μC/OS-II 운영체제의 부시스템별 CPU 전력 소비 분석

강경태[0] 심호준 박상수 성민영 신현식 장래혁
서울대학교 전기.컴퓨터공학부
(nicola, hjshim, sspark, minyoung)@cslab.snu.ac.kr, (shinhs, naehyuck)@snu.ac.kr

Energy Consumption Analysis of μC/OS-II Subsystems

Kyungtae Kang[0] Hojun Shim Sangsoo Park Minyoung Sung Heonshik Shin Naehyuck Chang
School of Computer Science and Engineering, Seoul National University

요 약

저전력 시스템에 대한 연구는 회로 수준에서부터 응용 소프트웨어 수준에 이르기까지 다양한 각도에서 이루어지고 있다. 본 연구에서는 특히 운영체제 수준, 즉 μC/OS-II(MicroC/OS - II) 커널 코드의 다양한 함수들에 대한 에너지 소비값을 추출하며 이 결과를 바탕으로 운영체제의 각 모듈별 에너지 소비를 분석한다. 이를 위하여 ARM7TDMI 마이크로 콘트롤러를 이용하여 제작된 시스템에 μC/OS-II 운영체제를 이식하고 SES(SNU Energy Scanner) 툴을 이용하여 수행된 운영체제 코드의 각 모듈 즉 태스크 관리, 인터럽트 처리, IPC등에 대한 에너지 소비를 얻고 이 결과를 분석한다.

1. 서 론

저전력 시스템에 대한 기존의 연구는 주로 칩이나 회로, 혹은 명령어 수준에 초점이 맞추어져 있었다. 최근에는 운영체제 수준의 연구가 비교적 활발히 진행되고 있는데 대부분 시스템의 유휴 시간을 이용한 저전력 모드로의 전환과 같은 전력 관리 모듈의 추가에 연구가 집중 되고 있다[4].

본 연구에서는 관심의 초점을 운영체제 자체 즉 시스템을 구성하는 운영체제 커널의 코드 수준으로 맞추어 실험을 진행하였다.

운영체제의 부 시스템별로 에너지 소비를 추출한 기존의 연구로는 μC/OS-II의 TCP/IP 스택의 전력 소비를 시뮬레이터를 이용하여 추출한 사례가 있다. 이밖에 운영체제의 전력 소비를 분석하기 위한 다양한 시도가 있었으나 대부분 시뮬레이션 수준에 머물고 있다[3]. 본 연구에서는 실제 시스템에서 동작하는 운영체제의 에너지 소비에 대한 실측 데이터를 얻어 내었으며 이 데이터를 이용하여 운영체제를 구성하는 각 모듈, 특히 태스크 관리, IPC(프로세스간 통신), 인터럽트등에 대한 코드 수준의 에너지 소비를 분석하고 이를 통해 저전력으로 동작할 수 있는 운영체제 개발의 가능성을 제시하였다.

운영체제는 다양한 기능을 수행하는 여러 모듈들이 상호 작용하여 여러 응용 프로그램이 효율적으로 수행될 수 있도록 도와주는 역할을 한다. 따라서 운영 체제의 커널 코드는 매우 복잡한 메커니즘으로 구성되어 있다. 본 연구에서는 운영체제를 수행하는 가장 작은 단위의 함수로부터 분석을 시작하여 이 함수를 포함한 상위 수준의 함수들로 그 결과를 상향식으로 확장해 나갔으며 이러한 방식을 사용하여 몇 가지 API 함수들에 대한 에너지 소비 결과도 얻어 내었다. 실험을 위하여 우리는 커널의 함수 트리 구조를 분석하였으며 함수의 호출 경로를 추적하였다.

2절에서는 실험을 위한 하드웨어와 소프트웨어의 구성에 대해 설명하고 3절에서 실험 결과에 대해 분석하도록 하겠다.

