자동 생성 기능으로 만들어진 코드를 변환하여, 바로 SCA에서 실행 가능한 웨이브폼 어플리케이션을 만드는 방법을 제안한다.

2. 웨이브폼 어플리케이션의 개발

SCA 시스템은 SDR을 위한 소프트웨어 구조이다. SCA 시스템 위에서 돌아가는 응용 프로그램을 웨이브폼 또는 웨이브폼 어플리케이션이라고 부른다. 이 웨이브폼 어플리케이션은 웨이브폼 컴포넌트라는 서브 모듈들의 조합이며, SCA의 베이스 어플리케이션 인터페이스(Base Application Interface)를 상속 받아 개발된 모든 컴포넌트는 웨이브폼 컴포넌트이다. XML로 작성된 도메인 프로파일(Domain Profile) 문서에 이러한 웨이브폼 어플리케이션과 웨이브폼 컴포넌트에 대해 기술하고 있으며, SAD(Software Assembly Descriptor) 도메인 프로파일을 읽어서 필요한 웨이브폼 컴포넌트를 실행시키고 연결한다. 웨이브폼 어플리케이션은 일반적인 통신을 위한 CDMA나 GSM과 같은 모듈일 수도 있고, 동영상을 재생하는 응용 프로그램일 수도 있다.

내장형 시스템은 가격과 전력 사용량의 제한 때문에 고성능의 프로세서를 사용할 수 없다. 많은 내장형 시스템들은 400MHz 정도의 처리 능력을 가진 프로세서를 사용한다. 대표적인 무선통신 내장형 시스템인 휴대폰은 주로 250MHz~400MHz의 ARM9 코어를 사용하고 있고, 스마트 폰은 400MHz~650MHz 정도의 ARM11 코어를 사용하고 있다. 따라서, 내장형 시스템에서 SCA 시스템을 사용하기 위해서는 부가적인 하드웨어의 사용이 불가피하다. 하지만 부가적인 하드웨어의 사용으로, 범용 프로세서만 사용하는 플랫폼에 비해서 웨이브폼 어플리케이션의 개발이 어렵게 된다. 하드웨어 위에서 실행할 컴포넌트의 개발이 어렵고, 해당 컴포넌트를 하드웨어에서 실행할 것인지 소프트웨어로 실행할 것인지를 결정해야 하는 파티셔닝(Partitioning) 이슈가 생기기 때문이다. 따라서 본 논문에서는 모델 기반 디자인 방법을 이용하여 시뮬링크로 자동 생성된 코드를 바로 SCA에서 실행 가능한 컴포넌트를 만드는 방법을 제시한다. 이렇게 만들어진 웨이브폼 컴포는트들을 이용하면 디자인 공간 탐색이 좀 더 쉽고 빠르게 진행되기 때문에 다양한 환경에서의 웨이브폼 컴포넌트 개발이 쉬워진다.

2.1 PIM/PSM

SCA 시스템에서 웨이브폼 어플리케이션은 OMG(Object Management Group)의 MDA(Model Driven Architecture) 패러다임을 적용하여 개발된다. MDA는 이기종 플랫폼 및 언어독립성을 위해서 시스템의 기능과 구현을 분리시키는 구조로, 웨이브폼 어플리케이션을 플랫폼에서 독립시켜 개발자는 비즈니스 로직(business logic)에만 집중할 수 있도록 해준다. MDA는 UML을 바탕으로 플랫폼 독립적인 모델(PIM, Platform Independent Model)과 플랫폼 의존적인 모델(PSM, Platform Specific Model)로 구분되는데, PIM은 시스템의 기능에 대한 명세로 인터페이스나 동작 등을 정의한 플랫폼 독립적인 설계 모델이며, 이러한 PIM을 타겟 플랫폼에 맞게 C/C++나 Java, VHDL과 같은 언어로 구현한 것이 PSM이다.

