http://dx.doi.org/10.3745/KIPSTD.2012.19D.4.281

UML 상태 기계를 이용한 임베디드 소프트웨어의 소모 전력 분석

이 재 욱[†] · 홍 장 의[††]

요 약

스마트 폰과 같은 모바일 기기에서 동작하는 응용 소프트웨어는 제한된 배터리 용량으로 인하여 안정적인 서비스를 지속적으로 제공하는데 어려움을 갖는다. 과거에는 배터리의 수명을 연장 시키거나 소모전력이 적은 하드웨어 장치들을 개발하는 것으로 모바일 기기의 소모전력을 관리하였다. 그러나 시스템에 탑재되는 소프트웨어가 복잡해짐에 따라 소프트웨어에 대한 소모전력 분석 연구도 관심을 갖게 되었다. 이러한 연구들 중에서 모델 기반의 소모전력 분석은 코드가 개발되기 이전에 미리 소모전력을 분석하고, 분석 결과를 기반으로 소프트웨어를 개발한다는 측면에서 소모전력 감소를 위한 하나의 중요한 전략으로 인식되어 오고 있다. 따라서 본 논문에서는 소프트웨어의 행위 모델인 UML의 상태 기계(State Machine) 다이어그램을 이용하는 소모전력 분석 기법을 제안한다. 제안하는 분석기법은 상태기계 다이어그램을 페트리 넷으로 변환하고, 이의 시뮬레이션을 통해 소모전력을 예측하게 된다.

키워드 : 소프트웨어 소모전력, UML 상태기계 다이어그램, 페트리 넷

Analysis of Power Consumption for Embedded Software using UML State Machine Diagram

Jae-Wuk Lee[†] · Jang-Eui Hong[††]

ABSTRACT

A wide variety of smartphone applications is increasing the usage time of smartphone. Due to the increased time, it becomes difficult to providing stable services to users with limited battery capacity. The past works have been performed the power management of mobile device toward long-lasting battery development or low-power electric devices. However as the complexity of software embedded into system are increased, the research interests of the software power analysis is also increased. Among these studies on the software power analysis, model-based analysis technique is one of major interests because it can be able to analyze the power consumption before the development of source codes, then the analysis result can be used in the development of the software system, This paper suggests a model-based power analysis technique using UML state machine diagram. Our proposed technique estimates the power consumption by the simulation of Perti-net which is transformed from the state machine diagram.

Keywords : Software Power Consumption, UML State Machine Diagram, Petri-net

1. 서 론

임베디드 소프트웨어에 대한 수요 증가와 스마트 폰과 같은 모바일 장비의 보급으로 인하여 임베디드 소프트웨어 개발에 대한 중요성이 증가하고 있다. 특히 매우 다양한 응용 프로그램을 개발하고, 다양한 지능형 서비스를 제공해야 하는 스마트 폰의 경우에 있어서는 보다 새로운 아이디어를 제공하는 다양한 소프트웨어들이 출현하고 있다. 그러나 이러한 응용 소프트웨어의 풍부함은 스마트 폰의 사용 시간을 증대시키고 있으며, 이로 인하여 제한된 배터리를 통해 서비스의 안정적인 제공을 보장하는 것이 쉽지 않게 되었다.

기존의 소모전력 분석에 관한 연구들은 하드웨어나 시스템 수준에서 소모전력을 측정하는 연구들이 대부분이었다[1,2,3]. 그러나 시스템에 탑재되는 소프트웨어가 더욱 복잡해지고 소프트웨어에 대한 소모전력의 연구도 관심을 갖게 되었다[4]. 1990년대 중반이후에 시작된 소프트웨어 소모전력에

※ 이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업(No.2012-0006426)과 기초연구사업(2011-0010396) 지원을 받아 수행한 것임.
† 준 회 원 : 충북대학교 컴퓨터과학과 박사과정
†† 종신회원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 2012년 2월 21일
수 정 일 : 1차 2012년 6월 26일
심사완료 : 2012년 7월 10일
* Corresponding Author : Jang-Eui Hong(jehong@chungbuk.ac.kr)

대한 연구는 크게 명령어 수준에서의 소모전력 분석[5,6], 소스 코드 수준에서의 분석[7,8], 그리고 모델 수준에서의 소모전력 분석 기법들[9,10]이 연구되었다. 이러한 과거의 연구들은 소모전력을 측정 또는 분석하기 위한 별도의 완전한 시스템을 개발하거나, 시뮬레이션을 수행하기 위한 인프라가 충분히 구축되어야만 소모전력 분석을 수행할 수 있었다. 이 중에서 모델 기반의 소모전력 분석은 코드가 개발되기 이전에 미리 소모전력을 분석해 보고, 이의 분석 결과를 기반으로 소프트웨어를 개발한다는 측면에서 소모전력 감소를 위한 하나의 중요한 전략으로 인식되어 오고 있다[11].

본 연구에서는 소프트웨어 개발 과정에서 생성되는 설계 모델을 이용한 소모전력 분석 기법을 제안한다. 최근 UML 기반의 행위 모델, 즉 시퀀스(Sequence) 다이어그램 등을 이용한 소모전력을 분석하는 연구[12]가 시도되어 왔다. 임베디드 소프트웨어 또는 모바일 소프트웨어 개발을 위해 UML 기반의 모델링을 수행하는 경우, 시퀀스 다이어그램을 통해 소프트웨어의 동작 시나리오를 표현하기도 하지만, 일반적으로 상태 전이를 중심으로 시스템에 대해서는 상태 기계(State machine) 다이어그램을 이용하여 모델링을 수행한다. 특히 동일한 이벤트라 할지라도 서로 다른 상태에서는 서로 다른 동작을 나타내는 시스템에서는 상태기계 다이어그램을 이용한 소프트웨어 모델링이 많이 수행된다.

