10

에너지 효율화를 위한 임베디드 소프트웨어 소모전력 분석 기술

충북대학교 ▌홍 장 의*
KAIST ▌배 두 환*

1. 서 론

최근 세계적으로 에너지 관련 문제가 많은 화제의 중심에 와 있다. 특히 에너지 문제는 그린 IT 측면에서 그 중요성이 더욱 강조되고 있는데, 그린 IT라 함은 환경에 미치는 마이너스 효과가 적은 IT 기술이나 IT 산업을 총칭하는 것으로써, 크게 그림 1과 같은 세 가지 영역에서 이러한 목적을 달성하고자 하고 있다[1].

가장 먼저, 제품 생산 측면에서의 그린 IT 분야이다. IT 제품의 생산단계에서 환경에 유해한 물질 사용을 제한하고, 폐 전자제품을 재활용하여 IT 제품을 생산하는 분야의 기술이다. 두 번째는 수요 측면에서의 그린 IT 분야로써, 유해물질의 저사용, 에너지 효율성 증대, 그리고 IT 제품 라사이클링 등이 이에 속하는 기술 영역이다. 마지막은 유통 및 폐기 측면에서의 그린 IT 분야로써, 이는 IT 제품의 유통과정에서 발생하는 비효율을 줄이고, 제품의 폐기과정에서 환경에 유해한 물질을 억제하고자 하는 기술 영역을 의미한다.

특히 경제, 사회, 문화 등 다양한 분야의 기술 발전과 환경 변화에 따라 방대한 IT 제품이 사회의 전반에 걸쳐 활용되고 있기 때문에 에너지 효율성의 증대는 매우 중요하게 인식되고 있으며, 특히 전력량을 감소시키는 분야가 그린 IT에서의 핵심적인 요소라고 인식되고 있다.

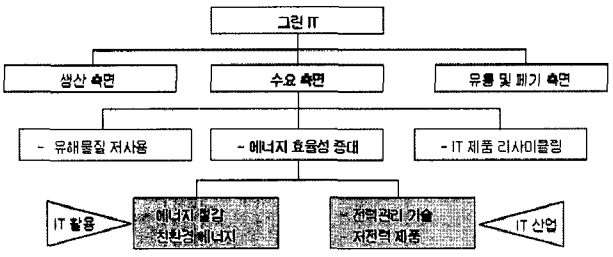

그림 1 그린 IT 추진 영역[1]

본 연구에서는 에너지 효율성을 높이기 위한 전략 중의 하나로써, 에너지 소모가 적은 소프트웨어를 어떻게 개발 할 것인가에 연구의 주안점이 있다. 통신, 우주 항공, 제어 및 로봇, 가전, 건축 등 대부분의 모든 산업 분야에서 다양한 유형의 소프트웨어들이 사용되고 있으며, 이들 임베디드 소프트웨어들은 IT 활용 측면에 있어서 매우 중요한 에너지 절감 요소가 되고 있다.

에너지 절감을 위한 기술은 제품 즉, 시스템 차원에서 고려되는 것이 일반적이다. 그러나 최근 임베디드 시스템의 기능 및 서비스가 다양해지고, 매우 다양한 멀티미디어 타입의 데이터를 처리해야 하는 요구가 급증하고 있다. 따라서 소프트웨어가 어떻게 동작하느냐에 따라 제품의 에너지 소모량을 달라질 수 있으며, 특히 제한된 배터리를 갖는 환경에서는 소프트웨어 소모전력 감소가 더욱 중요한 문제가 되었다.

기존에 임베디드 소프트웨어에 대한 에너지, 즉 전력 소모에 대한 연구는 대체적으로 소스코드 기반의 분석을 통해 수행되어 왔다. 그러나 소스 코드 기반의 소모전력 분석은 그림 2에서 보는 바와 같이 분석을 위해 소요되는 시간이 길고, 또한 분석 결과의 반영을 위한 피드 백 노력도 많이 요구되기 때문에 이러한 문제들을 극복하기 위한 연구들이 진행되고 있다[2].

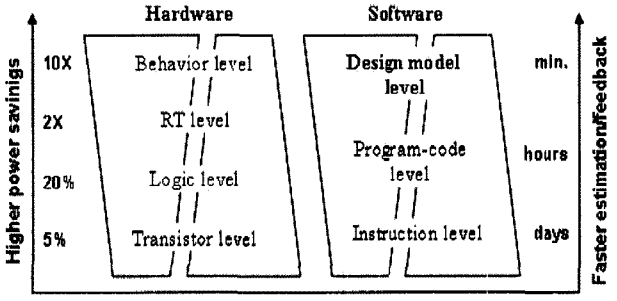

그림 2 추상화수준에 따른 소모전력분석 특성[2]

* 종신회원

† 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음(NIPA-2009-(C1090-0902-0032)).

