프로토타이핑 기술을 적용한 임베디드 시스템 가상 시뮬레이션†

선문대학교 정영진 · 이정배 · 이영란 · 권진백 · 임기욱
한국외국어대학교 조상영

1. 소 개

컴퓨터 기술의 발전에 따라 임베디드 소프트웨어 개발은 범용 시스템 소프트웨어 개발과 달리 저전력, 저메모리를 사용하여 제품 단가, 실시간성 등의 요구사항을 맞춰야 하는 매우 엄격한 제약 조건을 가진다. 뿐만 아니라 필수적으로 하드웨어와의 동시 설계 및 역할 분담 과정을 거쳐야 하며, 이로 인해 일반 소프트웨어 개발 방법론과는 차별화된 개발 절차를 요구한다[1]. 그리고 임베디드 소프트웨어는 범용 마이크로프로세서 및 IP(Intellectual Property)산업의 발전으로 임베디드 시스템의 요구에 따라 특수한 기능을 수행하는 설계로 산업 전반으로의 적용이 확대되고 있다. 그러나 점차 복잡도가 증가하는 임베디드 시스템 설계에서 크게는 요구 분석, 하드웨어/소프트웨어 역할 분담 및 동시 설계, 그리고 재통합의 과정을 필수적으로 포함해야 하며, 이 과정 속에서 하드웨어/소프트웨어에 대한 설계 및 구현, 그리고 제품의 기능 검증과 디버깅 작업까지 소모되는 시간과 비용의 부담은 시스템의 초기 개발 비용을 증가시킴으로써 제품의 경쟁력을 저하시킨다.

따라서 이러한 개발과정은 개발 프로젝트에 대한 생명 주기 모델 선택, 응용 분야, 혹은 개발 환경에 따라 다양하게 달라질 수 있으며, 개발자들에게는 문제에 대한 정확한 이해와 기술 변화에 대한 직시, 그리고 엄정한 기술 분석에 의한 도구 및 개발 환경 선정 등 현명한 전문적 판단이 요구된다[1].

본 고에서는 임베디드 소프트웨어 개발시의 차별성과 특수성을 반영한 프로토타이핑 개발 방법론에 대해서 설명하고, 가상 프로토타이핑 개발 방법론을 이용하여 임베디드 시스템을 시뮬레이션하는 방법에 대해서도 논하고자 한다.

2. 임베디드 시스템

임베디드 시스템이란 하드웨어와 소프트웨어로 하나의 시스템을 구성하고 있으며 시스템 내에서 특정 기능을 수행하기 위한 서브시스템으로, 마이크로프로세서에 탑재되어 수행되는 시스템을 말한다. 일반적으로 임베디드 시스템은 하나의 마이크로 컴퓨터의 ROM상의 소프트웨어로 구성되며, 전원이 들어오면 바로 특수한 목적을 위한 어플리케이션 프로그램이 실행되고, 전원이 공급되는 한 실행을 유지한다. 산업 및 군사용 제어기기, 로켓, 인공위성, 비행기, 의학용 기계등과 같은 특수 분야에서 자동판매기, 세탁기, 장난감과 같은 정보 가전을 포함한 거의 모든 기계를 임베디드 시스템이라 해도 과언이 아니다.

최근 정보 통신 선진국을 중심으로 급격히 확산되고 있는 편재형 컴퓨팅(pervasive or ubiquitous computing) 서비스 환경과 마이크로프로세서의 발전으로 스마트 핸드폰, PDA(Personal Digital Assistant), HDTV (High-Definition Television), 정보 가전 등 제한적 컴퓨터 자원을 가지면서 고성능, 고품질의 서비스를 요구하는 임베디드 시스템 개발에 대한 요구가 급증하고 있다. 또한 임베디드 시스템은 IT기술이 사회와 경제적 변화에 따라 다양한 욕구를 충족시키기 위해 복합화 및 광대역화 되고 있으며 이러한 추세를 통합하는 시장의 중심에 있다.

2.1 임베디드 소프트웨어 개발 요구 사항

임베디드 소프트웨어는 임베디드 시스템의 응용분야가 전 산업분야로 다양해지면서, 마이크로프로세서에 내장되어 복잡하고 다양한 기능을 수행하게 되었다. 특수한 목적으로 개발되는 다양한 임베디드 시스템에 한정된 최소한의 자원이 탑재되는데, 임베디드 소프트웨어는 이러한 최소한의 자원으로 실행될 소프트웨어이다. 따라서 임베디드 소프트웨어는 저전력, 저메모리, 실시간성 등의 엄격한 제약조건이 따른다.

또한 임베디드 소프트웨어 개발과정은 구현 목표 시스템에 따라 정확한 이해와 기술 변화에 대한 고난이도의 지식 및 기술 분석에 의한 개발 환경 선정 등의

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구 결과로 수행되었음.

판단이 요구된다. 이러한 과정을 통하여 개발된 시스템의 검증과 디버깅 작업 또한 필수적이며 프로젝트의 성격에 따라 다양한 방법론이 제시 될 수 있으며, 개발자의 정확한 이해와 기술 변화에 대한 인식 및 개발환경 선정 등의 광범위한 판단이 요구된다.

임베디드 시스템을 개발하기 위해 일반적인 소프트웨어 공학 이론을 적용하는 것은 의미가 없으며, 목표 시스템에 맞는 최적화된 개발 방법론을 제시하는 것이 현실적이라고 볼 수 있다. 이를 위하여 임베디드 시스템 개발도구의 개발은 여러 요건 중 특히, 아래와 같은 요건을 충족해야 한다.

① 사용자의 요구 분석 및 개발 시스템에 대한 성능 예측

사용자의 요구를 면밀히 분석하여 제품의 외형 및 성능을 결정한다. 특히 임베디드 시스템은 사용자의 요구분석과 성능예측에 따라 성패결과가 결정되며, 사용자들의 요구를 정확하게 도출하고, 내용의 구체화가 큰 과제이다. 하지만 사용자 입장에서 개발 시스템에 관한 기술적이고 경직된 문서는 거부감을 불러 일으켜 정확한 요구를 추출하기 어려움이 있으며, 사용자의 요구를 구체적으로 분석하고, 거부감을 주지 않는 소비자의 요구 추출방법으로 가상 프로토타이핑 기법이 활용된다.

② 응용분야 적응형 개발 방법론 요구

임베디드 시스템은 전 산업분야에 적용되면서 응용분야가 다양한 제약조건을 가지고 있다. 따라서 일반적인 소프트웨어 공학적 접근법을 임베디드 소프트웨어 개발에 적용시키는 것은 위험이 따르며, 응용분야의 특수성의 정확한 반영과 최적화된 개발방법의 적용이 제품개발의 성패를 좌우한다고 할 수 있다. 이러한 접근방법으로 적응형(Domain-Specific) 개발방법은 임베디드 소프트웨어 개발 분야의 연구방법으로 구체적 방법론에 대한 연구가 활발히 이루어지고 있다.

③ 재사용 가능성과 시장 진입 시점(Time-to-Market)을 단축

임베디드 시스템은 안정성과 고난도의 개발 기술을 요구하지만, 제품 출시까지 개발 기간이 짧은 최신 기술력을 적용한 첨단제품이 대부분이다. 따라서 상품 개발기간의 단축을 위한 방법론의 연구가 활발히 진행되고 있다. 결과적으로 단기간에 안정적 신제품의 개발을 위하여 임베디드 시스템 개발도구를 이용한 재사용 가능한 컴포넌트 기술의 이용이 효율적이라 할 수 있다.

④ 개발자들간의 하드웨어/소프트웨어 동시 설계를 위한 이해 요구

임베디드 소프트웨어는 제어 및 기계공학, 회로설계 분야의 전문 개발자와 소프트웨어 개발자가 공동작업을 통하여 프로젝트가 진행된다. 소프트웨어 개발자의 주 임무는 하드웨어 개발자가 제시한 제어, 신호처리와 같은 알고리즘에 대한 구현을 한다. 따라서 소프트웨어 개발자는 기본 이론에 이해가 부족하고, 하드웨어 엔지니어는 운영체제, 디바이스 드라이버, 통신 프로토콜 특성 등의 소프트웨어 플랫폼에 대한 이해부족으로 프로젝트에 부정적 영향을 초래할 수 있다.

⑤ 통합설계와 개발환경 요구

임베디드 시스템은 특수한 목적의 제한된 수행만을 위하여 제작되는 시스템으로 기술 특성상 하드웨어가 개발되고, 개발된 하드웨어를 기반으로 한 소프트웨어 개발이 진행된다. 이러한 문제로 인한 인력의 낭비와 개발비용 증가의 부담을 해결하기 위하여 최근 하드웨어의 주요기능을 충실히 수행하면서 쉽게 제작이 가능한 실물 프로토타이핑에 관한 연구가 활발히 진행되고 있다.

3. 임베디드 시스템 프로토타이핑

앞에서 언급한 바와 같이 임베디드 시스템 개발 프로젝트는 단순 범용 소프트웨어 개발 프로젝트에 비해 고난도의 기술을 연구할 뿐만 아니라, 개발 팀원들 또한 하드웨어와 소프트웨어에 대한 동시 이해 능력을 보유해야만 성공적인 프로젝트 수행이 가능하다. 특히, 범용 소프트웨어 개발 프로젝트에서는 필요에 따라 쉽게 단계적 개선 및 전 과정으로의 피드백(feed back)이 가능하나, 임베디드 시스템에서는 하드웨어와 소프트웨어의 동시 설계 및 동시 진행으로 인해, 전 과정으로 피드백이 불가능하거나 많은 위험요소를 내포할 수 있다. 이러한 위험 요소를 최소화하기 위해서 다양한 임베디드 소프트웨어 개발 경험을 바탕으로 한 프로토타이핑 개발 방법론이 필요하다.