본 논문에서는 이와 같이 상태기반의 소프트웨어 개발과정에서 작성되는 상태기계 다이어그램을 이용한 소모전력 분석 기법을 제안하고자 한다. 기존의 연구에서도 상태기계를 이용한 소모전력 분석에 대한 연구[14,15]가 제시되어 왔으나 응용 소프트웨어 보다는 프로세서(Processor) 모델링에 상태기계를 이용하여 소모전력을 분석하거나, 상태 전이에 대한 임의의 가중치를 부여하여 소모전력을 분석하고 있다. 본 연구에서는 기존의 연구와 다른 관점에서 응용 소프트웨어에 대한 상태기계 다이어그램을 사용하고, 상태 전이의 가중치 부여 대신에 Action Description을 사용하여 소모전력 분석에 이용한다. 이러한 연구 결과는 상태 기반의 소프트웨어 개발에 있어서 소모전력에 대한 요구사항을 설계 단계에 미리 고려할 수 있도록 하여 소모전력이 적은 소프트웨어 개발을 가능하게 할 수 있는 이점을 제공할 수 있다.

논문의 2장에서는 기존의 모델 기반 소모전력 분석에 대한 관련 연구들을 살펴보고, 3장에서는 본 논문에서 제안하는 소모전력 분석 절차에 대해 설명한다. 4장에서는 에너지 라이브러리를 참조하여 상태 기계의 전력 소모 특성을 나타내는 방법에 대해 기술하고, 5장에서는 상태 기계 모델을 CPN(Coloured Petri Net)으로 변환하기 위한 규칙을 제시한다. 6장에서는 6장에서는 간단한 예제를 통한 제안하는 분석 기법의 적용 과정을 제시하고, 7장에서 예제를 통한 제안한 분석기법의 유용성을 살펴본다. 그리고 마지막으로 8장에서는 결론 및 향후 연구 내용에 대해 기술한다.

2. 관련 연구

기존의 모델 기반 소모전력 분석에 관한 연구들은 Tan[11]의 연구와 Jun[13]의 연구, Hong[12]의 연구, 그리고 Nogueira[14]와 Carneiro[15] 연구 등이 있다. Tan의 연구에서는 소프트웨어의 소모전력 예측을 위하여 소프트웨어 태스크 모델에 대한 SAG(Software Architecture Graph)를 작성하고, 작성된 그래프의 리덕션(reduction) 과정을 거쳐 소모전력에 대한 절감 효과를 제시하고 있다. Jun의 연구[13]는 컴포넌트 기반의 소프트웨어 개발에서 소모전력을 분석하는 연구이다. 소프트웨어 컴포넌트 아키텍처를 오토마타로 정의하고, 정의된 오토마타를 시뮬레이션하기 위해 실행시간에 대한 가중치를 부여한 후, 오토마타 실행을 통한 소모전력을 예측하였다.

Hong의 연구[12]는 임베디드 소프트웨어에 대한 모델 기반의 소모전력 분석을 위한 에너지 매크로 모델링에 대한 연구이다. 이 논문에서는 명령어 기반의 소모전력 분석을 위한 시뮬레이션 도구를 이용하여, UML 모델요소에 대한 명령어 패턴을 정의하고, 이를 기반으로 에너지 매크로 모델을 개발하였다. 개발된 에너지 매크로 모델은 확장 가능한 라이브러리로 구축되어 모델 기반의 소모전력 분석을 위한 기반으로 활용하였다.

그러나 이러한 연구들은 SAG, 오토마타, UML 시퀀스 다이어그램 등을 이용하여 소모전력을 분석하였으며, 상태기반의 소프트웨어 모델링에 적합한 방법으로 볼 수 없다.

모델 기반의 소모전력 분석에서 특히 페트리 넷을 이용한 연구들은 Nogueira[14]와 Carneiro[15] 등에 의해 수행되었다. Nogueira의 연구에서는 CPN(Coloured Petri Net)과 DTMC(Discrete Time Markov Chains)을 이용하여 임베디드 시스템의 소모전력을 분석 하였다. 이 연구에서는 먼저 프로세서의 명령어(instruction)별로 에너지 소모량과 수행시간을 측정한 후, CPN으로 모델링된 프로세서 모델에서 각 명령어의 소모전력 값을 계산할 수 있게 하였다. 즉 분석 대상 프로그램 코드를 DTMC로 변환하여 각 명령어의 실행 빈도와 횟수를 구한 후, CPN 모델과 조합하여 전력 소모 값을 구한다.

Carneiro의 연구[15]는 SysML의 상태기계 다이어그램을 기반으로 시스템이 소모하는 전력을 평가 하였다. SysML 상태기계의 상태와 트랜지션에 대한 소모전력 값과 수행시간을 먼저 실측하고, 이를 주해(annotation)를 이용해 최대 값과 최소 값으로 정의하였다. 이를 페트리 넷(Petri Net)을 확장한 ETPN(Energy Timed Petri Net)으로 변환하여 시스템의 소모전력을 평가하였다.

Nogueira의 연구[14]는 소프트웨어 모델 보다는 프로세서를 모델링한다는 측면에서 본 연구와 차별성이 있으며, Carneiro의 연구는 상태 기계를 기반으로 소모전력을 분석하고 있으나, 상태 전이에서 발생하는 행위를 기반으로 분석하는 것이 아니라 상태 변화에 따른 실측값을 이용했기 때문에 다양한 상태 변화나 트랜지션의 변경이 발생하면 다시 측정해야 하는 문제점을 가지고 있다. 즉, 상태 변이에 대한 부분적인 변경이나 모델의 변경에 따라 분석기법의 적용이 어려워진다. 따라서 본 논문에서는 상태 기계에서 전

력 소모를 일으키는 대상으로 액션(action)을 분석하고, 이를 통한 소프트웨어 시스템의 소모전력 특성을 구하고자 한다.

3. 상태 기계를 이용한 소모전력 분석 절차

상태 기계 다이어그램은 임베디드 시스템의 상태 변화를 기반으로 소프트웨어의 동작을 표현하는 모델이다. 그러나 UML에서 제시하는 상태 기계 다이어그램의 표현과 상태기반 분석 기법만으로는 소프트웨어 행위를 통한 성능, 데드락과 같은 비기능적 속성을 분석하고 검증하기에는 충분하지 못하다[16]. 이러한 단점을 극복하기 위해 상태 기계 모델을 페트리 넷과 같은 정형 기법으로 변환한 후, 시뮬레이션을 수행하는 다양한 방법들이 제시되었다[17,18]. 특히 CPN은 다양한 종류의 파라미터를 정의할 수 있는 Color Set을 통해 시스템의 동적 행위를 보다 정밀하게 분석할 수 있게 지원하고 있다[19]. 따라서 상태 기계를 이용한 소모전력 특성을 분석하기 위해서는 CPN과 같은 모델로 변환하여 분석하는 것이 적합할 수 있다. 그림 1은 본 논문에서 제안하는 상태 기계 기반의 소모전력 분석 절차를 나타내고 있다.

