내장형 시스템 설계 : 개론

홍익대학교 정기석
한국과학기술원 김태환*

1. 내장형 시스템이란?

내장형 시스템이란 특정한 목적을 수행하도록 만든 컴퓨터 하드웨어와 소프트웨어의 결합체이다. 즉, 내장형 시스템에서 컴퓨터는 특정 목적을 위해 설계된 시스템의 한 구성요소이다. 예를 들어, 자동차에 쓰이는 ABS(Anti-lock breaking system)에 쓰이는 마이크로-프로세서(micro-processor) 및 하드웨어와 ABS를 위해 실행되는 프로그램의 결합이 내장형 시스템이다. 내장형 시스템의 종류는 너무나 다양해서 단순하게는 세탁기에서, 복잡한 로켓에 이르기까지 모두가 내장형 시스템이라 할 수 있다. 전 세계에 존재하는 컴퓨터시스템의 90%가 내장형 시스템이라는 보고서도 있는 것으로 보아, 얼마나 많고 다양한 생활 구석구석까지 이용되는지 알 수 있다. 제반 응용 프로그램의 전반적 고성능 실행을 목적으로 하는 범용 컴퓨터 시스템과는 달리 특정한 목적을 수행하기 위해 설계되고, 또 그 특정한 목적을 수행하기 위한 최적의 하드웨어와 소프트웨어로서 원하는 기능의 요구조건을 만족시켜야하는 점에서 범용시스템의 설계와는 크게 다르다고 할 수 있다.

내장형 시스템에 대한 연구와 개발은 이미 컴퓨터의 역사와 맥락을 같이 해 왔다고 볼 수 있을 정도로 오래 되어왔다. 하지만, 비교적 최근에 들어서야 많은 관심이 고조되고 있는 이유는 무엇일까? 그 대표적인 이유는 내장형 시스템이 복잡해지고 있다는 점이다. 반도체기술과 통신 기술의 급속한 발전은 인터넷의 일반적인 보급과 더불어 세상을 바꾸어 놓았다. 예를 들어 멀티미디어라고 불리는 다양한 정보 매개체는 단순히 한번에 한가지의 매개체만 전달하던 정보처리의 세계를 완전히 바꾸어 놓았다. 단순히 문자만을 교환하던 전자메일이나, 목소리만을 전달하던 전화선이 대량의 동영상이나 음악을 전달하는데 쓰이고, 이런 다중 매체 처리기는 실제로 하나의 정보만 처리하는 정보처리기기들의 결합이라고 볼 수 있기 때문에 내장형 시스템의 설계 복잡성은 빠르게 증가해 왔다.

이런 제반 기술의 급속한 발전은 내장형 시스템의 소형화/경량화, 고성능화, 다기능화, 및 다양화를 가능하게 하였다. 즉, 과거에는 단순한 하드웨어와 고정된 소프트웨어의 결합으로 보드(board) 상에서 구현되었던 것이 내장형 시스템의 일반적인 구현 방법이었다고 하면, 현재에 와서는 엄청난 양의 특화된 하드웨어와 아주 복잡한 소프트웨어가 효과적으로 실행되도록 설계된 고성능 프로세서가 하나의 칩 안에 집적되게 하는 SoC (system-on-chip) 개발이 일반화 되어가고 있다. 이는 사람들로 하여금 간단한 휴대폰 단말기 한 대로 전화, 인터넷, MP3 음향기기, 휴대 메모리 장치 뿐 아니라 디지털 카메라로도 사용을 가능하게 하고 있다.

이러한 기술적인 발전에 의한 내장형 시스템의 설계는 엄청난 시장의 성장을 예고하며 어림잡아도 3-4년 내에 내장형 시스템의 시장규모는 전 세계적으로 3000억 달러에 이를 것으로 전망되고 있다. 특히나 디지털 가전 기기와 멀티미디어 및 통신 기기의 시장은 가히 폭발적으로 증가될 것이 예상되고 있다. 또한 가전기기들과 정보통신 기기들이 하나로 통합되어 가고 있는 과도기에 있다고도 볼 수 있을 것이다.

이렇게 복잡한 기능을 갖고, 최고급의 성능을 유지해야하며, 더구나 빠른 시간 안에 다양한 대중의 요구에 맞는 제품을 만들어 내기 위해서, 내장형 시스템의 연구는 어느 때 보다 중요해졌다고 하겠다. 이는 과거의 하드웨어와 소프트웨어가 독립적으로

* 종신회원

개발된 후, 설계의 마지막 단계에서 결합되는 설계 방법으로는 도저히 시장의 고성능 및 빠른 생산 시간(fast time to market) 요구 조건을 만족시킬 수 없으며, 하드웨어와 소프트웨어간의 끊임없는 상호연관을 통하여 동시에 개발/검증되는 방법이 내장형 시스템 설계에 있어 매우 중요한 과정이 되고 있다. 이것이 내장형 시스템의 설계에 대한 전반적인 연구가 현재 주목받고 있는 가장 큰 이유라 하겠다.

2. 내장형 시스템의 특징

내장형 시스템은 너무나 다양해서 특징을 일일이 언급한다는 것은 어렵지만, 앞에서 열거한 내장형 시스템의 경향에 의거하여 몇 가지 특징을 요약하면 다음과 같다.

2.1 목적이 한정되어 있다

범용으로 만들어진 프로세서와 메모리는 어떤 프로그램이 수행될 지 정확히 알 수 없기 때문에 일반적인 프로그램이 빠르게 실행될 수 있는 구조를 갖게 되며, 이렇게 설계된 후 고정된다. 이렇게 일단 설계된 프로세서의 성능은 결국, 운영체제 및 컴파일러 등 많은 구성 소프트웨어에 의해 평가된다. 이는 먼저 하드웨어가 개발되고 나서 그 구조에 가장 적합한 소프트웨어 설계라는 순차적인 설계 방법론을 의미한다.