3.1 가상 프로토타이핑

가상 프로토타이핑(Virtual Prototyping)은 임베디드 시스템의 개발 초기 단계에서 사용자의 요구를 정확히 반영하고, 기능적, 비기능적 요구를 효과적으로 구현하기 위하여 고안된 시스템 제약조건 추출방법이다. 또한 컴퓨터 기술을 이용하여 제품의 프로토타입을 제공하고, 인터페이스가 복잡한 제품 설계 및 실제 제품으로 테스트를 하는 것과 같은 현실감을 제공하는 컴퓨터상의 데이터를 가상 프로토타이핑이라고 한다[2,3]. 컴퓨터 프로그램으로 프로토타입 모델을 작성하고 시연하기 때문에 사용자의 선호에 따라 가상 모형의 모양을 변경하거나 기능을 새로이 추가하는 등의 변화를 손쉽게 반영할 수 있다. 이는 시제품의 검증을

위한 실물 프로토타이핑 방법보다 전체 개발비용 절감과 생산 주기 등의 문제에 편리함을 제공한다. 따라서, 개발 초기에 사용자의 요구를 효과적으로 수용할 수 있는 개발 방법론으로 평가받고 있다. 현재 다양한 목적으로 개발되는 각각의 임베디드 시스템을 내장하여 출시되는 제품은 최첨단의 기술과 기능을 요구하는 디지털 산업 분야가 대부분이다. 이러한 제품 개발시 가상 프로토타이핑 개발방법을 이용하여 복잡한 인터페이스를 가진 제품의 개발 시간이 단축되고, 따라서 출시된 제품의 수명 주기도 점차 짧아지고 있다.

가상 프로토타이핑 개발 방법은 실물 프로토타이핑으로 제품을 구현하고, 테스트 하였을 경우와 비교하면 아래와 같은 장점을 제공한다.

① 컴퓨터 그래픽스 모델링으로 디자인의 변경 용이
- 실시간으로 변경된 디자인을 관찰함으로써 소비자에게 친근감이 가는 디자인으로 변경이 가능하다.

② 기능이 정형 명세로 표현되어 새로운 기능의 첨가 및 수정이 용이
- 시뮬레이션 가능한 정형명세 언어로 설계상의 오류를 가상 프로토타입 단계에서 검증함으로써 신뢰도 있는 제품의 생산과 제품의 생명주기를 단축할 수 있다.

③ 제품 개발 및 테스트 시간과 비용 절감 효과
- 실물 프로토타입 방법은 제품의 외형 및 기능 변경 시 처음 단계부터 다시 시작하지만, 가상 프로토타이핑은 컴퓨터상의 간단한 데이터 수정으로 즉시 제품 테스트와 수정이 가능하다.

가상 프로토타이핑을 지원하는 대표적인 임베디드 시스템 개발 도구로는 e-sim사의 RapidPLUS[4], MSC Software사의 MSC.SimManager[5], Opal-RT Tech. 사의RT-LAB[6]등이 있으며, 최근에는 UML 등 통합 개발 방법론과 결합된 형태의 CASE (Computer-Aided Software Engineering) 형태로 발전하고 있다. i-Logix사의 Rhapsody[7], 아이너스 기술의 PlayMo[8]등이 대표적인 통합 개발 환경이다.

3.2 도메인 적응형 프로토타이핑

도메인 적응형 프로토타이핑은 나선형 소프트웨어 생명주기 모델에 근거한 대표적인 개발 방법으로 소프트웨어 개발에서의 위험요소를 최소화하기 위해서 개발하려는 시스템의 주요기능을 동시에 개발하지 않고, 주요기능에 우선순위를 설정하여 우선순위에 따라 소규모로 핵심 기능들을 구현하고 기능 및 성능에 대한 정확한 평가 후에 추가 기능을 점차적으로 구현하는 개발방법론이다. 이러한 개발방법론을 기반으로 복잡하고 다양한 임베디드 시스템에서 요구되는 도메인 특성을 소프트웨어 구조에 첨가하여, 다양한 요구 조건과 제약 조건을 가진 임베디드 시스템 개발에 적합한 소프트웨어 개발 구조로 활용 할 수 있다. 이 방법은 폭넓은 연구를 통하여 개발 및 실용화단계에 있으며 정보가전 분야의 Philips의 Koala, Arcticus사에서 개발한 Time-Triggered 프로토콜을 기반으로 한 경성 실시간 시스템 개발을 위한 Rubus, 유럽의 GN&C에서 개발 중인 DSSA 프로젝트 등의 개발 사례가 대표적이다[9].

3.3 실물 프로토타이핑

임베디드 시스템에 공통적으로 활용되는 표준 센서와 엑츄에이터, 그리고 쉽게 조립할 수 있고 재사용 가능한 부품 및 블록들을 활용하여 실제 개발하고자 하는 실물과 유사한 모양과 기능을 가진 실물 프로토타입을 개발하여 소프트웨어 개발과정에 접목하면 다음과 같은 이득을 얻을 수 있다. 우선, 임베디드 시스템 특성상 공동 작업을 하게 되는 다양한 전공분야의 개발자들이 서로 간에 이해증진을 위한 도구로 활용할 수 있으며, 보다 효과적인 문제 해결 방법을 새로이 고안해 내기 위한 도구로써도 활용이 가능하다. 다음으로, 임베디드 시스템은 하드웨어 개발과 소프트웨어 개발이 함께 되어야 하나 종래의 개발 방법론에서는 하드웨어 개발이 완료된 후에야 소프트웨어 개발이 진행되므로 개발인력에 대해 효과적인 활용이 불가능하고, 개발 기간이 길어지며, 소프트웨어 개발 시 발견되는 하드웨어적 결함에 대한 대처 능력이 떨어지는 문제점이 상존하고 있다. 임베디드 시스템의 특성상 개발자들 간의 공동작업에서 다양한 전공분야에 대한 개발자들의 이해가 필수적이다. 따라서 보다 효과적인 문제 해결 방법으로 실물과 유사한 모양과 기능을 가진 프로토타입을 만들어 서로간의 이해를 증진시킬 수 있다. 실물 프로토타입을 이용하면 개발 중간에서부터 하드웨어 개발자와 소프트웨어 개발자가 동시작업을 진행하면서, 소프트웨어 적용 시 발생하는 문제점들을 바로 하드웨어 개발자에게 전달함으로써 최종 제품에 대한 품질을 개발 단계에서부터 보장할 수 있게 된다. 최근에 이러한 실물 프로토타이핑 개념의 중요성이 강조되면서 많은 개발 도구들이 연구 개발되고 있다. 대표적으로 UC Berkeley의 Mica Mote&TinyOS, EU "Disappearing Computer Initiative"의 Smart-Its, 경북대학교 Real-Time Systems 연구팀에서 개발한

Embedded System Prototyping Suit(ESPS)[10] 등이 있다.

4. 임베디드 시스템 가상 시뮬레이션

4.1 가상 프로토타이핑 개발 방법

최근 엔지니어들은 컴퓨터 기술이 발달함에 따라, 복잡한 시스템 개발 시 컴퓨터를 이용하고 있으며, 컴퓨터 기술의 발달 및 응용분야의 확대로 프로토타이핑의 장점을 극대화 할 수 있게 된 것으로 가상 프로토타이핑이 있다.

기존의 실물 프로토타입은 제품의 외형 변경 및 기능 추가 시 새로운 프로토타입을 만들어야 하지만, 가상 프로토타입은 컴퓨터에 데이터 형태로 존재하면서 GUI 방식의 모델링 방법을 이용함으로써 변경된 디자인을 손쉽게 확인할 수 있으며, 기능이 정형명세로 표현되어 외형과 연결되어 새로운 기능의 첨가가 쉽다. 이러한 가상 프로토타이핑은 제품 출시 전 하드웨어나 소프트웨어 분야의 완벽한 시뮬레이션을 위하여 컴퓨터에 데이터의 형태로 존재하면서, 실물 프로토타입을 대체할 수 있는 기술이다. 또한 가전 제품 개발이나 카오디오 설계, 가상 공장 구축 시스템, 공항 관제 시스템의 설계와 검증 분야에 이용되고 있으며[2] 추후 배부분의 생산 분야에서 가상 프로토타이핑을 통한 제품 개발로 생산성 향상을 기대할 수 있다. 가상 프로토타이핑 개발 방법으로는 다음과 같이 사용자 중심의 인터페이스 개발과 플랫폼 기반의 개발 방법이 있다.