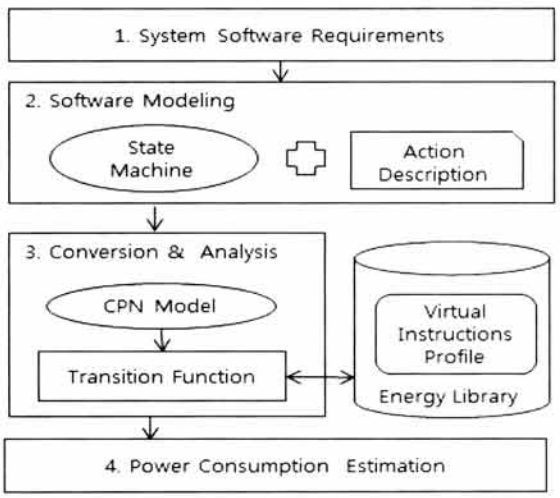


그림 1. 상태 기반의 소모전력 분석 절차
Fig. 1. State-based software power analysis process

3.1 시스템 소프트웨어 요구사항

소프트웨어 개발을 위해서는 개발될 시스템에 대한 요구사항 분석이 선행 되고, 이를 토대로 소프트웨어 설계 모델들을 작성한다. 따라서 상태 기계 다이어그램은 시스템의 행위와 관련된 요구사항을 만족할 수 있게 작성되어야 한다. 또한 전력 소모 절감을 위한 분석이 수행되어야 하므로, 소프트웨어가 실행되는 동안 상태별로 최대 허용 가능한 전력 소모량에 대한 요구사항을 포함하고 있어야 한다.

3.2 소프트웨어 분석/설계

주어진 요구사항을 기반으로 소프트웨어의 설계가 이루어진다. 소프트웨어는 특정 시점에서의 상태들의 연속으로 표현될 수 있으며, 이때 UML에서 제공되는 상태기계 다이어그램을 이용하여 소프트웨어 설계를 진행할 수 있다. 특히 임베디드 소프트웨어의 대부분은 이벤트 기반의 상태 전이 행위를 유발하는 시스템 모형화가 가능하기 때문에 상태기계 다이어그램을 통한 소프트웨어 설계가 가능하다. 특히 실제 산업체에서는 StatemateTM와 같은 상태 기계 다이어그램의 자동화 도구를 이용하여 소프트웨어를 설계하고 검증하며, 코드를 자동 생산하기도 한다. 상태기계 다이어그램에서 소프트웨어의 동작은 이벤트 발생과 이로 인한 상태 전이 유발 행위이다.

설계 단계에서의 행위 즉, 액션은 소프트웨어의 동작을 추상적으로 기술하기 때문에, 내부 행위를 알 수 있는 추가 정보가 필요하다. 따라서 상태 기계를 모델링 하는 과정에서 액션의 내부 행위를 기술하는 Action Description을 구성하였다. Action Description의 요소들은 실제 전력 소모를 일으키는 부분으로써, 상태 전이와 관련하여 발생하는 동작을 표현하게 된다. Action Description은 일반적인 프로그램 언어에서 쓰이는 단위 연산자들과 반복문, 분기문으로 구성된다.

3.3 상태 기계 모델의 변환

시뮬레이션을 위해 상태 기계 모델과 Action Description을 CPN 모델로 변환한다. 먼저 상태 기계 모델을 CPN 모델로 변환 하는데, 이때 추상적으로 기술된 액션들은 각각 CPN의 트랜지션으로 나타낸다. 액션에 대한 트랜지션들은 Action Description을 변환한 모델로 연결되어 실제 전력 분석을 수행하게 된다. 이를 위해 상태 기계 모델을 CPN 모델로 변환하는 방법과, Action Description을 CPN 모델로 변환하는 방법이 필요하다.

Action Description에 대한 전력 소모 값을 구하기 위해서는 에너지 라이브러리에 유지되고 있는 가상 명령어(Virtual Instructions)의 소모전력 프로파일을 이용한다. 가상 명령어라 함은 임베디드 리눅스에 사용하고 있는 명령어를 일반화하여 운영체제와 명령어간의 의존성을 배제하도록 한 것이며 Bammi의 의해 연구[20]되어 제시되었다. Action Description은 가상 명령어들의 조합으로 나타낼 수 있으므로, Action Description을 에너지 라이브러리에 유지 관리되는 가상 명령어로 대응시키기 위한 매핑 테이블을 필요로 한다. 매핑 테이블은 CPN의 트랜지션 함수로 작성할 수 있으며, 트랜지션 함수를 이용하여 Action Description에 나타난 행위들의 소모전력을 구할 수 있다.

3.4 소모전력 분석 및 예측

CPN은 계층형 구조의 모델링이 가능하도록 지원한다. 따라서 계층적으로 모델링된 상태기계 다이어그램에 대한 표현이 자연스럽게 이루어질 수 있다. CPN 모델에 대한 시나리오 기반의 시뮬레이션을 통해 토큰의 이동, 즉 상태 전이 과정을 추적(Trace) 할 수 있으며, 이러한 추적과정에서의 누적 소모전력이 곧 임베디드 소프트웨어의 소모전력 값으로 정의 할 수 있다.

4. 에너지 라이브러리

소모전력 분석을 위해서는 대상 시스템의 구성 요소에 대한 전력 특성 정보가 필요하다. 이러한 전력 특성 정보를 얻는 방법 중의 하나로 라이브러리를 구축하여 소모전력 특성을 관리하는 방법이 있다[12]. 본 논문에서는 기존에 수행한 연구 결과로 구축된 에너지 라이브러리를 이용하여 소모전력 분석에 활용한다. 라이브러리에 저장된 소모전력 특성들은 추후 재사용이 가능하며, 업데이트가 용이하다. 따라서 이를 통해 소모전력 특성을 체계적으로 관리할 수 있고, 소모전력 분석을 위한 시뮬레이션 수행의 속도를 향상시키는 효과가 있다.

