임베디드 시스템 소개

이 영 대*

1. 서 론

최근 IC 설계 및 제조 기술의 급격한 진보에 따라 CPU는 점점 저렴해지고 있으며 점점 많은 정보 가전제품에 임베디드 시스템이 적용되고 있다. 임베디드 시스템은 다양한 분야에서 응용된다. 예를 들어, 카메라, 캠코더, 마이크로 오븐와 같은 가전제품이나 엔진제어기, 엔티록(anti-lock) 브레이크 등과 같은 자동차분야, 산업용 로봇, 공정제어기 등과 같은 산전제어기, 미사일, 항공기 등의 군수분야, 프린터, 팩스 라우터 등과 같은 컴퓨터 및 통신관련 제품 그리고 PDA, 화상회의 장비서버, 대화형 게임기, TV, 세톱박스 등의 다양한 분야에 적용되고 있다[1].

오늘날 임베디드 시스템의 융성을 가져오게 된 이유는 0.3 미크론 이하로 소자 면적을 축소할 수 있는 집적화 기술의 발전으로 1 제곱인치의 다이에 20개의 386 프로세서 코어를 구현 가능하게 된 반도체 생산 기술의 발전과 CAD 및 소프트웨어 기술의 진보에 따라 설계 기술이 제품 품질을 좌우할 정도로 발전한 점을 들 수 있다.

1995년 통계로는 세계적으로 팔린 약 30억 개의 CPU 중 인텔 프로세서 수는 1% 정도인 데 비해 4비트, 8비트와 같은 소형 CPU의 수를 포함한 임베디드 프로세서 계열이 99% 정도로 이미 비 PC 계열의 CPU가 양적으로 다수를 차지한다. 2002년에 임베디드 시스템 시장은 약 310억 달러로 예상되며 이는 약 465억 달러 정도인 일반 데스크탑 컴퓨터 시장에 버금간다. 그리고 일반 컴퓨터 프로세서의 성장세는 약 10%인 반면 임베디드 프로세서의 성장세는 약 18%로 보다 더 클 것으로 예상된다. 어떤 시장전문가들은 현재에도 임베디드 시스템의 시장이 전형적인 PC CPU에 비해 10배 이상이며 향후 십 년 이내에 PC를 포함한 컴퓨터 시장을 압도할 것으로 예측하기도 한다[2-4].

본고에서는 근래에 각광을 받고 있는 임베디드 시스템의 주요 개념을 소개하는데 초점을 둔다. 먼저 임베디드 시스템에 주요 특징에 대해 알아본 후 실시간 특징이 중요한 임베디드 운영체계에 대해 살펴본다. 그리고 최근의 하드웨어 소프트웨어 병행 설계 및 해석이 중요한 이슈로 대두되고 있는 임베디드 시스템의 설계에 대해 간략히 기술 한다.

2. 임베디드 시스템의 특징

임베디드 시스템은 데스크탑 컴퓨터를 이용한 스프레드 쉬트나 워드프로세싱 대신에 머신 비전을 이용한 로봇 동작 제어와 같이 외부 환경과의 상호 작용하는 응용분야에 사용된다. 전통적으로 임베디드 시스템은 전기-기계시스템의 제어나

* 세명대학교 정보통신과 부교수

유한상태머신(finite state machine) 및 신호처리 알고리즘에 종종 사용되어 왔다.

임베디드 시스템에는 CPU와 메모리에 추가하여 외부 환경에 상호 작용하여 측정, 조작할 수 있도록 다양한 인터페이스가 존재한다. 그림 1은 전형적인 임베디드 시스템의 일례로 현장에서 프로그램이 가능한 FPGA, 특화된 응용목적의 ASIC, 아날로그 하드웨어 및 사용자 인터페이스가 주변의 전기기계 시스템에 대한 오류를 감지하고 반응하도록 시스템을 구성하고 있다[5].

이와 같이, 임베디드 시스템은 특정목적을 위해 하드웨어 및 소프트웨어가 밀접히 결합하는 것으로 일반적인 목적을 위해 사용되는 컴퓨터와 대조되며 시간 제한적인 환경에 신속히 대응해야 한다.