이러한 연구 방안중의 하나가 명령어나 소스 코드를 보다 상위 수준으로 추상화시켜 분석하는 것이다. 이러한 추상화는 소모 전력을 분석하기 위해 요구되는 시간을 줄이고 또한 에너지 절감 효과도 상대적으로 크다는 장점을 제공할 수 있다. 소스 코드를 추상화한 형태는 로직이나 추상화 데이터 타입 등으로 나타낼 수 있으나, 그 보다 한 차원 높은 추상화의 수준은 설계 모델이다.

개발 대상 소프트웨어에 대한 설계 모델을 이용하여 소모 전력을 분석하는 것은 분석을 위한 별도의 모델을 개발하지 않고도 이루어질 수 있기 때문에 많은 노력의 절감이 이루어질 수 있다. 특히 UML이나 태스크 다이어그램과 같이 기존 소프트웨어 개발 방법론에서 사용하는 표현 방법을 그대로 사용한다면 더욱 유리하다. 본 연구에서는 설계 모델을 근간으로 하는 임베디드 소프트웨어의 소모전력 분석 기법에 대하여 제시한다.

본 논문의 2장에서는 배경 지식으로 소프트웨어 소모전력 분석 기법을 명령어 수준, 소스코드 수준, 그리고 모델 수준으로 구분하여 관련연구들을 살펴보고, 3장에서는 특히 모델 기반의 분석 기법에 대하여 자세히 설명한다. 4장에서는 모델 기반의 소모전력 분석 기법을 이용하여 수행할 수 있는 공학적 활용 분야를 제시하고, 5장에서는 결론을 기술한다.

2. 소프트웨어 소모전력 분석기법

소프트웨어가 소모하는 전력량을 분석하기 위한 기법은 다양하며, 그 분석 시점들도 다양하다. 소모 전력 분석 시점을 소프트웨어 개발 절차를 중심으로 살펴보면 그림 3과 같다[3].

그림 3에서와 같이 소프트웨어의 개발 과정에서 소모 전력의 분석은 개발 활동의 산출물이 무엇인가에 의존적이다. 따라서 소프트웨어 개발 프로세스의 단계별로 생성되는 산출물을 이용하여 소모 전력을 분석할 수 있다. 특히 임베디드 소프트웨어의 개발 과정에서는 최종 타겟 시스템의 설치에 앞서 가상 시스템 기반의 테스팅 활동이 수행되는데, 이 경우에는 소모 전

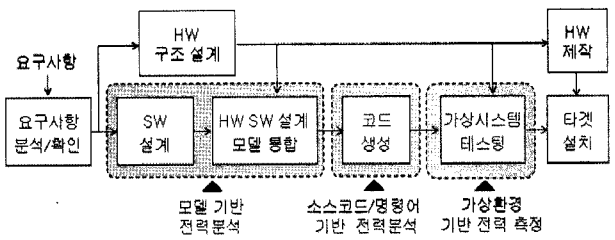

그림 3 개발 과정상의 소모전력 분석시점[3]

력을 분석, 예측하기 보다는 측정 장치를 이용한 계측 활동이 주로 이루어진다. 소프트웨어 개발 단계별 수행되는 소모 전력의 분석 기법에 대한 기존의 연구 동향은 다음과 같다.

2.1 명령어 기반의 분석 기법

임베디드 소프트웨어에 대한 소모전력 분석기법 중에서 가장 대표적인 최초의 연구는 1994년의 Tiwari [4,5]의 연구에서 찾아볼 수 있다. Tiwari는 소프트웨어에 대한 명령어 수준에서의 소모 전력을 분석하기 위하여 특정한 명령어나 짧은 명령어 순열을 실행할 때 프로세서에 흐르는 전류의 양을 측정하였다. Tiwari에 의해 제안된 명령어 기반의 소프트웨어 소모 전력을 측정하기 위해서는 크게 각 명령어를 수행하기 위해 요구되는 Base Energy Cost와 서로 다른 명령어를 수행하는 과정에서 발생하는 서킷(circuit) 변화에 의한 Inter-instruction Effect Cost[6]로 구분하였다.

Base Energy Cost, $E = I \times Vcc \times N \times \tau$에 의해 결정되며, I는 평균 전류, Vcc는 공급 전압, N은 명령어 클럭 사이클의 수, 그리고 τ는 클럭 주기이다. 명령어 기반의 소모전력 분석에서 공급 전압과 클럭 주기는 하드웨어 아키텍처에 따라 고정적이며, 클럭 사이클의 수 또한 명령어마다 일정하기 때문에 각 명령어를 수행할 때 프로세서에 흐르는 전류의 양만 측정하면 명령어에 의해 소모되는 에너지를 계산할 수 있다고 정의하였다. Inter-instruction Effect Cost는 서로 다른 명령어가 순차적으로 실행될 경우, 제한된 자원에 따른 영향으로 인해 파이프라인 스톨(pipeline stall), 쓰기 버퍼 스톨, 캐스 미스(catch miss) 등이 일어날 수 있기 때문에 고려된 것이다. 이와 같은 Base Energy Cost와 Inter-instruction Effect Cost는 다양한 형태의 명령어들에 대한 반복적인 실험을 통해 측정하고 평균값을 계산하여 산출하였다.