4.1.1 사용자 인터페이스 중심의 개발

현대사회는 일상생활 속에서 원활한 의사소통을 위한 관계와 인터페이스의 요구가 전반적으로 요구되고 있다. 특히 컴퓨터가 직접 또는 간접적으로 우리 생활 속에 관련되어 있는 현재 상황에서 컴퓨터와 인간의 상호 작용은 무엇보다 중요한 문제로 대두되었다. 이를 위한 연구의 한 영역으로 인간-컴퓨터 상호작용(HCI : Hunman-Computer Interaction) 분야가 탄생하였다. HCI 시스템에서 두 개체는 각 독립된 개체를 형성하지만, 그들을 하나의 통합된 시스템으로 구축 및 개발하기 위해서 사용자 인터페이스(User-Interfaces)의 필요성이 대두되었다. 사용자 인터페이스는 인간과 시스템 간의 공통점과 사용자와 각각의 시스템 사이의 정보채널로 표현되어 보다 사용하기 편리한 시스템을 만들기 위하여 사용자의 인지적 측면에서 디자인하고 사용의 편리성을 평가하는 개념으로 정의된다.

사용자 인터페이스를 중심으로 한 컴포넌트 기반의 가상 프로토타이핑은 CBD(Component-Based Development)의 개발방법론에 기초하고 있으며, 시스템의 수정과 업그레이드가 빈번한 임베디드 시스템 개발 시 기능의 첨가 및 수정이 간단하며, 컴포넌트화 되어 재사용이 가능하다. 많은 사용자 인터페이스기반의 개발도구가 시뮬레이터와 디버거, 프로젝트 관리자, 컴파일 기능을 포함하고 소스 내비게이션 기능을 통합함으로써 다양한 플랫폼을 지원하고 전문적인 프로그래밍 언어에 능숙하지 않아도 편리하게 시스템을 구현할 수 있다.

인터페이스 기술과 가상 프로토타이핑 기술을 통합한 개발 방법의 특성으로 제품의 수명주기가 짧고, 컴포넌트의 재사용을 통하여 기존의 기능에서 일부의 업그레이드가 잦은 임베디드 시스템의 초기 개발비용을 최소화하는 효과가 있다. 또한 하드웨어와 소프트웨어 동시 시뮬레이터 및 디버거가 통합되어 제공됨으로써 제품 개발 시 기존의 절차지향적 개발 방식보다 편리한 개발 모형을 제공한다. 이러한 통합 개발 환경은 사용자 혹은 개발 의뢰자가 현실감 있는 제품 또는 환경에서 최적화된 플랫폼을 가상의 기반으로 구현 가능한 멀티 플랫폼과 개발환경을 제공하고, 개발과정에서 생성된 정보는 실제 시스템 구현 시 참조가 가능한 문서로 저장된다. 이러한 사용자 인터페이스 중심 개발도구는 프로토타이핑을 컴퓨터로 하는 개발에서 사용범위가 급속히 확대되고 있으며, 실시간 임베디드 시스템 설계를 위한 모델링 도구로서 엔지니어 및 소프트웨어 개발자와 테스트 엔지니어에게 완벽한 개발과 검증을 위한 환경을 제공한다. 또한 시스템에 적합한 코드 생성과 테스트를 수행하며 임베디드 소프트웨어 개발과정을 표준화함으로써 시스템 및 소프트웨어 품질의 획기적 향상을 가져온다. 사용자 인터페이스를 기반으로 하는 임베디드 소프트웨어 개발도구로는 RapidPLUS, Rhapsody와 ASASDAL[11]등이 있다.

가상 프로토타이핑 개발 도구 중 RapidPLUS는 사용자 인터페이스 기반의 개발도구로서 모든 특징을 제공하며, 현실감 있는 개발 환경을 제공하며, 어떤 개발 단계에서도 테스트를 통한 오류의 즉시적 수정이 가능하다. 또한 시뮬레이션 기술과 가상 개발(Virtual Product) 환경의 개념을 기반으로 임베디드 제품의 기획에서 출시 후의 고객지원의 전 과정을 동시에 진행하며, 실물과 가상 프로토타이핑의 연동시 데이터 교환을 위하여 요구되는 ActiveX와 같은 외부 객체의 반입을 허용한다. 그리고 Rhapsody는 실시간 임베디드 시스템 설계를 모델링 방식으로 제공하며 UML과 MDD(Model-Driven Development) 환경을 제공하며 개발자에게 UML을 이용한 임베디드 시스템의 설계를 그래픽으로 정의하고 시뮬레이션과 자동 검증, 코

드 생성과 테스트를 수행한다. 이와 같은 개발 도구들은 가상 프로토타이핑의 장점을 이용한 시스템 설계와 개발 과정을 표준화함으로써 시스템 및 소프트웨어 품질을 획기적으로 향상 시킬 수 있다.

마지막으로 ASADAL은 실시간 시스템의 분석 및 설계를 위한 다양한 정형 명세화 기능을 제공하며 시뮬레이션을 통해서 검증되어진 정형 명세의 코드 생성을 자동으로 수행하는 프로토타이핑 설계도구로 강력한 기능을 제공한다.

4.1.2 플랫폼 기반의 개발도구

플랫폼 기반의 개발도구는 주로 표준이나 칩 사양이 변경될 여지가 많은 부분은 소프트웨어로 구현되고 이에 맞는 성능을 가진 프로세서가 선정이 되며, 최적화된 운영체제가 제공된다. 이러한 플랫폼 기반의 임베디드 시스템 개발 도구는 SoC(System on Chip)개발에 적용이 급속도로 확대되고 있으며, 임베디드 시스템 응용프로그램의 검증을 위하여 상위수준의 언어(예, C, SystemC)로 기술된다. 또한 목표 플랫폼에 따라 응용프로그램을 하드웨어와 소프트웨어로 구현될 부분을 구분하여 설계가 가능하다. 이러한 플랫폼은 사용자에 의하여 재정의 단계에서 좀더 구체적으로 설계 목적에 맞게 변경되고 최적화된다.

즉, 클럭의 속도 혹은 통신프로토콜이 결정되어 플랫폼 라이브러리에 등록되고, 등록된 플랫폼은 프로세서와 하드웨어 모듈, IP등과 통합되어 하드웨어와 소프트웨어의 통합 설계와 통합 시뮬레이션이 가능하다. 또한 시뮬레이션 후 FPGA 혹은 전용 에뮬레이터를 통한 최종 검증이 수행되는 모든 과정의 구현이 가능하다.

플랫폼 기반의 임베디드 시스템 개발 도구 역시 객체지향적 개발방법론을 기반으로 IP의 재사용과 같이 시스템의 자원을 활용하고, IP간의 연결과 버전 문제를 조율함으로써 어플리케이션 입장에서 시스템을 모니터링 할 수 있는 능력과 시스템 구현 시 사용할 IP 선정 능력을 동시에 지닐 수 있다. 이러한 플랫폼 기반의 개발도구는 애플리케이션의 변화에 대한 수용 가능한 일반적 구조 여부가 포인트가 되며, 주로 세 분야의 세부 플랫폼으로 구성되어 검증 영역을 포함한다.

① 하드웨어 플랫폼

하드웨어에 맞게 프로그램이 가능한 코어부분(programmable core), 입출력, 서브시스템(sub-system), 메모리로 구성된 마이크로아키텍처(Micro-architecture)로 재사용 가능한 소프트웨어를 장착할 수 있는 기반을 형성한다. 또한 사용이 빈번한 서브시스템의 구조를 정형화하고 SystemC, VHDL과 같은 하드웨어 컴포넌트로 구성된다.

② 소프트웨어 플랫폼

소프트웨어 플랫폼은 응용프로그램이 하드웨어와 소프트웨어 사이의 인터페이스를 추상화하는 플랫폼으로 API(Application Program Interfaces)가 이에 해당한다. 이는 임베디드 소프트웨어 개발자들과 산업체에서는 일종의 계층구조로 볼 수 있으며, API의 추상화를 통하여 하부 계층의 구현 내용이 상이하더라도 상위계층은 영향을 받지 않도록 함으로써 이식성과 호환성을 최대한 보장하는 개념이다. 이는 커널을 포함하는 운영체제, 컴포넌트 IP에 대한 디바이스 드라이버 등이 해당된다.

③ 시스템 플랫폼

앞서 소개된 두 플랫폼의 공통점을 찾아 시스템을 구현하기 위한 플랫폼이다.

플랫폼 기반의 개발도구는 하드웨어와 소프트웨어 및 시스템 구성 요소들을 사용 시 최적화 된 상태로 API에 포함하여 두고 사용자의 요구에 따른 선택이 가능하다. 각 컴포넌트는 재사용을 통한 편리한 임베디드 시스템의 구현 및 디버깅, 시뮬레이션과 검증테스트까지 개발의 전 과정을 지원한다. 또한 개발시 편리한 사용자 인터페이스 환경의 제공 및 컴포넌트의 재사용을 통한 작업량의 획기적 감소에도 불구하고 실제 사용은 미미한 수준이다. 이와 같은 현상은 학습용 혹은 개발용도로 아직은 개발 도구의 구입비용이 너무 크고, 개발을 위한 다양한 플랫폼이 갖추어지지 않아, 실제 개발 할 때는 편리함이 기대효과에 미치지 못하는 것으로 분석된다.

플랫폼 기반의 임베디드 시스템 가상 프로토타이핑 개발도구는 Virtio사의 Virtio[12], Ptolemy[13]등이 있으며, 이중 특히 Virtio의 경우 가상 플랫폼(Virtual Platform)에 기반하여 SoC의 가상 프로토타이핑 환경을 제공한다.

