임베디드 소프트웨어의 모델기반 전력분석을 위한 에너지 라이브러리 구축*

김두환, 김종필, 홍장의
충북대학교 전자계산학과
e-mail : {dhkim, kimjp}@selab.cbnu.ac.kr, jehong@chungbuk.ac.kr

Building an Energy Library for Model-based Power Analysis of Embedded Software

Doo-Hwan Kim, Jong-Phil Kim, Jang-Eui Hong
Dept of Computer Science, Chungbuk National University

요 약

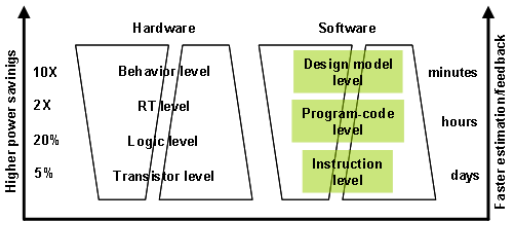
임베디드 시스템에서는 기능적 요구사항 뿐만 아니라 전력량, 응답시간, 견고성 등의 여러 가지 비기능적 요구사항들도 중요하다. 그중에서 전력량에 대한 비기능적 요구사항은 휴대형 임베디드 시스템의 운영에 있어서 핵심적인 요소이다. 임베디드 소프트웨어의 복잡도 및 크기 증가로 전력 소모량이 증가하고 있는 추세이며, 그로인해 소프트웨어 기반의 저전력 소모를 위한 임베디드 시스템 개발 기술이 활발히 연구되고 있다. 본 논문에서는 임베디드 소프트웨어 개발의 선행단계에 설계모델 기반으로 소프트웨어 전력소모량을 예측하기 위하여 요구되는 에너지 라이브러리를 구축한다.

1. 서론

임베디드 시스템을 구성하는 소프트웨어는 일반적인 소프트웨어의 품질요소 이외에도, 시스템의 특성에 의해 추가되는 여러 가지 품질요소가 있다. 그중에서도 휴대형 임베디드 시스템에서는 전력소모량이 시스템의 운영에 미치는 영향이 크기 때문에 제한된 전력량을 효율적으로 사용하는 것이 매우 중요하다. 그로인해 소프트웨어의 전력 분석에 대한 기술 도입이 필요하게 되었다.

현재 진행된 소프트웨어에 대한 대부분의 전력분석 방법들은 명령어 수준 또는 프로그램 코드 수준에서 분석을 수행하는 것에 초점을 맞추어 왔다[1, 2, 3]. (그림 1)에서 살펴볼 수 있듯이, 소프트웨어에 대한 전력분석은 추상화수준에 따라 명령어 수준, 코드 수준, 설계모델 수준 등으로 구분되며, 각 수준에서의 전력분석 방법들은 분석 시간 및 적용 대상 등에서 차이를 보인다[4].

명령어 수준과 프로그램 코드 수준의 전력분석 방법은 소프트웨어 구현단계 이후에 전력을 분석을 수행할 수 있으며, 분석결과의 정확도가 상대적으로 높다. 설계모델 수준의 전력분석 방법이 제공하는 분석결과의 정확도는 앞의 두 가지 방법보다 다소 낮지만, 구현 단계보다 이른 설계 단계에서 이루어지기 때문에, 다른 두 방법보다 전력분석 결과에 대한 feedback 노력이 감소된다. 특히 복잡하고, 다양한 기능을 갖는 임베디드 소프트웨어의 경우 저전력 요구사항을 만족시키지 못하는 경우, 소프트웨어



(그림 1) HW와 SW의 전력분석 추상화 수준에 따른 분석효과[4]

를 처음부터 재개발 하는 문제가 유발될 수도 있다. 이러한 문제를 극복하기 위해 본 연구에서는 임베디드 소프트웨어의 설계모델을 이용하여 소모 전력을 분석하기 위한 방법을 개발하고 있다. 이 전력분석 방법을 통해 UML 등과 같은 모델링 언어로 표현된 설계모델로부터 전력 소모량을 예측해 봄으로써 모델의 어느 부분에서 전력소모가 많은지, 전체적으로 소모 전력의 병목현상이 어디에서 발생하는지를 사전에 식별하여 설계모델의 품질을 향상시키고자 한다.