2.2 소스코드 기반의 분석 기법

Tan[7]은 소프트웨어를 구성하는 내장함수와 사용자 정의함수에 대하여 명령어 수준에서의 시뮬레이션을 수행한 결과를 이용하여 매크로 모델(macro-model)을 정의하고, 이를 기반으로 소프트웨어에 대한 소모 전력을 분석하였다. 매크로모델을 이용한 소모전력 도출 방법은 함수의 성격에 따라 (1) 복잡도 기반의 예측 방법과 (2) 프로파일링 기반의 예측 방법이 있다. 복잡도 기반의 분석기법은 멀티미디어나 데이터 처리를 수행하는 함수에 주로 사용한다. 매크로 모델의 변수는 함수에서 요구하는 입력 데이터의 크기 등에 의해 도출되며, 매크로 모델의 상수나 변수의 차수 등

은 Θ 표기법에 의한 알고리즘의 복잡도를 기반으로 도출된다.

한편 프로파일링 기반의 방법은 복잡도 기반의 방법으로 매크로모델을 도출하기 어려운 함수들에 대하여 소스코드를 프로파일링(profiling) 함으로써 매크로 모델을 도출한다. 프로파일링은 소스코드에 대한 반복적인 실행을 통해 다수의 소모전력 실측 데이터를 확보하고, 이를 회귀분석하여 얻은 계수 값을 이용하여 매크로 모델을 정의하는 방법이다. 매크로모델의 도출이 끝나면 해당 함수에 대한 매크로 모델과 이에 요구되는 변수를 이용하여, 소프트웨어의 소스코드 기반 전력 소모량 예측을 수행할 수 있다.

소스코드 기반의 전력 소모를 분석하기 위한 또 다른 방법은 소스코드에서 제공되는 함수들의 에너지 모델을 추출하기 위해 테스트 코드를 작성하고, 이를 이용하여 명령어 수준에서의 에너지 시뮬레이션을 수행하는 방법이다[8]. 시뮬레이션 결과는 데이터 저장소에 구축 관리되며, 프로그램을 입력받아 프로그램의 실행정보를 추출하고, 실행 정보에 해당되는 소모 전력량을 저장소로부터 참조하여 전체 소모 전력량을 산출한다.

2.3 모델 기반의 분석 기법

모델 기반의 소모 전력을 분석하기 위해 대표적인 연구도 Tan[2]에 의해 수행되었다. Tan은 임베디드 소프트웨어에 대하여 Software Architecture Graph(SAG)라는 소프트웨어 아키텍처 모델을 이용하여 아키텍처 레벨에서 에너지 소모에 영향을 미치는 인자들을 찾아내고 이를 바탕으로 에너지 효율적인 소프트웨어 아키텍처 모델을 설계하도록 하는 방법을 제안하였다. 에너지 효율적인 모델을 식별하기 위하여 SAG를 특정 소스 코드로 구현한 후, 소스 코드 기반의 에너지 시뮬레이션 도구를 사용하여 에너지 프로파일링을 수행한다. 그래프의 노드에 대한 통합 및 연산 이주(computation migration) 등을 통해 변형된 SAG를 생성하고, 이를 구현하여 에너지 프로파일링을 수행한다. 이렇게 함으로써, 그래프 변형 전후의 아키텍처 모델에 대한 전력 소모를 상호 비교할 수 있다.

비주얼한 모델을 이용하여 소모 전력을 분석하는 또 다른 방법은 컴포넌트가 갖는 내부행위를 오토마타로 모델링하고, 이의 도달성 그래프(reachability graph)를 생성하여 각 실행 패스(path)별로 에너지 소모량을 산정하는 방법이다[9]. 이를 위해서는 오토마타의 상태에서 요구되는 Time interval과 에너지 소모량을 추가하여 확장된 에너지 인터페이스 오토마타를 생성하고, 이를 이용하여 도달성 그래프를 생성한다. 또한 Yue[10]는 IP(Intelligence Property) 기반의 임베디드 소프트웨어 개발 과정에서, 각 IP의 내부에 소모전력을 측정하기 위한 명령어 주해(annotation)를 추가하고, 이를 기반으로 전체 소프트웨어에 대한 소모전력을 산출한다. 특히 IP의 Idle 상태와 Active 상태를 모두 고려하여 정적 에너지 모델과 동적 에너지 모델을 생성하고, 이를 이용하여 IP의 소모 전력을 산출한다.