컴퓨터 시스템이 어떤 특정 목적을 위해 설계될 때, 수행되는 기능이 거의 고정적이기 때문에, 범용 시스템의 설계 시 만큼, 소프트웨어만을 최적화시켜 얻을 수 있는 성능 향상이 크지 않은 경우가 많다. 그리하여, 내장형 시스템은 빈번한 연산의 보다 빠른 수행을 위하여 때론 그 기능을 하드웨어로 구현해야 될 필요가 있다. 즉, 소프트웨어와 하드웨어가 동시에 개발되면서, 어떤 기능이 하드웨어로 구현되어야 하는지 아니면 소프트웨어로 구현되어야 하는지의 문제를 먼저 고려해야 한다. 바꾸어 말하면, 이는 특정한 목적을 효과적으로 달성하기 위한 최소의 하드웨어와 최소의 소프트웨어를 이용한 시스템을 설계해야 한다는 문제를 내장형 시스템 설계자가 해결해야 함을 의미한다. 현실적으로는 하나의 범용 프로세서가 개발되었을 때, 이 프로세서가 내장형 시스템에 쓰이기 위해, 이 범용 프로세서를 기본으로 하는 여러 종류의 변종 프로세서가 상황에 맞추어 필요한 기능을 갖도록 특화되는 경우가 많다. 그 만큼 하나의 내장형 시스템은 각각의 개성을 갖게 되며, 다양하다는 말과도 일맥 상통 한다고 보아야한다. 혹자는 이를 "Each embedded system is unique." 이란 말로 표현하는데 내장형 시스템의 다양성을 잘 말해준다고 하겠다.

2.2 실시간 처리가 많다

범용 시스템의 처리는 일반적으로 주어진 자원을 최대한 효과적으로 활용하여, 되도록 빠르게 수행하는 것을 목적으로 한다. 이는 절대적인 처리 기한이 없이 주어진 시간 안에 되도록 많은 일을 수행하도록 설계되었음을 의미한다. 이에 반해 내장형 시스템의 경우는 과제(task)의 처리 기한이 주어지는 실시간 처리가 많다. 예를 들어 목소리와 동영상이 같이 멀티미디어로 실행되어야 하는 경우, 목소리와 동영상은 실행 기한(deadline)이 주어져 있어 그 실행 기한 안에 데이터가 처리되지 않으면, 동영상이나 인터넷 전화에서 정보 전달이 제대로 되지 않는다. 다중의 매체가 동시에 처리되는 멀티미디어 처리기에서의 어려운 점은 각 매체의 종류마다, 실행 기한이 각 과제마다 다를 수 있으며, 데이터 처리의 정밀도 및 질(QoS: Quality of Service) 등이 모두 다르기 때문에 효과적인 실시간 처리는 매우 어려운 문제이다.

이런 실시간 처리는 실행 기한 내에서만 처리된다면, 무조건 빨리 처리되는 것이 바람직하다고 할 수 없기 때문에, 처리해야하는 모든 작업들의 우선순위를 효과적으로 조정하여 모든 작업이 각각의 제한 시간 안에 처리되도록 하는 것이 중요하다. 이는 내장형 시스템의 주요 특징이다. 또한 때로는 처리되어야 하는 속도가 외부의 환경적인 event에 적절히 반응해야 하는 reactive적인 성격이 강해서, 지나치게 비관적이지 않으면서도 주기적이나 비주기적인 event에 적절히 대응하기 위한 조건을 만족해야 한다. 실시간 처리는 실행 제한 시간을 절대로 어겨서는 안되는 hard realtime 시스템과, 어느 정도 어겨도 되는 soft realtime system으로 크게 나뉘며 자세한 설명은 참고 서적 [Shin, Liu]를 참조하기 바란다.

2.3 대량 생산을 목적으로 한다

우리의 생활 속에서 쉽게 발견되는 몇 가지 내장형 시스템의 예를 보아도, 내장형 시스템이 얼마나 우리의 실생활과 밀접하게 연관되어 있는지 알 수 있다.

예를 들어, 세탁기, 냉장고, 셋-탑 박스, 게임기 등의 기기들을 보아도, 대량 생산을 목적으로 최소비용에 최대 효과를 내야하는 기기 등임을 쉽게 알 수 있다. 즉, 내장형 시스템의 성능평가는 최종 제품의 성능에 의해 평가되지, 그 안에 쓰이는 하드웨어와 소프트웨어의 성능에 의해 평가되지는 않는다. 그러므로 비용의 최소화와 소비자의 다양한 욕구를 최대로 충족시켜야 하는 것이 중요하다. 예를 들어, 세탁기를 설계함에 있어 인텔 펜티움 프로세서를 쓴다면, 그 회사는 아무리 고성능 세탁기를 만든다 하더라도, 경쟁에 살아남기 힘들 것이다. 그런 의미에서 내장형 시스템의 설계 철학은 상당부분 마켓팅에 의존하며, 범용 컴퓨터 시스템의 설계와는 많이 달라야 하고 또 다르다.

2.4 거친 환경에 강한 내구성을 가져야 한다

많은 내장형 시스템은 고온이나 다습한 환경, 또는 충격이 가해지거나, 일부 기능에 이상이 있어도 기본 기능은 계속 동작하도록 요구되는 경우가 많다. 예를 들어, 자동차에 쓰이는 ABS의 경우, 50도 이상의 고온과 -30도 이하의 저온에서도 고장이 없는 내구성을 보여야 한다. 이는 많은 내장형 시스템이 험난한 환경에서 동작함과 동시에 절대적으로 오작동을 허락하지 않는 미사일 제어 시스템이라든가 ABS 등의 아주 중요한 작업을 수행하는 경우가 빈번하기 때문에 더욱 더 중요하다고 할 수 있다.

