임베디드 소프트웨어 전력분석기법의 조사분석을 통한 특성 모델 도출 및 활용

(Extracting and Applying a Characteristic Model with Survey of Power Analysis Techniques for Embedded Software)

김 종 필 † 김 두 환 † 홍 장 의 ††
(Jong-Phil Kim) (Doo-Hwan Kim) (Jang-Eui Hong)

요 약 그린 IT의 중요성이 부각되면서 저전력의 소프트웨어 개발에 대한 요구사항이 증가하고 있다. 본 논문에서는 임베디드 소프트웨어 개발과정에서 사용되는 기존의 전력분석기법들을 살펴보고, 이들 기법이 제공하는 분석 접근방법의 특성을 추출하였다. 이들 특성을 분류하고 체계화하여 전력분석기법에 대한 특성 모델(characteristic model)을 제안하였다. 제안한 특성 모델을 임베디드 소프트웨어 개발단계와 매핑하여 단계별 전력분석의 주안점이 무엇인가를 살펴보고, 이들이 갖는 의미를 스파이더 다이어그램을 이용하여 해석하였다. 본 연구는 임베디드 소프트웨어의 전력분석 기법에 대한 이해를 높이고, 분석방법의 선택에 대한 가이드라인을 제공할 뿐만 아니라 향후 전력분석을 위한 적용 기술의 변화를 예측할 수 있도록 하는 장점을 제공할 것으로 보인다.

키워드 : 전력분석기법, 임베디드소프트웨어, 특성 모델

Abstract Increasing the importance of Green IT brings low-power consumption requirements for embedded software into relief. This paper focus on the power analysis techniques of embedded software along with the trend. We survey the existing research on the power analysis techniques performed during the last decade, and find out some features or characteristics from the analysis approaches of those techniques. Also we summarize those characteristics into a systematic model, and then apply the model to embedded software development process using spider diagram. Our suggestion gives such benefits as improving the understanding of power analysis techniques, guiding the choice of an appropriate technique to their power analysis, and forecasting the direction of technology changes in embedded software power analysis.

Key words : Power Analysis Technique, Embedded Software, Characteristic Model

1. 서 론

전 세계적으로 환경문제가 대두되면서, 에너지 절약 및 대체 에너지 개발에 대한 관심이 높아지고 있다. 2008년 3월 독일의 하노버에서 개최한 CeBiT 기술 전람회에서 "그린 IT"가 화두가 되면서, 특히 에너지에 대한 관심이 더욱 높아지고 있다. 그린 IT라 함은 환경에 미치는 마이너스 효과가 적은 IT 기술이나 IT 산업을 총칭한다. 그린 IT의 포함 범위에는 논의의 여지가 있지만 일반적으로 IT 산업의 공정이나 제품의 친환경화를 통해 환경에 미치는 유해함을 최소화하는 기술이나 제도, 시스템을 의미한다[1]. 이러한 범주내에서 중요한 부분을 차지하는 것이 IT를 활용한 에너지 효율성의 증대이며, 이를 위한 방안으로 전력 관리 기술 및 저전력 시스템의 개발 기술이 관심의 대상이 되고 있다.

· 이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-313-D00936)

† 학생회원 : 충북대학교 컴퓨터과학과
kimjp@selab.cbnu.ac.kr
dhkim@selab.cbnu.ac.kr
†† 종신회원 : 충북대학교 전기전자컴퓨터공학부 교수
jehong@chungbuk.ac.kr

논문접수 : 2008년 11월 7일
심사완료 : 2009년 3월 31일

Copyright©2009 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

본 연구에서는 저전력 시스템 개발과 관련하여 임베디드 소프트웨어의 소모전력 분석기법에 주안점을 두었다. 임베디드 소프트웨어의 개발 과정에서는 매우 다양한 접근 방법에 의한 전력 분석이 가능하다. 단순히 소프트웨어의 명령어 수준에서 소모 전력을 분석하기도 하고, 프로그래밍 언어 수준에서 전력을 분석하기도 한다. 또한 소모 전력의 산출 방법을 시뮬레이션 기반으로 수행하기도 하고, 실측을 통해 산출하기도 한다.

본 연구에서는 이와 같이 매우 다양한 접근 방법을 기반으로 제시된 기존의 전력분석 기법들을 살펴보고, 이들 분석 기법의 특성을 도출하여 분류하고 모형화 하였다. 이러한 모형화는 크게 7가지의 요소를 갖는 특성 모델로(characteristic model)로 표현하였으며, 각 요소들에 대하여 세부 분류 특성을 정의하였다. 제시하는 특성 모델은 임베디드 소프트웨어의 개발 단계에서 전력분석을 위한 접근 방법의 선택에 대한 가이드라인을 제공할 뿐만 아니라 향후 전력 분석을 위한 적용 기술의 변화를 예측하는데 도움을 줄 수 있다. 또한 추후 새로운 전력 분석기법을 개발하기 위하여 고려해야 하는 특성이 무엇인지를 식별할 수 있는 기초 정보를 제공할 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 임베디드 소프트웨어의 전력분석에 대한 연구 동향을 살펴보고, 3장에서는 전력 분석에 대한 기존의 연구들을 추상화수준에 따라 분류하여 분석하였다. 4장에서는 전력분석기법의 특성을 분석하여 모형화한 특성 모델에 대하여 정의한다. 5장에서는 특성 모델에서 제시하고 있는 구성요소(분류기준)별로 기존 연구들을 분류하고, 이들의 연구 변화 과정을 살펴보았다. 6장에서는 특성 모델이 임베디드 소프트웨어의 개발 절차와 어떠한 관계를 갖는지 설명하고, 마지막으로 7장에서 결론을 맺는다.

2. 전력분석기법의 동향

임베디드 소프트웨어의 전력분석은 1994년 프린스턴 대학의 Tiwari에 의해 연구[2]가 시작되었다. 그 이전의 연구들에서는 저전력을 소모하는 마이크로프로세서 하드웨어 설계나, 소프트웨어 컴파일을 통한 시스템 성능의 최적화 등에 대한 연구가 진행되어 왔다.