위에서 제시한 명령어, 소스코드, 모델 기반의 소모전력 분석들은 다양한 도구들에 의해 지원되고 있다. SimplePower[11], JouleTrack[12], SoftExplorer[12], 그리고 EMSIM[14] 등은 명령어나 소스코드를 기반으로 소프트웨어 소모 전력 분석을 지원하는 도구이며, ESUML-EAF[15]는 UML 모델을 입력으로 하는 모델 기반의 분석 도구이다.

3. 모델 기반 소모전력 분석

3.1 모델 기반 분석 개요

임베디드 소프트웨어의 설계 모델을 이용하여 소모전력을 분석하기 위해서는 다음의 정보가 선행적으로 제공되어야 한다.

- 분석 대상 소프트웨어 또는 소프트웨어 컴포넌트의 기능적 행위 모델이 제공되어야 한다.
  분석대상 소프트웨어 컴포넌트에 대한 기능적 행위 모델은 UML[16]을 이용하는 경우, 시퀀스 다이어그램과 Action Language(AL)[17]를 이용하여 표현할 수 있다. 시퀀스 다이어그램은 시간의 흐름에 따른 소프트웨어의 동작을 묘사하고 있으며, 특히 임베디드 소프트웨어의 경우 C나 C++와 같은 프로그래밍 언어 수준으로 소프트웨어의 동작을 표현할 수 있다. 또한 특정 매크로 함수 수준의 수학식은 AL을 이용하여 표현할 수 있다.
- 설계 모델을 구성하는 각 요소들에 대한 전력 소모 정보가 제공되어야 한다.
  설계 모델을 입력으로 소모 전력을 분석하기 위해서는 모델의 구성요소 - 예를 들어 UML의 시퀀스 다이어그램의 경우는 동기적 메시지 전달, 비동기적 메시지 전달, 루프(loop), 선택(alt), 병렬(par)을 표현하는 combined fragments 등 - 에 대한 분석을 통해 전력을 소모하는 요소들을 결정해야 한다. 예를 들면, 시퀀스 다이어그램에 나타나는 Lifeline이나 State_Invariant 등은 행위를 갖는 표현요소가 아니기 때문에 소모 전력의 대

상이 될 수 없다. 식별된 모델 구성 요소들에 대해서는 각 요소가 소모하는 전력량을 - 3.2절에서 설명하는 방법들을 이용하여 - 산출할 수 있도록 지원되어야 한다.

3.2 모델 기반 분석 프레임워크

3.1절의 설명을 기반으로 소프트웨어 설계모델 기반의 소모 전력을 분석하기 위해서는 그림 4와 같은 분석 프레임워크가 제공되어야 한다.

그림 4에 나타난 프레임워크의 구성요소에서 모델 요소별 에너지 산출기와 모델 파서에 대하여 설명한다. 행위중심의 설계 모델러는 일반적인 UML 다이어그램 작성기를 의미하며, 결과 분석기는 설계 모델의 소모 전력 분석 결과를 다양한 형태로 보여주기 위한 일반적인 기능을 갖는다.

3.2.1 모델요소별 에너지 산출기

모델 파서가 소프트웨어 설계 모델을 파싱하여 전력을 소모하는 모델 요소를 찾아내면 해당 모델 요소에 대한 소모 전력량을 산출해야 한다. 이때 파서는 해당 모델 요소의 식별자를 에너지 산출기에게 보내 전력 소모량을 반환받는다. 에너지 산출기가 모델요소의 소모전력을 결정하기 위한 방법은 크게 다음의 두 가지 방법에 의해 이루어질 수 있다.

가. 코드 구현을 통한 예측

구현을 통한 방법은 모델 요소를 C 언어와 같은 특정 언어로 구현하고, 구현된 언어를 명령어 수준으로 분석하여 명령어 패턴을 결정하고 이들에 대한 소모 전력량을 산출하는 방법이다. 예를 들면, UML 시퀀스 다이어그램의 "alt" combined fragment의 경우는 조건변수를 로드하는 load 명령어, 조건식을 비교하기 위한 compare 명령어, 그리고 비교 결과에 따라 분기하는 branch 명령어로 구성된다. 따라서 load, compare, branch 명령어가 소모하는 소모 전력을 산출하여 "alt"의 최종 소모전력 값을 결정한다. 이때 load, compare, branch 명령어에 대한 각각의 소모 전력은 EMSIM 2.0[14]과 같은 소스코드 기반의 소모전력 분석 시뮬레이터를 이용하여 구할 수 있다. 표 1은 UML 모델 요소에 대한 명령어 패턴에 대한 예를 보여주고, 표 2는 StrongARM 기반의 EMSIM 2.0 시뮬레이터를 이용하여 산출된 명령어별 소모 전력량을 보여준다.