4.1 에너지 라이브러리 구성

Kim[21]의 연구에서는 에너지 라이브러리를 구축하여 설계 모델 기반의 소모전력 분석을 지원하고 있다. 에너지 라이브러리의 구조는 그림 2와 같다.

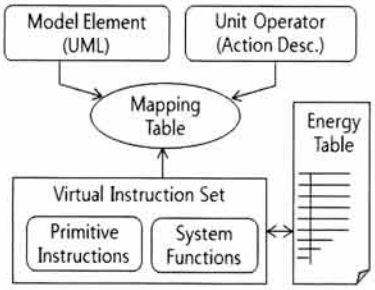


그림 2. 에너지 라이브러리의 구조
Fig. 2. The structure of energy library

에너지 라이브러리에는 UML 모델에서 전력 소모를 유발하는 모젤 요소들과 Action Description의 단위 연산들에 대한 목록을 유지관리하며, 이들은 가상 명령어와 일대일 또는 일대다 관계로 매핑되어 있다. 모든 가상명령어들은 최소 200번 이상의 시뮬레이션을 통해 산출된 명령어별 소모전력 특성 값을 가지고 있으며, 명령어의 피연산자에 대한 자료형에 따라 세부적으로 소모전력 특성 값이 제공된다. 이들에 세부 사항은 참고문헌 [21]에서 찾을 수 있다.

4.2 에너지 라이브러리 활용

상태 기계 모델에서 에너지 소비를 유발하는 동작 요소는 에너지 라이브러리에 존재하는 매핑 테이블을 통해 실제 명령어 수준에서의 소모전력 특성을 파악하게 된다. 특히 Action Description에 나타난 상태 기계의 동작은 에너지 라이브러리와 연계하여 소모전력 특성을 산출한다.

표 1은 Action Description의 단위 연산자(AD_PO)들을 VIS 명령어로 매핑한 것을 나타낸다. 이러한 정보는 에너지 라이브러리에 유지 관리되며, UML 모델 요소의 추가와 단위 연산자의 추가 및 수정이 쉽도록 라이브러리가 구축되어 있다. VIS 명령어에는 각 단위 연산자에 대해서 피연산자의 자료형별로 소모 전력량이 기술되어 있다[12].

표 1. AD_PO와 VIS의 매핑 예시
Table 1. Mapping example of AD_PO and VIS

단위 연산자	VIS
=	store
+	add
-	sub
*	mult
/	call divide
%	call modulo
<<	bitShiftLeft
>>	bitShiftRight
!	bitNOT
&	bitAND
|	bitOR
^	bitXOR

5. 모델 변환

5.1 상태 기계 모델의 메타 모델

UML 2.3[22]에서는 상태 기계 모델을 구성하는 요소들을 그림 3과 같이 메타 모델로 정의하고 있다. 그림 3에서 정의하는 것과 같이 상태 기계는 크게 State, Transition, vertex, 그리고 Region으로 구성된다. State는 시스템의 상태를 직접적으로 표현하는 simple state, composite state, submachine state, final state와 시스템의 상태를 표현하지는 않지만, 시스템의 흐름을 표현하기 위한 사용되는 Pseudo state로써, choice, entry point, exit point, history, initial, junction, terminate 상태가 존재한다[22].

Transition은 상태 변화를 위한 행위 수행 부분으로써, action, receive signal, send signal 등이 있다. 또한 Transition에는 상태 전이과정에서 발생할 수 있는 레이블이 존재한다. 이 레이블은 e[g]/a 와 같은 형태로 표현된다. 즉, 일정 조건 g 상에서 이벤트 e가 발생하면 액션 a를 수행한다는 의미이다. 여기서 액션 a가 단위 동작으로 표현되는 경우는 단위 연산에 의해 이루어질 수 있으나, 액티비티(Activity)와 같은 함수 호출로 이루어지는 경우에는 이 함수의 동작을 Action Description에 의해 정의한다. 마지막으로 상태 기계는 시스템이 갖는 병행 수행의 의미를 나타내기 위한 목적으로 Region을 정의 할 수 있다.

5.2 모델 구성 요소의 변환 규칙

앞서 정의한 상태 기계를 표현하는 구성요소인 State, Pseudo state, Transition, Vertex label, Region에 대하여 CPN으로 어떻게 변환하는지에 대한 규칙을 정의하면 다음과 같다.

5.2.1 State의 변환 규칙

1) simple state는 시스템의 특정 시점에 존재하는 하나의 상태를 의미하기 때문에 CPN의 place로 매핑한다.
2) composite state는 하위에 sub-states들로 구성된 복합

