논문 2006-43CI-5-1

저전력 마이크로컨트롤러를 위한 명령어 레벨의 소모전류 모델링 및 최적화에 대한 연구

(Study of Instruction-level Current Consumption Modeling and Optimization for Low Power Microcontroller)

엄 흥 식*, 김 건 욱**

(Heungsik Eom and Keonwook Kim)

요 약

본 논문에서는 임베디드 시스템에서 사용되는 대표적 저전력 마이크로컨트롤러인 ATmega128을 대상으로 명령어 레벨의 소모전류를 측정, 모델링하였다. 마이크로컨트롤러가 소모하는 전류는 메모리의 접근 유무에 의해 차이가 나며, 메모리 접근 명령어가 메모리 비접근 명령어에 비해 내부 메모리 기준으로 17% 더 높은 전류소모를 나타낸다. 프로그램의 메모리 접근 명령어 사용빈도가 높을수록, 메모리 계층구조에서 낮은 계층의 정보를 접근할수록 프로그램의 전력소모는 비례한다고 관찰된다. 본 논문에서는 명령어 레벨의 소모전류모델화를 통하여 실제 프로그램의 전력소모를 예측, 분석하고 메모리 접근 명령어의 비율을 줄이는 방향으로 프로그램의 전력소모를 최적화한다. 또한 마이크로컨트롤러 기반 시스템에서 프로그램 실행 전력을 최적화할 수 있는 기법을 하드웨어와 소프트웨어 측면에서 다양하게 제안한다.

Abstract

This paper presents experimental instruction-level current consumption model for low power microcontroller ATmega128. The accessibility of instruction for internal memory decides power consumption of the microcontroller as much as 17% of difference between access instruction and non-access instruction. The power consumption for the given program will be increased in the proportional to the ratio of memory access instruction and lower level memory access in the hierarchy. Throughout the current consumption model, the power consumption can be predicted and optimized in the direction of reducing the frequency memory access. Also, the various optimization methods are introduced in terms of software and hardware viewpoints.

Keywords: Microcontroller, Instruction-level current consumption, Power optimization

I. 서 론

임베디드 시스템에서 사용되는 마이크로컨트롤러의 전력소모는 시스템이 배터리 기반의 특성을 지니고 있을 때 매우 중요한 부분을 차지한다. 센서네트워크의 노드에 사용되는 마이크로컨트롤러는 센서네트워크 시스템의 특성상 불필요한 소모를 줄여 전력소모를 최소화하여야 한다. 마이크로컨트롤러의 전력소모 최소화는 여러 측면에서 이루어 질 수 있다. 첫째로, 하드웨어의 설계 시 반도체 특성을 이용하여 저 전력 마이크로컨트롤러를 설계, 제작하는 것이다. 두 번째 방법은 주어진 프로그램에서 마이크로컨트롤러의 효율적인 수면모드 관리에서 비롯된다. 세 번째 방법은, 소프트웨어 측면에서 명령어의 특성을 고려하여 전력소모를 최적화시키는 방식이다.

본 논문에서는 소프트웨어 측면에서의 전력소모를 모델링하고 소모되는 전력을 최소화하는데 중점을 두는데 이는 마이크로컨트롤러를 동작시키는 명령어의 소모전류가 균일하지 않기 때문이다. 명령어 특성에 따른

* 학생회원, ** 정회원, 동국대학교 전자공학과
(Department of Electronics Engineering, Dongguk University)

※ 본 연구는 과학기술부/한국과학재단 우수연구센터육성사업의 지원으로 수행되었음(R11-1999-058-01007-0)
접수일자: 2006년7월6일, 수정완료일: 2006년8월27일

전류소모의 주된 이유는 명령어가 실행될 때 내부에서 동작되는 형태가 각기 다름에 기인한다. 명령어의 addressing mode에 따라 사용하는 피연산자의 종류, 위치, 실행패턴 등이 결정되기 때문에 명령어의 소모전류가 차이가 난다. 그 중에서도 내부 메모리 접근 유무의 영향을 가장 많이 받는다고 본 논문에서 관찰된다. 즉, 마이크로컨트롤러가 내부 메모리를 접근하는 경우가 접근을 하지 않는 경우보다 더 높은 전류소모를 나타낸다. 그러므로 프로그램 작성 시 가능한 내부 메모리의 접근을 최소화시키면 상당한 전력효율을 얻을 수 있다. 이러한 고찰을 기반으로 소프트웨어 측면에서 전력 효율성을 높일 수 있는 방식을 다양하게 제안한다.