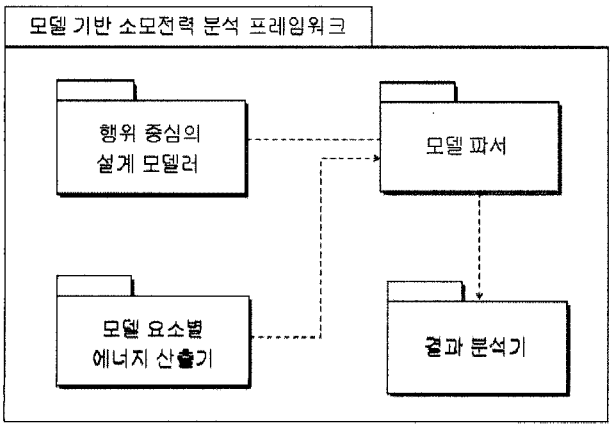

그림 4 모델 기반 소모전력 분석 프레임워크

표 1 모델 구성요소의 명령어 패턴

모델 요소	명령어 패턴
msg send	msgsnd()
sync. call	store + branch
asynch. call	branch + fork() + msgsnd() + msgrcv()
signal	signal()
alt	load + compare + branch
par	fork()
:	:
AL: >>	bitShifrRight()

표 2 명령어별 소모 전력량

구분	명령어	타입	소모전력(nJ)
명령어	load	int	13.016
	store	char	13.1765
	compare	int	10.1905
	branch	N/A	1.91309
	:	:	:
시스템 함수	함수명	파라메터	매크로(nJ)
	fork()	N/A	E=132202.6
	signal()	N/A	E=9460.6
	msgsnd()	c bytes	E=4.0c+4554.0
	fopen()	N/A	E=17886.6
	:	:	:

나. 산출식을 이용한 예측

산출식을 이용한 예측 방법은 각 모델 요소가 갖는 시스템 수준의 행위를 분석하고 이를 수학식으로 정형화하는 방법이다. 일반적으로 소프트웨어가 소모하는 전력량은 다음과 같은 식에 의해 표현될 수 있다.

$$E(S)=\sum_{t=1}^{l}\sum_{e=1}^{m}\sum_{i=1}^{n}(1+S_i+C_i)\times J_{t,e,i} \quad (1)$$

위 수식 (1)에서 보여주는 것과 같이 소프트웨어 소모 전력은 소프트웨어가 갖는 모델의 수(l), 각 모델에 포함된 모델 요소의 수(m), 각 모델 요소가 포함하고 있는 명령어의 수(n), 각각 명령어의 파이프라인 스

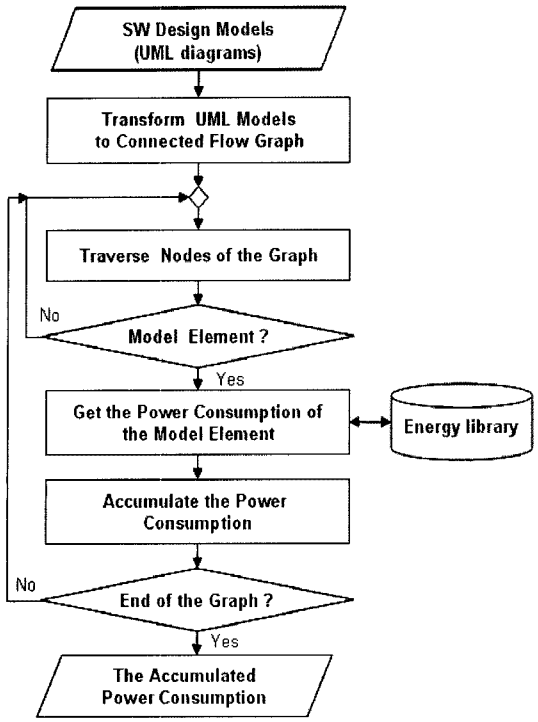

그림 5 모델 파서의 수행 절차

톨 확률(Si)과 메모리 캐시 미스 율(Ci), 그리고 해당 명령어의 소모 전력(J)의 이용하여 산출된다.

에너지 소모량을 구하기 위한 파이프라인 스톨 확률과 캐시 미스율 등은 하드웨어 아키텍처를 고려한 실측 데이터의 회귀분석을 통해 얻을 수 있다.

3.2.2 모델 파서

모델 파서의 역할은 UML을 사용하여 계층 구조로 작성된 임베디드 소프트웨어의 설계모델을 1차원적 평면으로 펼쳐(unfolding) 하나의 연결된 그래프(connected graph)로 변환하고, 이 그래프를 시작노드부터 종료노드까지 탐색해 가는 것이다. 탐색하면서 하나의 모델 요소를 식별하게 되면, 이에 대한 소모 전력량을 라이브러리로부터 얻어오게 된다. 그래프의 종료 노드에 도달하게 되면 해당 설계 모델의 소모 전력량이 산출된다. 이와 같은 모델 파서의 수행 절차는 그림 5와 같이 나타낼 수 있다.