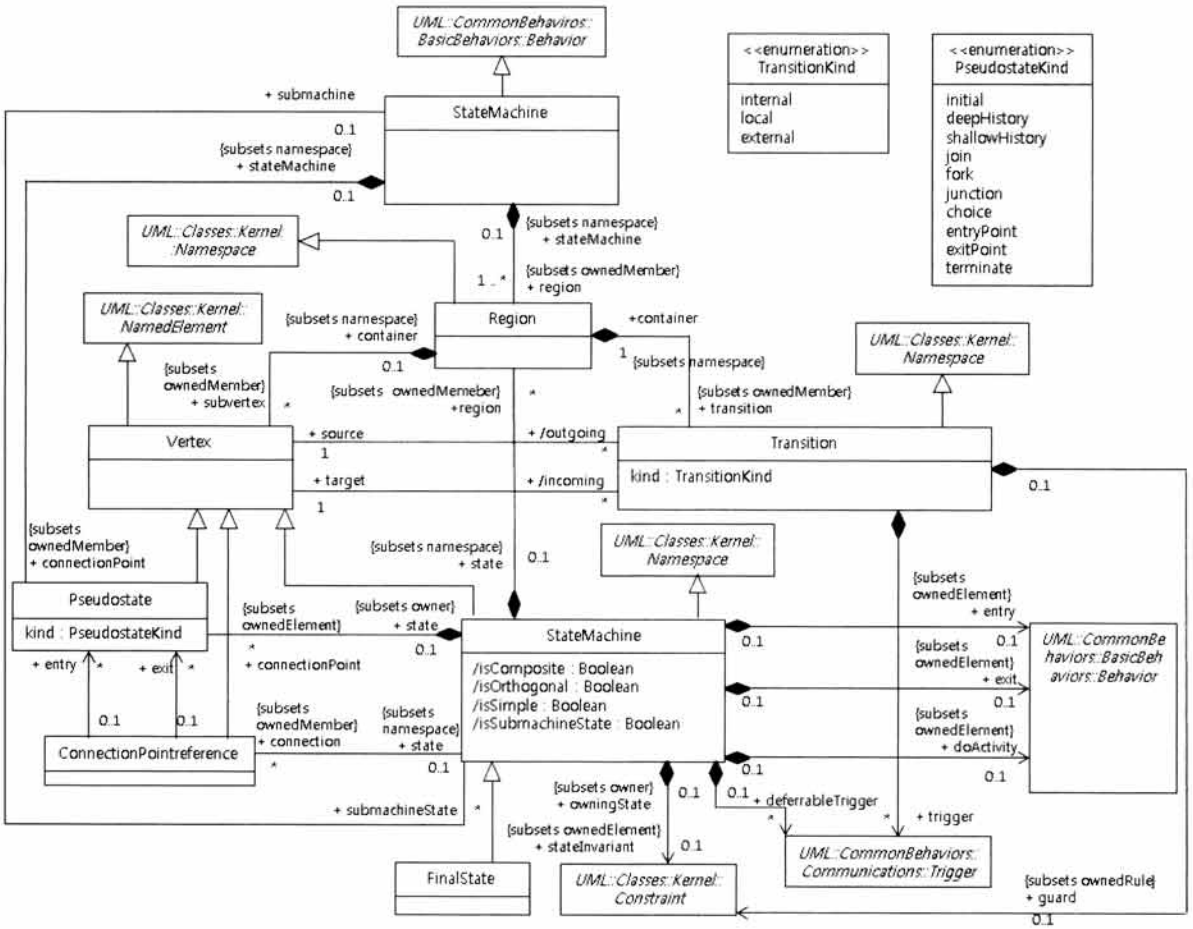


그림 3. UML 2.3에서 정의된 상태기계의 메타 모델[22]
Fig. 3. Meta-model of UML 2.3 state machine diagram[22]

상태를 의미하기 때문에 CPN의 fusion place로 매핑한다.

3) submachine state는 entry point와 exit point를 갖는 일종의 composite state로써, 역시 fusion place로 매핑한다.
4) final state는 시스템의 종료를 이미하는 상태로써, incoming arc만을 갖는 place로 매핑한다.

이와 같은 규칙에 의거하여 변환 형태를 심볼로 표기하면 표 2와 같이 나타낼 수 있다.

표 2. State의 변환 규칙
Table 2. Transformation rules of state

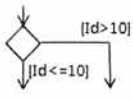
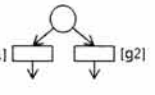
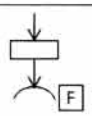
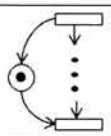
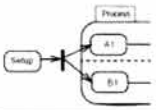
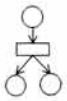
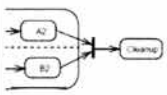
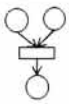
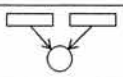
구성요소		상태기계 표현	CPN 표현
State	simple state	S	
	composite state	S	F
	submachine state	S, Monitoring activity	F
	final state		

5.2.2 Pseudo state의 변환 규칙

1) choice는 특정 조건에 따른 선택적 분기를 의미하며, 분기의 수만큼 트랜지션으로 매핑한다.
2) entry point와 exit point는 submachine state로의 진입을 위한 입출구를 의미하며, 이는 fusion place로 진입을 위해 요구되는 트랜지션으로 매핑한다.
3) history는 특정 시점에서의 상태를 저장하기 위한 용도로 사용되며, 토큰을 갖는 place로 매핑된다.
4) initial은 상태 전이의 시작을 의미하며 initial marking을 갖는 place로 매핑된다.
5) junction은 두 개 이상의 트랜지션으로부터 incoming arcs를 갖는 place로 매핑된다.
6) terminate는 final state와 동일하게 incoming arc만을 갖는 place로 매핑한다.
7) fork는 병렬 프로세스의 상태를 생성하기 위한 트랜지션을 거쳐 두 개의 place로 매핑된다.
8) join은 두 개의 상태로부터 하나의 상태로 전환하는 트랜지션을 거쳐 하나의 place로 매핑된다.

이와 같은 규칙에 의거하여 변환 형태를 심볼로 표기하면 표 3과 같이 나타낼 수 있다.

표 3. Pseudo State의 변환 규칙
Table 3. Transformation rules of pseudo state

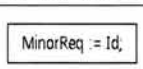
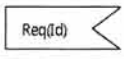
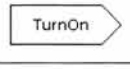
구성요소		상태기계 표현	CPN 표현
Pseudo State	choice		
	entry point		
	exit point		
	history		
	fork		
	join		
	initial		
	junction		
	terminate		

5.2.3 Transition의 변환 규칙

1) action : 상태전이에 나타나는 행위를 표현하는 것으로써, Transition function을 갖는 트랜지션으로 매핑한다.
2) receive signal : 외부로부터의 입력을 의미하며, outgoing arc와 토큰을 갖는 input place로 매핑한다.
3) send signal : 외부로의 출력을 의미하며, incoming arc와 토큰을 갖는 output place로 매핑한다.

이와 같은 규칙에 의거하여 변환 형태를 심볼로 표기하면 표 4와 같이 나타낼 수 있다.

표 4. Transition의 변환 규칙
Table 4. Transformation rules of transition

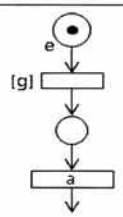
구성요소		상태기계 표현	CPN 표현
Transition	action		
	receive signal		
	send signal		

5.2.4 Transition Label의 변환 규칙

1) 이벤트 e : 토큰을 갖는 place로 매핑한다.
2) guard condition : guard condition을 갖는 트랜지션으로 매핑한다.
3) action : transition function을 갖는 트랜지션으로 매핑한다.

위와 같은 규칙에 의거하여 트랜지션 레이블 전체에 대한 CPN 표현은 표 5와 같이 나타낼 수 있다.