3. 내장형 시스템 설계 경향

앞에서도 언급한 바와 같이 내장형 시스템의 설계는 엄청난 변화를 겪고 있다. 이는 반도체 및 통신 기술의 눈부신 발전 속에, 과거에는 이룩할 수 없었던 "꿈의 기기"들이 이제는 현실로 가능해져 있음을 의미하는 것이기 때문에 이런 발전된 기술을 최대한도로 이용할 수 있는 설계 방법론에 생기고 있다. 이 중 몇 가지를 정리하면 다음과 같다.

3.1 HW/SW co-design co-verification

과거의 내장형 시스템의 설계는 하드웨어와 소프트웨어의 독립적인 개발팀에 의해 먼저 하드웨어가 설계되고, 소프트웨어는 이미 정해진 프로세서에 수행된다는 가정 하에 독립적으로 개발되는 형태로 많이 이루어져 왔다. 하지만, 이제는 그런 방법에 의한 설계는 경쟁력이 없다. 즉, 앞으로의 내장형 시스템의 설계는 하드웨어와 소프트웨어가 동시에 개발되면서, 가장 최적의 하드웨어 구조와 또 그 하드웨어를 최적으로 이용할 수 있는 소프트웨어 개발이 동시에 밀접한 상호 작용을 갖으며 이루어져야한다. 또한 하드웨어와 소프트웨어가 동시에 설계된다는 것은 결국 각 단계의 검증 작업도 하드웨어와 소프트웨어에서 동시에 수행되어야 한다는 것이고, 이런 설계의 중간 단계에서 하드웨어와 소프트웨어의 동시 검증 능력이 내장형 시스템의 빠르고 올바른 설계를 위해 필수적인 요건이 되고 있다.

3.2 Software portion의 증가

과거에는 특정 목적을 갖고 있는 내장형 시스템이 한 두가지의 목적만을 위해서 설계되었기 때문에 대부분 간단했다. 실제로 대부분의 소프트웨어는 하나의 기기의 수명이 다하는 동안 항상 일정한 기능만을 실행하기 때문에 ROM에 들어가 있는 경우가 일반적이었다. 현재에 들어 메모리 가격의 인하와 flash 메모리의 폭넓은 보급으로 소프트웨어의 활용이 더 용이해지면서 소프트웨어의 장점이라 할 수 있는 유연성과 기능 확장의 용이성이 가능한 점까지 더해져 점점 더 비중이 증가하고 있다. 더구나 최근의 휴대폰이나 멀티미디어 처리 기기는 소비자의 욕구가 자주 바뀌면서 유연한 기능의 개선이 가능해야 하는데 이를 위해서도 소프트웨어의 비중이 점점 커져야 하는 것이 현실이다.

3.3 System-on-chip

과거의 내장형 시스템은 일반적으로 보드(board)에 시스템이 구현되는 형태였으나 최근 들어 급속도로 발전된 반도체 기술은 엄청난 양의 복잡한 하드웨어를 하나의 칩 안에 넣을 수 있는 것이 가능해 졌다. 즉, 프로세서 뿐 아니라, 상당량의 메모리, 그리고 특화된 처리기 코어(core)와 주문형 반도체 회로, 아날로그 신호의 입출력을 위한 장치까지 다양한 종류의 하드웨어가 하나의 칩 안에 구현되어 이루어진 시스템으로 제품화되고 있다. 이는 초소형/초경량 시스템의 일반화를 의미하는 것으로 다방면에서 큰 파급효과를 주는 중요한 경향이라고 할 수 있다.

3.4 Faster-time-to-market (IP 사용의 증가)

내장형 시스템은 범용 컴퓨터의 설계에 비해 대량

생산을 하면서도 치열한 경쟁 속에 다양한 소비자의 욕구를 만족시키기 위해서는 빠른 시간 내에 경쟁력 있는 제품을 출시해야하는 부담이 있다. 이를 위해서는 먼저 빠른 설계가 필수적인데, 위에서도 언급한 바와 같이 내장형 시스템이 나날이 복잡해짐에 따라 빠른 시간 안에 모든 설계를 처음부터 시작해서 끝낸다는 것은 현실적으로 거의 불가능하다. 그래서 IP (intellectual property) block이라고 불리는 이미 설계된 모듈들을 재사용함으로써 설계의 시간을 단축시키는 방법이 점점 더 많이 사용되고 있다. 이 IP block들은 비교적 여러 종류의 vender 설계 기술과 설계 방법론 (design methodology)과 호환이 잘 되도록 설계되어 있다. 일반적으로는 과거 IP block은 하나의 회사에서 설계에 재사용되는 핵심 기술을 포함하는 의미하였으나, 현재에는, 결합이 용이하며, 유연하고, 기능에 대한 검증이 이루어진 IP block들의 보유가 빠른 시간 안에 전체 시스템의 검증이 용이한 시스템을 설계하는데 매우 중요한 일로 널리 인식되면서, IP block의 폭넓은 사용은 IP core만 전문적으로 설계하여 판매하는 회사들도 많이 늘어나고 있다.

4. 내장형 시스템 하드웨어

4.1 Processors

내장형 시스템의 성능이 복잡해진다는 것은, 처리과정 및 계산이 복잡해짐을 의미함으로 프로세서의 역할이 점점 더 중요해짐은 말 할 나위가 없다. 다만, 범용 시스템의 설계와는 달리, 내장형 시스템은 정해진 목적이 일정하기 때문에, 반드시 고성능일 필요가 없이 목적에만 가장 잘 맞는 선택을 해야 한다는 것이 중요하다. 범용 프로세서가 Intel, IBM, Motorola, AMD, Sun 등 비교적 소수의 제품들로 시장을 점유한 반면, 내장형 프로세서의 시장에서는 약 30년 전에 나온 Intel 8080부터 64-bit processor에 이르기까지 선택할 수 있는 프로세서의 종류가 무수히 많다.