임베디드 시스템에서 일반적으로 사용되는 범용 마이크로컨트롤러인 Atmel사의 ATmega128을 대상으로 명령어 레벨의 전류소모를 측정하였다. 또한 내부 메모리 접근 명령어의 비율에 따라 전체 프로그램이 사용하는 전류와 이 때 소모되는 전력의 변화를 모델링하였다. 본 연구를 통하여 특정 프로그램의 전력소모를 근사화하여 예측할 수 있고 프로그램의 내부 메모리 접근 명령어의 발생빈도를 줄여 전력효율에서 효과적인 결과를 얻을 수 있다.

Ⅱ. 관 련 연 구

마이크로컨트롤러의 전력소모가 임베디드 시스템 설계를 위한 중요한 연구 분야로 대두되면서 다양한 방법을 통한 전력소모 모델링이 이루어지고 있다. 과거에는 마이크로컨트롤러의 회로 레벨[1] 혹은 게이트 레벨[2]에서의 시뮬레이션을 통한 모델링이 활발히 이루어졌다. 그러나 하드웨어 레벨의 시뮬레이션을 통한 전력소모 모델링은 매우 느리다. 또한 마이크로컨트롤러의 회로 레벨과 게이트 레벨의 정보가 부족하여 비현실적이고 정확성이 결여되어 있다. 이로 인해 보다 상위계층인 마이크로컨트롤러의 구조적 레벨의 전력 시뮬레이션에 대한 연구가 진행되었다.[3][4] 시뮬레이션이 실행되는 동안 마이크로컨트롤러를 구성하고 있는 각 모듈에 흐르는 전류의 합을 통하여 전력을 예측하였다. 하지만 이 또한 회로의 상태, 연산자의 종류와 각 모듈의 연관성을 고려하지 않았기 때문에 신뢰성이 떨어졌다. 프로그램의 함수레벨에서도 전력소모의 연구가 진행되었는데 라이브러리 정의 함수, 사용자 정의 함수 등의 전력 소모를 데이터베이스화하여 마이크로컨트롤러의 전력소모를 모델링하였다.[5]

위의 방법은 단지 시뮬레이션에 의존한 연구였기 때문에 실용화에 있어서 한계를 가지고 있다. 이로 인해 실제 마이크로컨트롤러가 소모하는 전류량을 물리적으로 측정할 수 있는 연구의 필요성이 대두되었다. 최근에는 소프트웨어 차원에서 명령어 레벨의 소모전류를 실제로 측정하여 마이크로컨트롤러의 전력소모를 모델링하는 연구가 활발히 진행되고 있다. 명령어 레벨의 모델링 또한 다양한 방법으로 이루어지고 있는데 특정 마이크로컨트롤러의 명령어가 사용하는 전류량을 바탕으로 피연산자의 변화에 따른 전력소모, 인접 명령어의 변화로 인한 전류량의 영향, 파이프라인의 영향에 따른 변화 등의 관점에서 연구가 이루어지고 있다.[6][7]

Ⅲ. 마이크로컨트롤러 전력소모 모델링

1. 전력소모 모델링

본 논문에서는 명령어의 내부 메모리 접근 여부에 따른 마이크로컨트롤러의 전력소모를 모델링한다. 명령어의 내부 메모리 접근 유무는 addressing mode에 따라 결정된다. 그러므로 모드가 다른 명령어의 소모전류를 측정함으로서 내부 메모리 접근유무에 따라서 달라지는 전류량의 차이를 관찰할 수 있다. 명령어 레벨의 소모전류를 이용한 소비전력의 예측은 마이크로컨트롤러의 구동전압 V_{cc}과 프로그램을 구성하는 명령어들의 평균 전류량을 곱하여 이루어진다.

$$E[P_{instr}] = V_{cc} \times E[I_{instr}] \quad (1)$$

식(1)에서 V_{cc}는 마이크로컨트롤러의 구동전압을 나타내고 $E[I_{instr}]$는 전체 명령어의 소모전류 평균값이다. 따라서 $E[P_{instr}]$는 프로그램이 실행되는 동안의 소비전력의 평균을 나타낸다.