웨이브폼 컴포넌트의 PIM에서 관련 데이터와 컨트롤 인터페이스를 정의하고, 그에 따른 PSM은 여러 플랫폼을 타겟으로 하여 다양하게 구현될 수 있다. 웨이브폼 어플리케이션을 만들거나 사용하기 위해서는 필요한 웨이브폼 컴포넌트들의 PIM만 알고 있으면 PSM은 환경에 맞게 구현만하면 되는 것이다. PIM과 PSM의 매핑을 위해서 몇 가지 방법이 사용되고 있는데, SCA에서는

XML을 사용하여 플랫폼에 맞는 PSM을 가져다 쓰게된다. 웨이브폼을 실행 할 때에는 SAD 파일로부터 웨이브폼 어플리케이션을 구성하기 위한 웨이브폼 컴포넌트들의 리스트와 포트 연결 구조, 각 웨이브폼 컴포넌트의 PSM 정보를 기술하고 있는 SPD(Software Package Descriptor) 문서 위치 등을 읽어서 하나의 웨이브폼을 구성하고 실행한다. 각 웨이브폼 컴포넌트의 SPD 파일에서는 어떤 구현 프로그램을 사용할 것인지, 프로그램의 저장 위치는 어디인지 등이 기술되어있다.

따라서 우리는 PIM에 따라 웨이브폼 컴포넌트의 이름과 사용되는 데이터, 컨트롤 변수, 포트 등을 사용자로부터 입력 받아 비즈니스 로직을 제외한 코드를 자동 생성하는 웨이브폼 개발 프로그램(WavDeveloper)을 사용하였고, 필요한 비즈니스 로직은 시뮬링크의 코드 자동 생성 기능을 이용하여 다양한 플랫폼에서의 웨이브폼 개발을 보다 쉽게 하였다. 이러한 방법을 사용할 경우, OOP나 POSIX, CORBA와 같은 개념이나 SCA 코어프레임워크에 대한 깊은 이해 없이도 간단한 모델링만으로도 웨이브폼 어플리케이션을 만들 수 있으며, 다양한 하드웨어 플랫폼에서의 웨이브폼의 프로토타입을 빠르게 구현할 수 있다.

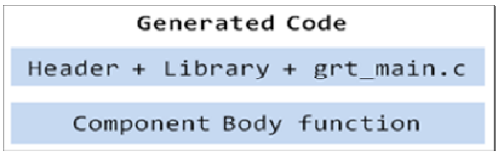


그림 1. Simulink에 의해 자동 생성된 코드의 구조.

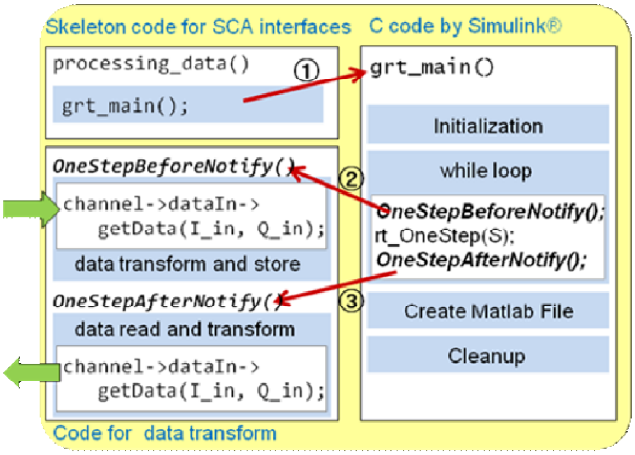


그림 2. 시뮬링크의 자동 생성 코드를 이용한 SCA 웨이브폼 컴포넌트의 구조

2.2 소프트웨어 웨이브폼 컴포넌트 자동생성

시뮬링크는 매스웍스(Mathworks) 사에서 개발한 다중 도메인 시뮬레이션(multidomain simulation)과 내장형 시스템을 위한 모델 기반의 디자인(Model-based design)의 도구이다. 시뮬링크는 그래픽 환경에서 커뮤니케이션, 컨트롤, 신호 처리, 영상 처리, 이미지 처리 등을 포함하는 다양한 모델을 디자인 및 시뮬레이션, 구현과 테스트를 위한 여러 가지 라이브러리를 제공한다. 그리고, 시뮬링크에서는 사용자가 디자인한 모델을 위한 C나 VHDL 코드를 자동으로 생성해주는 기능을 제공하고 있다. 이러한 특성에 따라, 매스웍스사 역시 매트랩(Matlab®)과 시뮬링크를 SDR을 위한 툴로서 적극 추천하고 있으며[7], 모델 기반 디자인 방법을 사용하여 쉽게 SCA 호환 가능한 SDR을 만들 수 있는 가능성을 제시하였다[8]. 이 논문에서는 시뮬링크로 모델링 후, 자동으로 생성된 코드를 웨이브폼 컴포넌트의 비즈니스 로직으로 바로 사용할 수 있도록 하기 위해서 다음과 같은 작업을 하였다.