4. 모델 기반 분석기법 평가 및 적용

4.1 모델 기반의 분석 기법의 평가

본 연구를 통해 제시된 UML 설계 모델 기반의 분석 기법은 Java Eclipse를 이용한 도구로 구현되었다. 그림 6은 구현된 에너지 라이브러리의 스크린을 보여주며, 그림 7은 파서에 의해 분석된 버블 정렬(bubble sort) 알고리즘에 대한 모델 요소별로 분석 과정에 대한 스크린, 그리고 그림 8은 분석 결과를 그래프 형태로 보여주는 스크린의 예이다.

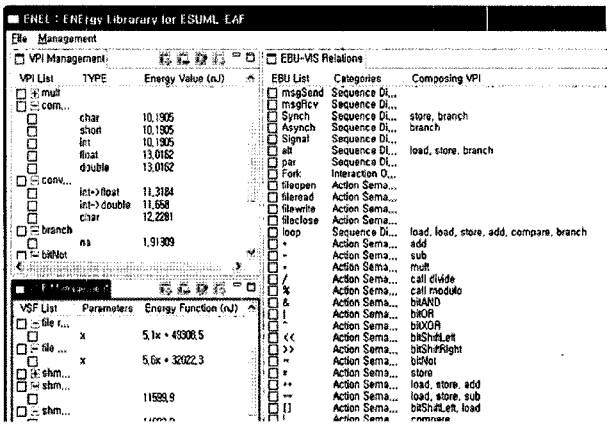

그림 6 에너지 라이브러리 화면

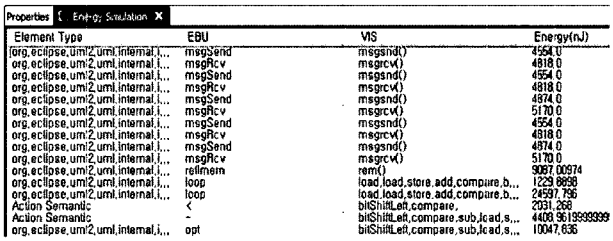

그림 7 모델 파서의 분석 과정 화면

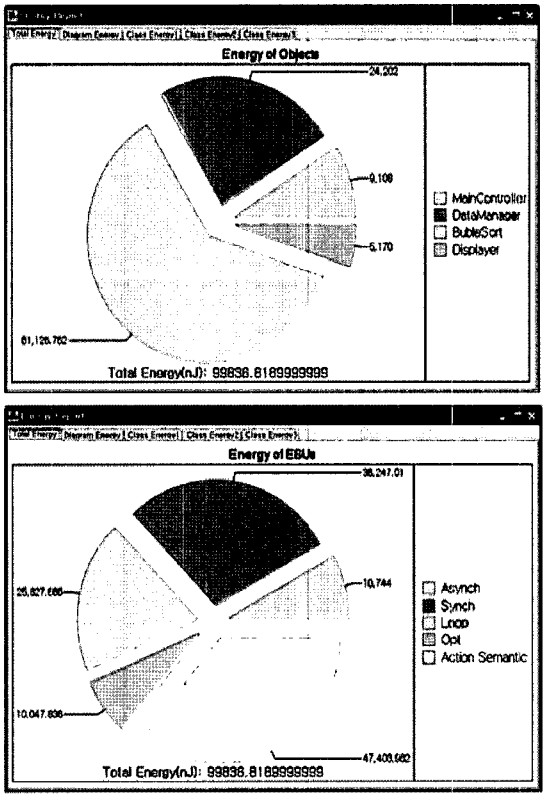

그림 8 소모전력 분석 결과 전시화면

구현된 도구에 의하여 모델링된 정렬 알고리즘의 소모 전력량은 99836.818999nJ이었으며, 이 알고리즘을 ANSI C 언어로 구현하여 시뮬레이션 한 결과 113203.616505nJ을 보였다. 더욱 많은 실험과 큰 시스템의 예제 적용을 통하여 최종 결과를 결정할 수 있

지만, 다수의 예제 적용을 통한 본 연구의 결과는 코드기반 소모 전력과 비교하여 10% 수준의 편차가 발생함을 알 수 있었다. 이는 구현 코드를 수정해야 하는 피드백 노력을 고려할 때 충분한 적용 가능성을 보여준 결과라고 판단된다.

4.2 모델 기반 분석 기법의 적용 영역

앞의 3장에서 설명한 소프트웨어 설계 모델기반의 분석기법은 다음과 같은 다양한 영역으로 활용될 수 있을 것으로 판단된다.