4.2 ARMulator 기반의 가상 프로토타이핑

앞서 살펴본 바와 같이 가상 프로토타이핑은 시뮬레이션 과정을 하드웨어 없이 신속하게 수행함으로써 제품의 실물 프로토타입을 제작하는데 소모되는 시간을 단축시키고 빠른 시간 내에 하드웨어와 소프트웨어를 가상으로 검증할 수 있는 시스템 구축이 가능하다. 이러한 환경 구축을 위하여 영국 Cambridge의 Acron Computers Limited에서 개발된 ARM 프로세서 코어 기반의 제품 프로토타입을 손쉽게 제작 가능한 가

상 프로토타이핑 환경을 구축한다. 이를 위하여, ARM 사의 ADS(Application Develop Suit) 1.2에 포함되어 있는 ARM 코어 명령어 집합 시뮬레이터인 ARMulator를 사용한다. 또한 ARMulator 상에서 소프트웨어를 실행시키기 위해서는 armsd나 AXD, 또는 서드파티 디버거를 이용해야 한다[14]. 그리고 ARMulator에서 제공하는 기본 메모리와 타이머 환경에서 특정 IP들을 구현하고 이를 위한 OS로 초경량의 RTOS인 uC/OS-II를 포팅한다. 따라서 개발 환경에 적합한 개발 보드 없이 휴대형 단말 장치를 위한 LCD와 Keyboard를 개발하기 위한 프로토타이핑 환경을 구축한다.

4.2.1 ARMulator의 구조

ARMulator는 ARM 프로세서 코어와 캐쉬 모델 그리고, 어드레스 디코딩을 포함하는 기본 메모리 모델을 중심으로 그림 1과 같이 네 개의 콤포넌트로 구성되어 있다[15].

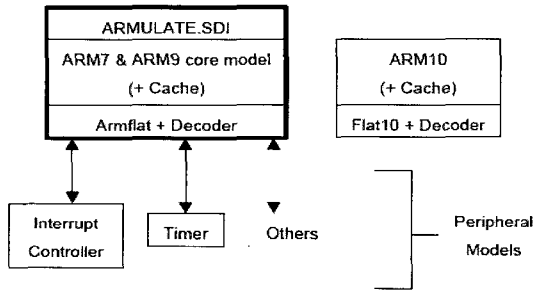


그림 1 ARMulator의 구조

1) 캐시를 사용하는 ARM 프로세서 모델
2) 어드레스 디코딩과 기본 메모리(Armflat) 통합 모델
3) 파일 설정을 통한 활성 또는 비활성되어 기본 메모리 모델과 통신하는 주변장치 모델
4) 실행환경을 제공하기 위한 OS 모델

제공되는 모델들을 편집 또는 새로운 모델을 작성하여 다른 주변장치를 손쉽게 추가 할 수 있다. 주변장치는 모델이 초기화 되는 동안 ARMulif_ReadBusRange()와 bus_registerPeripFunc()로 호출되어 등록된다. 그림에서 ARMULATE.SDI는 ARMulator 컴포넌트의 핵심부분이다.

ARMulator는 Dynamic Link Library(.dll)와 Shared Object(.sdi)와 같은 모듈로 구성되어 있으며, 메인 모듈은 ARM 프로세서 코어모델과 프로세서에 의해 사용되는 메모리 모델이다. 이것은 각 부분을 위해 미리 정의되어 있으며, 메모리 모델과 프로세서 모델 사이에서 사용하고자 하는 것을 선택 가능하다. 미리 정의된 메모리 모델은 mapfile로 시뮬레이터된 메모리 시스템을 나타내 주고 있으며, 부족한 메모리와 대기 상태를 명시한다. 미리 정의된 모듈의 다른 조합과 다른 메모리 맵을 적절히 사용함으로써 정의된 메모리 모델이 요구 사항에 맞지 않을 경우 새로운 모듈을 추가하거나 수정이 가능하다[15].

4.2.1.1 ARMulator 확장 키트

ARMulator는 모듈의 집합으로 구성되어 있고, ARM 코어와 메모리에 관한 에뮬레이션을 할 수 있으며 추가적 주변장치에 관한 모듈을 삽입할 수 있다[16]. 표 1은 두 가지 모델 그룹의 소스코드로 지원되는 것을 보여주고 있다.

표 1 ARM 모델의 기본 소스 코드

Basic models	Peripheral models
Tracer.c	Intc.c
Profier.c	Timer.c
Pagetab.c	Millisec.c
Stackuse.c	Watchdog.c
Nothing.c	Tube.c
Semihost.c	
dcc.c	
Mapfile.c	
Flatmem.c	

새로운 모델을 추가하는 가장 좋은 방법은 기존의 모델 중 하나를 카피 한 뒤 그것을 수정하는 것이다. ARMulator는 .ami와 .dsc 설정 파일을 읽고 어떤 모델인지 파악한다. ARMulator에서 새 모델을 사용하기 전에 .dsc파일에 새 모델을 추가 해 야하고, default.ami와 peripherals.ami. 설정 파일에 추가되어져 있어야만 참조할 수 있다[15,16].

4.2.1.2 ARMulator 설계

ARMulator는 armsd 혹은 AXD 상에서 실행가능하며ARM 사에서 제공되는 ADS 1.2 버전에 포함되어 있는 AXD로 실행하였다. ARMulator는 기본적으로 인터럽트 컨트롤러와 두 개의 타이머를 지원하며[17][18], 출력물을 별도의 화면으로 보여줄 수 있는 LCD 모듈을 추가하였으며, ARMulator를 지원하는 uC/OS-II를 포팅한 후 OS가 정상적으로 포팅되었는지 검증가능한 애플리케이션 프로그램을 추가하였다. LCD와 Keyboard는 ARM 기반의 시스템 개발 환경에서 입출력을 테스트해 볼 수 있는 간단한 모듈로 ARM 에서 제공되는 Application Note 92[19]를 참조하여 테스트 하였다. ARM에서 제공되는 간단한 소스파일은 C와 ARM 어셈블러로 구현되었으며, AXD 디버거로실행하며, ARMulator의 상세한 실행 환경은 그림 2와 같다.

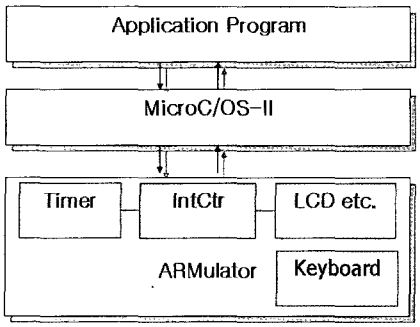


그림 2 구현 시스템 구성도

ARMulator에서 uC/OS-II 포팅에 사용되는 IP는 다음과 같다.

가) Timer

ARMulator는 기본적으로 2개의 16bit down counter Timer형 타이머를 제공한다. 시스템 동작 주파수를 낮추기 위한 Prescale unit에 의해 1, 1/16, 1/256으로 클럭을 낮추고 로드 레지스터에 특정한 값을 설정하고 타이머를 동작시키면 타이머 클럭마다 설정된 값을 감소시키고, 값이 0이 되면 타이머 인터럽트를 발생시켜 인터럽트 제어기에게 전달한다.

나) Interrupt controller

ARMulator의 인터럽트 컨트롤러는 간단한 소프트웨어 인터페이스를 제공한다. FIQ(Fast Interrupt Request)와 IRQ(Interrupt Request)의 두 가지 레벨의 인터럽트가 있으며, 각 인터럽트 소스를 컨트롤하기 위한 32bit 사용이 가능하며 최대 32개의 인터럽트 소스를 처리할 수 있게 확장 가능하다.

- Enable Register : 읽기 전용으로 프로세서에 해당 인터럽트를 요청하게 하는 데 사용.
- Enable Set : 쓰기 전용으로 Enable register에 해당 bit 세팅을 위해 사용.
- Enable Clear : 쓰기전용으로 Enable register에 해당 bit를 클리어하기 위하여 사용.
- Source Status : 읽기 전용으로 인터럽트 콘트롤러에 인터럽트 소스의 상태를 알림
- Interrupt Request : 읽기 전용으로 마스킹 후에 인터럽트 소스의 상태를 제공.
- Programmed IRQ Interrupt : 읽기 전용으로 이 레지스터로 인터럽트를 set 혹은 clear.

기본적으로 아래 표 2와 같이 software programmed interrupt 와 communication channel, timer 두 개가 정의되어 있다. Bit 0은 IRQ 컨트롤러에서는 정의 되어지지 않고 FIQ의 인터럽트 소스를 위해 사용되어 진다.

표 2 인터럽트 컨트롤러 정의 비트

Bit	Interrupt Source
0	FIQ scurce
1	Programmed Interrupt
2	Comms Rx
3	Comms Tx
4	Timer1
5	Timer2

Timer1을 셋팅하는 것으로 타이머와 인터럽트 컨트롤러에 사용법을 보면

- Timer1의 Control과 Clear를 0으로 초기화
- Timer1Load에 값을 넣고 (최대값은 16bit , 0xffff)
- Timer1Control에 bit 7, 6, 3을 1로 셋팅 해주면 시스템 클럭의 1/256의 클럭으로 주기적으로 동작
- 여기서 IRQEnableClear를 초기화 해주고 IRQEnableSet에 Timer1에 해당하는 Bit 4를 1로 셋팅해 주면 Timer1에 의한 인터럽트가 허용.
- 그 전에 인터럽트 발생 시 인터럽트 핸들러에서 각 인터럽트에 해당 ISR를 벡터로 등록해 놓아야 그 인터럽트가 정상적으로 처리 가능

4.2.2 uC/OS-II

uC/OS-II는 RTOS (Real Time Operating System)로 Micro Controller Operating System Version 2를 말한다. 임베디드 시스템을 위한 작고 간단한 RTOS로 다양한 프로세서 아키텍쳐에 포팅 하기 쉬운 구조로 되어 있다[20,21].