2. 마이크로컨트롤러의 addressing mode

마이크로컨트롤러는 정해진 몇 가지의 addressing mode를 가지고 있다.[8] 피연산자로 즉치(immediate constant)를 사용하는 immediate addressing과 레지스터를 사용하는 register direct addressing, 데이터의 내부 메모리 영역을 직접 지정하는 direct addressing이 있다. 또한, 데이터 내부 메모리 영역을 어드레스 포인터 레지스터 X, Y, Z를 이용하여 지정하는 indirect addressing이 존재한다.

본 논문에서 사용되는 프로세서는 RISC(Reduced

Instruction Set Computer)구조를 가지고 있기 때문에 addressing mode에 따라 명령어는 참조하는 피연산자의 종류나 위치가 달라지는데, 크게 특정 레지스터 사용 명령어와 내부 메모리 사용 명령어 등으로 나뉜다. 즉치를 입력하여 피연산자로 사용을 하는 경우 또한 즉치가 레지스터나 메모리에 저장이 되면서 레지스터 혹은 내부 메모리 접근 연산이 병행된다. 물론 마이크로컨트롤러 내부에서도 명령어에 따라 각각 다른 addressing mode를 가질 수 있다. 그러나 근래 대부분의 마이크로컨트롤러는 직교 명령어 집합(orthogonal instruction set)을 사용하여 명령어마다 거의 모든 addressing mode가 구현이 되어 있어 다양한 addressing mode의 사용이 가능하다.

IV. 마이크로컨트롤러의 소모전류 측정

1. 실험환

본 논문의 실험에 사용된 ATmega128은 Ateml사의 8bit 마이크로컨트롤러로서 RISC구조를 가지고 있으며 5V, 16MHz에서 평균적으로 16MIPS의 명령처리 속도를 내고 4KB의 데이터 저장용 SRAM을 내부메모리로 가지고 있다.[8]

그림 1은 Agilent사의 Device Characterization software로 전원공급기의 도선을 통해 흐르는 전류를 확인 및 수집할 수 있다. 또한, 전원공급기를 원격으로 제어할 수 있는 기능을 가지고 있다.[11]

또한 세밀한 명령어 레벨의 소모전류를 측정하기 위하여 Agilent사의 MSO6034A 오실로스코프와 1147A 전류측정 프로브를 사용하였다. MSO6034A 4채널 오실로스코프는 300Mhz의 대역폭과 최대 2GSa/s의 샘플링

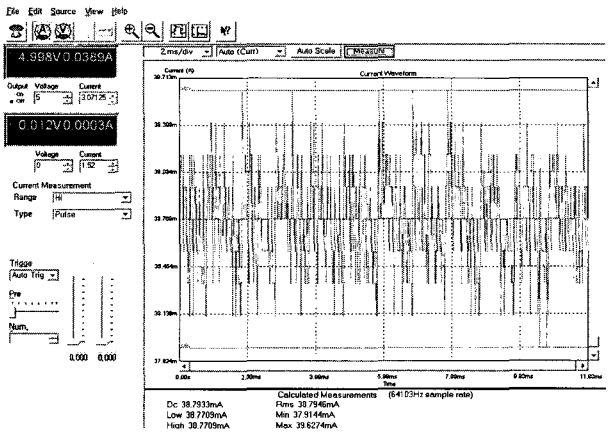


그림 1. 소모전류 측정 소프트웨어
Fig. 1. Software for Current Consumption.

