실행시간 추정 가능한 RTOS 시뮬레이터의 구현

김방현* · 류성준* · 김종현** · 남영광** · 이광용***

Implementation of RTOS Simulator With Execution Time Estimation

Kim Banghyun · Ryu Sungjoon · Kim Jonghyun · Nam Youngkwang · Lee Kwangyong

요 약

실시간 운영체제(Real-Time Operating System: 이하 RTOS라 함) 개발환경에서 제공하는 도구 중에 하나인 RTOS 시뮬레이터는 타겟 하드웨어가 호스트에 연결되어 있지 않아도 호스트에서 응용프로그램의 개발과 디버깅을 가능하게 해주는 타겟 시뮬레이션 환경을 제공해 줌으로서, 개발자로 하여금 빠른 시간 내에 응용프로그램을 개발할 수 있도록 지원하며 하드웨어 개발이 완료되기 전에도 응용프로그램을 개발할 수 있게 해 준다. 그러한 이유로 현재 대부분의 상용 RTOS 개발환경에서는 RTOS 시뮬레이터를 제공하고 있다. 그러나 현재 상용 RTOS 시뮬레이터들은 대부분 RTOS의 기능적인 부분들만 호스트에서 동작하도록 구현되어 있어서 RTOS나 RTOS 응용프로그램이 실제 타겟에서 실행될 때의 실질적인 시간 추정이 불가능하다. 이러한 문제점은 실시간 시스템이 정해진 시간 내에 결과를 출력해야 하는 시스템임을 감안한다면 RTOS 시뮬레이터의 가장 큰 결점이 되기 때문에 실행시간 추정 기능을 가지면서 실용화도 가능한 RTOS 시뮬레이터가 필요하다.

본 연구에서는 이러한 문제점을 해결하여 RTOS와 RTOS 응용프로그램이 실제 타겟에서 처리될 때의 실행시간 추정이 가능하고 상용화가 가능한 기계 명령어 기반(machine instruction-based)의 RTOS 시뮬레이터를 연구 개발하였다. 나아가 실행시간의 주요 요소인 파이프라인과 캐쉬의 영향도 고려함으로서 실행시간 추정의 정확도를 향상시켰다. 본 연구에서 사용된 RTOS는 한국전자통신연구원(ETRI)에서 2000년에 개발된 Q+이고, Q+가 동작하는 타겟 하드웨어는 ARM 계열의 StrongARM SA-110 마이크로프로세서와 21285 주제어기가 장착된 EBSA-285 보드이다.

1. 서론

최근 정보가전기기가 급속히 발전함에 따라 소형의 내장 컴퓨터시스템(embedded computer system)들을 위한 고성능 저가격 실시간 운영체제(Real-Time Operating System: 이하 RTOS라 함)의 개발이 필수적이다. 또한 타겟 하드웨어가 개발되기 전에 시스템 디버깅이나 응용프로그램의 개발이 가능할 수 있도록 해주는 개발환경의 개발도 병행되어야 한다. 이러한 개발환경은 RTOS의 선택에 중요한 조건이 되기 때문에 독립성, 개방성, 신뢰성 및 사용자의 편의성을 충족하는 개발환경에 관한 연구는 필수적이라고 할 수 있다[1].

RTOS 개발환경에서 제공하는 도구 중에 하나인 RTOS 시뮬레이터는 타겟 하드웨어가 호스트에 연결되어 있지 않아도 호스트에서 응용프로그램의 개발과 디버깅을 가능하게 해주는 타겟 시뮬레이션 환경을 제공해 줌으로서, 개발자로 하여금 빠른 시간 내에 응용프로그램을 개발할 수 있도록 지원하며 하드웨어 개발이 완료

* 연세대학교 전산학과 대학원
** 연세대학교 전산학과 교수
*** 한국전자통신연구원 선임연구원

되기 전에도 응용프로그램을 개발할 수 있게 해 준다. 그러한 이유로 현재 대부분의 상용 RTOS 개발환경에서는 RTOS 시뮬레이터를 제공하고 있다.

현재 상용화되어 보급되고 있는 RTOS 시뮬레이터들로는 미국의 VxWorks의 VxSim[2]과 VRTX의 MIPS XRAY Simulator[3], 그리고 RT-Linux의 Carbon-Kernel[4] 등이 있다. 이러한 RTOS 시뮬레이터들은 대부분 RTOS 의 기능적인 부분들만 호스트에서 동작하도록 구현되어 있어서, RTOS나 RTOS 응용프로그램이 실제 타겟에서 실행될 때의 실질적인 시간 추정이 불가능하다. 이러한 문제점은 실시간 시스템이 정해진 시간 내에 결과를 출력해야 하는 시스템임을 감안한다면 RTOS 시뮬레이터의 가장 큰 결점이 되기 때문에, 실행시간 추정 기능을 가지면서 실용화도 가능한 RTOS 시뮬레이터가 필요하다[5].