4.2.2.1 uC/OS-II 포팅

uCOS-II 소스 프로그램은 Processor-Independent Code와 Processor-Specific Code, Application-Specific Code로 나뉜다[15,16]. Processor-Independent

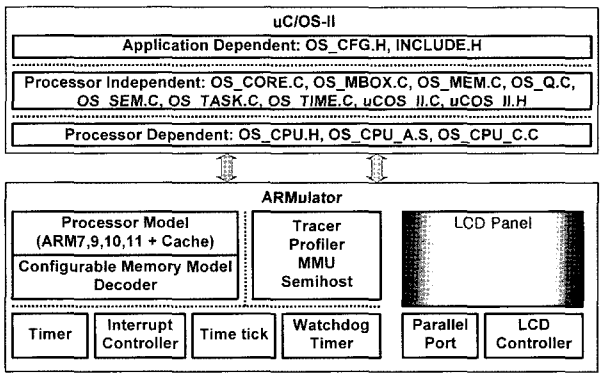


그림 3 uC/OS-II가 이식된 ARMulator 환경

Code는 포팅 대상 프로세서 아키텍처에 독립적이므로 포팅 시에는 특별한 경우를 제외하고는 수정할 필요가 없으며 포팅을 하기 위한 주요 파일은 OS-CPU.H, OS_CPU_A.S, OS_CPU_C.C이다. 다음 그림 3은 uC/OS-II가 이식된 ARMulator 환경을 나타낸다.

4.2.3 ARMulator를 이용한 가상 시뮬레이션

구현된 시스템의 동작 검사를 위하여 간단한 응용 프로그램을 작성하여 정상적인 이식을 확인하였다. 포팅이 성공적인지 판단하기 위해 테스트 해야 할 것은 우선 Timer와 인터럽트 컨터롤러가 정상적으로 작동하는지 살펴야 한다. 테스트 결과

성공이면 스케줄링이 가능하며 OS의 가장 중요한 요소인 스케줄링 테스트 단계로 진행된다. 먼저 유저 모드에서 OS가 실행되기 위해 모드를 전환한 상태에서 SWI나 Interrupt를 통해서만 PSR를 변경할 수 있고 context switching을 할 수 있다.

다음 단계로 OSTick에 사용할 Timer1을 초기화 시키고 가동 시킨다. 3개의 태스크 priority를 차례대로 부과하여 생성하였다. 메인에 의해 기본 적인 태스크가 생성이 되고 그 태스크에서 TASK2와 TASK3을 생성하였다. 테스트는 아래 그림 4의 좌측과 같이 TASK1에서 무한 루프를 돌면서 100Ticks의 딜레이를 갖는다. TASK2는 200Ticks의 딜레이를 가지면서 세마포어 pend를 하게 하였고, TASK3에서 400Ticks의 딜레이를 가지면서 세마포어 post를 하게 하였다. current는 현재의 PSR 값이다. user 모드로 변경되어 있고 인터럽트가 비활성 상태지만 SWI를 통해 인터럽트를 활성 상태로 변경하고, Timer1의 Load값에 최대값 0xffff를 설정하였다.

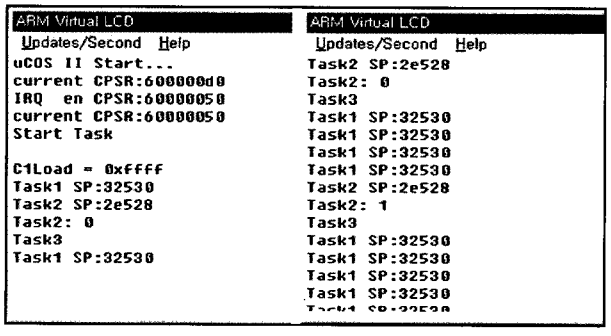


그림 4 context switching을 위한 실행 화면

위 그림 4의 우측은 실행 결과 TASK1이 실행되고 SP값을 보여주고 있다. TASK1이 OSTimeDly를 통해 100ticks 동안 waiting 상태로 들어가고, TASK2가 실행되며, 같은 과정으로 OSTimeDly에 의해 200Ticks 동안의 waiting에 들어가게 된다. 그 후 TASK3가 실행된다. 타이머가 정상적으로 작동되고 또 그 타이머에 의한 인터럽트를 정상적으로 처리해야 OSTimeDly에 의한 waiting이 가능하다. 그리고 그림 5와 같이 가상 프로토타이핑을 이용하여 개발보드의 IP의 일부를 구현하고 그 위에 RTOS인 uC/OS-II를 포팅하고 어플리케이션 프로그램을 실행해 보았다. 고가의 개발보드가 없어도 ARM 코어상의 OS 포팅 및 어플리케이션을 실행시켜 정상적인 동작을 테스트하였다.

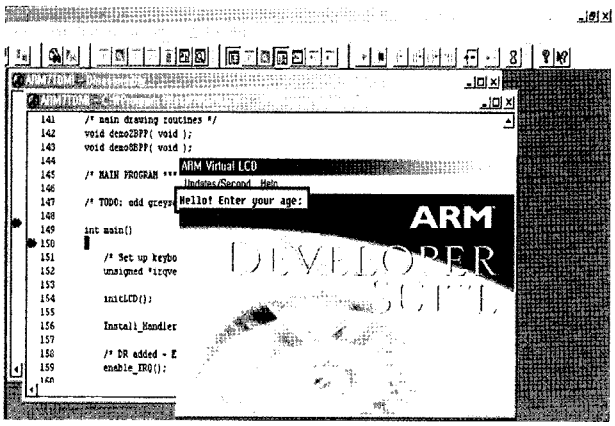


그림 5 키보드 문자입력 및 출력 화면

4.3 Virtio를 이용한 Platform 기반의 가상 프로토타이핑

플랫폼 기반의 가상 프로토타이핑 개발 도구인 Virtio는 제품의 개발 주기 단축을 목표로 초기 개발 단계에서 완벽한 임베디드 시스템의 소프트웨어 모델을 제공함으로써 신속한 임베디드 소프트웨어 개발을 위해 플랫폼에 맞는 소프트웨어 모형을 제작, 실제 플랫폼 기반의 개발환경을 제공한다. 그리고 선택사항에 따라 쉽게 플랫폼의 구성 및 변경이 가능하고, 프로세서와 시스템 프로토콜로 구성되어져 프로세서의 종류, 개수, 버스구조와 계층, 데이터 종류까지 구성하는 개방형과 응용모듈이 부가되어 약간의 변형 혹은 기능 추가를 위한 기본 디자인 형이 있다. 또한 모듈별 설계를 통한 검증과 하드웨어와 소프트웨어 모듈을 동시에 개발 가능한 환경을 제공한다. 본 절에서는 PXA255 칩셋 중 UART 입출력을 프로토타이핑하기 위해서 Intel의 Xscale PXA255 기반의 칩셋을 가상 플랫폼 기반으로 구현하기 위하여 PXA255 칩셋 중 UART 입출력을 프로토타이핑한다. 이를 위한 시뮬레이션 스케줄링은 기본적으로 시분할 스케줄링으로 구현하며, VRE(Virtio Runtime Environment) 마스터들은 각각 라운드로빈 스케줄링 알고리즘으로 실행한다. UART 입출력 인터페이스는 VSP(Virtual Serial Port)를 사용한다.

4.3.1 Platform기반 가상프로토타핑 개발도구 특징

임베디드 시스템에서 주어진 시스템의 기능을 하나의 칩에 구현하기 위한 SoC의 개발에 사용이 확대되고 있다. 하나의 칩에 시스템을 완성하기 위하여 제조 공정이 다른 여러 기능 블록들을 집적하는데 주요 블록은 다음과 같다[22].

- Embedded Microprocessor
- 메모리
- 외부 시스템과 통신을 위한 장치
- 데이터 전송 블록
- Analog, RF, MEMS 블록 등

플랫폼 기반의 임베디드 시스템 개발 도구 역시 객체지향적 개발방법론에 기반한 IP의 재사용을 통하여 시스템의 자원을 활용하고 IP간의 연결과 버전 문제를 조율함으로써 애플리케이션 입장에서 시스템을 모니터링 할 수 있는 능력과 시스템이 사용할 IP 선정 능력을 동시에 지닐 수 있다. 이러한 플랫폼 기반의 개발도구는 애플리케이션의 변화에 대한 수용 가능한 일반적 구조 여부가 포인트가 되며, 세 개의 세부 플랫폼으로 구성되어 검증 영역을 포함하는 플랫폼을 제공한다.