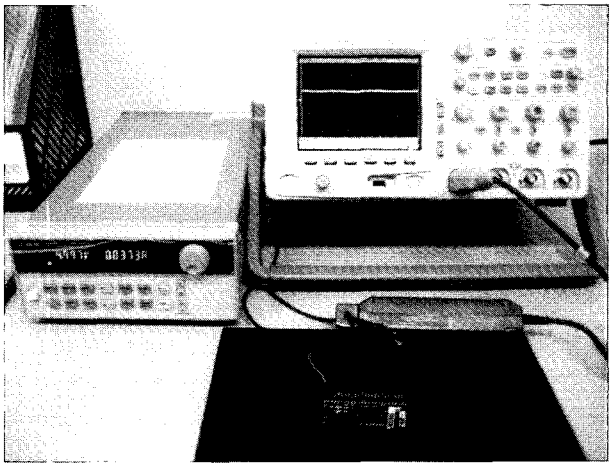


그림 2. 전류 측정 실험 사진
Fig. 2. Picture for Current Measurement Experiment.

주기를 가지고 있다.[9] 1147A 전류측정 프로브는 50Mhz의 대역폭으로 동작을 하고 1㎷의 전원을 가할 때 최대 ±1%의 오차로 전류 측정이 가능하다. 전류측정 프로브는 도선을 통하여 흐르는 전류가 발생시키는 도선 주위의 전자기장 세기를 감지하여 오실로스코프가 읽을 수 있는 전압값으로 변환하여줌으로써 전류측정을 가능하게 한다.[10]

열거한 장비를 통하여 마이크로컨트롤러와 소모되는 전류를 밀접하게 연결시켜 명령어 수준의 전류측정이 가능하며 실험방법은 다음 절에서 언급된다.

2. 명령어 레벨의 전류 측정

명령어 레벨의 소모전류는 대상이 되는 하나의 명령어를 연속으로 실행시킨 뒤 실행시간 동안의 소모전류 평균값으로 얻을 수 있다.[7] 하나의 명령어가 무한하게 반복 실행되더라도 소모되는 전류는 명령어 하나가 실행될 때의 소모전류값을 유지하므로 단일 명령어의 소모전류를 얻을 수 있게 된다. 더욱이 실험에 사용된 마이크로컨트롤러인 ATmega128은 내부에 명령어 또는 데이터 캐쉬를 가지고 있지 않기 때문에 위의 방법으로 측정된 결과는 마이크로컨트롤러가 레지스터 혹은 내부 메모리를 접근할 때 소모되는 전류를 나타낸다. 다양한 방법을 통해 명령어의 무한반복을 구현할 수 있지만 반복루프의 영향을 최소화하기 위하여 분기명령어를 사용하였다. 분기명령어 자체의 영향이 있으나 루프 내에 삽입되는 측정대상 명령어의 양이 많아질수록 분기명령어의 영향은 무시할 수 있는 수준으로 낮아지게 된다. 정확한 측정을 위해 하나의 명령어마다 소모전류값을 총 5회 측정하여 측정결과의 평균값으로 대표적 명령어

```
loop : add r0, r1
       add r0, r1
       add r0, r1
          :
       add r0, r1
       jmp loop      — 분기명령어
```

그림 3. 분기명령어 jmp를 이용한 명령어 무한반복
Fig. 3. Infinite Loop Program with Simple Branch Instruction jmp.

표 1. 명령어 레벨의 소모전류
Table 1. Instruction-level Currnet Consumption.

분류	명령어	소모전류(㎃)
산술연산 논리연산	내부 메모리 비접근 명령어	
	add rd, rs	34.9
	sub rd, rs	35.0
	and rd, rs	35.0
	or rd, rs	34.1
	subi rd, k	36.3
비트연산	set	34.5
	cli	34.4
	bclr s	34.5
	평균	34.8
데이터이동	내부 메모리 접근 명령어	
	ld rd, (x,y,z)	41.0
	st (x,y,z), rd	40.5
	lds rs, k	40.8
	sts k, rs	40.6
	평균	40.7

의 소모전류를 획득하였다.

ATmega128의 명령어는 산술연산, 논리연산, 비트연산, 데이터 이동, 분기연산, MCU 제어 등 총 6가지로 분류할 수 있다. *Jump, Branch, Call* 등의 분기연산과 *Break, Sleep* 등의 MCU 제어 명령은 프로그램 내에서 사용빈도가 높지 않기 때문에 본 논문의 분석대상에서 제외하였다.