4.2.1 소프트웨어 설계 대안 결정

흑자는 소프트웨어 모델링이 예술(art)적인 창작활동이라고 말한다. 이는 아마도 비슷한 능력의 서로 다른 소프트웨어 엔지니어라도 동일한 기능에 대한 모델링 결과가 상이할 수 있다는 것을 의미한다. 따라서 특정 기능에 대한 소프트웨어의 설계 모델은 매우 다양한 모습으로 나타날 수 있다. 그러나 개발된 설계 모델들이 요구사항을 모두 충족한다고 하더라도 소모 전력의 측면에서는 서로 다를 수 있다.

본 연구에서 제시하는 모델 기반의 소모전력 분석 기법은 이와 같이 서로 다른 모습으로 작성되는 설계 모델에 대한 소모 전력을 분석함으로써, 보다 에너지 효율적인 설계 대안을 결정할 수 있다. 또한 작성된 설계 모델이 많은 전력을 소모한다고 하면, 이 모델을 수정하여 보다 적은 전력을 소모하는 설계 모델을 개발할 수 있다.

4.2.2 CBD 기반의 소모전력 분석

모델 기반의 소모전력 분석의 적용이 가능한 또 하나의 유용한 영역은 컴포넌트 기반의 소프트웨어 개발 과정이다. 컴포넌트는 재사용 가능한 소프트웨어 모듈로써, 자신에 대한 기능 명세를 가지고 있다. 수용 가능한 신뢰성을 갖는 컴포넌트를 이용하여 소프트웨어를 개발할 때, 노력과 시간의 절감이 가능하기 때문에 매우 유용하다. 그러나 현재까지는 소프트웨어 컴포넌트들이 기능적인 명세만을 가지고 있지, 컴포넌트가 얼마나 많은 소모 전력을 유발하게 될 지에 대한 정보는 가지고 있지 못하다.

본 연구에서 제시하는 모델 기반의 소모전력 분석 기법은 재사용 가능한 컴포넌트의 명세에 추가하여 전력 소모에 대한 명세를 포함할 수 있도록 지원할 수 있다. 컴포넌트의 소모 전력의 측정은 코드를 기반으로 수행할 수 있겠지만, 소프트웨어 아키텍처 수준에서 재사용 컴포넌트를 식별하고자 할 때는 모델 기반의 소모 전력 분석기법을 적용하여 소모 전력에 대한 요구사항 충족 여부를 결정할 수 있다.

4.2.3 저전력을 위한 태스크 분할

지능형 서비스를 요구하는 임베디드 시스템의 경우 대체적으로 MPSoC와 같은 멀티프로세서 아키텍처를 사용한다. 멀티프로세서 아키텍처를 기반으로 하는 임베디드 시스템의 경우 성능상의 이득을 제공하기는 하지만, 소모 전력의 측면에서는 이득이 감소할 수 있다. 따라서 수용 가능한 수준에서의 성능을 보장하면서 소모 전력을 감소시킬 수 있는 방법이 필요하다.

모델 기반의 소모 전력의 분석을 프로세서에 할당할 태스크를 식별하는 과정에 적용하여 식별되는 태스크들이 보다 적은 전력을 소모하도록 유도할 수 있다. 소모 전력을 유발하는 요인들을 정하고, 이 요인들을 태스크 분할 과정에 적용한다. 특히 설계 모델을 기준으로 태스크를 분할하는 MDD(Model-Driven Development) 기반의 소프트웨어 개발에서는 태스크 분할을 위한 기준으로 전력 소모량을 고려할 때, 분할된 태스크 모델을 입력으로 소모 전력을 분석할 수 있다.

5. 결 론

최근 그린 IT의 중요성이 강조되면서, 에너지 절감형 IT 기술 개발에 노력하고 있다. 특히 많은 전력을 소비하는 데이터 서버의 소모 전력 감소기술, 지능형 빌딩에서의 에너지 절감 기술, 자동차, 항공 분야와 같은 융합 IT 분야에서의 에너지 활용 기술 등이 대표적인 에너지 절감 잠재력 분야 영역이다. 그러나 이러한 응용 영역에서 중요하고 공통적으로 요구되는 기반 기술은 임베디드 소프트웨어이다. 많은 시스템에 내장되는 소프트웨어의 소모 전력을 감소시키기 위한 노력은 중요하다.

본 연구는 임베디드 소프트웨어의 소모 전력을 분석하기 위한 연구의 일환으로써, 그동안 수행되어 왔던 코드 기반의 소모전력 분석 기법을 기반으로 모델 수준에서의 소모 전력 분석을 지원하기 위해 연구하였다. 모델 수준에서의 분석은 코드 수준에서의 분석보다 소프트웨어 개발의 앞선 단계에 이루어질 수 있기 때문에 설계 모델에 대한 대안 분석을 가능하게 하며, 또한 소모전력 요구사항에 대한 충족 여부를 보다 사전에 점검할 수 있다는 장점을 제공한다.