2. 실험 환경

2. 1. 하드웨어 구성

운영체제의 각 구성 요소별 에너지 소비를 분석하기 위한 SES 하드웨어는 그림 1과 같다. 에너지 측정의 대상이 되는 CPU는 ARM7TDMI이며 프로그램의 주소와 데이터를 저장하기 위한 공간을 위하여 16MB의 SDRAM, 그리고 프로파일 데이터를 저장하기 위한 공간으로 4MB SGRAM을 사용하였다. 또한 사이클별 ARM7TDMI CPU 의 에너지 소비 결과를 측정하기 위한 별도의 ADC(A-D Converter)회로를 구성하였다[5].

ADC회로는 CPU가 수행되는 각 사이클별로 프로파일 데이터를 추출하여 SGRAM에 저장한다. 저장된 데이터는 PCI 버스를 통해 초당 16MB 단위로 호스트에 전송된다. (그림2) 프로파일 데이터는 현재 수행되는 프로그램의 주소와 데이터, 제어 신호 및 두개의 ADC값 (rising edge, falling edge)으로 구성되어 있다. 이 ADC값과 아래의 식을 이용하여 소비된 전압과 에너지를 얻어낸다[6].

$$V = \frac{11 \times V_{ref} - 1.5 - \frac{x}{511.5}}{10} \quad (\text{단위} : V)$$

$$E_{rising}, E_{falling} = \frac{1}{2} \times C_L \times (V_{initial}^2 - V_{final}^2)$$

$$E_{MCLK} = E_{rising} + E_{falling}$$

호스트에 전송된 프로파일 데이터로부터 각 주소별 CPU에서 소비된 에너지를 위의 세 식으로부터 계산하고 이 값을 코드의 주소와 대응시켜 커널 소스 코드 수준의 에너지 소비를 분석할 수 있다.

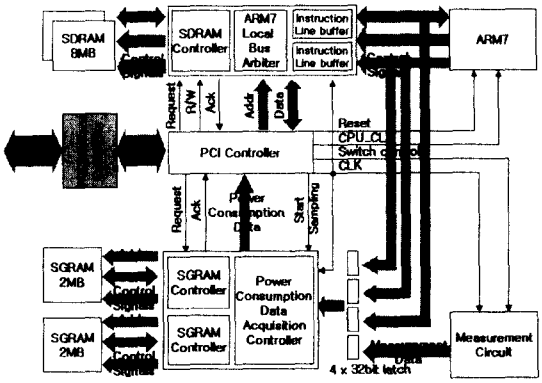

그림 1. SES 하드웨어 구성

2.2. 소프트웨어 구성

그림 1의 하드웨어 플랫폼에 에너지 소비 측정의 목표가 되는 운영체제로 μC/OS-II를 이식하였다. 커널 이미지를 생성하기 위해 ARM SDT 2.5를 사용하여 커널 소스를 컴파일 하였다. 전체적인 소프트웨어의 구성은 그림 3과 같다. 호스트의 모니터 프로그램의 명령에 의해 μC/OS-II의 커널 이미지는 PCI 버스를 통해 호스트로부터 SES하드웨어의 메모리 영역으로 전송된다. 반대로 운영체제와 태스크가 수행되면서 소비된 전력 데이터는 마찬가지로 PCI 버스를 통해 호스트로 전송되고 전송된 데이터는 호스트의 분석 소프트웨어에 의해 어셈블리나 C 언어 수준 혹은 함수나 모듈 수준으로 에너지 소비 결과가 계산되어 호스트의 메모리에 저장된다.(그림2)