표 1에서는 명령어의 내부 메모리 접근 유무에 따라 전류소모가 다름을 알 수 있다. 내부 메모리 비접근 명령어의 평균전류소모자 34.8㎃인 반면, 내부 메모리 접근 명령어의 평균전류소모는 40.7㎃로 내부 메모리 접근 명령어가 약 17%가 더 높은 전류소모를 보이고 있다. 이는 ATmega128이 일반적인 RISC구조를 가지고 있다는 점에서 분석이 가능하다. RISC구조 마이크로컨트롤러의 산술, 논리, 비트 연산과 같은 내부 메모리 비

```
for(i=0; i<10000; i++){
   a+=i;
 }
```

(a)

```
ldi  rd, k
ldi  rd, k
cpi  rd, k
ldi  rd, k
cpc  rd, rs
brge k
add  rd, rs
adc  rd, rs
rjmp k
```

(b)

그림 4. 덧셈 프로그램 (a) C언어; (b) 어셈블러
Fig. 4. Simple Accumulating Program (a) C-Language Implementation; (b) Compiled Program (Assembler Language Format).

접근 명령어는 파이프라인 후반부의 메모리 접근 부분을 생략하고 간단한 레지스터 파일을 접근하여 연산의 결과를 기록하기 때문에 메모리 접근에 필요한 만큼의 전류가 덜 소모된다.

우선 표 1의 측정결과를 일반화하기 위하여 1부터 9999까지 더하는 간단한 프로그램을 실행시킨 뒤 전류소모를 측정하였다.

그림 4(b)와 같이 간단한 덧셈 프로그램을 어셈블러로 컴파일된 결과를 통하여 명령어 레벨의 전류소모를 34.8㎃로 예측을 하였다. 실제로 위의 프로그램을 실행한 후 소모되는 전류를 측정한 결과 35.6㎃의 소모량을 보임으로써 표 1의 결과와 2.3%의 오차범위 내에서 일치하였다. 그러나 그림 4의 프로그램은 내부 메모리 비접근 명령어로만 이루어진 프로그램이므로 외부/내부 메모리 접근 명령어의 영향을 판단할 수 없다. 내부 메모리 접근 명령어의 영향을 알아보기 위해, 특정 프로그램의 내부 메모리 접근 명령어와 비접근 명령어 비율을 일정하게 변화시켜 이에 따른 전류소모의 변화를 모델링/측정하였다. 식(2)을 이용하여 내부 메모리 접근 명령어의 비율에 따른 프로그램 전체의 평균소모전류량을 모델링할 수 있다.

$$E[I_{avg}] = E[I_{MEM}] \times x + E[I_{NMEM}] \times (1-x) \quad (2)$$

$E[I_{MEM}]$와 $E[I_{NMEM}]$는 각각 내부 메모리 접근 명

령어와 비접근 명령어가 소모하는 전류의 평균값을 나타내고 x는 전체 프로그램에서 내부 메모리 접근 명령어가 차지하는 비율이다.

예를 들어 전체 프로그램에서 내부 메모리 접근 명령어가 차지하는 비율이 30%라고 가정한다면 내부 메모리 비접근 명령어의 비율은 70%를 차지하게 된다. 내부 메모리 접근 명령어와 비접근 명령어의 전류량을 실험결과와 같은 34.8㎃와 40.7㎃로 대입한 후 식(2)을 이용하여 전체 프로그램의 평균소모전류량을 구할 수 있다.

같은 방법으로 식(2)를 이용하여 내부 메모리 접근 명령어의 비율을 0~100%까지 변화시키면서 평균소모전류량을 예측하고 이를 실제 프로그램 내의 내부 메모리 접근 명령어 비율을 10%씩 감소시키며 측정한 소모전류량과 비교하였다. 더불어 내부 메모리 접근 명령어와 비접근 메모리의 순서에 따른 영향을 알아보기 위하여, 순서를 일정하게 하여 측정한 결과와 무작위로 배치하여 측정한 결과를 비교하였으나 두 개의 결과가 미미한 오차범위 내에 일치함을 확인하였다. 내부 메모리 접근 명령어가 프로그램 전체를 차지할 경우 프로그램의 전류소모가 39.9㎃로 표1의 내부 메모리 접근 명령어가 소모하는 평균전류량과 1.97%의 오차를 보였다.