1) 목적하는 모델을 시뮬링크에서 디자인한 후, 코드를 생성한다.
2) 웨이브폼 컴포넌트의 정보를 입력 받아 SCA 표준 인터페이스를 위한 코드를 생성한다.
3) 시뮬링크 코드와 SCA 표준 인터페이스 모듈 사이의 데이터 호환을 위한 콜백(Callback)함수를 작성한다.
4) 위 1,2,3에서 생성된 코드를 컴파일 하고 실행한다.

제일 먼저, 시뮬링크를 이용하여 필요한 SDR을 모델링 한다. 시뮬링크에서 제공하는 라이브러리 만으로도 다양한 웨이브폼 어플리케이션을 구성할 수 있다. 필요한 라이브러리를 열고 어떤 컴포넌트들로 나눌 것인지 결정해야 한다. 컴포넌트들로 나눌 때에도 여러 가지 기준이 사용될 수 있는데, sample rate나 symbol rate, frame rate와 같은 클럭 도메인으로 나눈 다음에 프로세싱 블록에 따라 다시 나누는 방법도 많이 사용된다[9]. 주의해야 할 점은, 시뮬링크 라이브러리에서는 디자인된 모델이 어떤 타입의 입력과 출력 값을 가지는지 알지 못하기 때문에, 웨이브폼 컴포넌트 개발을 쉽게 하기 위해서 모든 데이터 형을 복소수 형태로 지정하여야 한다는 것이다. 그런 다음, 분할된 컴포넌트들의 코드를 생성한다. 코드를 생성하기 전에 컴포넌트를 나누는 것이 코드 생성 후에 코드를 컴포넌트 별로 분할하는 것보다 빠르고 정확하다. 이렇게 자동 생성된 코드는 그림 1과 같이 여러 헤더 파일과 라이브러리를 포함하고 있으며 신호 처리를 위한 함수를 가지고 있다.

시뮬링크로부터 비즈니스 로직에 해당하는 코드를 얻었으므로, 이제는 사용자로부터 웨이브폼 컴포넌트의 이름, 포트와 데이터의 종류와 이름, 필요한 속성과 그 값, 등을 입력 받아 SCA 표준 인터페이스를 위한 코드를 생성한다. 이 코드에 비즈니스 로직만 추가하면 되는데, 이를 위해서 SCA 표준 인터페이스의 데이터와 시뮬링크 코드가 사용하는 입출력 데이터의 변환을 위한 콜백(callback) 함수를 작성한다. 이 콜백 함수에서는 SCA 표준 인터페이스의 포트로 들어오는 데이터를 시뮬링크 코드에서 필요한 데이터 형태로 변환하여 전달하며, 계산된 결과 데이터를 읽어와서 다시 SCA 표준 인터페이스 포트로 내보내는 일을 한다. 이 과정은 그림 2를 통해서 쉽게 이해가 가능하다. 시뮬링크에서 생성