성공적인 모델 기반의 소모전력 분석이 이루어진다면, 임베디드 소프트웨어의 개발에 있어서 MDD 방법론의 유용성이 증대될 수 있으며, 소프트웨어 아키텍처를 기반으로 하는 제품 계열 생산 방식에서의 활용성도 높아질 수 있을 것으로 판단된다.

참고문헌

[1] 이은민, 임순옥, "그린 IT 추진을 위한 규제 및 대응방안", 정보통신정책, 20권 12호, pp. 1-21, 2008년 7월

[2] T. K. Tan, et. al., "Software Architectural Transformations: A New Approach to Low Energy Embedded Software", Proc. the Design, Automation and Test in Europe Conference and Exhibition, pp. 1046-1051, 2003.

[3] J. Kim, D. Kim, and J. Hong, "Extracting and Applying a Characteristic Model with Survey of Power Analysis Techniques for Embedded Software", Journal of KIISE: Software and Applications, Vol. 36, No.2, pp.376-385, May 2009.

[4] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software : A First Step Towards Software Power Minimization," in IEEE trans. VLSI syst., Vol. 2, No. 4, pp. 437-445, 1994.

[5] M. T. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," in IEEE Trans. VLSI Syst., Vol. 5, No. 1, pp. 123-135, 1997.

[6] B. Klass, et. al, "Modeling Inter-instruction energy effects in a digital signal processor," in Power Driven Microarchitecture Workshop and 25th Int'l Symposium on Computer Architecture, 1998.

[7] T. K. Tan, A. Raghunathan, "High-Level Energy Macromodeling of Embedded Software", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 9, Sep. 2002.

[8] G. Qu, N. Kawabe, K. Usami, et al., "Code Coverage-Based Power Estimation Techniques for Microprocessors", Journal of Circuits, Systems, and Computers, Vol. 11, No. 5, pp. 1-18, 2002.

[9] Hu Jun, et. al., "Modelling and Analysis of Power Consumption for Component-based Embedded software", Proc. EUC Workshop, pp.795-804, 2006.

[10] X. Yue, et. al., "OOEM: Object-Oriented Energy Model for Embedded Software Reuse", Proc. International Conference on Information Reuse and Integration, pp.551-558, 2003.

[11] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," In : Proc. Design Automation Conf.(DAC'00), pp.340-345, 2000.

[12] A. Sinha and A. P. Chandrakasan, "JouleTrack - A Web based Tool for Software Energy Profiling," in Proc. 38th IEEE Conference on Design Automation (DAC'01), pp. 220-225, June 2001.

[13] E. Senn, et. al., "SoftExplorer: Estimating and Optimizing the Power and Energy Consumption of a C Program for DSP Application," EURASIP Journal on Applied Signal Processing, Vol.16, pp.2641-2654, 2005.

[14] T. K. Tan, A. Raghunathan, et al, "EMSIM: An Energy Simulation Framework for an Embedded Operating System," Proc. of International Symposium of Circuits and Systems, pp. 464-467, 2002.

[15] ESUML, "ESUML: Embedded Software Modeling using UML 2.x", [Online]. Available: http://selab.cbnu.ac.kr/projects/mpsoc/index.html/

[16] OMG, "Unified Modeling Language: Superstructure" V2.1.2, 2007, [Online]. Available: http://www.omg.org.

[17] OMG, Concrete Syntax for a UML Action Language, Aug. 2008.

홍 장 의

2001 한국과학기술원 전산학(공학박사)
2002 국방과학연구소 선임연구원
2002~2004 (주)솔루션링크 기술연구소장
2005 한국소프트웨어진흥원 외부자문교수
2006 한국국방연구원 자문위원
2004~현재 충북대학교 컴퓨터공학 부교수
관심분야 : 소프트웨어공학, 임베디드 소프트웨어, 소프트웨어 품질공학, 소프트웨어 프로세스
E-mail : jehong@chungbuk.ac.kr

배 두 환

1980 서울대학교 조선공학(학사)
1987 미국 Univ. of Wisconsin-Milwaukee 전산학(석사)
1992 미국 Univ. of Florida 전산학(박사)
2002~현재 ITRC SPIC 센터장
2008~현재 한국정보과학회 소프트웨어공학소사이어티 회장
1995~현재 한국과학기술원 전산학과 교수
관심분야 : 소프트웨어 프로세스, 객체지향 프로그래밍, 컴포넌트기반 프로그래밍, 임베디드 소프트웨어 설계, 관점지향 프로그래밍
E-mail : bae@se.kaist.ac.kr