- 하드웨어 플랫폼 : programmable core, I/O subsystem Memory
- 소프트웨어 플랫폼 : 커널을 포함하는 OS, 컴포넌트 IP에 대한 디바이스 드라이버 등
- 시스템 플랫폼 : 앞서 소개된 두 플랫폼의 공통점을 찾아 시스템을 구현하기 위한 플랫폼

각 플랫폼을 미리 만들어 API에 등록된 하드웨어와 소프트웨어의 속성에 따라 시스템을 구현하고 시뮬레이션을 통한 검증 기능까지 가능하다. 플랫폼 기반의 개발도구는 하드웨어와 소프트웨어 및 시스템 구성 요소들을 사용 시 최적화 된 상태로 API에 포함하여 사용자의 요구에 따라 선택 가능하며 각 컴포넌트는 재사용을 통한 편리한 임베디드 시스템의 구현 및 디버깅, 시뮬레이션과 검증 테스트가 가능하다.

4.3.2 하드웨어 및 소프트웨어 디자인

임베디드 시스템의 개발, 디버깅, 실행의 가상 컴포넌트와 플랫폼을 제공하는 Virtio innovator는 Microsoft의 Visual C++ 컴파일러와 유사한 인터페이스로 제공된다. 툴 컴포넌트는 디자인 브라우저와 Graphical Magic-C 에디터와 디버거, Code Generator를 통하여 C++코드가 생성되며 플러그앤 플레이로 테스트 할 수 있는 Test Bench Editor로 구성되어져 있다. 그림 6은 Innovator로 하드웨어를 구성하는 과정을 보여주고 있다.

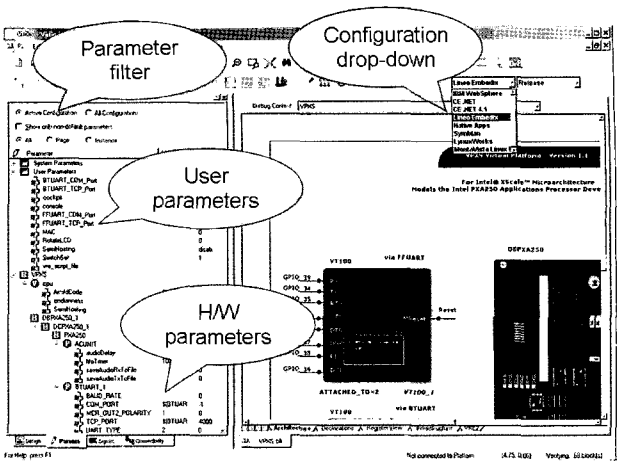


그림 6 하드웨어 모델링

UART는 하나의 명령을 처리하기 위한 블록 컴포넌트로 PXA250 프로세서는 3개의 UART를 포함하는데 BlueTooth UART(BTUART), Full Function UART(FFUART)와 Standard UART(STUART)로 구성된다. 가상 VPXS-WM Platform은 시리얼 Port와의 채널 인터페이스를 제공하며 데이터 전송은 윈도우 API 함수로 실행한다.

4.3.3 시스템 입출력 인터페이스 설계

그림 7과 같이 Serial COM Port와 인터페이스를 구현하기 위한 가상 Port로 VSP(Virtual Serial Port)를 사용한다. VSP는 DDI(Device Driver Interfaces) 표준 윈도우의 통신 Port(COM Port)를 표시하는 것으로, UART,Modem과 같은 실제 하드웨어를 통하여 수행되는 데이터로 다른 애플리케이션이 VSP로 액세스를 허용하는 I/O 인터페이스의 설계가 가능하다.

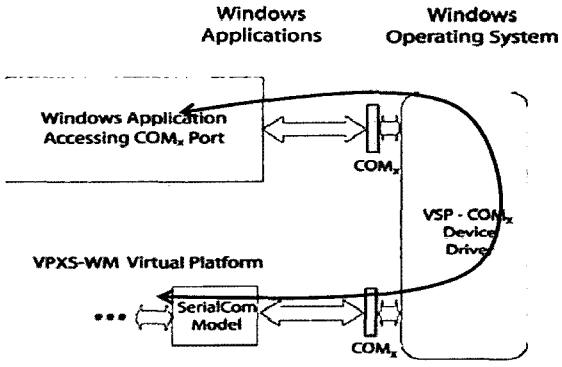


그림 7 시스템 입출력 VSP 동작

4.3.4 하드웨어와 소프트웨어 통합 디버깅

아래 그림 8은는 통합 디버깅 환경을 보여준다. 디버깅은 VRE로 시뮬레이션, 컴포넌트의 로딩과 스케줄

링 통합 관리 및 VDI(Virtio Debugger Interfaces)를 제공한다. 또한 VRE는 ISSs, SystemC, C로 변환된 HDL등과 같은 써드파티 디버거와 쉽게 통합이 가능하게 하는 역할을 한다.

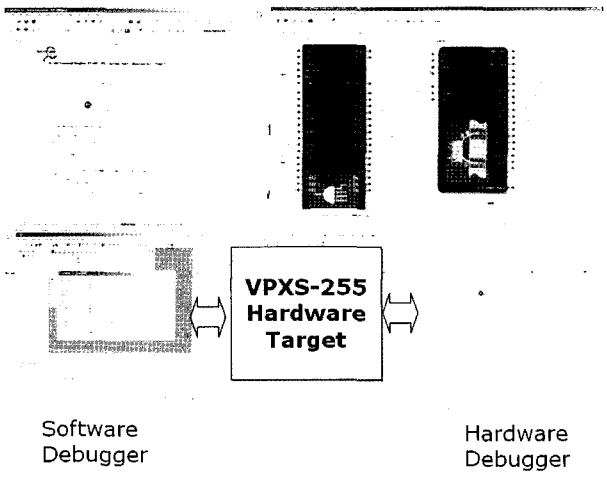


그림 8 Virtio Debugger Interfaces

4.3.5 UART 입출력 가상 시뮬레이션

VPXS-255 보드의 3개 UART중 FFUART와 BTUART의 Port를 각각 2와 5로 셋팅하고 BAUD를 각각 19200으로 설정하고 호스트와 타겟의 파라미터의 값을 설정한다. 아래 그림 9는 VPXS-255보드의 UART 설정후 실행했을때 타겟보드의 가상 시스템 콘솔 메뉴이다. 그림 10은 타겟보드의 Port를 VSP를 사용하지 않은 입출력 현황을 보여준다.

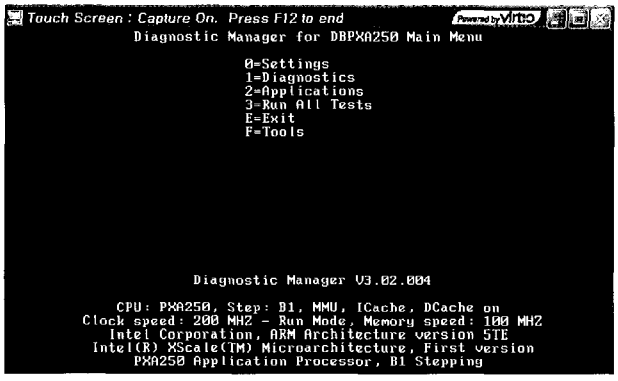


그림 9 VPXS-255 타겟보드 입출력 메뉴

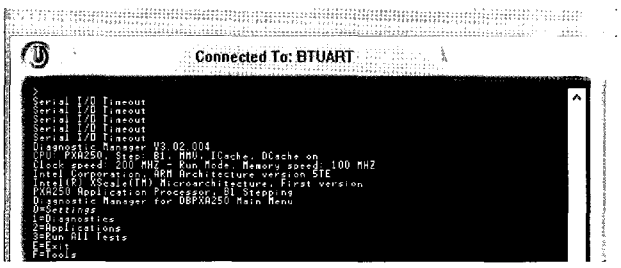


그림 10 VSP를 사용하지 않은 시스템 입출력

아래 그림 11은 각 Port의 값을 5와 2로 설정하고 우측의 실제 Port를 각각 VSP와 같은 값의 Port로 설정하여 VSP를 이용하여 가상으로 시스템 입출력을 시뮬레이션 한 화면이다.

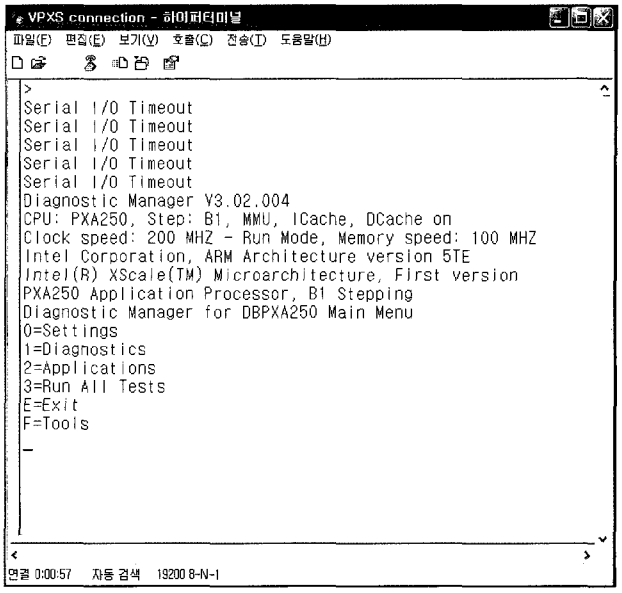


그림 11 VSP를 사용한 시스템 입출력

I/O 및 각각의 인터페이스는 VSP를 사용하지 않아도 무관하나, 가상 플랫폼에서 가상의 시리얼Port의 연결에서 VSP는 임베디드 칩셋의 가상 프로토타이핑 및 임베디드 컴퓨팅에 적용이 가능하다.