하나의 프로세서 선택은 그와 같이 동작해야하는 버스의 구조 및 제반 주변 장치의 선택에도 지대한 영향을 주는 것은 물론, 프로세서에 수행되는 소프트웨어의 구조에도 막대한 영향을 미치기 때문에 아주 치명적이라 할 수 있다. 프로세서의 선택은 일단 마이크로-프로세서와 마이크로-콘트롤러의 선택으로 크게 나눌 수 있다.

마이크로-콘트롤러는 마이크로-프로세서와 아날로그 인터페이스나 메모리 같은 장치들이 하나에 집적된 형태로 값이 저렴하고, 또 이미 필요한 장치들이 결합되어 있는 상태로 여러 부품들의 결합 시에 생길 수 있는 문제가 거의 없어 널리 쓰인다. 하지만, 이런 결합된 형태의 마이크로-프로세서는 확장성이 떨어지며, 기능의 점진적인 개선에 따른 확장성이 약한 단점이 있다. 또한 마이크로-콘트롤러는 저렴한 가격에 간단한 시스템의 제어기로서 주로 쓰이므로, 고성능이나 저전력 등 특정 목적에 최적화된 성능을 보이기 어렵다. 그리하여 최근 들어 고성능 내장형 시스템에서는 주로 최첨단 마이크로-프로세서를 사용하는 경우가 많다.

고성능 마이크로-프로세서의 선택도 다양한데 크게 harvard/SHARC 구조, MIPS나 ARM같은 RISC (Reduced Instruction Set Computer)구조, Intel x86이나 Motolora 68000 series 같은 CISC (Complex Instruction Set Computer) 구조, 신호처리 기능이 최적화된 DSP (Digital signal processor), Network processor와 Multimedia Processor등으로 크게 나눌 수 있다. 이 분류는 편의상 나눈 것이고 실제로 요즘의 고성능 마이크로-프로세서 구조는 하나의 구조를 주로 이루면서도 다른 구조의 장점을 많이 갖고 있는 복합적인 형태를 갖고 있다고 보아야한다. 즉 Intel의 Pentium processor도 기본적으로는 CISC 형태이나 많은 부분이 RISC의 특징을 갖고 있고, 멀티미디어 처리의 강화를 위한 기능들이 많이 첨가되어 있다. 프로세서의 선택은 설계하려는 내장형 시스템의 주요 계산이 어떤 것이냐에 많이 좌우되는데, 그 밖에도 메모리, 클럭 스피드, 인터럽트 처리과정 등의 제반 성능, 운영체제 및 프로그램 개발 환경의 용이, 전력소모, 및 가격 그리고 주변 하드웨어와의 결합 용이성 등 여러 가지를 고려하여 이루어진다. 최근 들어서 멀티미디어 처리기능이 아주 중요해지면서 멀티미디어 처리에 효과적인 VLIW (Very Long Instruction Word), Superscalar, 또는 SIMD (Single Instruction stream Multiple Data Stream)의 구조를 갖는 마이크로-프로세서가 많이 쓰이고, 또 개발되고 있는 추세이다. 가장 대표적인 마이크로-프로세서로는 Motorola 68000 series와 PowerPC series, Intel사의 8051 series, x86 series, 최근에 개발된 Xscale. 그리고 MIPS와 strongARM도 널리 쓰이는 마이크로-프로세서라고 할 수 있다.

4.2 ASICs/Cores

반도체의 급속한 발전은 하나의 칩 안에 마이크로-프로세서 뿐 아니라 여러 가지 특화된 하드웨어의 포함을 가능하게 만들었다. 특화된 core의 대표적인 예는 음성처리를 위한 core, 동영상처리를 위한 codec회로, 또 음악소리의 처리를 위한 core등이 있으며, 이들은 프로세서에서 자주 수행되는 기능을 하드웨어적으로 구현함으로써 실행 속도를 크게 향상시키려는데 목적이 있다. 이런 core라고 불리우는 하드웨어는 흔히 IP (intellectual property)라고 불리는 것으로 일반적으로는 하나의 회사에 다시 재사용이 가능한 이미 설계된 블록을 의미하는 것인데 빠른 설계를 위해서도 많이 쓰이고, 또한 다른 시스템과 비교하여 월등한 성능을 내기 위한 핵심적인 기술을 포함하는 경우가 많아, 내장형 시스템의 설계에 있어 차지하는 비중이 매우 크다고 하겠다.

4.3 Interfaces

반도체기술의 발전은 과거에는 비교도 할 수 없을 정도로 다양한 하드웨어들을 하나의 칩 안에 포함시키는 것이 가능해 졌다. 그에 따라, 설계의 초점이 각 부분들의 최적 성능구현 못지않게 각 블록간의 호환적인 상호 작용이 중요해지고 있다. 즉, 여러 다양한 종류의 설계 기법과 방법론에 의해 설계된 다른 부분들이 어떤 종류의 버스에 의해 연결되며, 또한 어떤 종류의 I/O기법에 의해 통신하며, 만일 서로간의 통신 프로토콜 (protocol)상 서로 호환이 안 되는 경우가 있을 때, 이 문제를 어떻게 해결하느냐가 중요하다. 대표적인 I/O기법으로는 별도의 I/O 명령어에 의한 I/O와 I/O 장치들에게 일정한 메모리를 할당하고 그 메모리 주소로부터 읽고, 그 메모리 주소로 써서 I/O 장치와 통신하는 Memory -Mapped I/O라는 방법이 널리 쓰인다.

내장형 시스템에서 널리 쓰이는 버스의 구조는 VMEbus, compactPCI, I2C, USB, Firewire, SCSI, IEEE-488(1394) 등이 있으며, 전통적인 표준으로는 비동기식 직렬 (asynchronous serial) 전송의 RS-232C serial port와 parallel port interface 등이 있다. 특히, 무선 통신 인터페이스로 주목해야 하는 것은 근거리 무선 통신 표준인 Bluetooth와 802.11b이다.