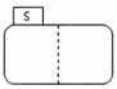
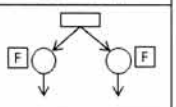
표 5. Transition label의 변환 규칙
Table 5. Transformation rule of transition lable

구성요소		상태기계 표현	CPN 표현
Transition Label	e[g]/a		

5.2.5 Region의 변환 규칙

Region은 병행적으로 존재할 수 있는 AND submachine을 의미하는 것으로 두 개의 fusion place로 표현하되 하나의 트랜지션으로부터 incoming arcs를 갖도록 표 6과 같이 매핑한다.

표 6. Region의 변환 규칙
Table 6. Transformation rule of region

구성요소		상태기계 표현	CPN 표현
Region	AND submachine		

5.2.6 Action Description의 변환 규칙

상태 기계의 트랜지션에 나타나는 액션은 크게 단위 동작과 함수 호출로 구분된다. 단위동작은 명령어 수준에서 쪼개질 수 없는 단위 연산을 의미하며, 함수 호출은 다수의 기본 동작을 포함하는 액티비티(Activity)로써, 가상명령어 집합에 포함된 Virtual System Function이나 사용자가 정의한 함수들로 나타날 수 있다. 사용자 정의 함수는 Action Description을 이용하여 기술한다.

Action Description은 상태기계 다이어그램을 지원하는 Rhapsody나 Tau G2 등과 같은 상용 도구에서 C 언어나 C++과 같은 언어의 문법을 사용하여 작성하는 것이 일반적이다[23]. 이들에 대한 변환 규칙은 다음과 같다.

1) 단위 연산: 트랜지션의 표현식으로 매핑된다.
2) Virtual System Function: CPN Transition function으로 매핑한다.

3) 사용자 정의 함수: CPN Transition function으로 매핑하거나 사용자 정의 함수를 CPN의 Sun-Net으로 모델링하는 Fusion 트랜지션으로 매핑한다.

참고로 사용자 정의함수를 기술하는 Action Description이 CPN의 Transition Function을 매핑될 때, CPN ML에서 제공하는 Expression과 Function의 구성요소로 표현될 수 있다. Design/CPN에서 제공하는 ML 스펙[24]에서는 일반적인 산술, 관계 연산의 표현식뿐만 아니라 참조 연산, Timer 연산, 조건연산 등을 포함하고 있다.

6. 모델 기반 소모전력 분석 및 적용

앞의 4장과 5장에서 제시한 에너지 라이브러리 구성과 모델 변환 기법을 기반으로 CPN 기반의 소모전력 예측이 가능하다. 따라서 본 절에서는 예제 시스템의 적용을 통하여 CPN 기반의 소모전력 분석 방법에 대하여 보다 구체적으로 제시한다. 예제 시스템은 상태 기계 다이어그램으로부터 CPN의 변환을 잘 보여줄 수 있는 예제로 소모전력의 예측이 간단한 사칙연산 프로그램으로 선택하였다.

6.1 사칙 연산자 모델

그림 4는 아주 간단하게 2개의 데이터와 연산자가 입력되면 해당 연산을 수행하고 결과를 저장하는 상태기계 모델이다. n1과 n2의 값과 연산자 op를 받아서 op 값에 따른 사칙연산중의 하나를 수행하는 행위를 보여준다. 그림 5는 그림 4로부터 5장에서 제시된 변환 규칙을 기반으로 생성된 CPN 모델이다.

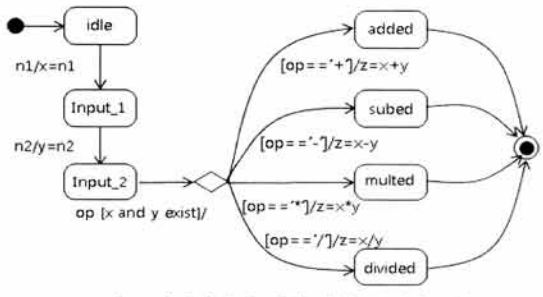


그림 4. 사칙연산자 상태 기계 다이어그램
Fig. 4. State machine diagram of arithmetic operators

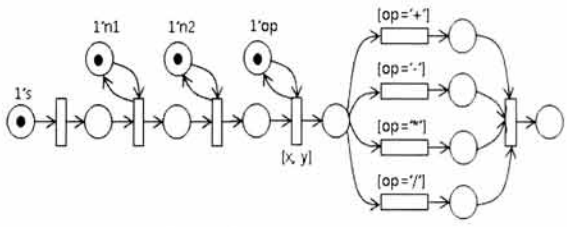


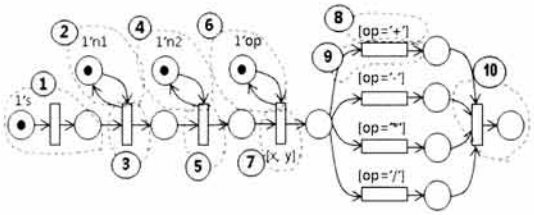
그림 5. 사칙연산자의 CPN 모델
Fig. 5. CPN model of arithmetic operators

6.2 CPN 모델의 소모 전력량 산출

그림 5의 CPN 모델을 기반으로 소모전력을 산출할 수 있다. 즉 CPN 모델은 초기 상태에서 시작하여 각각의 트랜지션을 거쳐 최종의 상태에 도달하는 과정에서 모든 선언과 동작에 해당되는 에너지 소모 값을 라이브러리로부터 얻어오고 이를 합산하게 된다. 이 과정을 구체적으로 살펴보면 표 7과 같이 정리할 수 있다.

표 7에서 얻어진 값에 대한 소모전력 값의 적정성을 확인하기 위하여 주어진 두 수에 대하여 연산자에 따라 사칙 연산중에 하나를 선택하는 C 언어 프로그램을 작성하고, 이를 에너지 라이브러리 구축을 위해 사용했던 EMSIM 2.0 도구[25]를 사용하여 소모전력을 시뮬레이션 하였다. 시뮬레이션 결과는 사칙연산중 add 연산의 경우를 선택하였을 때,

표 7. 사칙연산자의 에너지 소모량 산출과정
Table 7. Energy calculation of arithmetic operators



Step	Operation	Instructions	Energy Consumption (nJ)
1	1's	set	13.016
2	1'n1	load n1 store n1	13.016 12.489
3	x=n1	load n1 move n1 store x	13.016 12.489 12.489
4	1'n2	load n2 store n2	13.016 12.489
5	y=n2	load n2 move n2 store y	13.016 12.489 12.489
6	1'op	load op store op	14.036 13.176
7	op[x,y]	compare x,y load op move op store op	10.191 14.036 13.176 13.176
8	[op='+']	load op compare op branch	14.036 10.191 01.914
9	z=x+y	load x load y add x, y move result store z branch	13.016 13.016 10.869 12.489 12.489 01.914
10	return	load 0 store branch	13.016 12.489 01.914
Sum of Energy Consumption			335.158

506.06791nJ의 값을 얻었다. 그러나 EMSIM 2.0을 이용한 시뮬레이션을 수행하기 위해서는 다음과 같은 함수를 정의하게 되는데, 이는 EMSIM 시뮬레이터를 시작하고 종료시키기 위한 시스템 함수이다.