이 후의 결과 또한 그림 5와 같이 오차 2.3% 이내에서 일치함을 나타낸다. 이와 같은 오차의 이유는 실험환경 특성상 측정 시 발생하는 유도전류의 미세한 편차에 기인한다. 이와 같이 특정 프로그램의 내부 메모리 접근 명령어 비율을 줄임으로써 특정 프로그램의 평균 소모전류량을 감소시킬 수 있고 마이크로컨트롤러가 프로그램을 실행하는 동안 소모하는 전력을 최적화 할 수 있음을 관찰하였다.

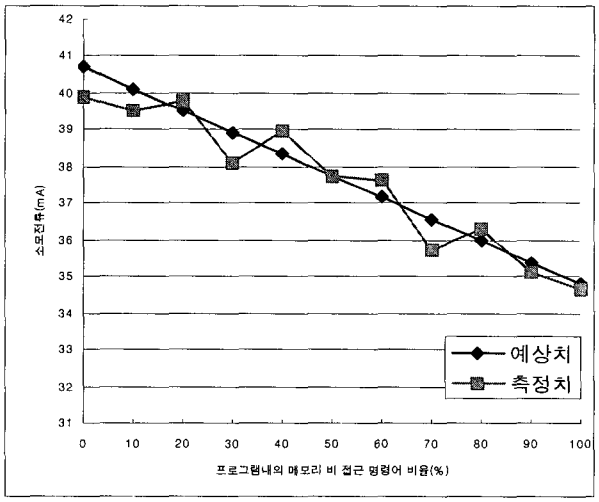


그림 5. 내부 메모리 접근 명령어의 비율에 따른 소모전류
Fig. 5. Current Consumption with Memory Access Ratio.

3. 마이크로컨트롤러 전력소모의 최적화

본 논문의 명령어 레벨의 소모전류 모델링을 실제로 적용하기 위해 무선 센서네트워크의 TinyOS 예제 프로그램 중 하나인 SMACTest의 데이터 전송 함수인 sendDATA를 컴파일한 결과를 통하여 소모전력 예측과 최적화하였다.[12]

그림 6은 TinyOS의 예제프로그램 SMACTest의 모듈흐름도이다. 위의 흐름으로 sendDATA함수는 일정시간마다 SMACTestM 모듈에 의해 호출되어 실행되게 된다. sendDATA 함수 전체 사용 명령어 19개 중 약 53%에 해당하는 10개의 명령어가 메모리 접근연산을 수행한다. 그러므로 함수가 한번 호출될 때마다 전체 720.0㎃의 누적전류를 소모를 하게 된다. 하지만 메모리 접근을 최적화하여 메모리 비접근 명령어의 비율을 높인다면 함수가 소모하는 전력은 낮아질 것이다. 100% 최적화가 이루어져 소모 누적전류를 661.2㎃로 낮춘다면 총 8%의 소모전력 효율을 얻을 수 있다. 다중호출의 특성을 충분히 이용한다면 상당한 최적화가 이루어 질 것이다, 또한 예제 프로그램에서 수초 혹은 수 십초에 한 번씩 호출되는 sendDATA함수의 빈도를 고려한다면 마이크로컨트롤러가 소모하는 전력이 현저히 낮아지는 것을 예상할 수 있다.

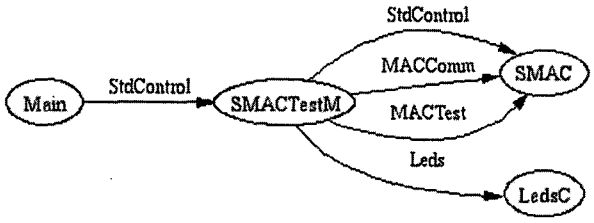


그림 6. TinyOS 예제 프로그램 SMACTest 모듈흐름도
Fig. 6. Module Flow Chart for SMACTest in TinyOS Application.