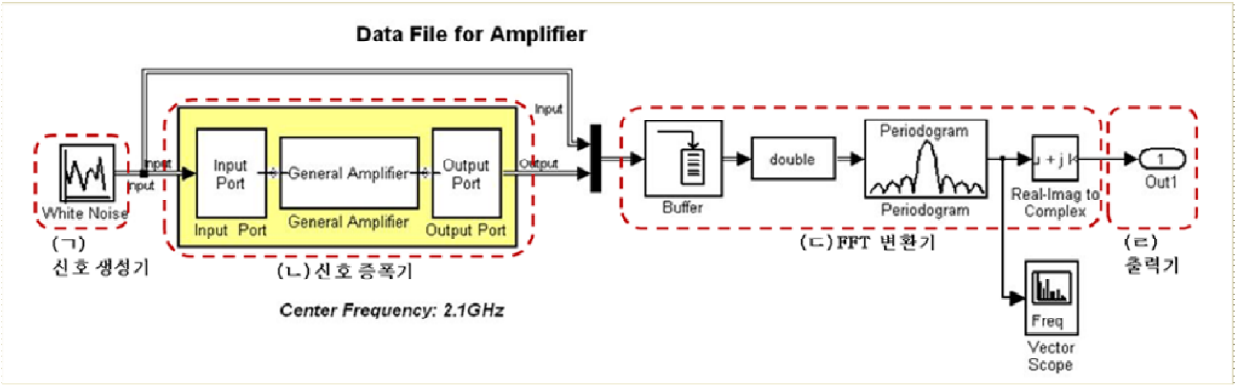


그림 3. 시뮬링크를 이용한 모델 기반 디자인 예제

된 코드는 공통적으로 grt_main() 함수를 통하여 동작한다. grt_main()에서는 모델의 동작에 필요한 초기화작업을 한 이후에, rt_OneStep()함수를 반복적으로 호출하여 1회 실행에 샘플 하나를 처리하기 때문에, 바로 이 rt_OneStep() 함수 실행 전후로 우리가 작성한 콜백 함수인 OneStepBeforeNotify()와 OneStepAfterNotify()를 추가한다. OneStepBeforeNotify() 함수에서는 SCA 인터페이스로부터 받은 데이터를 creal_T라는 시뮬링크에서 사용하는 복소수 데이터 구조를 이용하여 저장한 후에, rt_OneStep()에서 사용할 데이터에 이 값을 써준다. rt_OneStep()의 계산 결과는 OneStepAfterNotify()함수를 통해서 SCA 표준 인터페이스의 데이터형으로 변환되어 전송된다. 복소수 데이터는 실수와 허수를 각각의 데이터로써 포트를 통하여 전달한다. 이 콜백 함수 역시, 기본적인 코드는 자동생성이 가능하며, 모델에 따라 샘플 기반의 통신에서 프레임 기반 통신으로의 변환과 같은 코드의 추가 작성이 필요하다. 마지막으로, 이렇게 만들어진 코드들을 컴파일하여 웨이브폼 컴포넌트를 만들면, SAD파일을 통해서 SCA 시스템에서 웨이브폼 어플리케이션을 실행시킬 수 있다.

2.3 하드웨어 웨이브폼 컴포넌트 자동생성

FPGA와 같은 하드웨어를 이용하는 웨이브폼 컴포넌트를 만드는 과정도 소프트웨어 컴포넌트를 만드는 과정과 크게 다르지 않다. 어댑터 디자인을 이용해서 만들면 플랫폼 독립적인 모델을 이용하여 비즈니스 로직을 제외한 코드를 자동 생성한 다음에, 하드웨어를 다루기 위한 코드를 넣어주면 된다. 그리고, CPU가 주변장치와 통신하기 위한 방법 중에 memory-mapped I/O 방식을 이용하면, 어댑터에서 하드웨어로 데이터를 전달하는 코드의 작성이 쉬워진다. Memory-mapped I/O란 CPU가 사용하는 물리 주소 공간의 일정 영역을 주변장치와 통신을 위해서 사용하여, 해당 장치에게 할당된 물리 주소에 데이터를 읽거나 써서 데이터를 전달하는 방식이다. 따라서, 접근하고자 하는 하드웨어에게 할당된 주소 정보를 가지고 있으면, 사용자로부터 메모리 주소, 데이터 크기 등을 입력 받아 하드웨어를 컨트롤 하는 코드를 만들 수 있다.

3. 구현 및 결과

우리는 3장에서 제시한 방법을 이용하여 화이트 노이즈를 앰프를 통해 증폭한 다음 증폭이 잘 되었는지 FFT(Fast Fourier Transform)를 통해서 변환한 결과를 보여주는 웨이브폼 어플리케이션을 만들고, 그 실행 결과를 시뮬링크의 시뮬레이션 결과와 비교하였다.