본 연구에서 개발 중인 전력분석 방법의 수행과정 중에는 소프트웨어의 모델요소에 대한 전력특성 정보를 담고 있는 기반 라이브러리가 필요하다. 물론 이러한 라이브러리를 사용하지 않아도 수학적 모델을 개발하여 소모 전력을 직접 예측할 수도 있지만, 모델의 변화에 기민하게 적용하기 어렵다는 문제가 있다. 에너지 라이브러리의 사용은 두 가지 장점을 갖는데, 첫 번째가 시뮬레이션 기

* 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임.(NO.R01-2008-000-20485-0)

반의 전력분석 프레임워크에서의 소요시간 문제를 개선할 수 있다는 것이고, 두 번째는 소프트웨어 설계모델과 하드웨어 설계모델을 분리할 수 있도록 하여, 하나의 소프트웨어 설계모델에 대해서 다양한 하드웨어 플랫폼에 대한 분석 결과를 얻을 수 있다는 것이다. 덧붙여 임베디드 소프트웨어의 동작이 하드웨어와 밀접한 관계가 있기 때문에 하드웨어 아키텍처를 고려한 소모 전력이 분석되어야 하지만, 본 연구에서는 하드웨어 플랫폼의 특성을 배제한 저전력 소모 소프트웨어 개발에 주안점을 두고 있다.

2. 관련연구

최근 많은 연구가 이루어지고 있는 저전력 소모 소프트웨어 개발방법의 유형을 살펴보면 프로그램 코드 기반의 분석방법[2, 3], 시뮬레이션 수행에 의한 측정방법[5, 6, 7], 매크로모델을 이용한 분석방법[6, 8, 9, 10] 등이 있다. 본 연구는 모델기반으로 전력분석을 수행하는데, 이를 위해서는 고수준의 명령어 레벨에서 전력이 얼마나 소모되는지 알 필요가 있다. 이러한 정보를 전력분석에 사용하기 위해서는 전력특성 모델을 포함 하는 에너지 라이브러리가 필요하다. 소프트웨어 전력 분석에 라이브러리를 사용한 기존에 진행된 대표적인 연구에는 Gang Qu의 연구와[3] A. Muttreja의 연구[6], T. K. Tan의 연구[8, 10]가 있다.

Qu의 연구에서는 프로그램 코드를 "Built-in library functions", "User-defined function"의 두 가지 종류로 분류하고, 각각에 대한 프로그램코드 수준에서의 전력 소모량을 측정·저장하여, "Power data bank"라는 라이브러리를 구축하였다. 이렇게 구축된 라이브러리는 코드기반의 시뮬레이션을 통한 전력분석을 위해 사용된다.

Muttreja의 연구는 임베디드 소프트웨어의 전력분석을 두 단계로 나누어, 첫 번째 단계에는 주어진 프로그램에 대한 에너지 특성을 기존에 제시된 방법[9]에 의해 매크로모델링을 수행하고 생성된 매크로 모델은 "Macromodel-library"에 저장한다. 두 번째 단계는 시뮬레이션 환경을 구축하고, 프로그램 코드와 정의된 매크로 모델을 이용하여 시뮬레이션을 수행하는 단계이다. 이 방법은 소프트웨어와 하드웨어의 모델간의 의존도가 높기 때문에 시뮬레이션을 수행할 때마다 매크로 모델을 생성한다.

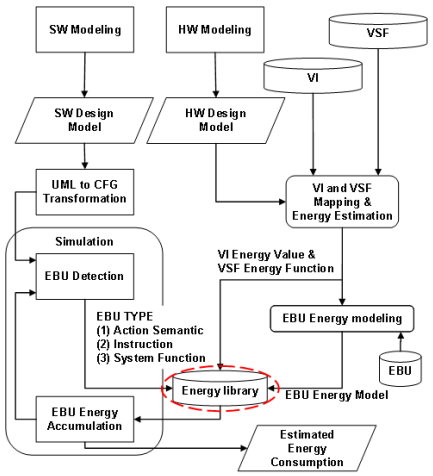
Tan의 연구는 OS의 소스코드를 에너지 특성에 따라 분석한 후, 테스트 프로그램을 통해 에너지 시뮬레이션을 수행하여 OS가 가지는 에너지 특성에 대한 매크로모델을 정의하였다.