대부분의 버스 표준은 고유의 전기적 특성과 기계적 특성을 정의하며, 또한 동기 또는 비동기, 또는 반동기 등의 통신 프로토콜을 정의한다. 이러한 버스 프로토콜의 특징은 주로 같이 사용되는 프로세서의 인터페이스와도 밀접하게 연관이 되어, 어떤 버스에 어떤 프로세서가 같이 사용될 수 있는지도 바로 전기/기계적인 호환성과 통신 프로토콜의 유사성에 의해 결정된다.

특히, 내장형 시스템에서 반드시 고려해야할 인터페이스는 아날로그 장치를 위한 인터페이스이다. 즉 어떻게 아날로그 입력을 처리가 용이한 디지털 형태로, 또 다시 아날로그 형태의 출력을 위해 디지털에서 다시 아날로그로 바꾸어주는 interface도 중요한 내장형 시스템의 구성요소이다. 멀티미디어 처리가 가장 중요한 성능 평가 요인의 하나이므로, 모뎀, 카메라, 음향기기, 무선 통신 기기 등 아날로그 입출력을 위한 연구는 매우 중요하며 끊임없이 연구되고 있다.

4.4 Memories

최근 몇 년 사이에 메모리 시장의 추세는 메모리 가격의 급격한 인하와 flash 메모리 시장의 급성장을 들 수 있다. 또한 메모리의 접근 시간 (access time) 면에서의 성능 향상은, 소프트웨어의 비중이 커지는 현대 내장형 시스템에서 중요한 역할을 한다. 메모리는 크게 EDO, DDR, RDRAM, SDRAM 등으로 불리우는 DRAM (dynamic random access memory)과, Cache등에서 쓰이는 SRAM (static random access memory) 그리고 ROM (read only memory)로 나눌 수 있다. ROM에는 크게 OTP(one-time-programmable) ROM, EPROM, EEPROM등이 있고, EEPROM의 일종으로 flash 메모리가 있다. 특히 flash 메모리는 읽고 쓰는 것이 매우 자유로우면서도 전원이 없어져도 내용을 저장하는 non-volatile 메모리로서 디지털 카메라나 휴대용 메모리 카드 등에 널리 쓰이면서 시장이 빠르게 성장하고 있다.

내장형 시스템의 설계에 있어 메모리의 선택은 크게 크기의 결정과 종류의 결정으로 나눌 수 있으며, 크기는 대량생산에 따른 원가 절감의 차원에서 필요한 기능을 수행하기 위해, 성능을 크게 떨어뜨리지 않으면서도 가격을 낮출 수 있는 정도에서 결정된다. 메모리의 크기는 크게 운영체제 및 수행 프로그램이 차지하는 공간과 데이터가 차지하는 공간, 그리고 운영체제의 메모리 관리 능력에 의해 결정된다. 프로그램의 크기에 따른 고려라고 함은 내장형 시스템이 수행하는 기능이 어떤 언어 (C나 어셈블리 또는 Java)

로 사용되었느냐에 따라 프로그램이 저장되어야 할 공간이 달라지며, 또한 추후에 기능이 추가됨으로써 프로그램의 크기가 커질 때를 대비하여 여유의 공간을 두어야하는 것을 의미한다. 또 데이터의 공간은, 내장형 시스템의 기능상 얼마나 많은 정적 또는 동적인 임시 저장장소가 할당되어야 하는지에 따라 메모리의 크기가 결정되어야 한다. 이는 운영체제가 MMU(memory management unit)가 있어 효과적으로 메모리를 관리할 수 있는지에 따라서도 영향을 받는다. 메모리의 성능에 따른 구분은 가장 빠른 순서부터 레지스터 파일, SRAM, DRAM, Hard-disk 순이며, 일반적으로 성능대 가격의 trade -off를 고려하여 결정된다. 대표적인 예가 graphic card이며, 고성능을 유지하기 위해서 고가의 제품에서는 빠른 SRAM을 쓰며, 중저가의 제품에서는 DRAM의 일종인 DDR (double data rate) 등의 메모리가 널리 쓰인다.

4.5 주변 장치

내장형 시스템과 연결되는 주변 장치의 종류는 너무 다양하여 일일이 열거하기가 쉽지 않다. 대표적으로 카운터 및 타이머, 그리고 DMA (direct memory access) 및 인터럽트 처리기, 디스크, 키보드, 모니터 같은 입출력 장치, 무/유선 통신 장치, 그리고 아날로그 장치 등으로 크게 나누어 생각해 볼 수 있다. 카운터/타이머는 말 그대로 어떤 일들의 순서를 정하거나 동기화를 위한 참조용, 또는 어떤 event의 발생 회수 등을 기록하는 데 쓰인다. 하드-디스크간의 데이터 전송 같은 대량 정보의 전송을 주 프로세서의 관여 없이 원활히 수행하기 위한 DMA controller가 있으며, 다양한 장치로부터의 인터럽트 우선순위에 의한 효과적인 인터럽트 처리를 위한 인터럽트 처리기가 있다. 또한, 잘 알려진 키보드나 디스크, 그리고 모니터 등의 장치들이 주변장치라고 할 수 있으며, 모뎀이나 넷트워크 라우터, 무선 통신용 access point등도 그 자체로 내장형 시스템이면서 다른 내장형 시스템의 주변장치에 포함된다고 하겠다. RF(radio frequency), 동영상과 음성/음악 처리 등의 다양한 아날로그 장치들 또한 중요한 내장형 시스템의 구성요소이다. 내장형 시스템의 기능이 복잡해지고 고성능이 요구되는 현실에서 과거 하나에 모든 것이 집적되어 있는 간단한 micro-controller에 비해서, 다양한 종류의 장치들이 고성능 내장형 시스템의 주 장치에 부착되어 내장형 시스템의 기능성과 유연성을 높이는데 기여하고 있다.