V. 마이크로컨트롤러 전력소모 최적화 기법

본 논문에서는 내부 메모리 접근 명령어의 비율에 따른 마이크로컨트롤러의 전력소모를 모델링하였다. 내부 메모리 접근 명령어의 비율을 줄일수록 소모전류량이

낮아지고 식(1)에 따라 마이크로컨트롤러가 소모하는 전력이 줄어든다. 내부 메모리 접근 명령어의 비율을 줄여 마이크로컨트롤러의 전력소모 최적화를 이룰 수 있는 3가지 기법을 제안한다.

1. 마이크로컨트롤러 구조의 변화

첫 번째 방법은, 증가된 수의 레지스터로 내부 메모리 접근 빈도를 감소시키는 것이다. 대부분의 명령어는 연산을 수행하기 위해 피연산자를 사용한다. 내부 메모리에 저장되어 있는 피연산자보다 레지스터에 있는 피연산자에 접근하는 것이 전력소비가 덜하다는 것은 본 논문의 실험을 통하여 증명을 하였다. 증가된 레지스터의 수는 데이터 캐쉬와 같은 역할을 수행하여 공간적/시간적 지역성을 이용하여 하부 계층의 메모리에 접근을 감소시켜 전력소모를 줄일 수 있다.

두 번째 방법은, 마이크로컨트롤러의 내부 메모리 접근 대역폭을 증가시키는 것이다. 향상된 대역폭으로 인해, 보다 많은 데이터를 한 번에 가지고 올 수 있다면 내부 메모리 접근 명령어의 비율을 줄일 수가 있다.

그림 7(b)는 한 번의 내부 메모리 접근을 통해 2개의 피연산자를 가져와 연산을 수행한다. 또한 연산의 결과를 동시에 저장함으로써 내부 메모리 접근을 그림 7(a) 보다 반으로 줄일 수 있다. 따라서 마이크로컨트롤러의 전력소모도 내부 메모리 접근에 비례하여 줄일 수 있다.

```
load r0
load r1
use r0, r1 → r2

load r3
load r4
use r3, r4 → r5

save r2
save r5
```

(a)

```
load r0 || r1
use r0, r1 → r2

load r3 || r4
use r3, r4 → r5

save r2 || r5
```

(b)

그림 7. 병렬 접근을 통한 내부 메모리 접근 제어의 예 (a) 일반적인 메모리 접근; (b) 메모리 병렬접근

Fig. 7. Pseudo-code for Parallel Memory Access (a) Sequential Memory Access; (b) Parallel Memory Access.

2. 프로그램 내에서의 최적화

제안하는 마지막 방법은, 프로그램의 재배치를 통해 전력소모 최적화를 달성하는 것이다. 그림 8에서 job A와 job B의 의존성이 존재하지 않는다고 가정한다. 하지만 job B를 수행하기 위하여 내부 메모리로부터 새로운 정보를 레지스터에 저장하는 과정에서 job A의 정보를 손실하게 된다. 의존관계에 있는 job A′의 실행은 손실된 정보의 복원과정이 필연적으로 수반되며 추가적인 메모리 접근이 요구된다. job A가 가져온 피연산자를 job A′도 사용을 하고 있으므로, job B와 job A′의 위치를 바꾼다면 단일 내부 메모리 접근을 통해 job A와 job A′의 연산이 가능해진다. 재배치를 통하여 job A′에서 필요한 6번의 메모리 접근을 줄일 수 있고, 이는 30%의 메모리 접근을 감소시키는 결과로, 이에 상응하는 소모 전력을 낮출 수 있다. 명령어수준의 프로그램 재배치는 개발자의 수동적인 작업이기 보다는 지능적 컴파일러를 통하여 달성될 수 있으며 다양한 시도를 기대할 수 있다.

```
load r0, r1,..., r6
use r0, r1,..., r6 → r7
save r7 // job A

load r1, r2,..., r5
use r1, r2,..., r5 → r6
save r14 // job B

load r0, r1,..., r5
use r0, r1,..., r5 → r6
save r6 // job A′
```

그림 8. 프로그램의 재배치를 통한 내부 메모리 접근 제어 예제

Fig. 8. Pseudo-code Example for Code Relocation.