4.4 RapidPLUS를 이용한 기능 및 UI(User Interface) 기반의 가상 프로토타이핑

임베디드 시스템 개발에서 개발 도구를 이용한 UI 기반의 개발 방법은 기존의 범용 프로그래밍 툴인 Visual Basic, Visual C++, Delphi, Java 등의 개발도구를 이용 개발하는 방법보다 새로운 모델 구현이나 다른 모델의 프로토타이핑시 편리함을 제공한다. 또한 UI 기반의 개발 도구는 다음과 같은 특징을 갖는다.

- GUI 기반의 point-and-click 환경으로 요구되는 컴포넌트와 동작을 생성
- State Chart를 이용한 동작의 설계와 분석 및 문서화 가능
- 하드웨어와 소프트웨어의 동시 시뮬레이션이 가능하며 개발 초기부터 오류수정과 실시간 동작 테스트를 통하여 최종제품의 강력하고 완벽한 시뮬레이션이 가능
- 외부 오브젝트를 위한 인터페이스 코드의 최적화
- 다양한 플랫폼과 실시간 OS가 제공됨으로써 목적에 맞는 시스템 구현을 위한 환경 제공
- 소스레벨 디버거 기능 및 써드파티 디버거 지원

본 절에서는 임베디드 시스템 가상프로토타이핑 개발 도구인 RapidPLUS를 이용하여 컴포넌트의 재사용의 유연함과 확장 가능성을 지원하는 UI 기반의 세탁기 가상프로토타이핑 모델을 구현한다.

4.4.1 사용자 인터페이스 설계

RapidPLUS를 이용한 사용자 인터페이스를 설계하는데는 아래 그림 12에서 보듯이 크게 시뮬레이션을 하는데 있어서 보여주기 위한 Object부분과 Object들의 이벤트나 기능, 행동등의 기능을 정의 부분(Modes, Transitions, Triggers, Activities)으로 나눠서 설계한다.

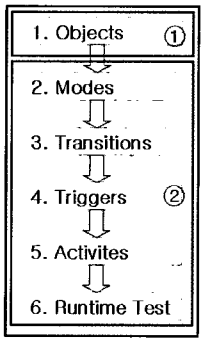


그림 12 인터페이스설계

- Object : 어플리케이션의 UI를 위한 가시적 타입 표현
- Modes : 어플리케이션의 동작가능 상태정의를 위한 계층적 형태
- Transitions : 어플리케이션 모드 사이의 전이(transitions) 생성
- Tirggers : 전이가 발생할 때의 트리거 정의
- Activities : mode의 동작이 만들어졌을 때 발생
- Runtime Test : 프로토타이퍼 내부의 구현 애플리케이션 테스트

사용자 인터페이스를 모델링하기 위해서 개발 도구에서 지원해주는 Object Layout 창에서 Class들의 Object들을 이용해서 가정용 전기 세탁기를 설계할수 있다. 가정용 전기 세탁기 가상 프로토타이핑을 시뮬레이션하기 위해서는 다음과 같은 역할을 하는 Object들을 그림 13과 같이 배치한다.

① 가정용 전기 세탁기의 전원을 킨다.
② 코스를 선택한다(표준, 강력, 헹굼...)
③ 냉·온수, 물높이, 물살의 특성을 선택한다.
④ 세탁, 헹굼, 탈수의 조건을 선택한다.

가시적 타입을 위한 그래픽 Object는 그래픽 라이브러리에 등록되어 있으며, 사용자가 필요로 하는 Object가 없을 경우 직접 만들어 라이브러리에 등록하여 사용가능한 UDO(User-Object Librery)를 제공한다.

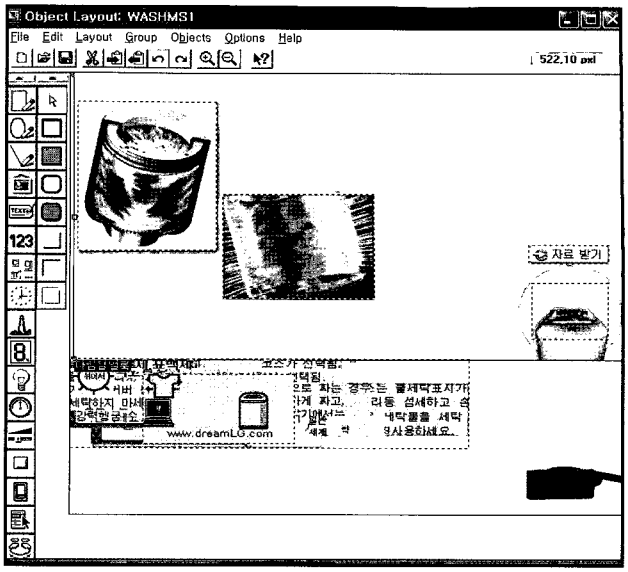


그림 13 세탁기 정적 모델링

4.4.2 Object 기능 정의

위에서 정의한 Object들을 기반으로 한 기능을 정의한다. RapidPLUS에서는 기능 정의는 다음 그림 14와 같은 논리 흐름을 따른다.

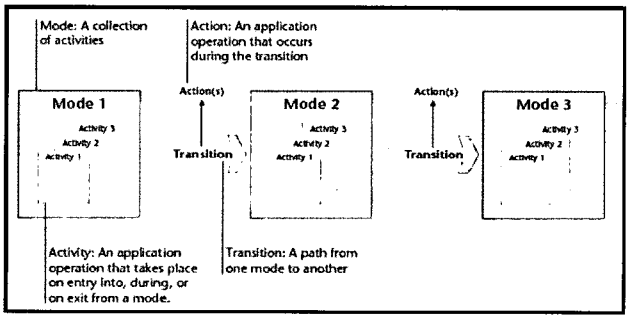


그림 14 기능 정의를 위한 논리 흐름

- **Mode** : activities의 집합
- **Activity** : 한 모드로부터 빠져나가거나 한 모드에 있거나, 한 모드로 진입했을때 일어나는 RapidPLUS 애플리케이션 동작
- **Transition** : 한 모드에서 다른 모드로 가는 경로
- **Action** : 트랜지션 동안에 일어나는 RapidPLUS 애플리케이션 동작

위와 같은 기능 정의를 위한 논리 흐름은RapidPLUS에서 제공하는 Logic Editor와 Logic Palette를 이용해서 설계 가능하다.

4.4.3 동적 모델링

위의 절에서 정의된 기능을 통해서 아래 그림 15는 세탁기 애플리케이션의 동작의 상세 구현을 보여준다.

tree의 mode를 선택하고 Logic Editor를 통하여 transition, tirgger, event, activity를 Logic Pallet에서 선택하여 구현하고, Prototyper로 테스트 후 오류 및 동작 수정이 가능하다.

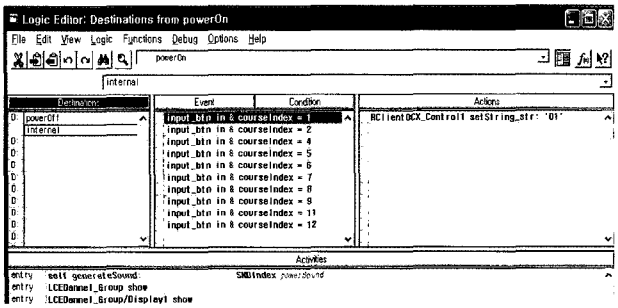


그림 15 기능과 동작의 상세 구현

구현을 위한 function과 property는 API에 등록이 되어 있으며, UDO와 같이 시스템이 요구하는 함수가 API에 없을 경우 사용자가 제작하여 API에 등록하여 사용할 수 있는 UDF (User-Define Function)를 제공한다. 그리고 트리구조에서는 최상위 객체인 Washms1로부터 각 UI와 동작들이 상속되는 구조를 볼 수 있다. 모든 동작은 상속 관계에 있으며, 복잡한 설계 과정이 계층적으로 구성되어 mode tree를 통하여 전체 시스템의 구조를 볼수 있다. 트리구조에서 세탁기 어플리케이션의 최상위가 powerOff/powerOn 상태로부터 파생되는 것과 같이 동시에 실행이 불가능한 상반된 역할의 같은 계층에서 화살표로 표시하며 Exclusive mode임을 표시한다.

4.4.4 시뮬레이션될 시스템의 모델링

그림 16은 시뮬레이션될 세탁기 프로토타이핑 시스템의 전체를 모델링하였다.

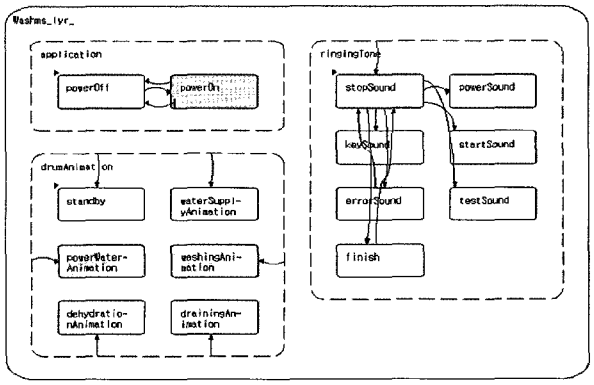


그림 16 세탁기 모델링

Washms1에 각각 application drumAnimation, ringingTone으로 구성된 3개의 mode가 있다. application은 세탁기의 상세 설계 동작을 구현하고 있으며, drumAnimation은 세탁조의 물이 채워지는 상태와 세탁이 실행되고 멈추는 상태를 표현한다. ringingTone은 세탁기 실행 동작 상태를 음성안내 동작을 구현하고 있다.