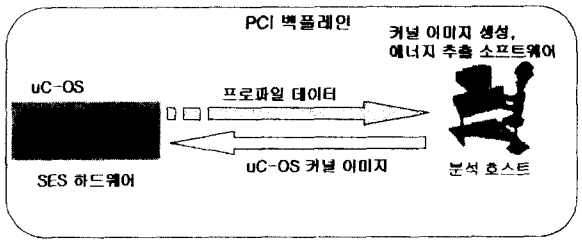

그림 2. 에너지 소비 분석을 위한 호스트 인터페이스

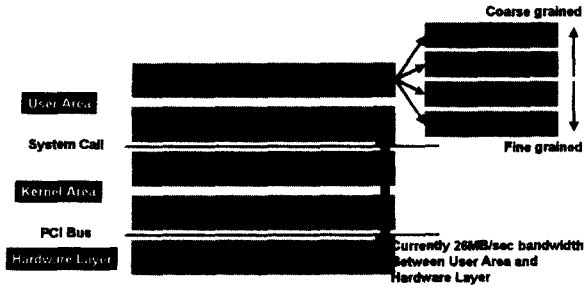

그림 3 소프트웨어 구성

2.1절에서 언급했듯이 SES 하드웨어로부터 얻은 데이터는 초 16MByte 단위로 프로파일 데이터를 추출한다. 따라서 256Mbyte의 주 메모리를 사용할 경우 약 10초 정도의 수행 결과에 대한 데이터를 얻을 수 있다. 본 연구에서는 10초 시간창(Time Window) 안에 충분히 수행 가능한 태스크를 생성하여 실행시켰으며 이 태스크의 구조는 아래 그림과 같다.

```
void Task(void* pdata)
{
  for (::)
  {
/* wait for the semaphore */
    OSSemPend(Sem2, 0, &Reply);

    /* wait a short while */
    OSTimeDly(2);

    /* signal the semaphore */
    OSSemPost(Sem1);
  }
}
```

그림 4.(a). PING 태스크

```
int main(void)
{
  TimerInit( )

  OSInit();
  OSTimeSet(0);
  Sem1 = OSSemCreate(1);
  Sem2 = OSSemCreate(1);
  OSTaskCreate(Task1, (void *)&Id1, (void
  *)&Stack1[STACKSIZE - 1], 1);
  OSTaskCreate(Task2, (void *)&Id2, (void
  *)&Stack2[STACKSIZE - 1], 2);
  OSStart();
  /* never reached */
}
```

그림 4.(b). main 함수

그림 4의 코드와 같이 실험을 위해 세마포어를 이용해 두개의 태스크가 랑데부 통신하는 간단한 프로그램을 실행하였다.

3. 운영 체제 수준의 전력 소비 분석

2절에서 언급한 실험 환경을 이용하여 운영 체제의 각 함수별 혹은 부 시스템별 에너지 소비 결과를 얻었다. 특정 함수가 호출될 때 이 함수의 에너지 소비는 전달된 인자와 레지스터의 값에 따라 다르기 때문에 본 연구에서는 이들 값들의 평균을 사용하였다.

3.1 운영체제의 각 API함수별 전력 소비 분석

그림 5는 μC/OS-II에서 제공하는 API 함수 및 중요한 운영체제 함수들에 대한 에너지 소비 비교이다. 그림 5.(a)는 함수가 한번 호출되었을 때 평균적으로 소비하는 에너지이다. 물론 한 함수 내에서도 실행된 부분과 실행되지 않은 부분이 있을 수 있지만 그림 5는 오직 수행된 코드 영역에 대한 에너지 소비 총합만을 나타낸다. 몇몇 운영체제 함수를 제외한 많은 함수들은 프로그램에 에러가 없다면 대부분 정해진 코드 루틴을 수행한다. 설령 그렇지 않은 함수라 하더라도 태스크를 오랫동안 실행한다면 잦은 빈도수로 수행되는 코드 영역이 분명히 존재한다. 따라서 수행된 코드영역에 대한 에너지 합만을 이용해서도 운영체제가 소비하는 에너지 소비 경향을 충분히 파악할 수 있다.