본 연구에서는 이러한 문제점을 해결하여 RTOS와 RTOS 응용프로그램이 실제 타겟에서 동작할 때의 실행시간 추정이 가능하고 상용화가 가능한 기계 명령어 기반의 RTOS 시뮬레이터를 구현하는 것을 목표로 하였다. 이를 위해 기계어 단계의 명령어 시뮬레이션 방법을 사용하여 RTOS의 실행 이미지(executable image)를 호스트 상에서 수행되도록 하였다. 따라서 타겟 하드웨어에서의 실시간 운영체제의 동작이 호스트에서 시뮬레이션 되고, 응용프로그램의 동작도 시뮬레이션 된 RTOS 상에서 실행된다. 본 연구에서 사용한 방법은 C와 같은 상위 수준의 명령어를 호스트 기반으로 시뮬레이션 하거나, VHDL과 같은 하드웨어 설계 언어 위에서 시뮬레이션 하는 기존의 방법과는 달리 타겟의 마이크로프로세서의 명령어들을 타겟의 하드웨어 특성을 반영한 가상 하드웨어 위에서 시뮬레이션 한다는 점에서 구분된다. 나아가 실행시간의 주요 요소인 파이프라인과 캐쉬의 영향도 고려함으로서 실행시간 추정의 정확도를 향상시켰다.

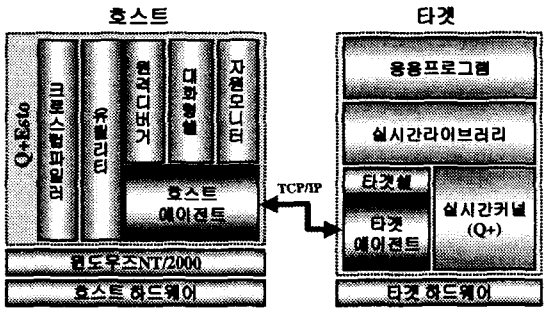
시뮬레이션 대상으로 한 RTOS는 한국전자통신연구원(ETRI)에서 2000년에 개발된 Q+이고, Q+가 동작하는 타겟 하드웨어는 ARM 계열의 StrongARM SA-110 마이크로프로세서와 21285 주제어기가 장착된 EBSA-285 보드이다. 그리고 본 연구에서 개발한 RTOS 시뮬레이터의 동작환경은 윈도즈 NT/2000이며, Q+의 통합개발환경인 Q+Esto의 도구들 중의 하나로서 포함된다.

2. 시뮬레이션 모델

2.1 Q+와 Q+Esto

Q+는 ETRI에서 2000년에 개발된 정보가전용 RTOS로서, 디지털 TV와 인터넷 TV 등에 활용이 가능한 기능을 갖추고 있으며, 국제 표준 및 업계 표준과의 호환성을 최대한 보장한다. 또한 Q+는 다양한 정보가전기기들에 쉽게 활용될 수 있게 하기 위하여 조립성 및 확장성을 가질 수 있는 다중 계층 구조로 구성되어 있다[6].

Q+의 통합개발도구 Q+Esto는 원격지 호스트에서 목적프로그램을 개발하여 타겟 보드에 적재하여 실행하고, 실행상태와 자원사용을 추적 관찰하며 발생하는 오류의 원인을 찾아 제거하도록 크로스 컴파일러, 유틸리티, 원격 디버거, 대화형 쉘, 자원모니터 등의 도구와 타겟서버, 디버그 에이전트 등으로 구성되는 서브시스템이다[7]. <그림 1>은 Q+Esto의 구성과 타겟과의 관계를 보여주고 있다.



<그림 1> Q+Esto와 타겟

2.2 테스트용 플랫폼 EBSA-285

Q+와 Q+Esto의 테스트용 플랫폼으로는 ARM 계열의 StrongARM SA-110 마이크로프로세서와 21285 주제어기(Core Logic Controller)가 장착된 인텔의 EBSA-285 보드를 사용한다. EBSA-285 보드는 내장시스템에 들어가는 SA-110과 21285 주제어기를 평가하기 위한 보드로서, SA-110을 이용하는 내장 시스템에 사용되는 하드웨어와 소프트웨어를 개발할 때에 프로토타입으로 많이 사용되며, PCI 카드 하나에 마이크로프로세서, 시스템 제어기, 기억장치, 입출력 인터페이스 등과 같이 모든 관련 구성 요소를 갖추고 있는 단일 기판 컴퓨터(single-board computer)이다[8].