4.4.5 세탁기 가상 시뮬레이션

아래 그림 17은 Object Layout을 이용해서 세탁기 정적 모델링을 하고 세탁기 어플리케이션의 실행을 통한 시뮬레이션 화면을 보여주고 있다. 정적 모델링을 통해서 인터페이스를 설계하고, 설계된 Object들의 기능 정의와 동적 모델링을 통해서 시뮬레이션을 하여 실제 세탁기가 하는 기능들을 그대로 구현하여 컴퓨터 상에서 가상으로 시뮬레이션을 할 수 있다.

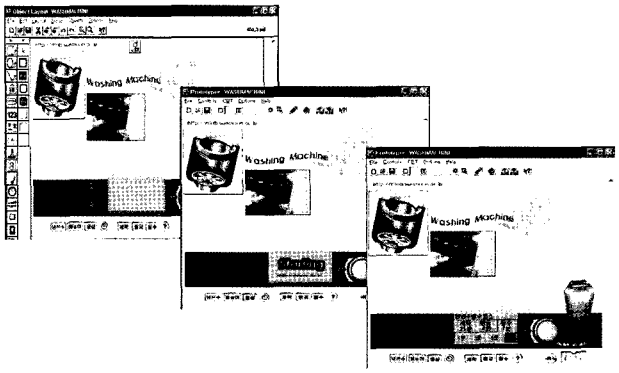


그림 17 세탁기 가상 시뮬레이션

5. 결 론

본 원고에서는 임베디드 소프트웨어 개발 시의 차별성 및 특수성과 문제점을 살펴보고, 이를 해결하기 위한 여러 관련 연구들을 소개하였다. 그리고 임베디드 소프트웨어 개발 과정에서 필요성이 대두되고 있는 프로토타이핑 기술을 이용한 개발 방법과 이 프로토타이핑 개발 방법을 이용한 임베디드 시스템 가상 시뮬레이션하는 과정을 소개하였다. 임베디드 시스템 가상 시뮬레이션은, 가상 프로토타이핑 모델을 컴퓨터 상에서 완제품이 출시되기 전에 먼저 설계 및 구현하여 사용자들의 생생한 요구를 수렴하고 이를 이용하여 기본적인 요구 분석 및 설계 작업이 시행된다. 이러한 과정을 지속적으로 거치면서 프로그램 구조를 설계하여 구현하는 과정을 점증적으로 반복하게 된다. 따라서 설명된 개발 도구와 개발 방법론을 이용하여 임베디드 시스템을 시뮬레이션하여 응용 모델에 알맞은 점증적인 개발 과정을 거칠 수 있다.

본 프로토타이핑 개발 방법론을 이용하여 하드웨어와 소프트웨어 개발자들이 임베디드 시스템 제품 개발에 적용하면 하드웨어와 소프트웨어를 동시에 설계 및 구현할 수 있는 기회를 가지게 되고, 다양한 형태의 임베디드 시스템 모형을 제작해 재사용이 가능하게 되어

복잡도가 빠르게 증가하는 임베디드 시스템 제품 설계에서 제품의 기능 검증까지 소모되는 시간과 비용의 부담의 최소화로 제품의 경쟁력을 강화시킬 수 있고 시장 진입 시점(Time-to-Market)을 단축시킬 수 있을 것이다.

참고문헌

[1] 정기훈, 채화영, 김정길, 이재진, 강순주, "임베디드 시스템 프로토타이핑 기술", 정보처리학회지, 제11권, 제6호, pp.86-99, 2004년 11월

[2] 경희대학교 CANN 연구실 금지수, Virtual Prototyping, http://myhome.hitel.net/~tbno/work1/VirutalProtyping.ppt

[3] 원종혁, 한상용 "사용자 인터페이스 향상을 위한 3차원 VP시뮬레이터 설계", http://nlpwww.sogang.ac.kr/~hci/i-3-2.pdf

[4] RapidPLUS, http://www.e-sim.com

[5] MSC.SimManager, http://www.mscsoftware.com/products/simmanager.cfm?Q=131&Z=288

[6] RT-LAB, http://www.opal-rt.com/productsservices/softwarecomponents/rtlab/index.html

[7] Rhapsody, http://www.ilogix.com

[8] PlayMo, http://www.playmo.co.kr

[9] Anders Moller et al., "An Industrial Evaluation of Component Technologies for Embedded Systems," MRTC RePort ISSN 1404-3041 ISRN MDH-MRTC-155/2004-1-SE, Malardalen Real-Time Research Centre, Malardalen University, February 2004.

[10] ART System, http://www.artsystem.co.kr

[11] ASADAL, http://selab.postech.ac.kr/realtime/public_html/

[12] Virtio, http://www.virtio.com

[13] Ptolemy, http://ptolemy.eecs.berkeley.edu/

[14] ARM Cop. ARM DUI0068D ARM Developer Suite Debug Target Guide

[15] ARM Cop. ARM DAI0032E Application Note 32 The ARMulator

[16] ARM Cop. ARM DAI0052A Application Note 52 The ARMulator Configuration File

[17] ARM Cop. ARMDDI0062D Reference Peripherals Specification

[18] ARM Cop. ARM DUI0207A Realview ARMulator ISS User Guide

[19] ARM Cop. ARM DAI0092B Application Note 92 LCD and Keyboard ARMulator model, http://www.arm.com/documentation/

[20] JEAN J. LABROSSE, MicroC/OS-II Real Time Kernel 2/E R&D Technical Books, 2002.

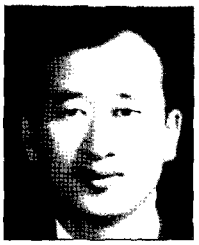
[21] Micrim Technologies Cop. ARM Porting 소스, http://www.ucos-ii.com/

[22] Virtio Cop. Virtual Prototyping http://www.virtio.com/products/page/0,2573,21,00.html

정 영 진

2004 선문대학교 컴퓨터정보학부(학사)
2006 선문대학교 전자계산학과(석사)
2006~현재 선문대학교 컴퓨터정보학과 박사과정
관심분야 : 실시간 시스템, 임베디드시스템
E-mail : yjjung.kr@gmail.com

이 정 배



1981 경북대학교 전자공학과 전산전공(학사)
1983 경북대학교 전자공학과 전자계산전공(석사)
1995 한양대학교 전자공학과 정보통신(박사)
1982~1991 한국전자통신연구원 선임연구원
1996~1997 U.C.Irvine 객원교수
1991~2002 부산외국어대학교 컴퓨터·전자공학부 부교수
2002~현재 선문대학교 컴퓨터정보학부 교수
관심분야 : 실시간 시스템, 임베디드 시스템, 실시간 통신 프로토콜
E-mail : jblee@sunmoon.ac.kr

이 영 란

2000 한국방송통신대학교 교육학과(학사)
2002 부산외국어대학교 컴퓨터·전자공학부(석사)
2006 선문대학교 컴퓨터정보학과(박사)
2002~현재 선문대학교 IT교육원 전임강사
관심분야 : 실시간 시스템, 임베디드 시스템
E-mail : yrlee@sunmoon.ac.kr

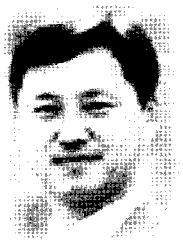
권 진 백

1998 한국외국어대학교 정보통계학과 (학사)
2000 서울대학교 전산과학과(석사)
2003 서울대학교 컴퓨터공학부(박사)
2003~현재 선문대학교 컴퓨터정보학부 조교수
관심분야 : 임베디드 시스템, 분산 시스템, 멀티미디어 시스템
E-mail : jbkwon@sunmoon.ac.kr

임 기 욱

1977 인하대학교 전자공학과(학사)
1987 한양대학교 전자계산학(석사)
1994 인하대학교 전자계산학(박사)
1977~1983 한국전자통신연구원 선임연구원
1983~1988 한국전자통신연구원 시스템 소프트웨어 연구실장
1988~1989 U.C.Irvine 방문연구원
2001~2003 한국전자통신연구원 컴퓨터 소프트웨어 연구소장
2000~현재 선문대학교 컴퓨터정보학부 교수
ITRC(임베디드 S/W 개발환경 연구센터) 소장
관심분야 : 실시간 데이터베이스시스템, 운영체제, 시스템 구조
E-mail : rim@sunmoon.ac.kr

조 상 영



1988 서울대학교 제어계측공학과(학사)
1990 KAIST 전기전자공학과(석사)
1994 KAIST 전기전자공학과(박사)
1994~1995 KAIST 정보전자연구소 위촉연구원
1995~1996 U.C.Irvine Post Doc.
1996~1997 삼성전자 선임연구원
1997~현재 한국외국어대학교 컴퓨터공학 전공 부교수
관심분야 : 임베디드시스템, 병렬처리, 컴퓨터 구조
E-mail : sycho@hufs.ac.kr