이상에서 제시한 연구들은 모두 프로그램 코드 수준에서의 전력분석을 수행하기 위해 라이브러리를 구축하였으며, 구축된 라이브러리는 프로그램 코드 기반의 시뮬레이션 수행에 사용하였다. 그러나 본 연구에서는 그와 같은 라이브러리를 구축하기 위해 추가적인 명령어 집합을 정의하였으며 특히 전력분석을 통해 얻은 라이브러리를 그보다 높은 수준인 설계 모델에서의 전력분석에 활용한다는 점에서 앞서 제시한 연구들과 차이를 보인다.

3. 모델기반의 소프트웨어 소모 전력 분석

3.1 분석 개요

모델기반의 소프트웨어 소모전력을 분석하기 위해서는 기존의 소스코드/명령어 수준의 전력분석이 아닌 명령어 및 시스템 함수들을 추상화시켜서 설계모델을 구성하는 요소들과 매핑하기위한 방법이 필요하다. 따라서 본 연구에서는 명령어 및 시스템함수들을 추상화한 Virtual Instruction(VI)과 Virtual System Function(VSF) 개념을 적용하여 소프트웨어의 전력분석을 수행하며, 개략적인 절차는 (그림 2)와 같다.



(그림 2) 모델 기반의 전력 소비 예측 과정

UML등을 이용한 소프트웨어의 모델링이 완료된 후에는 모델의 구성요소들을 VI, VSF에 매핑 시키는 작업을 수행한다. VI는 실행에 영향을 받지 않으므로 스칼라 값을 가지며, VSF는 실행조건에 따라 전력소모량이 달라지므로 매크로모델로 정의한다.

시뮬레이션을 수행할 때, 에너지 단위가 되는 모델 원소들을 Energy Behavioral Unit(EBU)라고 한며, 각 EBU는 “Instruction Type”, “System Function Type”, “Action Semantic Type”의 세 가지 타입으로 구분된다. Action Semantic 타입인 경우에는 해당하는 VI들을 기반으로 에너지를 측정하고, 그 외의 경우는 각각의 EBU마다 에너지 모델을 만들어 서비스의 OS종속성 여부에 따라 VI 집합 또는 VSF 집합으로 대응되어 전력 분석을 지원한다.

플랫폼에 독립적인 소프트웨어 설계 모델을 시뮬레이션 하기 위해 본 연구에서는 시뮬레이션 모델 Control Flow Graph(CFG)를 이용한다. CFG를 통한 시뮬레이션

수행은 그래프를 구성하는 모든 요소들을 각각 EBU로 매핑하고, 정의된 EBU 에너지 함수를 이용하여 소모 전력을 구한다.

3.2 에너지 라이브러리

본 연구에서 구축한 에너지 라이브러리를 정형화하여 표현하면 **[정의 1]**과 같다.

[정의 1] 에너지 라이브러리 L_e = $<C_G, T_E, M_e, \alpha>$으로 구성된다. 여기서

- C_G : CFG에 나타난 모델 요소
- T_E : EBU Type = {VI | VSF | ActionSemantic}
- M_e : 에너지 정보를 나타내는 모델 요소(EBU)
- α : 매핑($T_E \mapsto M_e$)을 수행하는 액션의 집합

[정의 2] 에너지 모델요소 M_e는 다음과 같은 요소들의 집합이다.

M_e = {I, S, A}

- I : VI에 대한 에너지 값들의 집합
- S : VSF에 대한 에너지 함수들의 집합
- A : Action Semantic에 대한 에너지 함수들의 집합

명령어는 하드웨어의 제어와 직접적인 관련이 있다는 점에서, 시스템 함수는 OS가 응용 소프트웨어에서 요청한 서비스를 제공하기위해 사용된다는 점에서[11], 명령어와 시스템함수의 전력소모량에 대한 데이터는 모델기반의 전력분석과정에서 중요한 기반정보가 된다. 본 연구에서는 에너지 라이브러리를 구축하기 위해, 명령어와 시스템 함수에 대한 VI와 VSF를 정의한 후, 그것들이 소모하는 전력량을 측정하였다. 시스템 소프트웨어와 하드웨어 플랫폼은 종류가 여러 가지가 있기 때문에, Jwahar R. Bammi[12]에 의해 소개된 VI, 그리고 그와 유사한 형태의 VSF를 정의하여 라이브러리 구축에 활용하였다.