이 함수가 소모하는 소모 전력량을 제거하기 위하여 sa110_terminate() 함수가 소모하는 전력량을 계산한 결과 171.91757nJ의 값을 얻었으며, 결과적으로 순수하게 add 연산자가 소모하는 전력량은 506.06791 - 171.91757 = 334.15048nJ의 값을 얻었다. 표 8은 이의 결과를 비교한 것이다.

```
void sa110_terminate()
{
 //        printf("Terminating\n");
#ifndef HOST
        ___asm___ ___volatile___ (
        "mov    r6, #0\n"
        "mov    pc, r6\n"
        );
#endif
}
```

표 8. 사칙연산자 소모전력의 정확도 비교
Table 8. Energy comparison of arithmetic operators

연산	EMSIM 2.0 소모전력	CPN기반 소모전력	편차
add(+)	334.150	335.158	0.3%
subtract(-)	333.158	335.498	0.7%
multiply(*)	335.752	335.804	0.2%
divide(/)	364.078	353.898	2.7%

표 8으로부터 CPN 모델로부터 예측된 소모 전력의 값은 소스코드 기반의 소모전력 예측 값과 비교하여 평균 0.9% 수준의 편차를 보였다. 이는 매우 단순한 사칙연산을 위한 프로그램으로부터 측정된 것으로 기인한다.

7. 실험 및 결과 분석

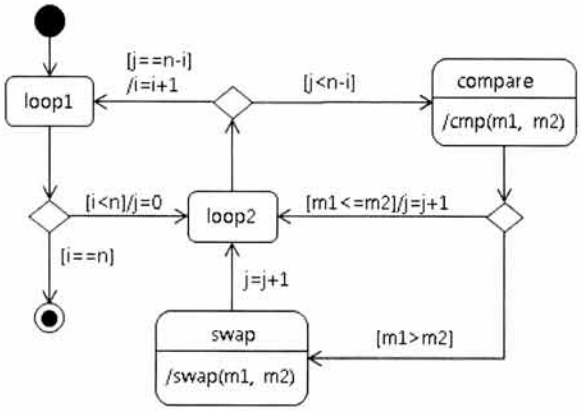
7.1 실험 환경

먼저 실험 도구로 사용한 CPN 툴은 덴마크의 Aarhus 대학과 Meta Software Co,에서 개발한 Design/CPN 툴을 이용하였다. 이 툴은 예제 시스템에 대한 상태 기계 모델을 CPN으로 변환한 후 시뮬레이션을 통해 CPN 모델에 문제가 없다는 것을 확인하기 위해 사용하였다. 실험을 위해서는 Open Source 사이트로부터 확보한 버블정렬을 사용하였다.

실험의 일관성을 위하여 예제 시스템에 대한 C 언어 기반의 프로그램은 EMSIM 2.0을 기반으로 하는 소모전력 시뮬레이터를 이용하여 산정하였다. EMSIM 2.0 분석 도구는 인터넷으로부터 구할 수 있는 널리 알려진 소모전력 분석도구이며, 작은 규모의 C 언어 분석에 많이 사용된다. 이 도구는 StrongARM 계열의 SA110 프로세스 코어를 갖는 하드웨어 플랫폼을 기반으로 소모전력을 분석하며, 본 연구에서 제시하는 에너지 라이브러리도 동일한 계열의 하드웨어 플랫폼에서 분석된 소모전력 프로파일 정보를 이용하여 실험하였다.

7.2 실험 내용

그림 6은 임베디드 소프트웨어의 구현에서 널리 사용되고 있는 버블정렬(Bubble Sort) 알고리즘에 대한 상태기계 다이어그램의 예시이다. 버블정렬의 알고리즘에 따라 두개의 반복문을 수행하는 상태와 데이터의 순서를 바꾸는 swap() 함수를 내부 행위로 갖도록 모델링 되었다. swap() 함수는 그림 6의 (b)와 같이 Action Description으로 정의하였다.



(a) Bubble Sort 상태기계 다이어그램

```
Action Description
swap(n1, n2) {
   var temp;
   temp = n1; n1 = n2;  n2 = temp;
}
```

(b) Swap 함수의 Action Description

그림 6. 버블정렬 알고리즘의 상태기계
Fig. 6. State machine of bubble sort

그림 6에 대한 상태기계 다이어그램에 대한 CPN 모델의 변환은 그림 7 및 그림 8과 같다. 그림 7의 CPN 모델의 선언부에 대한 Color Set을 정의한 부분이다.

그림 7에서와 같이 CPN의 선언부는 세 개의 그룹으로 나누어 선언하였다. Color Set 그룹은 CPN에서 각각의 place에 입출력 되는 token의 형태를 정의한다. Variable Set은 CPN의 arc를 통해 place와 transition간에 전달되는 token들을 정의한다. 세 번째 그룹은 CPN 모델에서 소모전력 값을 도출하기 위한 Func Set을 정의하였다. Func Set은 CPN 모델을 시뮬레이션 하는 동안 소모전력 값을 산출하는데 사용되는 함수들로, 에너지라이브러리의 소모전력 프로파일을 참조하여 각 트랜지션에서 발생하는 전력소모 값을 연산한다.

```
▼Color Set
  ▼val n = 10;
  ▼val cnt = 1`(23)
  ▼val energy=1`(0)
  ▼colset I = INT;
  ▼colset J = INT;
  ▼colset E = INT;
  ▼colset ExI = product E*I;
  ▼colset COLOR = product E*I*J;
▼Variable Set
  ▼var e:INT;
  ▼var i:INT;
  ▼var j:INT;
  ▼var jdx:INT;
  ▼var idx:INT;
  ▼var inp:INT;
  ▼var data:INT;
  ▼var data1:INT;
▼Func Set
  ►fun addNum
  ►fun subNum
  ►fun assign
  ►fun cmp
  ►fun swap
```