임베디드 소프트웨어의 개발에 있어서 성능의 최적화는 전력 소모량을 줄이는 효과를 가져왔고, 이를 인식한 후에는 소모 전력 분석에 대한 연구가 많이 진행되었다[3]. 전력분석에 대한 초기의 연구들에서는 DSP 기반의 프로세싱에서 데이터 연산의 효율성을 높이거나 메모리 접근 횟수를 감소시킴으로써 소모 전력을 줄이기 위한 방향으로 연구[4,5]가 집중되었다. 즉 대부분의 경우 펌웨어나 운영체제 측면에서의 소모 전력에 대한 연구가 진행되어 왔다.

그러나 2000년도 이후의 연구들에서는 임베디드 응용 어플리케이션에서의 소모 전력 분석에 대한 연구가 본격적으로 시작되었다. 초기에는 명령어 수준(low-level instruction)에서의 소모전력 분석에 대한 연구[6-10]가 진행되었으며, 점진적으로 소스 코드 수준에서의 소모전력 분석 기법[11-14]이 제안되었다. 국내의 소모전력 분석에 대한 연구는 2002년도 서울대학교의 연구[15]를 시작으로 임베디드 소프트웨어의 소모 전력을 분석하기 위한 다양한 방법들에 대한 연구가 진행되고 있다.

이러한 연구 중에서 많은 부분들이 프린스턴 대학에 의해 수행되었는데, 그 중에서 소프트웨어 중심의 전력 분석에 대한 연구는 T.K. Tan에 의해 제시된 소프트웨어 매크로 모델 기반의 소모전력 분석기법[13]이다. 이 연구에서는 일반적으로 잘 알려진 알고리즘, 즉 정렬(sorting), CheckSum, EdgeDetection 등의 알고리즘 단위로 매크로 모델을 정의하고, 이를 통해 소모 전력을 계산하도록 제안되었다. 예를 들면, 삽입정렬이 경우 $c1+c2N+c3N^2$이라는 매크로 모델을 정의하였는데, 여기서 N은 입력 데이터의 크기, c1, c2, c3은 삽입정렬 알고리즘의 구현 및 실행을 통하여 반복적으로 측정된 소모전력의 계수 값을 나타낸다. 따라서 입력 데이터의 크기만 주어지면 라이브러리에 저장된 c1, c2, c3 값을 이용하여 소모 전력을 바로 계산할 수 있다. T. K. Tan은 제시한 매크로 모델 기반의 전력 측정기법[13]을 실제 구현된 코드 기반의 측정과 비교하여 10% 미만의 오차율을 보인다고 제시하였다.

T.K. Tan은 그의 논문[16]에서 그림 1과 같이 임베디드 시스템 개발에서 소모 전력을 감소시킬 수 있는 영역이 매우 다양함을 설명하였다.

그림 1에서 보는 것과 같이 하드웨어의 경우, 트랜지스터 레벨에서 전력 감소를 시도하는 것보다 상위의 하드웨어 동작 수준에서 전력 감소를 시도하는 것이 10배 이상의 절감 효과가 있으며, 분석을 위해 요구되는 시간도 매우 빠르다고 설명하고 있다. 소프트웨어의 경우에 있어서도 명령어 수준에서 소모 전력을 분석하는 방법

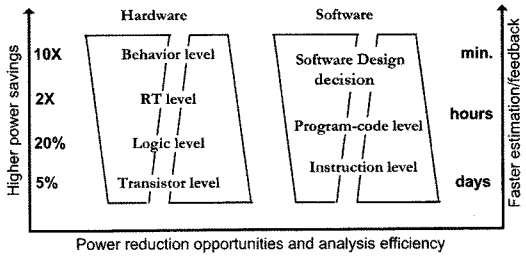


그림 1 임베디드 시스템의 전력 분석 효율성에 대한 범주[16]

은 전력 소모의 절감이 명령어에 의존적인 5% 정도에 불과하고 분석을 위해서도 하루 이상의 시간이 소요되는 경우가 종종 있지만, 설계 수준에서는 상대적으로 10배 이상의 소모 전력 절감이 가능해지며, 또한 매우 빠른 분석이 이루어질 수 있다고 제시하였다[16].

이와 같은 전체적인 연구의 동향으로 볼 때, 임베디드 소프트웨어에 대한 소모 전력의 분석은 명령어 수준에서의 분석 기법[2,5,7]부터 설계 모델을 근간으로 하는 전력 분석 기법[4,16,17]까지 매우 다양하다고 볼 수 있으나, 실제로 소프트웨어의 설계 모델을 근간으로 하는 전력 분석기법에 대한 연구는 이제 초보적인 연구[16,18]가 진행되고 있는 실정이다.

3. 추상화 수준에 따른 기존 연구 분석

임베디드 소프트웨어에 대한 소모 전력 분석을 수행하는 기존의 연구들을 분류할 때, 일반적으로 추상화 수준에 따라 분류한다. 다시 말해서 전력 분석의 기준이 되는 단위가 명령어, 함수, 소스 코드, 그리고 컴포넌트 단위인지에 따라 서로 다른 기법 및 분석 접근 방법이 적용될 수 있다.

3.1 명령어 수준 전력분석

명령어수준 전력분석(ILPA: Instruction-level Power Analysis)은 명령어들이 실행 시 소모되는 전류(또는 전력)를 직접 측정하거나, 고안된 측정도구[6]를 이용 또는, 하위수준(예, Gate-Level)에서의 시뮬레이션을 통해서 얻은 데이터를 기반으로 전력 모델을 수립하며, 이를 기반으로 소프트웨어의 전체 소모 전력량을 추정한다.

$$E_p = \sum_i (B_i N_i) + \sum_{i,j} (O_{i,j} N_{i,j}) + \sum_k E_k \quad (1)$$

식 (1)은 명령어 수준의 전력 모델의 한가지 예로써, 응용 프로그램의 전체 소모 에너지(E_p)을 추정하기 위해 사용된다[19]. B_i는 명령어 i가 실행 시 소모되는 전류이고, N_i는 명령어 i의 실행 횟수, $O_{i,j}$는 명령어 i, j가 연속실행 시 회로상태 변화에 따른 오버헤드 비용, $N_{i,j}$는 명령어 i, j의 연속실행 횟수, 그리고 E_k는 그 외의 파이프라인 스톨(pipeline stall), 캐쉬미스(cache miss)와 같은 자원 제약사항에 따른 추가 발생 비용을 나타낸다.