5. 내장형 시스템 소프트웨어

여태까지는 내장형 시스템의 하드웨어에 관해 살펴보았다. 이제는 내장형 시스템의 소프트웨어에 대해 살펴본다. 현대에 있어 내장형 소프트웨어의 비중은 엄청나게 증가하고 있으며 하드웨어와 소프트웨어의 통합 개발이 일반화되고 있어 과거에는 단순한 기능만을 요구했던 내장형 시스템 설계자들에게 많은 관심이 되고 있는 분야이다. 이에 대해 대략적으로 중요한 주제에 대해 살펴보자.

5.1 실시간 Operating System (운영체제)

운영체제라고 함은 사용자와 하드웨어에 구조에 의거한 저 단계 인터페이스 간의 중간자 역할을 하는 소프트웨어 환경이라 하겠다. 즉, 운영체제의 역할로 인해, 사용자는 하드웨어 지식 없이도, 소프트웨어 작성 이 가능하고, 하드웨어를 효과적으로 사용할 수 있게 된다. 특히 파일 시스템의 관리와 장치를 제어하는 일, 프로그램에서의 파일 개폐와 입출력 처리, 과적으로 가상메모리와 실제 메모리상의 처리를 관리하는 일, 그리고 인터럽트 처리 등은 사용자가 직접 프로그램으로 해결하기 힘든 일들을 운영체제가 해결함으로써 프로그래머의 부담을 줄여 준다. 현대의 운영체제는 동시에 여러 작업을 처리하는 멀티태스킹 OS가 일반적이다. 이는 동시에 여러 작업을 처리하게 해줄 뿐 아니라, 각 처리해야하는 작업의 우선순위를 부여하여 중요한 작업이 빨리 끝날 수 있도록 preemption할 수 있는 기능이 갖추어 져야 한다.

앞에서도 언급하였듯이 내장형 시스템에서의 작업은 실시간 처리를 요구하는 경우가 많다. 이와 같이 각 작업들이 처리해야 하는 최소 처리 속도나 처리기한이 주어져 있을 때, 작업 처리의 기준이 모든 작업의 실행 제약시간에 어김없이 처리하도록 작업을 실행하는 기능을 가진 운영체제를 실시간 운영체제 (real-time operating system)이라 한다. 크게 실시간 처리에는 제약시간을 어겨서는 절대로 안 되는 항공기 제어와 같은 hard real time 처리와 동영상 처리와 같이 때때로 여겨도, 질적인 저하 정도의 문제를 발생시키는 soft realtime 처리가 있다. 제약시간을 맞추어 주는 기능을 수행하기 위해 대부분의 운영체제는 기본적으로 멀티태스킹이 가능해야 하고, 시

간 분할, preemption 기능이 갖추어져야 한다. 이는 대부분 interrupt에 의해 구현되기 때문에, 실시간 OS에서 인터럽트의 처리 능력도 매우 중요하다고 하겠다. 또한 작업의 중요성이 외부 환경에 따라 변하는 event-driven 스케쥴링 기능도 갖추어야한다.

널리 쓰이는 내장형 시스템의 운영체제는 매우 다양하나, 대표적인 예로 MS-DOS, VxWORKS, Windows-CE, 그리고 linux를 들 수 있다. 이들 중 특히 open source인 linux는 내장형 시스템의 운영체제로서 많은 개발자들에게 관심의 대상이 되고 있다.

5.2 Retargetable/Cross 컴파일러

과거의 내장형 시스템은 기능이 간단하고, 프로그램의 저장 공간을 최소화하기 위해 어셈블리 언어로 소프트웨어가 구현되는 것이 일반적이었다. 기능이 간단할 뿐 아니라 하드웨어도 고정되어 있고, 또 기능이 장치의 수명이 다할 때까지 거의 바뀌지 않는 것이 일반적이라서 일단 실행 코드가 만들어지면 가장 값이 싼 ROM에 저장되는 것이 일반적이었다. 이는 현실적으로 과거의 내장형 시스템이 먼저 하드웨어가 설계되어 고정되어 있고, 이 고정된 하드웨어에 맞는 소프트웨어를 개발하는 설계 방법이 대부분이었기 때문에 가능했던 일이다. 또한 하드웨어가 복잡하지 않고, 기능 역시 복잡하지 않아 바로 하드웨어를 직접 제어할 수 있는 어셈블리 언어로의 소프트웨어 구현이 많은 장점이 있었다고 할 수 있다. 하지만 현대에 들어, 앞에서도 지적한 바와 같이 내장형 시스템이 복잡해지고, 또한 보다 고성능의 내장형 시스템 설계를 위해 이제는 소프트웨어와 하드웨어가 같이 동시에 개발되면서, 고정된 하드웨어를 가정하기 어렵고, 개발 중 기능도 끊임없이 변하기 때문에 어셈블리로 작성한 소프트웨어의 부분은 프로그램의 확장과 디버깅에 어려움이 많아 차지하는 비중이 점차 줄어들고 있다. 즉 다시 말해 C나 Java와 같은 상위-단계 언어(high-level language)에 의한 프로그램 작성이 일반화 되어가고 있다.