임베디드 시스템은 데이타 획득 및 처리부, 통신부, 시스템 로직 및 제어 알고리즘, 인터페이스부 및 디스플레이, 저장장치, 모니터링과 보호, 테스트와 진단을 담당하는 보조 유닛부로 구성된다. 임베디드 시스템에서의 하드웨어는 프로세서 코어, 특정 응용을 위한 게이트 메모리, I/O 가 포함되고 DSP를 채용하기도 한다. 하나 또는 몇 개의

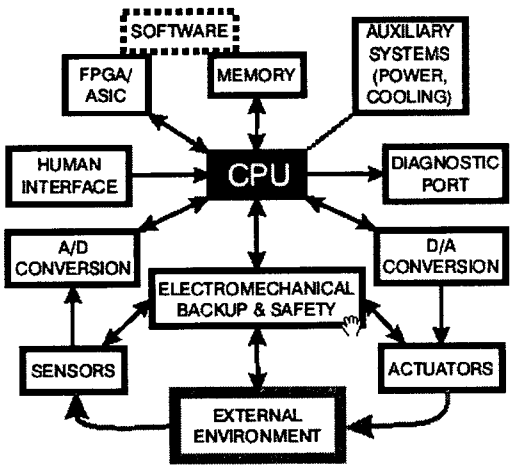

그림 1. 임베디드 시스템 일례

긴밀히 작용하는 기능들을 수행해야 하고, 점차적으로 고 기능화 및 실시간 제한적으로 변화하는 추세로 전력, 비용 및 신뢰성이 설계에 영향을 주는 중요한 인자이다. 응용 분야를 특화한 임베디드 프로세서의 장점은 칩 면적과 비용을 줄일 수 있고, 저 전력을 구현할 수 있다. 반면 하드웨어 및 소프트웨어 개발에 대한 오버헤드가 가중되고 출시가 지연될 수 있는 단점을 가진다.

3. 임베디드 운영체계

PC나 워크 스테이션의 경우 호스트 플랫폼이 곧 타겟 플랫폼으로 윈도우 NT, 윈도우 95/98, 솔라이스, 리눅스 등이 호화로운 환경을 제공하나 임베디드 운영체계의 경우 현재도 어떤 임베디드 운영체계가 사용되는 지 다 파악할 수 없을 수 정도로 다채롭다. 그 이유는 수많은 임베디드 시스템의 제조업체가 독자적인 운영체계(proprietary OS)를 사용하고 이 들이 임베디드 OS 시장의 반 이상을 차지하기 때문이다. 그 외에도 상용 OS (commercial OS)라 불리는 운영체계가 있는 데 독자적인 OS와 이것과의 비율은 현재 약 53 대 47로 파악되고 있다[6].

PC 플랫폼에서는 몇 개의 운영체계가 대부분의 시장을 장악하고 있는 반면 임베디드 시스템에서는 그렇지 못한 이유는 몇 개의 운영체계가 수많은 임베디드 시스템 제품의 요구 사양과 조건을 모두 최적으로 만족시키기 어렵기 때문에 하나 하나의 시스템에 가장 적합한 임베디드 운영체계가 채택되기 때문이다. 임베디드 운영체계 또한 간단하고 빠른 쓰레드로만 구성된 단일 운영체계로부터 멀티 태스킹과 마이크로 커널 구조를 갖는 운영체계에 이르기까지 다양하다. 이와는 달리 하나의 기능만을 수행하므로 아예 운영체계가 없는 임베디드 시스템도 무수히 많다.

임베디드 시스템의 운영체계는 관점에 따라 내장된 실시간 시스템이 경성인지 연성여부, 시스템이 오류에 대해 안전을 보장여부, 응답 마감을 보장하는 지 또는 최선의 노력(best effor)만 기울이는지, 지원된 자원(resource)의 적정 여부 및 사건 유발 또는 시간 유발 시스템 인 지에 따라 다차원적으로 분류될 수 있다. 이 중 실시간 시스템은 매우 중요한데 경성(hard) 실시간 시스템과 연성(soft) 실시간 시스템으로 구분된다.

경성 실시간 시스템은 반드시 시간 제약 사항을 만족시켜야 하는 시스템으로 예를 들어 항공기, 위성, 우주탐사, 미사일 방어 시스템, 핵 반응로, DRAM이나 비디오 리프레쉬 등에는 실행 마감 시간을 놓치면 치명적인 문제가 발생하며 결함에 대해 높은 신뢰도를 가져야 한다.