표 1은 명령어 수준의 전력분석 기법들로써, 명령어별 전력모델 개발 및 전력모델의 정확성 향상에 공통적인 연구 목적을 두고 있다.

명령어 수준의 전력분석 기법의 단점은 DSP(Digital Signal Processor)와 같이 아키텍처 복잡도가 높을수록 전력 모델 수립 비용이 많이 소요된다는 것이다. 따라서 이러한 문제를 극복하기 위해서는 내부적인 복잡함을 한 단계 추상화 시키고 이를 이용하여 전력 모델을 정의하는 함수 수준의 전력분석기법을 수행하는 것이다.

표 1 ILPA 기반의 전력 분석 기법

저 자	분석 방법	연도
Tiwari [2]	최초로 소프트웨어 관점에서 시도된 측정 기반의 전력 분석기법으로, 명령어를 중심으로 하는 전력 모델을 이용.	1994
Lee [5]	DSP에서의 명령어 실행에 대한 소모전력 측정을 통한 전력분석 모델을 제안.	1997
Klass [4]	DSP에서의 명령어간 처리 과정에서 생기는 에너지 효과에 의한 전력 모델을 수립하고, NOP 모델 제안을 통해 분석 비용 및 정확성을 향상시킴.	1998
Sarta [7]	명령어의 데이터 종속성에 의해 생기는 영향을 효과적으로 극복하기 위한 모델을 제시	1999
Shinha [8]	명령어간의 기능적 유사도에 따라 프로그램을 논리적으로 분류하고, 분류된 그룹별로 전력 모델을 구축	2001

3.2 함수 수준의 전력 분석

함수수준의 전력분석(FLPA, Functional-Level Power Analysis)은 프로세서 내부 구성요소들간의 행위와 전력소모에 영향을 주는 파라미터를 식별한 후, 이를 토대로 전력소모 패턴을 모델링 하고, 이를 전력모델로 이용하는 방법[19-21]이다. 그림 2는 FLPA 전력모델의 정의과정을 보여준다.

그림 2의 단계 1에서는 우선 프로세서가 특정 명령을 수행할 때 공통 행위를 갖는 구성요소들에 대해서 블록 단위(Functional Block)로 묶는다. 그런 다음에 기능 블

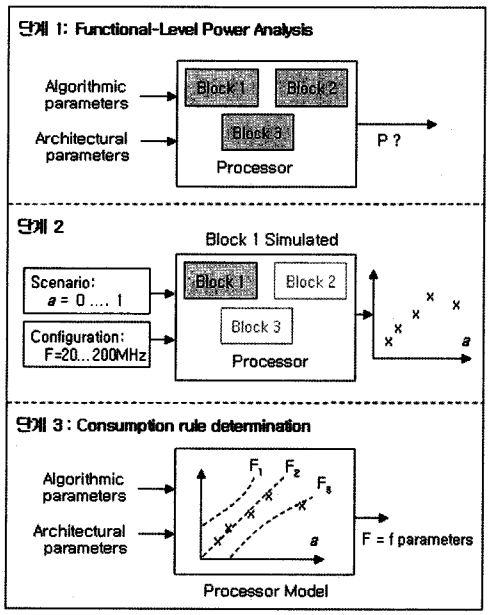


그림 2 FLPA 전력 모델 정의 과정[20]

록들의 행위와 전력 소모에 영향을 주는 요소들을 식별하게 되는데, 그러한 요소들은 두 종류의 파라미터들로 구분된다. 먼저 알고리즘 파라미터는 프로세서의 기능 블록간의 실행에 종속적인 속성들로써, Parallelism rate, Cache miss rate 등을 포함하며, 아키텍처 파라미터는 프로세서의 형상 정보 즉, Clock frequency, Voltage, Data mapping 등의 속성들을 포함한다.

단계 2에서는 정의된 각 기능 블록들에 대하여 실행을 위한 입력을 제공하고, 이에 대한 실행 결과를 수집한다. 수집된 실행 결과는 벤치마크 프로그램을 이용해서 기능 블록의 전력 소모와 파라미터 실제 값들의 변화를 정확히 표현할 수 있도록 회귀 분석 기법을 통해 처리하고(단계 3), 그 결과를 기반으로 전력 소모 규칙을 정의한다[20].

FLPA는 ILPA 분석기법에 비해 전력 모델 수립 및 관리 비용을 현저히 절감할 수 있는 장점을 갖는다. ILPA는 명령어간 상호작용 및 자원제약으로 인한 오버헤드를 모두 고려해야 하는 반면, FLPA 기반의 전력 모델은 그러한 추가 전력소모 요소들이 내부적으로 반영되기 때문에 별도의 추가 분석이 필요하지 않다. FLPA 기법에 대한 연구간의 상호 차이점은 대상 프로세서의 내부 구성요소들과 그것들의 행위 수준을 나타내는 파라미터 종류에 따라 달라질 수 있다[20].

3.3 소스코드 수준 전력분석

명령어 수준에서 소프트웨어의 소모 전력을 분석하기 위해서는 많은 시간이 요구된다. 따라서 C나 포트란과 같은 상위수준 프로그래밍 언어로 표현된 소스 코드의 소모 전력을 목적코드 생성없이 직접 예측할 수 있는 소스코드 수준 전력분석(SLPA, Source code-Level Power Analysis) 기법들[12-14]이 제안되고 있다. 표 2는 상위수준 프로그래밍 언어를 기반으로 하는 소스 코드 수준에서의 전력분석 기법들을 보여주고 있다. 이러한 기법들을 이용한 분석시간 측면의 효율성은 실제 명령어 또는 하위수준에서의 분석시간보다 현저히 개선됨을 보여주고 있다.

표 2 소스코드 기반 전력 분석 기법

저 자	분석 방법	연도
Qu [12]	라이브러리 함수와 명령어들의 소모 전력과 실행시간 정보를 사전에 Power Data Bank에 저장한 후, 전력분석에 이용	2002
Tan [13]	함수에 대한 실행 프로파일을 생성하고, 이를 기준으로 함수에 대한 복잡도 및 입력 데이터의 크기에 따라 소모전력 예측을 위한 에너지 매크로 모델을 정의하여 이용	2003
Senn [14]	C 코드에 대한 FLPA 전력모델을 구축하여, 프로그램 실행 없이 정적 프로파일 정보만을 이용하여 소모 전력을 예측	2005

3.4 설계모델 수준 전력분석

기존 설계모델 수준의 전력분석 기법들[16,17,22,23]이 갖는 공통적인 특징은 소프트웨어 설계모델의 기본 구성 요소(예, 태스크, IP 등)에 대응되는 전력소모 모델을 구축한 후, 이를 이용하여 소프트웨어 전체 전력소모를 추정하는 것이다.