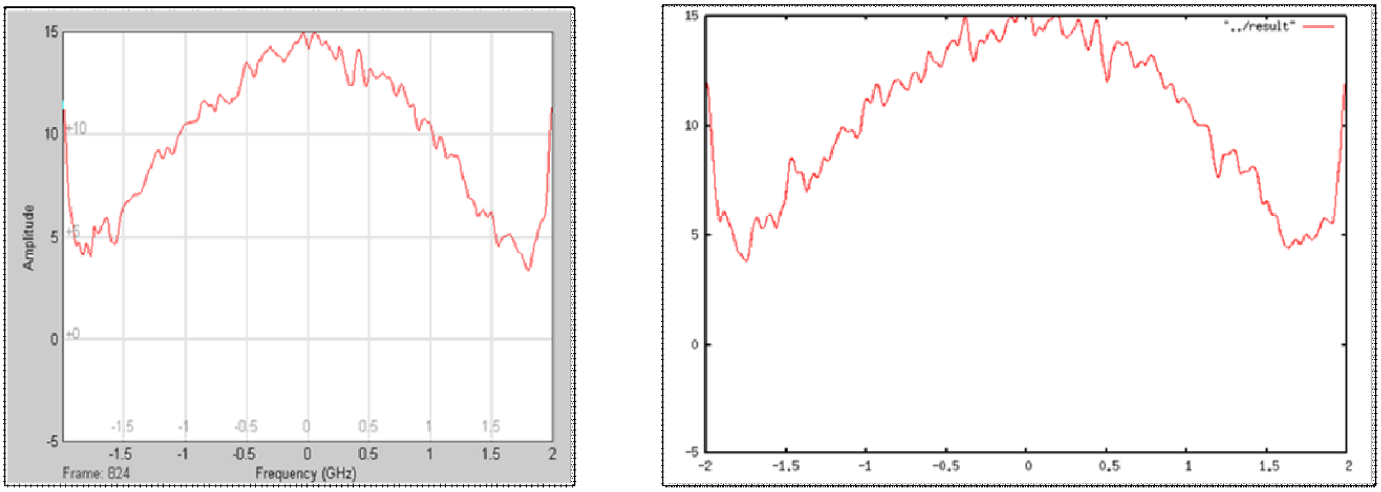
3.1 구현 플랫폼

범용 프로세서로는 400MHz의 PXA255를 사용하였다. PXA255는 인텔의 XScale 패밀리로, ARMv5TE 아키텍쳐를 사용하고 있다. 인스트럭션 캐시와 데이터 캐시로 각각 32KB를 가지고 있으며, MMU를 내장하고 있다. 그 외에도 128MB의 SDRAM, 32MB 플레시 메모리가 사용되었다. SCA 시스템은 리눅스 기반의 OSSIE(Open Source SCA Implementation – Embedded)[10]를 기반으로 구현되었으며, 임베디드 시스템에서의 성능 향상을 위해서 SCA의 Core Framework 내에서의 코바 통신을 제한하는 방법을 적용하였다[11].

3.2 시뮬링크 모델링과 웨이브폼 컴포넌트의 생성

디자인할 모델을 각 컴포넌트의 작업 로드와 컴포넌트간의 연결을 고려하여 모델을 나누어준다. 여기에서는 그림 3에서 보는 것과 같이, 신호를 생성하는 (ㄱ)신호 생성기와 받은 신호를 증폭해주는 (ㄴ)신호 증폭기, FFT 변환을 위한 (ㄷ)FFT 변환기, 그리고 그 결과를 그래프로 그려주는 (ㄹ)그래프 출력기와 같이, 총 4개의 컴포넌트로 모델을 나눈 후, 각각에 대하여 코드를 생성하였다. Vector Scope 모듈은 시뮬레이션을 위한 것으로, 결과 그래프를 위해 추가되었으므로, 코드 생성에는 제외하였다. 이 과정에서 생성된 각 컴포넌트의 코드를 3.2장에서와 같이 수정한 라이브러리와 함께 컴파일한다. 생성된 코드는 윈도우 기반의 C 라이브러리를 사용하는 소스이므로, 리눅스 기반의 SCA에서 테스트하기 위해서는 리눅스 기반의 C라이브러리 형태로 포팅 해야만 했다.

각 컴포넌트의 데이터형은 복소수형으로 지정하였



(ㄱ) 시뮬링크 모델의 시뮬레이션 결과 (ㄴ) SCA 웨이브폼의 실행 결과

그림 4. 신호 확장기 예제 결과 그래프

으며, 시뮬링크에서 생성된 코드는 샘플 기반 모델이므로, SCA 시스템에서 실행하기 위해서는 프레임 기반 모델로의 변환이 필요했다. 한 프레임이 512개의 샘플을 처리한다는 가정 아래, FFT 변환기에서는 512개의 샘플을 버퍼링 한 후에 그래프 출력기로 데이터를 넘겨주는 작업을 OneStepAfterNotify()함수에 추가하였다.

3.3 시뮬레이션 및 실험 결과

위에서 만들어진 웨이브폼 어플리케이션이 제대로 동작하는지 확인하기 위해서, 그래프 출력기에서는 최종적으로 받은 데이터를 파일에 저장하였고, 호스트 머신에서 이 파일을 이용하여 그래프를 그려 시뮬링크의 시뮬레이터에서 보여주는 그래프와 비교하였다. 그림 4에서 볼 수 있듯이, 시뮬링크 모델의 시뮬레이션 상의 그래프와 SCA 시스템 상의 웨이브폼 어플리케이션에서 받은 데이터로 그린 그래프 모양이 같은 것으로, SCA 플랫폼에서 실행한 웨이브폼 어플리케이션이 제대로 동작했음을 알 수 있다.