SA-110 마이크로프로세서는 DEC과 공동으로 ARM사에서 개발한 StrongARM 계열의 프로세서로, 분리 명령어(separate instruction)와 데이터 캐쉬를 사용하는 수정-하버드(modified Harvard) 구조가 처음으로 적용된 ARM V4 구조의 ARM 프로세서이며 1998년부터는 인텔에서 생산되고 있다. SA-110은 휴대용 제품이나 대화형 디지털 비디오와 같은 내장시스템에 사용될 수 있는 저전력, 고성능의 범용 32비트 RISC 마이크로프로세서로서, 레지스터 포워딩(register forwarding)을 통한 5단계 파이프라인 구조를 갖고 있어 64비트 곱셈 명령어와 레지스터간 다중 전송 명령어, 그리고 메모리와 레지스터간 교체 명령어를 제외한 모든 명령어들이 한 사이클 내에 실행될 수 있다. 캐쉬는 16K바이트의 명령어 캐쉬와 16K바이트의 쓰기 지연(write-back) 방식의 데이터 캐쉬가 내장되어 있고, 사상방식은 32-way 집합 연관 사상 방식을 사용하며, 교체방식은 라운드 로빈(round robin)을 사용한다[9].

21285 주제어기는 SA-110 마이크로프로세서를 위한 칩으로, SA-110과 PCI, SDRAM, ROM, 그리고 X-버스 사이의 인터페이스 역할을 한다. 또한 전원 관리, DMA 제어, 인터럽트 제어, 버스 중재, RS232 제어 등의 역할도 수행하는 다기능 단일 칩 반도체로서 EBSA-285 보드 전체를 제어한다[10].

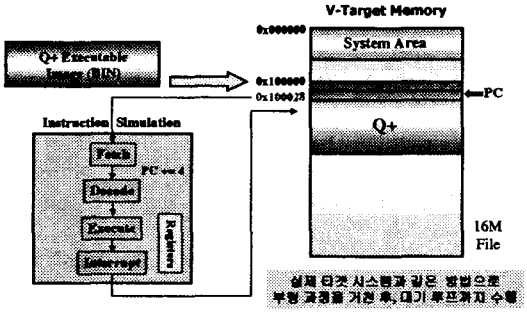
3. 시뮬레이터의 설계

3.1 설계 방법

본 연구는 정보가전용 RTOS인 Q+를 위한 시뮬레이터(Q+ Simulator: 이하 Q+Sim이라 함)를 실제 상용화가 가능하도록 구현하는데 중점을 두었다. 즉 RTOS 개발환경인 Q+Esto에서 하나의 도구로서, EBSA-285 보드와 같은 타겟이 호스트에 연결되어 있지 않더라도 타겟을 연결하여 작업할 때와 동일한 결과를 개발자에게 제공할 수 있게 하는 것을 목표로 Q+Sim을 설계하였다. 또한 기능면에서는 기존 상용화된 RTOS 시뮬레이터에서 제공하는 기능들은 Q+Sim에서도 가능한 모두 지원하도록 하고, 이와 더불어 기존의 제품들이 가지지 못한 실질적인 수행시간 추정 기능도 지원하도록 설계하였다. 이를 위해 본 연구에서는 기계어 단계의 명령어 시뮬레이션 방법을 사용하여 RTOS의 실행 이미지를 시뮬레이션 함으로서 RTOS를 시뮬레이션 한다. 따라서 타겟 하드웨어에서의 RTOS의 동작이 호스트에서의 시뮬레이터 안에서 동일하게 구현되고, 응용프로그램의 동작도 시뮬레이션 된 RTOS 안에서 실행된다.

명령어 시뮬레이션 방법은 실제 타겟에서 명령어가 실행되는 방법과 거의 동일하다. 먼저 타겟 하드웨어에서 실행될 실행 이미지 파일을 사용하여 타겟 하드웨어의 실제 메모리의 내용과 동일하게 가상 메모리를 파일로 구성한다. 이 가상 메모리에서 PC(program counter)의 주소에 있는 4바이트의 이진 명령어를 인출하고, 실제 타겟 하드웨어와 동일한 방법으로 명령어의 비트 패턴을 분석하여 명령어를 해독한다. 그리고 해독된 명령어의 기능을 소프트웨어로 구현한 가상 하드웨어에 적용하는 방법을 사용한다. 이렇게 타겟 하드웨어의 동작과 동일한 방법으로

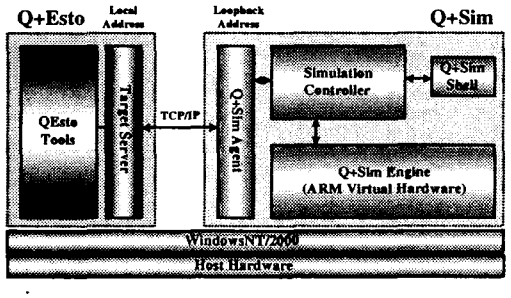
명령어를 시뮬레이션 함으로서 실행된 기계어 명령어의 계수를 통하여 어느 정도의 실질적인 수행시간 추정 기능의 구현이 가능하다. 또한 파이프라인이나 캐쉬와 같은 타겟 하드웨어의 특성들을 반영할수록 추정된 실행시간의 정확도는 높아진다. <그림 2>는 이러한 설계 방법을 개념적으로 보여준다.