표 3은 기존의 소프트웨어 설계모델 수준의 전력분석 기법들을 보여주고 있다. 이들 방법은 기존의 실행을 기반으로 작성된 소모 전력 프로파일을 구축하고, 이를 사용한 소모 전력 예측 기법을 제안하고 있다. 사용하는 설계모델은 태스크 상호작용 다이어그램(Task Interaction Diagram)이나 모듈의 행위를 표현하는 오토마타 등으로 비주얼한 그래픽 기반의 모델[16,22]이기는 하지만, 모델이 담고 있는 정보는 실행 단위를 기반으로 하는 상세모델이다. 그럼에도 불구하고 이들의 장점은 하위 수준에 대한 전력 분석을 추가적으로 수행하지 않는다는 것이다.

일반적인 UML 또는 SysML과 같은 모델링 언어를 이용하여 명세된 소프트웨어 설계모델 기반의 전력 분석에 대한 연구[18]는 아직도 매우 미흡한 실정이다.

표 3 설계모델 수준의 전력분석 기법

저 자	분석 방법	연도
Tan [16]	SAG(Software Architecture Graph)를 이용한 아키텍처 수준의 전력분석 기법으로서, 그래프 노드간의 통신에 대한 에너지 모델을 미리 구축하고, 이를 이용하여 전체 전력 소모을 추정	2003
Yue [17]	재사용 가능한 컴포넌트/ IP(Intellectual Property) 들에 대해 VI(Virtual Instruction)을 이용하여 인터페이스 중심으로 소모 전력을 모델링 후 소프트웨어 전체 전력소모 추정에 이용	2003
Talarico [23]	소프트웨어 모델과 하드웨어 구성요소들에 대한 모델을 개발하여 통합하고, 이로부터 실행단위 시뮬레이션을 통한 전력분석	2005
Jun [22]	형식언어 오토마타를 확장하여 에너지 인터페이스 오토마타를 정의하고, 이를 이용하여 컴포넌트 단위의 전력소모를 분석할 수 있는 방법 제안	2006

이상에서와 같은 기존의 전력분석 기법에 대한 연구들을 명령어 수준, 함수수준, 소스코드 수준, 그리고 설계모델 수준으로 구분하여 살펴보았다. 이를 기반으로 각 분석기법이 어떠한 영역에 적용되는지를 표현하기 위한 특성 모델을 정의한다.

4. 전력분석기법 특성모델

4.1 특성 모델 정의

명령어 수준에서의 소프트웨어 전력분석부터 매크로 모델 기반의 전력분석까지 매우 다양한 연구들이 진행되었다. 이러한 기존의 연구들을 유형화하여 분류하기 위한 목적으로 다음과 같이 특성 모델을 정의하였다.

정의 1. 소프트웨어 전력분석 기법의 특성 모델(characteristic model), M_E = {A, C, I, G, P, U}의 6가지 속성으로 표현되며, 각 구성요소는 다음과 같다.

(1) A(Abstraction) = $\{A_I \mid A_F \mid A_C \mid A_M\}$: 분석 대상의 추상화 수준에 의한 분류로써, 명령어 수준(A_I), 함수 수준(A_F), 소스 코드 수준(A_C), 설계 모델 수준(A_M)으로 구분.

(2) C(Construction) = $\{C_S \mid C_E \mid C_M\}$: 전력분석 모델을 구축하기 위한 방법에 의한 분류로써, 통계 기법(C_S), 시뮬레이션(C_E), 실측(C_M)으로 구분.

(3) I(information) = $\{I_S \mid I_D \mid I_H\}$: 소모 전력을 분석하는 과정에서 필요한 정보로 구분하며, 정적 속성(I_S), 동적 속성(I_D), 하이브리드 속성(I_H)으로 구분.

(4) G(Generality) = $\{G_H \mid G_D \mid G_I\}$: 전력분석의 기법의 하드웨어 특성 의존도에 따른 구분으로, 매우 의존적(G_H), 단순 의존적(G_D), 아니면 하드웨어 독립적(G_I)으로 구분.

(5) P(Platform) = $\{P_S \mid P_M \mid P_X\}$: 전력분석이 고려하는 하드웨어 플랫폼에 따라 구분되며, 단일프로세서(P_S), 멀티프로세서(P_M) 인가, 아니면 플랫폼 특성과 무관(P_X)으로 구분.

(6) U(Unit of Estimation) = $\{U_I \mid U_F \mid U_O \mid U_C\}$: 전력 분석 결과의 정보 제공 단위에 따른 구분으로, 명령어(U_I), 함수/메소드(U_F), 클래스/오브젝트(U_O), 컴포넌트/IP(U_C) 등으로 구분.

이상에서 정의한 특성 모델에 대한 각 구성요소를 그림으로 표현하면 그림 3과 같이 나타낼 수 있다.

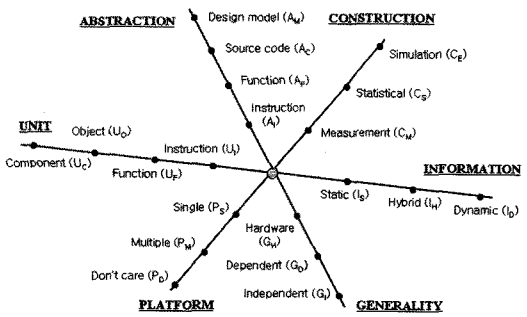


그림 3 전력분석 기법에 대한 특성 모델

4.2 특성 모델 구성 요소

본 절에서는 정의 1에서 제시한 특성모델의 구성요소들에 대하여 설명한다. 참고로 첫 번째 구성요소인 추상화 요소에 대해서는 3장에서 이미 설명되었기 때문에 생략하기로 한다.

(1) 전력모델 구축 방법(CONSTRUCTION)

소프트웨어의 소모전력 분석을 위해 요구되는 전력모델을 구축하기 위하여 소모전력 프로파일을 생성하는 방법에 따라 분석 기법들을 분류할 수 있다.