연성 실시간 시스템은 실행 마감 시간이 있기는 하지만 만족시키지 않아도 큰 피해를 입히지는 않으며 멀티미디어 시스템 등에 사용한다. 예를 들어 레이저 프린터는 분당 인쇄하는 페이지로 정격 시간 데드라인이 있으나 이는 인쇄하는 페이지의 복잡도에 따라 의존하므로 연성 실시간 시스템으로 충분하다. 전자 레인지, 전자 밥솥, 주문자 비디오(VOD),개인 휴대 정보 기기(PDA), 휴대폰, 라우터, 임베디드 웹 서버 기기 등이 이에 해당한다.

구조적인 측면에서 임베디드 운영체계를 멀티 쓰레드와 멀티 프로세스 모델로 구분하기고 한다. 멀티 쓰레드 실시간 운영체계의 경우 커널부과 응용 프로그램의 구분이 없이 하나로 동작하는 구조로 코드 크기가 작고 소형 시스템에 쉽게 구현할 수 있는 장점이 반면 사소한 버그가 시스템을 파괴하기는 단점이 있다. VxWorks, OSE, VRTX, pSOS, Nuclear Plus, Super Task, MicroC/OS-II 등이 이에 속한다.

VxWorks의 경우 현재까지는 세계적으로 점유율이 매우 높은 것으로 많은 마이크로 프로세서를 지원하고 있으며 VRTX는 몇 년전만해도 국내에서 시장 점유율이 상당히 높았었다. Nuclear Plus는 운영체계의 풀소스를 지원하고 제품당 로열티가 없으며 인공위성 우리별 1,2호에도 탑재된 것으로 알려져 있다. MicroC/ OS-II(uC/OS-II)는 Jean J. Labrosse라는 개인이 만들어 배포한 것으로 작은 크기의 실시간 운영체계이며 책을 구입시 풀소스를 제공하고 로열티가 없으며 업그레이드를 통해 낳은 종류의 프로세서를 지원하고 있다.

멀티 프로세스 실시간 운영체계의 경우 커널과 응용 프로그램이 독립적으로 동작하도록 설계되어 각 응용 프로그램의 메모리가 보호된다. 따라서 응용 프로그램의 변경이 용이하고 모듈화가 쉬우며 대형 시스템 개발에 적합하지만 운영체계의 코드 크기가 커서 작은 시스템에는 부적합한 단점이 있다. QNX, OS-9, LynxOS, Windows CE, RTLinux 등이 이에 속한다.

QNX는 해외에서 많이 사용되는 편으로 유닉스와 호환 가능하며 OS-9도 해외에서 높은 인지도와 점유율을 보인다. Windows CE는 MS사에서 판매하는 임베디드 OS이다. 근래에 매우 각광받고 있는 임베디드 리눅스는 국내에서도 많은 회사에 의해 개발되어 있으며 통신 라우터나 스위치 및 인터넷 어플라이언스(appliance)와 자동차 등에 적용되고 있다. 그 외에도 다양한 상용 또는 독자 임베디드 운영체계가 있으나 더 이상은 언급하지 않기로 한다.

임베디드 실시간 운영체계는 앞에선 언급한 두 모델이 가진 장단점을 고려하여 크기가 비교적 작고 덜 복잡한 시스템에는 멀티 쓰레드 모델을 사용하고 대규모의 복잡한 시스템의 개발 시에는 멀티 프로세스 모델을 사용하는 것이 적정한 것으로 알려져 있다.

최근의 임베디드 운영체계는 CISC 계열인 x86에서 ARM에 이르기까지 매우 제한된 자원을 가

진 CPU 구조를 주 타겟으로 하고 있으며 하드웨어를 최적으로 사용할 수 있도록 커널의 소형화 및 모듈화 경향을 보이고 있으며 재사용이 가능하고 재구성이 가능하도록(resuable and reconfigurable) 개발이 진행되는 추세이다. 하드웨어 관점에서 임베디드 시스템의 CPU는 실시간적으로 구성되는 추세인데 예를 들어 전력소비를 감소시킬 수 있도록 명령어(operand)의 비트폭을 줄일 수 있게 하기도 한다. 따라서 재구성 가능한 임베디드 하드웨어 플랫폼에 적합한 새로운 임베디드 운영체계의 개발이 요구되고 있다. 소프트웨어 관점에서 임베디드 운영체계를 시스템내에서 불변으로 고정되는 것과, 요구시 다운로드나 업로드되어 갱신되는 것으로 나눌 수도 있는 데 후자의 경우인 임베디드 자바가 이런 추세로 발전하고 있다.