<그림 2> Q+Sim의 설계방법

3.2 Q+Sim의 구조

Q+Sim은 <그림 3>과 같이 Q+ 에이전트(Q+ agent), Q+Sim 쉘(Q+Sim shell), 시뮬레이션 제어기(simulation controller), 그리고 가상 하드웨어(Q+Sim engine)로 구성되어 있으며, 가상하드웨어는 이식성을 위해 ANSI C로 설계되었고 나머지는 윈도즈 프로그램으로 설계되었다.



<그림 3> Q+Sim의 구조

3.3 실행시간 추정기능

Q+Sim은 기계어 단계의 명령어를 시뮬레이션하기 때문에 실행된 명령어의 계수를 통해 기본적인 실행시간 추정을 제공한다. 이는 SA-110이 하나의 CPU 클록에 평균 하나의 명령이 수행되었다고 가정하여 제공한다. 예를 들어 A라는 함수가 수행되는 동안 1000개의 명령어가 수행되었다면, 실행시간은 1000클록이며 200㎒의 SA-110의 경우에는 5㎲가 된다. 그러나 이것은 하나의 CPU 클록에 평균 하나의 명령이 수행되었다고 가정을 했을 때이고, 실제로는 실행되는 응용프로그램에 따라 파이프라인과 캐쉬의 효율이 달라지기 때문에 정확한 실행시간이 아니다.

따라서 본 연구에서는 파이프라인과 캐쉬의 효율을 실행시간에 반영하기 위하여 실제 SA-110의 동작과 동일하게 Q+Sim 가상 하드웨어에 파이프라인과 캐쉬를 소프트웨어로 설계하였다. 파이프라인은 5단계로 구성되어 명령어의 종류에 맞게 동작하도록 설계되었고, 캐쉬는 명령어 캐쉬와 데이터 캐쉬를 각각 16K바이트 파일로 구성하여 32-way 집합 연관 사상 방식과 라운드 로빈 교체방식으로 동작하도록 설계되었다.

4. 결론

본 연구에서는 현재 상용 RTOS 시뮬레이터들이 가지고 있는 기능적 시뮬레이션의 문제점을 해결하여 실제 타겟에서 동작할 때의 실질적인 시간 추정이 가능하고 상용화가 가능한 명령어 기반의 RTOS 시뮬레이터를 구현하였다. 이를 위해 기계어 단계의 명령어 시뮬레이션 방법을 사용하여 Q+의 실행 이미지를 시뮬레이션 할 수 있는 Q+ 시뮬레이터를 구현하였다.

Q+Sim은 Q+의 각 모듈별 함수를 정상적으로 수행하였으며, 이를 통해 Q+의 정상적인 시뮬레이션을 확인할 수 있었다. 또한 파이프라인과 캐쉬의 영향을 고려하여 RTOS의 특정 함수나 응용프로그램의 실행시간을 추정할 수 있었다. 그러나 이렇게 추정된 실행시간은 특정 상황에서

측정된 것이므로, 일반적인 상황에서 최악 실행 시간(worst case execution time: WCET) 예측을 위한 방법이 추후 연구되어야 할 과제이다.

참고문헌

1) 한국전자통신연구원, "조립형 실시간 OS 사용자 요구사항 정의서 1.0", 1998. 12.
2) WindRiver, VxWorks 5.3.1 Programmer's Guide, 1997. 4.
3) http://www.mentor.com/embedded, 2001. 1.
4) Realiant Systems, "Carbon Kernel User Manual 1.2" 2000. 1.
5) 김방현, 이종은, 김종현, "실시간 운영체제 시뮬레이터의 구현", 한국시뮬레이션학회 2001 춘계 학술대회, 2001. 5.
6) 한국전자통신연구원, "실시간 OS 커널 상세 설계서 1.0", 1999. 7.
7) 한국전자통신연구원, "사용자개발도구 서브시스템 설계서 1.0", 1999. 3.
8) Intel, StrongARM EBSA-285 Evaluation Board Reference Manual, 1998. 10.
9) Intel, SA-110 Microprocessor Technical Reference Manual, 2000. 12.
10) Intel, 21285 Core Logic for SA-110 Microprocessor Datasheet, 1998. 9.