- 측정(C_M) : 직접 소프트웨어 구성요소(예, 명령어)의 소모 전력(또는 전류)을 측정하고, 이를 기반으로 전력모델을 정의하는 방법
- 통계(C_S) : 소프트웨어 전력소모에 영향을 주는 파라미터들의 변화 패턴과 실제 측정된 전력과 관계를 회귀분석(regression analysis)하여 전력모델을 구축하는 방법
- 시뮬레이션(C_E) : 하위수준(Gate, Register 등) 시뮬레이터를 이용하여 플랫폼 구성요소들의 전력소모 정보를 획득하고, 이를 토대로 소프트웨어의 전력모델을 정의하는 방법

(2) 분석시 요구 정보(INFORMATION)

응용프로그램에 대한 소모 전력 추정 시 요구되는 정보의 성격에 따라 전력분석 기법들을 분류할 수 있다.

- 정적(I_S) : 소모전력 추정 시, 프로그램을 직접 실행하지 않고 코드 또는 설계모델 정보 등과 같은 소프트웨어 표현의 정적 프로파일 정보만을 이용하는 방법
- 동적(I_D) : 소프트웨어의 실행과정 중에 관찰할 수 있는 정보(예, 실행제어 경로)를 기반으로 전체 전력소모를 추정하는 방법
- 하이브리드(I_H) : 소프트웨어의 정적구조에 대한 정보와 실행경로 등의 동적 정보를 모두 사용하여 추정하는 방법

(3) 전력분석 기법의 일반성(GENERALITY)

소프트웨어 소모전력은 플랫폼 종속적이기 때문에, 전력분석 기법은 플랫폼 특성(구조적 복잡도, 아키텍처 속성 등)에 적합해야 한다. 그러나 정확한 소모전력 예측보다는 소프트웨어의 상대적인 소모전력 비교를 위하여 플랫폼과 독립적인 측면에서 소프트웨어 전력분석을 수행할 수 있다.

- 하드웨어 중심적(G_H) : 하드웨어 아키텍처 중심의 동작만을 고려하여 소프트웨어의 소모전력을 패턴화하고, 이들로부터 소모전력을 분석하는 방법
- 하드웨어 의존적(G_D) : 소프트웨어 실행과 관련하여 전력소모가 많은 하드웨어 구성요소들의 특성을 고려하여 소모전력을 분석하는 방법
- 하드웨어 독립적(G_I) : 하드웨어 특성은 고려하지 않거나, 또는 동일한 하드웨어로 가정된 조건에서 소프트웨어의 정보만을 이용하여 분석하는 방법

(4) 하드웨어 플랫폼 복잡도(PLATFORM)

플랫폼 아키텍처의 복잡도에 따라 소프트웨어의 전력 분석 방법은 달라질 수 있다. 전력분석에서 기반이 되는 플랫폼의 복잡도에 따라 다음과 같이 분류할 수 있다.

- 단일 프로세서 아키텍처(P_S) : 특정 시점에 하나의 단일 스레드만이 실행되는 플랫폼
- 다중 프로세서 아키텍처(P_M) : 다수의 스레드가 병렬 수행하는 플랫폼
- 고려하지 않음(P_D) : 플랫폼 아키텍처를 고려하지 않는 경우

(5) 전력 분석의 예측 단위(UNIT)

소프트웨어 전력분석의 기본 추정 단위에 따라 전력 분석 기법들을 분류할 수이다. 속성으로는 다음과 같은 단위들을 갖는다.

- 명령어(U_I) : 전력분석의 결과를 명령어 또는 기본 블록(연속 실행되는 일련의 명령어들) 단위로 산출하여 제공하는 방법
- 함수(U_F) : 분석 결과를 정의된 또는 내장된 함수 단위로 산출하여 제공하는 방법
- 객체(U_O) : 설계 모델을 표현하는 구성요소에서 모델을 구성하는 기본 단위로 전력 분석 결과를 제공하는 방법
- 컴포넌트(U_C) : 정렬(Sorting), 인텍싱(Indexing) 등과 같이 일반화된 컴포넌트 단위로 전력 분석 결과를 제공하는 방법

5. 특성 모델에 근거한 기존 연구 분석

임베디드 소프트웨어의 전력 분석을 위한 기존의 다양한 연구들에 대하여 정의 1에 제시한 전력 분석의 특성 모델을 중심으로 분석하면 표 4와 같다.

표 4에 나타난 기존의 분석들은 1994년에 수행한 Tiwari의 연구[2]로부터 2001년도에 제시된 JouleTrack[8], 2003년도에 연구 제시된 SoftExplorer[20], Tan[13], OOEM[17], 2005년도의 SoftExplorer v.2[14], 그리고 2007년도의 Muttreja[11]의 연구들이다. 이와 같은 연구의 년차적인 변화 과정에서 고찰할 수 있는 사항은 다음과 같이 정리할 수 있다.

(1) 명령어 수준에서의 소모 전력분석 보다는 추상화 수준이 높은 소스코드기반의 전력분석 기법으로 발전하고 있다. 이러한 추세는 분석 결과에 대한 사용자의 이해도를 높이고, 소스코드와 분석 결과의 상관성을 쉽게 알 수 있기 때문으로 해석된다.

(2) 임베디드 시스템 개발에서 널리 사용되는 하드웨어의 구조는 일반적으로 알려진 상용화된 스펙을 기준으로 한다. 따라서 전력 모델을 구축하기 위한 방법으로 소모전력을 직접 측정하거나 시뮬레이션을 수행하기 보다는 수십 또는 수백 번의 실측 데이터를 통계 분석하여 전력 모델을 생성하고, 이를 기반으로 소모 전력량을 예측하고 있다. 이는 반복해서 수행하는 전력 분석에서 하위 수준의 소모 전력을 재 측정하기 위한 노력을 줄이기 위함이다.

(3) 소프트웨어의 소모 전력을 예측하기 위해서 동적 정보를 기반으로 하는 명령어 수준의 시뮬레이션 방법은 소프트웨어의 규모에 따라 소모 전력을 분석하기 위한 매우 많은 시간을 요구하기 때문에 소프트웨어가 갖는 정적 정보를 중심으로 전력 분석을 수행하는 추세이다.