상위 단계의 언어로 작성 시 GNU gcc와 같은 널리 쓰이는 컴파일러를 사용해서 실행 코드를 생성하게 된다. gcc와 같은 컴파일러는 일반적인 범용 프로세서를 가정하고 만든 것이라 내장형 시스템의 개발 도구로서는 몇 가지 한계가 있다. 첫째, gcc는 일반적으로 잘 알려진 컴퓨터 구조에서 효과적으로 실행되는 코드를 만들어 내지, 현재 개발 중인 (즉, 아직은 존재하지 않는) 구조에 대한 코드를 만들어 내는 데 제한적이라는 문제가 있다. 그래서 하나의 프로그램을 컴파일 하면서, 컴파일러가 실행되는 기계가 아니라 다른 기계에서 실행이 되는 코드를 만드는 cross compiling이 내장형 시스템에서는 무척 중요하다고 할 수 있다. 둘째로 내장형 시스템은 몇 가지 특수 목적을 수행하기 위한 매우 독특한 하드웨어 구조를 갖고 있기 때문에 컴파일러가 실행될 타겟 구조의 특성을 잘 이해할 수 있도록 많은 패러미터들에 의해 customized될 수 있어야 하고, 또한 이런 기능들을 통해 개발 중이나 또는 향후에 upgrade를 위해서 하드웨어가 변할 때, 그 개선된 부분을 효과적으로 활용할 수 있는 코드의 생성이 중요하다고 하겠다. 이렇게 계속 점진적으로 변하거나, 어떤 특정 목적을 효과적으로 수행하기 위해 특정 core 하드웨어 첨가 시 그에 따른 효과적인 코드를 생성할 수 있는 컴파일러를 retargetable compiler라고 부르는데, 이는 내장형 시스템의 설계에 있어 아주 효과적인 소프트웨어의 설계를 위해 아주 중요한 부분을 차지하고 있다.

6. Modeling 기법 및 Embedded System 설계 언어

과거의 내장형 설계의 특징 중의 하나는 하드웨어의 비중이 크고, 또 하드웨어와 소프트웨어의 설계가 순차적이면서 독립적으로 수행되어 왔다는 것이다. 현재에도 통합 설계가 일반화되지 않은 많은 내장형 시스템들이 소프트웨어와 하드웨어의 독립적인 설계에 의해 구현되어 왔다. 따라서 하드웨어의 설계를 위한 기능을 표시하는 언어는 대부분 VHDL이나 Verilog와 같은 HDL(hardware description language)가 널리 쓰이고, 소프트웨어는 C와 같은 언어로 구현되는 것이 일반적이다. 그리하여, 개발 도중에 하드웨어와 소프트웨어의 성능 및 기능 검증을 위해 혼합 언어의 시뮬레이션 (예: Verilog의 PLI)을 수행하는 것이 일반적이다. 하지만 앞으로 통합설계가 보다 일반화되는 상황에서는 어떤 기능이 하드웨어로, 또는 어떤 기능이 소프트웨어로 구현된다는 것이 설계 진행 단계에 변할 수 있기 때문에, 서로 다른 언어로 소프트웨어와 하드웨어를 구분한다는 것에 많은 제약이 따른다. 그리하여, 내장형 시스템을 통합하여 모델링 할 수 있는 언어의 표준화가 매우 시급

한 문제라고 하겠다. 그의 일환으로 UML (Unified Modeling Language)가 많은 관심을 끌고 있다. 이는 원래 object-oriented system의 기능을 설명하기 위한 표시 방법이었으나 내장형 시스템의 하드웨어와 소프트웨어를 모두 모델링 하는데 유용하다고 할 수 있다.

최근에 기본 UML을 확장하여 실시간 내장형 시스템을 위한 실시간 UML에 대한 연구가 활발하다. 또한 C++/C의 확장을 통한 하드웨어의 모델링인 SystemC란 언어, 그리고 Verilog와 C의 확장을 시켜 개발한 언어인 Superlog등이 점차 많은 관심을 끌고 있다. 이러한 modeling의 기본은 소프트웨어의 모델링에서부터 실행 시간에 예상이 가능하며, 통신 메카니즘이 강화되고, 또한 내장형 소프트웨어에 적합한 schedulability와 같은 개념들을 모델링할 수 있도록 확장시킨 것들인데 아직 많은 연구가 진행되어야 할 단계이다.

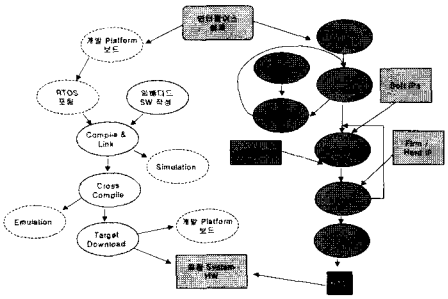

그림 1

7. 설계 단계와 개발 환경 및 검증 과정

7.1 설계 단계

설계의 절차에 대한 방법론의 중요성은 때로는 간과되기 쉽지만, 시스템이 복잡해질수록 더욱더 중요해 짐은 말할 나위가 없다. 이는 설계 방법론 (design methodology)이 설계시간을 줄이고 (time-to-market), 비용을 최소화하며 (low cost), 또한 최선의 품질 (best quality)을 만들어 내는데 있어 절대적인 영향을 미치기 때문이다. 현재로서 가장 일반적으로 받아들여지는 설계의 과정은 (그림 1) 먼저 설계의 기능 및 요구조건의 명백한 정의가 되고, 전체적인 구조, 즉 소프트웨어와 하드웨어 중 어떤 것으로 어떤 기능과 어떤 요구조건을 만족시킬 것인가 (HW/SW partitioning)에 대한 전체적인 구조를 결정하고, 그 다음에는 병렬적으로 하드웨어와 소프트웨어의 개발이 동시에 진행된다. 실제적인 설계의 검증에 걸리는 시간을 단축시키기 위해 계속적으로 소프트웨어와 하드웨어의 설계 중간 결과가 서로 같이 검증되어 각자 설계가 끝나면, 하나의 시스템으로 결합 (system integration) 단계가 있고, 각각의 테스팅이 아닌 하나의 시스템으로서 검증되는 system testing의 단계를 거치면 하나의 시스템의 설계가 완성된다고 하겠다.