4. 에너지 라이브러리 생성

4.1 EMSIM2 시뮬레이션 환경

본 연구에서는 에너지 라이브러리 구축을 위해 T. K. Tan이 개발한 시뮬레이터 EMSIM2를 사용하였으며[5], EMSIM2의 시뮬레이션 환경은 StrongARM 코어와 EBSA-110 플랫폼을 기준으로 구성되어있으며, OS는 임베디드 리눅스가 사용되었다.

4.2 전력 분석을 위한 EBU 에너지 모델 요소

전력 분석을 위한 EBU 에너지 모델 요소는 “VI value”, “VSF Energy Function”, “EBU Energy Model”들의 집합으로 구성되지만, VI는 VSF와 동일한 방법으로 측정이 되며 EBU Energy Model은 위의 두 가지를 합성하여 만들어지므로, 본 논문에서는 VSF Energy Function을 측정한 결과만을 설명하도록 한다.

VSF는 다음의 네 가지 서브시스템으로 분류하여 정의하였다.

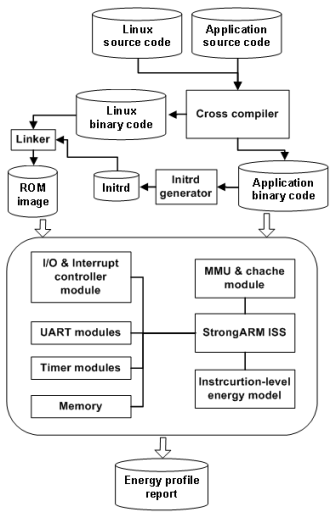
- Process Manager : fork()등의 프로세스를 관리하는 함수
- File System : file open()등의 파일시스템과의 인터페이스를 수행하는 시스템 함수
- Memory Manager : shmget()등의 메모리 관리 함수
- Inter Process Communication : msgsnd()등의 프로세스간의 통신을 위한 함수

시뮬레이션을 통해서 얻은 VSF의 전력 소모량에 대한 데이터는 회귀분석을 통해 매크로 모델을 생성한다.

4.3 VSF 전력소모 특성 측정 방법

EMSIM2의 개략적인 시뮬레이션 환경의 구성과 시뮬레이터를 사용하기 위한 과정은 (그림 3)과 같다.

EMSIM2를 통해 시뮬레이션을 수행할 경우, 사용자가 시뮬레이션을 수행하고자 하는 프로그램 코드를 작성하여, EMSIM2를 실행하면 미리 정의된 절차에 의해 어플리케이션 소스코드와 리눅스 소스코드를 크로스 컴파일하게 된다. 각각의 바이너리 코드는 사용자 메모리영역과, 커널영역에 할당될 수 있도록 RAMDISK 이미지 생성을 마친 후 ROM 이미지를 생성한다. 생성된 ROM 이미지는 시뮬레이션 수행에 사용된다. EMSIM2는 시뮬레이션이 끝난 후에 시뮬레이션 수행결과를 사용자에게 보이는데, 이때 전달되는 수행결과 중에서 사용자가 원하는 함수나 명령어에 대한 전력소모량 데이터만 선택적으로 사용한다.



(그림 3) EMSIM2의 전력분석 프레임워크[5]

5. VSF 전력소모 특성 측정 결과

에너지라이브러리를 구축하기 위해 정의한 VSF들에 대해서 시뮬레이션을 수행한 결과, <표 1> ~ <표 4>와

같은 에너지 매크로 모델을 얻을 수 있었다.

<표 1 > Process Manager의 에너지 매크로 모델

System Functions	Parameters	Macro-models(nJ)
fork()	-	E = 132202.6
waitpid()	-	E = 25077.3
wait()	-	E = 24986.6
signal()	-	E = 9460.6

<표 2 > File System의 에너지 매크로 모델

System Functions	Parameters	Macro-models(nJ)
File open()	-	E = 17886.6
File close()	-	E = 8264.2
File read()	x bytes	E = 5.1 x+49308.5
File write()	x bytes	E = 5.6 x+32022.3

<표 3 > Memory Manager의 에너지 매크로 모델

System Functions	Parameters	Macro-models(nJ)
shmget()	-	E = 126358.83
shmat()	-	E = 11599.9
shmdt()	-	E = 14603.2
shmctl()	IPC_SET	E = 3211.5
	IPC_STAT	E = 4010.1
	IPC_RMID	E = 17760.3