(4) 소프트웨어에 대한 전력 분석의 결과는 명령어 수준보다는 함수나 컴포넌트 단위로 정보를 제공하는 추세이다. 이러한 이유는 전력 모델의 구축이 용이하고, 분석을 위한 시간을 대폭 줄일 수 있는 효과가 있을 뿐만 아니라, 전력 분석의 결과를 재사용할 수 있는 기회를 제공하기 때문이다. 물론 명령어 수준으로 분석된 결과를 함수나 컴포넌트 단위로 종합하

표 4 특성모델에 의거한 기존 연구의 분석

Analysis Dimensions	Tiwari [2]	JouleTrack [8]	SoftExplorer [20]	Tan [13]	OOEM [17]	SoftExplorer v.2 [14]	Muttreja [11]
Abstraction (A)	A_I	A_I	A_I	A_C	A_M	A_C	A_C
Construction (C)	C_M	C_E	C_S	C_S	C_S	C_S	C_S
Information (I)	I_D	I_D	I_D	I_S	I_S	I_S	I_S
Generality (G)	G_H	G_D	G_D	G_I	G_I	G_D	G_I(or G_D)
Platform (P)	P_S	P_S	P_M	P_M	P_X	P_M	P_M
Unit (U)	U_I	U_I	U_I	U_F	U_C	U_F	U_I + U_F

여 그 결과를 제시하기도 한다.

위와 같은 고찰을 통하여 기존의 연구들에 대한 분포 그래프를 그려보면 그림 4와 같다.

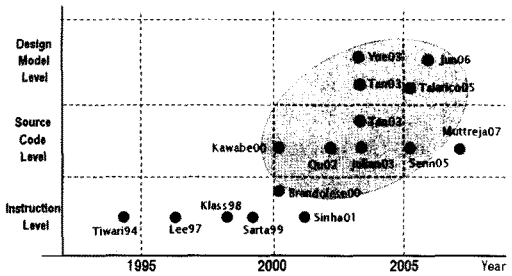


그림 4 추상화 수준에 따른 기존 연구의 분포도

그림 4에서 보여주는 기존 연구의 분포들도 2000년 전에는 명령어 기반의 연구에서 최근에는 소스코드 기반의 연구들로 그 흐름이 바뀌었으며, 모델 기반의 연구도 2003년 이후, Tan[16], Yue[17] 그리고 Jun[22] 등에 의해 수행되고 있다.

6. 소프트웨어 개발 절차와 특성 모델

본 장에서는 최근 변화하고 있는 임베디드 소프트웨어 개발 방법과 이러한 과정에서 요구될 수 있는 전력 분석 기술들에 대해 논의해 본다.

임베디드 소프트웨어는 일반 범용 소프트웨어 개발과 달리 응용분야에 따라 매우 다양한 설계 및 구현상의 제약사항들을 포함하고 있다. 임베디드 소프트웨어의 대표적인 개발 방법은 하드웨어 소프트웨어 동시설계(Hardware Software Codesign) 기법[24]이며, 이의 절차는 그림 5와 같다.

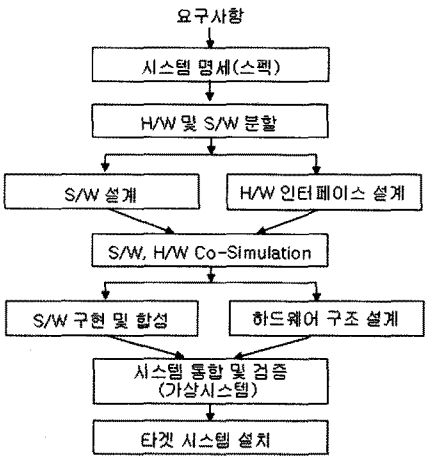


그림 5 H/W-S/W Co-design 개발절차[24]

그림 5와 같이 하드웨어 소프트웨어 동시설계 기법은 시스템 명세를 소프트웨어 부분과 하드웨어 부분으로 분할한 후, 각각에 대하여 설계한다. 소프트웨어 설계모델과 하드웨어 설계모델은 동시시뮬레이션(Co-simulation)을 통해 검증되고, 구현 과정을 거쳐 통합된다. 이러한 과정에서 전력분석은 시스템 통합 및 검증의 일부로써 수행되며, 가상 프로토타입 상에서 이루어진다.

임베디드 소프트웨어 개발의 또 다른 방법은 폭포수(Waterfall) 프로세스를 기반으로 하는 모델 기반의 개발 방법이다. 이 방법은 그림 6과 같이 하드웨어와 소프트웨어 요구사항을 분리하여 식별하고, 하드웨어 설계와 소프트웨어 설계를 서로 다른 개발 경로로 진행하는 것이다.

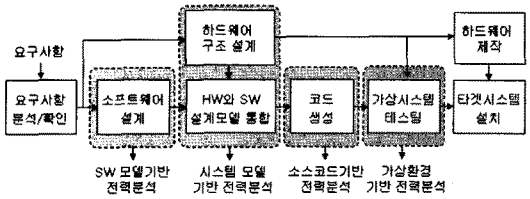


그림 6 모델 기반의 소프트웨어 개발 방법

이러한 개발 경로는 하드웨어 부분과 소프트웨어 부분이 함께 검증되어야 하는 시점에서 상호 통합된다. 그림 6에서는 특별히 이러한 모델 기반의 임베디드 소프트웨어 개발 단계에서 전력 분석을 수행하기 위한 기준점을 (a) 소프트웨어 모델기반 전력분석, (b) 시스템 모델기반 전력분석, (c) 소스코드 기반 전력분석, 그리고 (d) 가상시스템기반 전력분석의 4가지 유형으로 구분하였다.

이러한 각각의 기준점에서 소프트웨어의 소모 전력을 분석하기 위한 접근방법이 어떠한 특성을 갖는지 표현하기 위하여 4장에서 제시한 특성 모델을 이용하여 스파이더 다이어그램으로 표현하였다. 그림 7은 모델 기반의 임베디드 소프트웨어 개발에서 전력 분석 기준점에 대한 특성을 스파이더 다이어그램으로 표현한 것이다.