앞으로는 제조업체의 독자적인 임베디드 운영체계 개발보다는 상용 임베디드 운영체계의 사용율이 높아질 것이다. 가중되는 적시 시장 출하 압력과 정보가전 제품의 소프트웨어 지원 서비스를 위한 표준화 때문에 점점 더 독자적인 운영체계보다는 상용 임베디드 운영체계의 사용 비율이 높아질 것으로 기대된다.

4. 임베디드 시스템 설계

임베디드 시스템은 마이크로 프로세서, DSP 코어, 네트웍 및 버스 인터페이스 등의 다양한 컴포넌트가 임베디드 프로세서에 같이 내장되어지면서 하드웨어 및 소프트웨어 설계를 병렬로 진행하며 하드웨어와 소프트웨어의 설계의 각 단계 사이에 상호 작용이 이루어지고 있다. 또한 앞의 설계 단계는 다음 단계의 결과에서 피드백을 받아 개선이 이루어지는 방법을 취하고 있다.

이는 개발자의 요구 사양에 적합한 성능(속도, 전력, 비용, 신뢰성)을 만족하는 제품을 시장에 적시에 출시하고 다양한 제품군으로 시장을 선점하기 위함이다.

기존의 생산 개념에서는 필요 이상의 기능을 가진 범용 디지털 게이트 들을 시장에 양산 출시하고 개발자가 이 들을 이용해 제품을 탄생시키는 방법을 사용하였으나 현재는 매일 다양하고 새로운 기능 요구 사양이 시장에 분출되므로 이를 만족하기 위해 적시에 적합한 임베디드 제품을 시장에 출하하는 것이 중요하다. 이를 위해 최신 기술의 임베디드 컴포넌트를 조립하여 시스템을 신속히 구성하는 추세이다.

임베디드 시스템의 설계의 주요 과정을 살펴보면 모델링 단계, 기능 구분 및 세분화 단계, 하드웨어와 소프트웨어 모듈의 구분 및 할당 단계, 스케쥴링 단계 및 장착단계로 구분된다[1]. 모델링 단계에서는 시스템에 내포할 알고리즘과 더불어 설계되고 실험된다. 기능 구분 및 세부화 단계에서는 임베디드 시스템에 요구되는 기능들이 더 작은 조각으로 세분되고 상호 연계된다. 하드웨어/소프트웨어 구분 및 할당 단계에서는 세분화 된 요소들을 하드웨어 유닛이나 상용 마이크로 프로세서에서 소프트웨어를 실행한다. 스케쥴링 단계에서는 각 기능들이 수행될 시간을 배정하게 되는데 이것은 하나의 하드웨어 유닛을 몇 개의 모듈들이 공유할 때 중요하다. 마지막으로 장착 단계에서는 개발된 기능들을 하드웨어 및 프로세서에서 소프트웨어로 실행될 수 있도록 이식하게 된다.

임베디드 시스템과 같은 극단적인 다양성이 존재하는 상황에서는 설계의 일반화가 매우 어렵다고 할 수 있다. 기존의 임베디드 시스템의 개발방법은 하드웨어 및 소프트웨어의 구분 및 세부화(partitioning)를 초기에 결정하고 이후로는 두 부분을 별도로 진행하는 과정을 취하였다. 그리고 CAD로 하드웨어를 드로잉(drawing)하고 해석

(synthesis)하는 과정을 취하였다.

최근의 임베디드 시스템 설계에서 전반적으로 부가되고 있는 흥미로운 부분 중 하나는 하드웨어 및 소프트웨어의 병행설계에 대한 것이다[6,7]. 이 방법에서는 하드웨어 및 소프트웨어 개발이 별도로 진행되지 않으며 이들의 이질적인 혼합과 병행이 새로운 설계 패러다임으로 정착되고 있다. 임베디드 시스템을 위한 CAD는 병행설계(co-design)에 따라 하드웨어 및 소프트웨어를 연계하여 최적화 할 수 있고 병행해석(cosynthesis)으로 사양에 따른 각종의 설계안들을 검증하여 요건을 만족하는 설계의 대안들을 마련할 수 있게 하고 있다.