그림 5.(b)는 함수의 호출 빈도수까지 고려한 에너지 소비 분석 그래프이다. 정해진 시간 단위로 스케쥴러가 호출되며 스케쥴러는 문맥 교환 여부를 결정한다. 당연히 다른 함수들에 비해 OSSched, OSTimeTick ,문맥교환 함수의 호출 빈도수가 크며 소비한 에너지의 양도 다른 함수들에 비해 압도적으로 많은걸 확인할 수 있다. 물론 이 차이는 시간이 흐름에 따라 더욱 커질 것이다.

그림 5의 그래프를 보면 한번 수행될 때 OS_TASK_SW가 소비하는 에너지에 비해 OSTimeTick의 에너지 소비가 더 크지만 어느 정도 시간이 흐른 뒤에는 OS_TASK_SW가 소비하는 에너지의 양이 오히려 더 커졌다는 사실을 확인할 수 있다. 문맥 교환이 대부분 OSTimeTick함수가 호출될 때

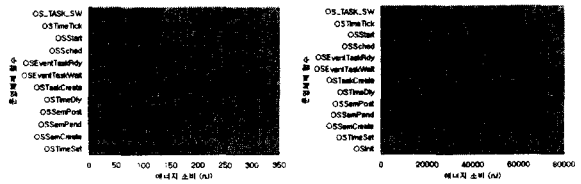

그림 5.(a). 단일 수행 그림 5.(b). 함수별 총 에너지 소비

그림 5. μC/OS-II의 함수별 에너지 소비

수행된다는 사실을 고려 한다면 문맥 교환의 오버헤드로 인한 에너지 소비가 상당히 크다는 사실을 알 수 있다.

OSInit, OSTaskCreate, OSSemCreate, OSTimeSet, OSStart 등의 함수는 태스크가 초기화될 때 한번 수행되고 더 이상 수행되지 않기 때문에 전체 운영체제가 소비하는 에너지의 양에는 영향을 주지 않는다. 그림 6은 운영체제의 각 부 시스템별 에너지 소비 총합이다. 10초간 시스템을 구동하였을 때 시스템이 소비한 에너지의 양은 5882844.9 nJ이고 이중 76%에 해당하는 4484670.16 nJ 을 Idle 태스크가 소비하고 있다. 이것은 실험에 사용한 테스트 프로그램이 생성한 태스크가 간단하여 주어진 주기 안에 실행을 마치고 나머지 유휴 시간에 Idle 태스크가 수행되기 때문이다.

3.2 운영체제의 각 부 시스템별 전력 소비 분석

그림 6은 μC/OS-II의 부 시스템별 전력 소비 결과이다. μC/OS-II는 임계 영역을 보호하기 위하여 임계영역 전후에 OS_ENTER_CRITICAL과 OS_EXIT_CRITICAL함수를 호출하는데 두 함수가 소비한 전력은 전체 시스템이 소비한 전력의 26%에 해당하는 1175823.81 nJ 이다. 결국 전체 에너지 소비의 92%가 Idle 태스크 및 임계 영역 보호 함수에 의해 소비되고 있다. 비록 두 함수의 코드는 간단한 어셈블리 언어로 작성되어 있지만 호출이 빈번하게 발생하기 때문에 누적된 에너지 소비 총합이 차지하는 비중이 커졌다. 위의 두 함수는 스케쥴러나 문맥 교환, OSTimeTick과 같이 수행 빈도수가 큰 함수에 의해서도 호출되지만 사실 그 외의 다른 많은 운영체제 함수들에 의해 호출되기도 한다. 따라서 태스크를 구동하기 위해 사용된 모든 함수들의 수행 횟수만큼 호출되기 때문에 소비되는 에너지도 그에 비례하여 클 수밖에 없다. 결국 임계영역 보호를 위한 코드를 호출하는 함수들은 1175823.81nJ 중 자신이 호출한 빈도수에 비례하여 추가적으로 에너지를 더 소비하게 되므로 태스크 스케쥴과

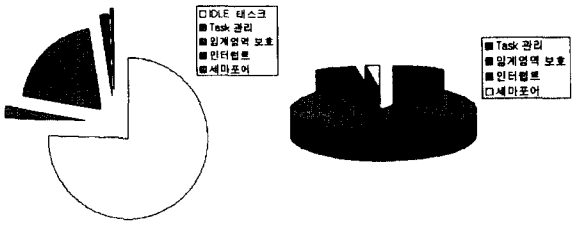

그림 6. μC/OS-II의 부 시스템별 전력 소비

문맥교환 그리고 OSTimeTick함수가 소비하는 전체 에너지의 양은 더욱 커지게 된다.