그림 7에서 보여주는 스파이더 다이어그램의 의미는 그림 1에서 나타내는 바와 같이 분석 시간 및 소모전력 절감 효과의 크기를 상대적으로 표현하는 것이며, 어떤 요소에 주안점을 두고 소모전력을 분석할 것인가는 현재 개발 과정에서 확보한 소프트웨어 정보가 무엇인가에 따라 결정될 수 있음을 보여준다. 또한 분석의 결과로 제시되는 정보가 다이어그램의 면적과 비례하여 미립자(fine grain) 수준으로 분석 결과를 얻을 수 있는지, 아니면 그 반대인지를 직관적으로 보여준다고 할 수 있다.

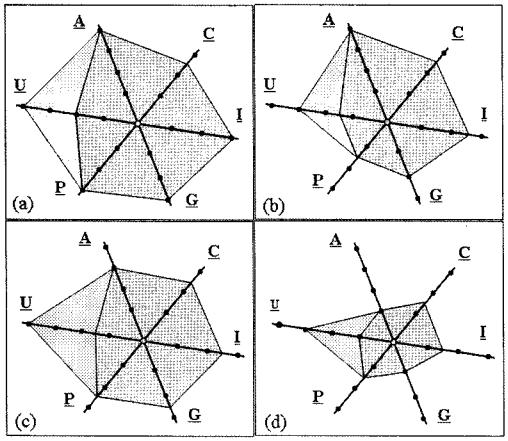


그림 7 개발 단계별 전력분석 특성에 대한 스파이더 다이어그램

(1) 소프트웨어 설계 단계

임베디드 소프트웨어의 설계단계에서는 설계모델을 근간으로 하는 전력분석이 가능하다. 이 단계에서는 기존에 프로그램/코드 중심의 소모 전력을 시뮬레이션이나 통계적으로 분석한 기반 정보를 활용한다. 설계모델이 UML의 시퀀스 다이어그램과 같은 행위를 표현하고 있기 때문에 소프트웨어의 설계 모델로부터 실행경로 등에 대한 동적인 정보가 확보 가능하며, 하드웨어 플랫폼에 대해서는 독립적이라고 볼 수 있다. 그림 7의 (a)에서 보는 바와 같이 특성 모형이 평가되며, 다른 단계에서의 전력분석보다 소요 시간이나 전력절감 이득 측면에서 가장 크다고 볼 수 있다. 다이어그램의 왼쪽에 보이는 점선 영역은 분석 결과의 표현 방법이 유동적일 수 있음을 나타낸다.

(2) 설계 모델 통합 단계

하드웨어 설계 모델과 소프트웨어 설계 모델이 통합되어 시스템 모델 기반으로 전력을 분석하는 경우는 하드웨어 아키텍처 특성을 포함하는 소프트웨어 행위를 중심으로 전력을 분석하는 경우이다. 이러한 경우, 하드웨어에 대한 플랫폼은 단일 또는 다중 프로세서 환경으로 고려할 수 있으며, 분석 결과에 대한 출력은 함수 및 객체단위로 이루어질 수 있다. 그러나 실제로 수행하는 실행기반이 아니기 때문에 객체의 인스턴스를 모두 고려할 수 있는 컴포넌트 단위의 결과를 얻기는 어렵다고 할 수 있다.

(3) 코드 생성 단계

소스 코드를 입력받아 전력을 분석하는 단계에서는 그림 7의 (c)와 같이 설계 모델기반의 분석과 유사하게 하드웨어에 대한 정보 보다는 소프트웨어 자체에 대한 정보를 더 많이 활용하여 이루어진다. 따라서 하드웨어 특성 및 하드웨어 아키텍처에 대한 의존도가 비교적 적은 편이다. 최근 소스코드를 기반으로 하는 전력분석이 가장 일반적으로 이루어지고 있는데, 이는 분석 속도나 소모전력 절감 측면에서 명령어 기반 분석보다 다소 높은 효과를 볼 수 있으며, 또한 구체적인 코드 기반으로 이루어져 결과의 정확도도 상대적으로 높다고 할 수 있기 때문이다.

(4) 가상시스템 기반의 검증 단계

이 단계에서는 하드웨어 아키텍처 및 운영체제 등의 기반 구조상에서 소스코드를 시뮬레이션하여 시스템의 행위를 확인하는 단계이며, 이 과정에서 소프트웨어의 소모 전력을 분석할 수 있다. 이 단계에서는 소스 코드 단위의 시스템 시뮬레이션이 이루어지기 때문에 상대적으로 전력 분석을 위한 시간의 많이 요구되나, 하드웨어 아키텍처의 세부 메커니즘 - 예를 들면, 파이프라인 처리 및 메모리 접근 등 - 을 고려하여 분석이 이루어지기 때문에 비교적 정확한 전력 분석이 가능하다고 볼 수 있다.

이상에서와 같이 모델 기반의 임베디드 소프트웨어 개발 절차상에서 소모전력 분석을 위한 기준점을 정의하고, 이들 시점에서 전력 분석이 어떠한 특징을 갖는지 살펴보았다. 최근에는 소스코드 기반의 전력분석과 함께 설계 모델 기반의 전력분석이 연구되고 있는데, 이는 코드 기반의 분석결과와 설계 모델 기반의 분석 결과가 많은 차이를 보이지 않기 때문이다. 대표적으로 T. K. Tan의 연구에 의하면 소프트웨어 매크로 모델을 근간으로 하는 전력 분석의 결과가 명령어 수준의 분석과 비교하여 SPARClite 환경에서 0.3%에서 5% 수준의 오차율을 보인 반면 분석시간 측면에서는 대략적으로 3만배 이상의 시간 단축이 이루어졌음을 제시한바 있다[14].

7. 결 론

임베디드 시스템의 비기능적 요구사항의 하나로 저전력 소모가 중요한 이슈로 부각되고 있다. 따라서, 임베디드 소프트웨어에 대한 소모 전력을 분석하기 위한 다양한 연구들이 진행되어 왔다. 본 연구에서는 1994년 Tiwari[2]에 의해 연구된 명령어 수준의 전력분석부터, 2006년 Jun[22]에 의해 연구된 컴포넌트 기반의 임베디드 소프트웨어 소모전력 분석, 그리고 2007년 Mutreja[11]에 의한 소스코드 기반의 전력분석 기법까지 살펴보았다. 이러한 연구들이 서로 다른 특성과 제한조건하에서 다양한 전력분석 기법을 제안하고 있음을 인식하고, 이들을 체계적으로 분류하기 위하여 분석 기법의 특성을 모형화하고, 임베디드 소프트웨어 개발 과정과 이들 특성과의 관계를 살펴보았다.