임베디드 시스템의 개발환경은 빠르게 진화하고 있다. 임베디드 시스템 개발에 소요되는 하드웨어와 소프트웨어 성능향상과 가격하락으로 90년대에만 해도 8비트 및 16비트 CPU에, 프로그램 코드의 길이도 64KB-512KB 정도가 보편적이었다. 그러나 2000년대에 들어 32비트 CPU가 보편화되고 메모리 가격의 하락함에 따라 화상 통신, 주문형 비디오와 같은 고성능 임베디드 시스템의 개발이 용이하게 되었다.

또한 가전 제품에 멀티미디어와 인터넷 연결을 접목하는 가전 제품의 정보화 추세에 따라 이러한 복잡한 디바이스를 지원하기 위한 구동 프로그램 개발이 많아짐에 따라 최근에는 임베디드 시스템의 개발자 수에서 소프트웨어 개발자의 수가 하드웨어 개발자 수의 증가율이 5배에서 6배에 달할 정도로 임베디드 시스템에서 소프트웨어가 중요해지고 있다[6].

5. 결 론

임베디드 시스템은 데스크 탑 컴퓨터와 전반적으로 다른 요건을 가지고 있으며 특히 응용 분야 특화와 외부 장비와의 인터페이스가 임베디드 시스템의 설계 사양을 좌우한다. 오늘날 임베디드 시스템은 산업 및 정보 사회의 각종 기기에 쓰이지 않는 곳이 없으며 기존의 시스템들도 임베디드화 되고 있다. 이에 따라 국내의 임베디드 시스템 개발 역시 급격히 증가하고 있는 바 앞으로 이 분야 연구자 및 개발자가 많이 질 것으로 사료된다.

데스크탑 컴퓨터의 운영체계와 달리 임베디드 시스템의 운영체계는 실시간적인 측면이 중요하다. 앞으로 홈 오토메이션을 지원하는 정보가전제품의 증대로 앞으로는 임베디드 운영체계도 표준화 문제가 중요한 이슈가 될 것으로 기대된다.

최근 임베디드 시스템에 도입되고 있는 하드웨어 및 소프트웨어 병행 설계 기법은 시스템의 성능 대 비용을 최적화하여 하드웨어에 소프트웨어를 실장하고 적시에 시장에 제품을 출하하려는 노력이다. 따라서 임베디드 시스템을 개발하기 위해 소프트웨어 개발자는 하드웨어 특성을 이해하고, 하드웨어 개발자는 소프트웨어적인 측면을 잘 이해할 필요가 있다.

참 고 문 헌

[1] Rajesh K. Gupta, "Introduction to Embedded Systems", ICS 212, 2002 Winter Workshop.

[2] Jim Turley, "Embedded System by the Numbers", *Embedded System Programming*, May, 1995.

[3] 최병욱, "임베디드 운영체계의 특징과 선정", *Electronic System*, Nov. 2001, vol. 78, pp26-40

[4] Bernard Cole, "Architectures Overlap Applications", *Electronic Engineering Times*, March 20, 1995, pp.64-65.

[5] Philip J. Koopman, Jr "Embedded System Design Issues (the Rest of the Story)", *Proc. of the International Conf. on Computer Design (ICCD 96)*

[6] 윤상진, "임베디드의 현재와 미래, 왜 임베디드

시스템인가", *마이크로소프트웨어*, 12월호, 2000, pp.239-249.

[7] Don Thomas & Rolf Ernsr (eds.), *Proceedings: Fourth International Workshop on Hardware/ Software Co-design*, IEEE Computer Society, Los Alamitos CA, 1996.

[8] Frank Vahid and Tony Givargis, "Embedded System Design: A Unified Hardware/Software Introduction", *John Wiley & Sons*, January 2001.

이 영 대

- 1987~1989 : 대우중공업 중앙연구소
- 1989~1991 : KPC 자동화 사업부
- 1998~1999 : KIST 휴먼로봇 센터
- 1999~현재 : 세명대학교 정보통신과 부교수
- 주관심분야 : 임베디드 시스템, 로봇공학, 지능시스템