<표 4 > IPC의 에너지 매크로 모델

System Functions	Parameters	Macro-models(nJ)
msgsnd()	x bytes	E = 4.0 x+4554.0
msgrcv()	x bytes	E = 4.4 x+4818.0
msgget()	-	E = 4351.019417
msgctl()	IPC_SET	E = 3334.7
	IPC_STAT	E = 3835.7
	IPC_RMID	E = 3603.6
semget()	-	E = 4557.2
semctl()	GETPID	E = 2573.4
	GETALL	E = 2938.0
	GETZCNT	E = 2795.5
	IPC_RMID	E = 3924.2
	IPC_SET	E = 2784.4
	IPC_STAT	E = 4021.4
	SET_ALL	E = 2976.4
	SETVAL	E = 2660.8
semop()	lock	E = 7322.8
	unlock	E = 3688.1
pipe()	-	E = 31947.2
pipe read()	x bytes	E = 4.7 x+18439.8
pipe write()	x bytes	E = 2.8 x+14720.0

정의된 VSF들 중에는 처리 데이터가 있는 경우와, 여러 가지의 Operation mode를 갖는 경우가 있다. 이러한 경우들에 있어서, 처리데이터의 크기나 Operation mode에 따라 전력소모량이 달라진다. 이 두 가지 사항에 대해서는 각각이 영향을 미치는 요인을 표의 'Parameters'란에 기술하였다.

6. 결론 및 향후연구

본 연구에서는 모델기반의 소프트웨어 소모전력 분석 과정 중에서 모델요소의 전력소모량을 분석하기 위한 에너지 라이브러리를 구축하였다. 이를 위해 명령어와 시스템 함수를 추상화한 VI와 VSF를 정의하고, 그것들에 대한 전력소모량을 EMSIM2를 통해 시뮬레이션 하여 측정하였다. 그 결과, VI는 스칼라 값으로, VSF는 회귀분석을 통한 매크로 모델의 형태로 얻을 수 있었다. 구축한 에너지 라이브러리는 임베디드 소프트웨어의 설계모델 기반 시뮬레이션에서 전력 분석을 위한 환경으로 사용할 것이며, 그 활용도를 높이기 위하여 설계모델의 추상화 수준에 따른 전력정보 지원방법이 요구된다.

참고문헌

[1] J. Ou, et al, "Rapid energy estimation for hardware-software codesign using FPGAs", EURASIP Journal on Embedded Systems, 2006

[2] T. K. Tan, et al, "Software Architectural Transformations: A New Approach to Low Energy Embedded Software", in Proc. of DATE, 2003

[3] G. Qu, et al, "Code Coverage-based power estimation techniques for microprocessors", Journal of Circuits, Systems, and Computers, Vol. 11, No. 5, pp. 557-574, 2002.

[4] E. Senn, et al, "Power Consumption Estimation of a C Program for Data-Intensive Applications", in Proc. of the ICD, 2002

[5] T. K. Tan, A. Raghunathan, and N. K. Jha, "EMSIM: An Energy Simulation Framework for an Embedded Operating System", in Proc. ISCAS 2002, May 2002.

[6] A. Muttreja, et al, "Hybrid Simulation for Energy Estimation of Embedded Software", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2007

[7] E. Senn, et al, "Software Explorer : Estimating and Optimizing the Power and Energy Consumption of a C Program for DSP Applications", EURASIP Journal on Applied Signal Processing 2005:16, 2641-2654, 2005

[8] T. K. Tan, A. Raghunathan, and N. K. Jha, "Energy Macromodeling of Embedded Operating Systems", ACM Transactions on Embedded Computing Systems, Vol. 4, Issue 1, pp. 231-254, 2005

[9] A. Muttreja, A. Rahunathan, et al, "Automated energy / performance macromodeling of embedded software", in Proc. on DAC, 2004

[10] T. K. Tan, A. Raghunathan, and N. K. Jha, "Embedded Operating System Energy Analysis and Macro-Modeling", in Proc. of ICCD, 2002

[11] R. P. Dick, et al, "Power Analysis of Embedded Operating Systems", in Proc. of the DAC, 2000

[12] J. R. Bammi, Edwin Harcourt, et al, "Software Performance Estimation Strategies in a System-Level Design Tool", in Proc. of the CODES'00, 2000