그림 6의 두 번째 그래프는 Idle 태스크를 제외한 나머지 요소들의 에너지 소비 비교이다. 1398174.74 nJ 가운데 태스크 관리를 위해 9.36 % , 인터럽트 처리에 8.94%, 그리고 세마포어를 이용한 랑데부 통신에 2.5%의 에너지를 소비하였다. 태스크를 관리하는 부 시스템이 소비한 에너지는 130874.33 nJ이며 이 에너지의 53%는 스케쥴러에 의해, 그리고 27%는 임계영역 보호에 사용되었다. 또한 스케쥴러에 의해 소비된 69472.19 nJ 가운데 문맥교환이 43932.24 nJ로써 63%를 차지하였다. 인터럽트 처리시에 소비되는 에너지의 대부분은 타이머 인터럽트로써 주로OSTimeTick함수(57%)와 핸들러로 인터럽트를 전달해주는 처리기(29%)에 의해 소비된다.

4. 결론 및 향후 연구과제

지금까지 우리는 ARM7TDMI를 이용하여 제작한 시스템에 μC/OS-II 운영체제를 이식하고 간단한 태스크를 수행하면서 운영체제의 각 코드가 소비한 에너지에 대한 실측 데이터를 추출하였고 이를 바탕으로 에너지 소비에 대한 경향을 분석하였다. 저전력 운영체제를 위해 코드를 수정 할 때는 수정 했을 때 그 효과가 가장 큰 부 시스템이 우선적인 고려 대상이 되어야 할 것이다.

운영체제가 소비하는 에너지의 양을 좌우하는 것은 코드의 복잡도가 아니라 수행 빈도수이다. 특히 실시간 운영체제의 경우 스케쥴러의 호출 빈도수가 크며 이것은 곧 문맥 교환의 빈도수가 크다는 사실을 의미한다. 따라서 효율적인 스케쥴러를 개발할 수 있다면 운영체제가 소비하는 전체 에너지의 양을 크게 줄일 수 있을 것이다.

그러나 본 실험에서 얻어낸 데이터는 단지 CPU 코어가 소비한 에너지만을 대상으로 하였다는데 한계가 있다. 시스템이 소비하는 에너지 소비를 얻기 위해 메모리-주메모리, 캐시를 고려한 시뮬레이터를 제작하여 이를 반영해야 할 것이다. 더불어 보다 크고 복잡한 태스크를 여러 개 실행하여 상대적으로 Idle 태스크의 영향을 덜 받도록 태스크의 수준을 조정할 필요가 있다.

참고 문헌

[1] ARM, " ARM7TDMI Data Sheet"
[2] JEAN J. LABROSSE, " MicroC/OS-II"
[3] Robert P. Dick and Ganesh Lakshminarayana and Anand Raghunathan and Niraj K. Jha, " Power Analysis of embedded operating systems" Design Automation Conference, 2000
[4] Yung-Hsiang Lu, Luca Benini, Giovanni De Micheli, " Operating-System Directed Power Reduction" International Symposium on Low Power Electronics and Design, 37-42, July, 2000
[5] N. Chang, K. Kim and H. Lee, " Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI" International Symposium on Low Power Electronics and Design, July, 2000
[6] Computer System Laboratory, SNU(Seoul National University) " SES(Seoul National University Energy Scanner) Tools" DAC 2001, University booth, LA, USA