본 논문에서 제시하고 있는 임베디드 소프트웨어의

전력분석 기법에 대한 특성 모델은 소프트웨어 개발 단계에서 작성되는 산출물을 기반으로 어느 수준의 전력 분석을 수행할 수 있는가에 대한 가이드라인을 제공하고, 또한 상대적으로 전력분석을 통해 얻는 이득과 분석 결과의 상세함을 직관적으로 보여주기 위해 스파이더 다이어그램을 사용하였다. 그림 7과 같이 소프트웨어 개발의 앞 단계에서 수행하는 전력 분석은 다른 단계에서 보다 상대적으로 많은 이득을 가질 수 있음을 스파이더 다이어그램의 면적 비교를 통해 직관적으로 알 수 있다.

최근의 연구들은 명령어 수준의 전력분석보다는 비교적 시간이 적게 걸리고, 소모 전력의 절감 효과도 상대적으로 높은 소스코드나 모델 수준에서 전력 분석을 시도하고 있으며, 이를 통해 보다 앞선 단계에서 저전력 소모라는 비기능적 요구사항을 충족시키기 위한 방법을 개발하고 있다. 따라서 향후에는 이득이 많은 임베디드 소프트웨어 개발의 앞 단계에서 얼마나 정확한 소모전력 측정결과를 제시할 수 있는 가, 또는 얼마나 유용한 결과를 제시할 수 있는가에 대한 심도있는 연구가 진행되어야 할 것으로 보인다.

참 고 문 헌

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

[2] V. Tiwari, S.Malik, and A.Wolfe, "Power analysis of embedded software: A first step towards software power minimization," IEEE Transactions on VLSI systems, Vol.2, No.4, pp. 437-445, Dec. 1994.

[3] C. H. Gebotys and R. J. Gebotys, "An Empirical Comparison of Algorithmic, Instruction and Architectural Power Prediction Models for High Performance Embedded DSP Processors," in IEEE International Symposium on Low Power Electronics and Design, pp. 121-123, 1998.

[4] 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.

[5] M. T. Lee, et. al., "Power Analysis and Minimization Techniques for Embedded DSP Software," IEEE Trans. On VLSI Sysems, Vol.5, No.1, March1997.

[6] N. Chang, K. H. Kim, and H. G. Lee, "Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI," In: Proc. Of the International Symp. On Low Power Electronics and Design, July, 2000.

[7] D. Sarta, D. Trifone, and G. Ascia, "A Data Dependent Approach to Instruction Level Power Estimation," IEEE Alessandro Volta Memorial Workshop on Low Power Design, pp.182-190, 1999.

[8] 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.

[9] S. Steinke, et. al., "An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimization," In: Proc. Int. Workshop Power and Timing Modeling, Optimization and Simulation(PATMOS'01), pp. 311-321, 2001.

[10] 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.

[11] A. Muttreja, A. Raghunathan, S. Ravi, and N. K. Jha, "Hybrid Simulation for Energy Estimation of Embedded Software," IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, Vol.26, No.10, Oct. 2007.

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

[13] T. K. Tan, et. al, "High-Level Energy Macro-modeling of Embedded Software," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.21, No.9, Sep. 2003.

[14] 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.

[15] D. Shin et. al., "Energy Monitoring Tool for Low-Power Embedded Programs," IEEE Design and Test of Computers, pp. 7-17, July, 2002.

[16] T. K. Tan, A. Raghunathan, and N. K. Jha, "Software Architectural Transformation: A New Approach to Low Energy Embedded Software," In Proceedings of the Design, Automation and Test in Europe Conference and Exhibion, 2003.

[17] X. Yue, Z. Xuehai, L. Xi and G. Yuchang, "OOEM: Object-Oriented Energy Model for Embedded Software Reuse," IEEE International Conference on Information Reuse and Integration, 2003.

[18] 임형인 외, "임베디드 소프트웨어 설계 모델의 추상화 수준에 따른 전력 소모 예측 기법", KCSE 2008, 10권 1호, pp.113-120, 2008년 2월.

[19] C. Brandolese, et. al., "An Instruction-level Functionality-based Energy Estimation Model for 32-bits Microprocessors," DAC 2000, pp.346-351, July 2000.

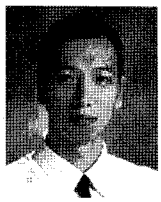
[20] N. Julian, et. al., "Power Consumption Modeling and Characterization of the T1C620," IEEE Micro., Vol.23, No.5, pp. 40-49, 2003.

[21] N. Kawabe, et. al., "Function-level Power Estimation Methodology for Microprocessors," DAC 2000,

pp. 810-813, July 2000.
[22] H. Jun, et. al., "Modelling and Analysis of Power Consumption for Component-Based Embedded Software," EUC Workshop 2006, pp. 795-804, 2006.
[23] C. Talarico, et. al., "A New Framework for Power Estimation of Embedded Systems," IEEE Computer, pp. 71-78, Feb. 2005.
[24] M. Balarin, et. al., Hardware-Software Co-Design of Embedded Systems: The Polis Approach. Kluwer Academic Press, June 1997.

김 종 필

2006년 충북대학교 컴퓨터공학과(학사) 2008년 충북대학교 컴퓨터과학과(공학석사). 2008년~현재 충북대학교 전자계산학과 박사과정. 관심분야는 Aspect 기반 컴퓨팅, 임베디드 소프트웨어, 소프트웨어 품질공학



김 두 환

2007년 충북대학교 컴퓨터공학과(학사) 2007년~현재 충북대학교 컴퓨터과학과 석사과정. 관심분야는 임베디드 소프트웨어 모델링, 모델기반 전력분석, 소프트웨어 품질공학

홍 장 의

1988년 충북대학교 전산학과(이학사). 1990년 중앙대학교 컴퓨터공학과(공학석사) 2001년 한국과학기술원 전산학(공학박사) 2002년 국방과학연구소 선임연구원. 2002년~2004년 (주)솔루션링크 기술연구소장. 2004년~현재 충북대학교 컴퓨터공학 부교수. 관심분야는 객체지향 모델링, 임베디드 소프트웨어, 소프트웨어 품질공학, 소프트웨어 프로세스