4. 결론

SCA는 SDR을 위한 표준 인터페이스를 제공하는 기반 시스템으로, 유연성과 상호 운용성 등을 높이기 위해서 제안되었다. 하지만 SCA 시스템을 위한 웨이브폼 어플리케이션을 개발하기 위해서는 SCA의 내부 구조뿐만이 아니라, SCA 표준에서 규정하고 있는 CORBA에 대한 지식도 필요하다. 더군다나, 내장형 시스템과 같이 범용 프로세서의 계산 능력이 낮은 시스템에서는 FPGA와 같은 부가적인 하드웨어가 함께 사용되어야 하는데, 이 때에는 하드웨어 소프트웨어 파티셔닝 이슈까지 생겨나게 된다. 따라서 본 논문에서는 SCA 플랫폼에서의 웨이브폼 어플리케이션의 프로토타입을 빠르게 생성하는 방법을 제안하였다. 시뮬링크를 이용하여 필요한 모델을 디자인하고 코드를 생성한 다음에, 코드에 대한 이해 없이도 콜백 함수를 추가하는 것만으로 SCA에서 바로 사용할 수 있게 하였다. 그리고, 모델을 이용해서 디자인을 하기 때문에, 개발 속도를 높일 수 있을 뿐만 아니라 검증까지도 용이 하게 할 수 있다.

그러므로, 이러한 방법을 사용하면 SCA에 대한 이해나 라디오 모듈에 대한 깊은 이해 없이도 필요한 웨이브폼 어플리케이션을 쉽고 빠르게 개발할 수가 있고, 이를 바탕으로 다양한 하드웨어 플랫폼 상에서의 하드웨어 소프트웨어 파티셔닝을 위한 디자인 공간 탐색도 효과적으로 수행할 수도 있다.

5. 향후 연구

본 연구는 SCA 시스템에서 바로 실행 가능한 웨이브폼 어플리케이션의 자동 생성 기법에 대한 것으로, 소프트웨어 모듈의 구현 방법에 대해서 자세히 다루었다. 하지만 3.3장에서 언급한 것과 같이, 하드웨어를 위한 웨이브폼 컴포넌트의 코드 생성에 대한 가능성도 살펴보았고, 현재 간단한 VHDL 코드를 소프트웨어 모듈에서 제어하는데 성공하였다. 이를 바탕으로 우리는 하드웨어를 위한 웨이브폼 컴포넌트를 자동 생성하는 방법에 대한 연구를 계속 진행하고자 한다.

6. 참고 문헌

[1] JTRS Joint Program Office, “Software Communications Architecture Specification”, Version 2.2, November 17,2001.

[2] C. Serra, B. Sourdillat, E. Nicollet, “Waveform Portability – Return of Experience on Implementing the SCA”, SDR’05 Technical Conference, 2005.

[3] JTRS Joint Program Office, “Specialized

Hardware Supplement to the Software Communication Architecture (SCA) Specifications", JTRS-5000 SP, V3.0, 27 August 2004.
[4] JTRS Joint Program Office, "Extension for component portability for specialized hardware to the JTRS Software Communication Architecture (SCA) Specification, V3.1x, 20 January 2005.
[5] Mark Hemeling, "Component-Based support for FPGA and DSP", SDR'06 Technical Conference, 2006.
[6] http://www.mathworks.com/products/simulink/
[7] Dr. Don Orofino, "Accelerating Software-Defined Radio Development with MATLAB and Simulink", Tutorial of SDR'03 Technical Conference, 2003.
[8] Zoran K. and Alex Rodriguez, "SDR-Targeted Design Flow: From Executable Specification To Signal Processing Subsystem Code Generation and SCA-Focused Tool Integration", SDR'05 Technical Conference, 2005.
[9] J. Neel, P. Robert, J. Reed, "A Formal methodology for estimating the feasible processor solution space for a software radio", SDR Forum Technical Conference 2005.
[10] http://ossie.mprg.org/
[11] 최민, 김한준, 김재극, 맹승렬, "임베디드 SDR 단말을 위한 SCA Core Framework 최적화 기법", 2007년도 대한전자공학회 소사이어티 추계학술대회 제 30권 제2호.