7.2 설계의 검증

하드웨어 설계의 꽃이라 할 수 있는 마이크로-프로세서의 설계에 있어 이미 설계에 쓰는 인적/시간적 비용보다 설계 검증에 쓰이는 인적/시간적 비용이 훨씬 더 많이 소요되는 현실은 이미 잘 알려져 있다. 앞으로도 설계 검증이 차지하는 비중은 점점 더 커질 것이 분명하다. 전통적으로 하드웨어의 설계의 검증에는 시뮬레이션과 에뮬레이션이 쓰였다. 또한 소프트웨어의 검증에는 gdb같은 디버깅 도구들이 널리 사용되어 왔다.

시뮬레이션이라는 것은 각 부분의 기능을 표현하는 소프트웨어와 설계된 구조를 소프트웨어로 표현하여 원하던 기능이 잘 구현되었는지를 프로그램을 통하여 확인하는 절차인데, 장점으로는 비용이 적게 들고, 실행이 용이한 반면, 가능한 모든 입력 값에 대한 시뮬레이션을 돌리는 일이라든가 또는 실제 설계를 바탕으로 하는 것이 아니기에 실제 설계에서 나타날 수 있는 많은 문제점을 발견하기에는 미흡한 면이 있다. 반면 에뮬레이션이라고 하는 것은 실제로 구현하고자 하는 설계를 하드웨어로 prototyping하여 비용은 많이 들지만 빠른 시간에 실제 실행하면서 문제점들을 발견하고자 하는 기법이다. 특히나 실시간 처리를 테스팅하기 위해서는실제 target system과 거의 일치하는 하드웨어에서 테스팅하는 ICE (in-circuit-emulation)이 많이 쓰이며 테스팅 방법으로는 SCAN기법과 JTAG 등 기법이 널리 쓰인다. 더불어 현대에 들어 소프트웨어와 하드웨어가 같이 존재하여 동시에 개발되는 현재의 내장형 시스템 설계에서 설계과정의 동적인 변화, 즉 어떤 기능이 하드웨어로 또는 소프트웨어로 구현될 것인가에 대한 결정이 가변적인 상황에서, 계속 개발되는 과정에 동시에 중간단계에서 이루어지는 시뮬레이션 기법 (co-

simulation)이 중요하다. 또한 보다 검증에 필요한 노력을 최소화하기 위해 이미 설계 및 사용이 되어 검증이 끝나있는 IP block들을 많이 사용함으로써 새로운 설계에서 야기하는 많은 실수들을 줄이는 노력이 많으며, 아예 설계의 품질이 얼마나 테스트가 쉽게 되어있는지에 의해 결정되는 design for testability에 대한 연구도 무척 활발하다고 하겠다.

참고문헌

[1] Stuart R. Ball, "Embedded Microprocessor Systems : Real World Design(2nd Ed.)", Newnes, 2000

[2] Hermann Kopetz, "Real-Time Systems:Design Principles for Distributed Embedded Applications", Kluwer Academic Publisher, 1997

[3] Michael Barr, "Programming Embedded Systems in C and C+-", O'Reilly, 1999

[4] Wayne Wolf, "Computers as "Components", Morgan Kaufmann Publishers, 2001

[5] Alan Clements, "Microprocessor Systems Design-68000 Hardware, Software, and Interfacing(3rd Ed.)", PWS Publisher, 1998

[6] Jan Rabaey, "Digital Integrated Circuits - A Design Perspective", Prentice Hall, 1997

[7] Steve Heath, "Embedded Systems Design", Newnes, 1997

[8] Phil Lapsley외 3인, "DSP Processor Fundamentals-Architectures and Features", Berkeley Design Technology, Inc.

[9] Clifford Liem, "Retargetable Compilers for Embedded Core Processor", Kluwer Academic Publisher

[10] Ken Short, "Embedded Microprocessor Systems Design: An Introduction Using the Intel 80C188EB", Prentice Hall, 1998

[11] Jane Liu, "Real-Time Systems", McGraw Hill, 2001

[12] www.embedded.com

[13] www.linuxdevices.com (내장형 리눅스 포탈 사이트)

[14] Kang Shin 외 1인 "Real-Time Systems", McGraw Hill, 1996

[15] Rainer Leupers, "Code Optimization Techniques for Embedded Processors: Methods, Algorithms and Tools", Kluwer Academic Publisher, 2000

[16] Jack Ganssle, "The Art of Programming Embedded Systems", Academic Press, 1992

[17] Jean J. Labrosse, "Embedded Systems Building Blocks: Complete and Ready-to-Use modules in C", CMP books, 2000

[18] Ed Sutter, "Embedded Systems Firmware Demystified", CMP books, 2002

[19] Borko Furht, "Multimedia Systems and Techniques", Kluwer Academic Publisher, 1996

[20] Keshab Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation", Wiley- Interscience, 1999

[21] Keshab Parthi, "Digital Signal Processsing for Multimedia Systems" Wiley-Interscience, 1999

[22] www.bdti.com

[23] www.cecs.uci.edu

[24] Michael Keating 외 1인, "Reuse Methodolgy Manual for System-On-A-Chip Designs", Kluwer Academic Publisher, 1999

[25] S. B. Furber 외 1인, "ARM System-On-chip Architecture (2nd ed.)", Addison-Wesley Pub. 2000

정 기 석

1989 서울대학교 컴퓨터공학과(학사)
1998 미국 일리노이 주립대(박사)
현재 홍익대학교 컴퓨터공학과 조교수
E-mail:kchung@cs.hongik.ac.kr

김 태 환

1985 서울대학교 계산통계학과(학사)
1987 서울대학교 계산통계학과(석사)
1993 일리노이주립대(박사)
현재 한국과학기술원 전산학과 부교수
E mail:tkim@cs.kaist.ac.kr