VI. 결 론

본 논문은 명령어 레벨에서의 메모리 접근 유무에 따른 마이크로컨트롤러의 전력소모를 예측하였다. 이를 바탕으로 특정 프로그램의 내부 메모리 접근 명령어의 비율을 낮춤으로써 마이크로컨트롤러에서 프로그램이 실행되는 동안 소모되는 전력을 최적화하였다.

실험 결과 내부 메모리 접근 명령어의 평균 소모전류는 40.7㎃, 내부 메모리 비접근 명령어의 평균 소모전류는 34.8㎃로 측정이 되었다. 프로그램 내의 내부 메모리 접근 명령어 비율의 감소가 프로그램이 실행되는 동안 마이크로컨트롤러가 소모하는 평균전류량의 감소로 이어짐을 보였다. 이를 바탕으로 특정 프로그램이 실행되는 동안 마이크로컨트롤러가 소비하는 전력을 근사화하여 예측할 수 있었고 더불어 내부 메모리 접근 명령어의 비율을 감소시킴으로써 마이크로컨트롤러의 전력소모를 최적화할 수 있음을 증명하였다.

레지스터 증가, 피연산자에 대한 병렬접근, 코드 재배치를 통하여 마이크로컨트롤러의 전력소모를 최적화할 수 있는 기법에 대해서도 제안하였다. 시스템은 경우에 따라 메모리 계층이 레지스터, 캐쉬, 내부 메모리, 외부 메모리 등으로 세분화될 수 있다. 좀 더 복잡한 메모리 계층에 있어서 전력소모를 단계적으로 최적화하는 연구가 향후 필요하다.

참 고 문 헌

[1] C.X. Huang, B. Zhang, An-Chang Deng, and B. Swirski. "The design and implementation of PowerMill", Proceedings. 1995 International Symposium on Low Powr Design. pp. 105-108, 1995.

[2] F.N Najm. "Transition Density : A New Measure of Activity in Digital Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Vol.14, No. 1, pp. 310-323, 1993.

[3] T. Sato, M. NagamAtsu, and H. Tago. "Power and performAnce simulator : ESP and its application for 100MIPS/W class RISC design" in Proc. Symp. Low Power Electron. San Diego, pp. 46-47, Oct. 1994.

[4] T. Sato, Y. Ootaguro, M. NagamAtsu, and H. Tago, "Evaluation of architecture-level power estimAtion for CMOS RISC processors", in Proc. Symp. Low Power Electron. San Jose, CA, pp. 44-45, Oct. 1995.

[5] Gang Qu, Naoyuki Kawabe, Kimiyoshi Usami, and Miodrag Potkonjak, "Function-Level Power EstimAtion Methodology for Microprocessors", in Proc. Design Automation Conf, pp. 810-813, June 2000.

[6] Mike Tien-Chien Lee, Vivek Tiwari, Shard mAlik, and mAsahiro Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software", IEEE Trans. VLSI System. Vol. 5, no. 1, pp. 123-135, mAr. 1997.

[7] Vivek Tiwari, Sharad mAlik, and Andrew wolfe, "Power Analysis of Embedded Software : A First Step Towards Software Power minimization", IEEE Trans. VLSI System. Vol. 2, No. 4, pp.437-445, Dec. 1994.

[8] Atmel Corp., 8-bit microcontroller ATmega128 complete.

[9] Agilent Technologies Corp., 6000 Series Oscilloscopes User's guide

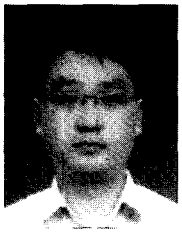
[10] Agilent Technologies Corp., 1147A 50㎒ Current probe User's guide.

[11] Agilent Technologies Corp., 14565A Device Characterization Software Quick Start Quide.

[12] http://sourceforge.net/projects/tinyos

저 자 소 개

엄 홍 식(학생회원)
2006년 동국대학교 전자공학과 학사
2006년 동국대학교 전자공학과 석사과정
<주관심분야 : 컴퓨터 구조, 센서 네트워크>



김 건 욱(정회원)-교신저자
1995년 동국대학교 전자공학과 학사
1997년 University of Florida 석사
2001년 University of Florida 박사
2001년~2003년 Florida State University 조교수
2003년~현재 동국대학교 전자공학과 교수
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>