그림 7. 버블 정렬의 CPN 모델 선언부
Fig. 7. CPN declaration part of bubble sort

버블 정렬의 상태기계 다이어그램의 CPN모델은 Design CPN 툴을 사용하여 그림 8과 같이 생성되었다. 이는 5장에서 제시된 변환 규칙에 의해 생성하였으며, Design CPN툴의 모델 시뮬레이션을 지원하기 위해 그림 7에서 선언된 Color Set을 할당 하였다. 또한 버블 정렬 알고리즘의 best case, mean case, worst case를 제어하기 위한 CNT place를 추가하여 각 case별로 소모전력 값을 도출할 수 있게 하였다. 또한 swap()과 같이 Action Description으로 기술된 행위들은 sub-net model로 작성하였다.

7.3 소모전력 예측 및 분석

C 언어로 작성된 버블 정렬 프로그램을 EMSIM 2.0을 이용한 소모전력 예측 값과 CPN 모델을 이용한 소모전력 분석 값을 표 9와 그림 9에 정리하였다. EMSIM 2.0과 CPN 모델은 동일한 테스트 케이스를 사용하여 실험을 진행하였다. 표 9는 입력 데이터의수를 변화시켜가면서 버블 정렬 수행 결과를 나타낸다. 그림 9는 표 9에 나타난 실험 결과에 대한 그래프 표현이다.

표 9의 실험 결과에 나타난 오차를 살펴보면, 음수로 표현된 값은 CPN으로 분석한 결과가 코드 기반의 분석(EMSIM 이용) 결과보다 더 많은 전력을 소모하는 경우이다. 전체적으로 보면 평균 2.64%의 오류가 존재하는 것을 알 수 있다. 6장의 사칙 연산자에 비해 오차가 높게 나타난 것은 C 컴파일러에서 코드 최적화를 적용한 Instruction Set을 생성하여, Action Description으로 기술된 행위와 차이를 보였기 때문으로 해석된다.

그림 9에 나타난 그래프를 볼 때, 코드 기반의 분석과 모델 기반의 분석에 대한 소모전력의 변화가 다소 차이가 있기는 하지만, 동일한 변화 추이를 보이고 있다. 이와 같은 결과는 소프트웨어 개발자 입장에서 코드가 생성되기 전에 설계 모델을 입력으로 전체적인 소프트웨어 소모전력을 예측해 본다는 측면에서 의미가 있다. 비록 코드에 비하여 정확한 값을 제공하지는 않지만, 다양한 설계 모델의 소모전력 예측을 통해, 소모전력 절감이 가능한 상태기계 다이어그램을 생성할 수 있다는 측면에서 도움을 받을 수 있을 것이다.

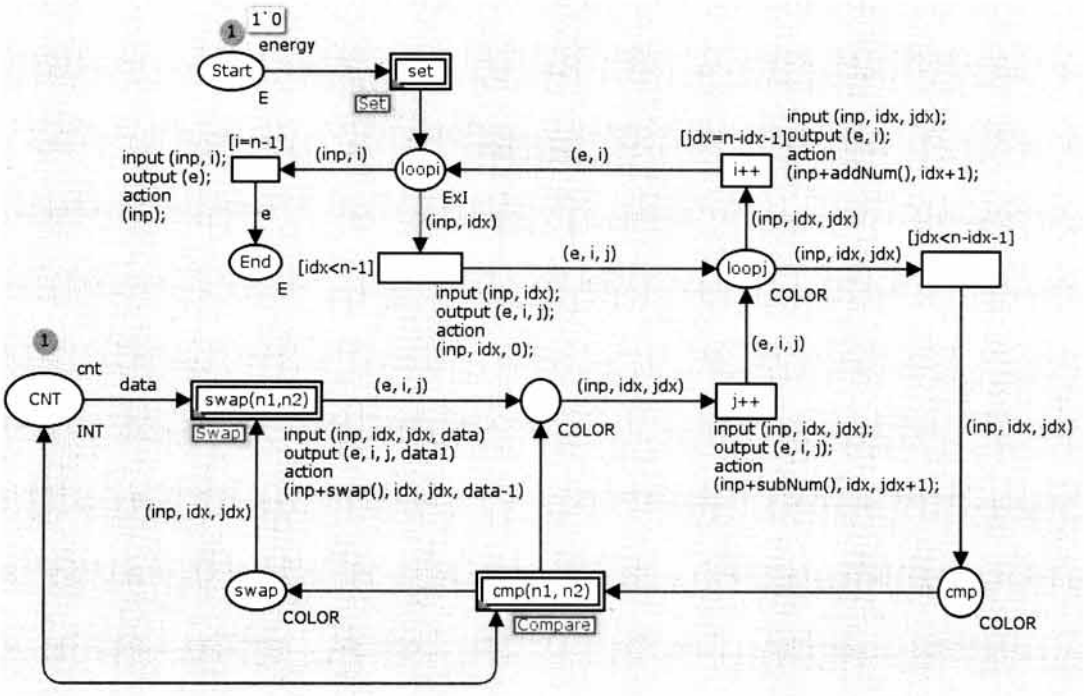


그림 8. Bubble Sort의 CPN 모델
Fig. 8. CPN model of bubble sort

표 9. 버블정렬에 대한 소모전력 분석결과(nJ)
Table 9. Energy consumption analysis results for bubble sort

# inputs		EMSIM	CPN	오차
5	best case	1266.335	1152.855	8.96
	mean case	1642.937	1527.525	7.03
	worst case	1924.005	1902.195	1.13
10	best case	3669.612	3227.994	12.03
	mean case	5114.272	4827.276	5.61
	worst case	6294.757	6357.024	-0.99
30	best case	29103.058	26515.665	8.89
	mean case	41003.447	41604.543	-1.47
	worst case	53651.505	56762.955	-5.80
50	best case	79589.709	72399.294	9.03
	mean case	114256.602	114954.102	-0.61
	worst case	148546.893	157578.444	-6.08
100	best case	315533.374	287060.895	9.02
	mean case	454982.403	459157.545	-0.92
	worst case	593886.189	631254.195	-6.29

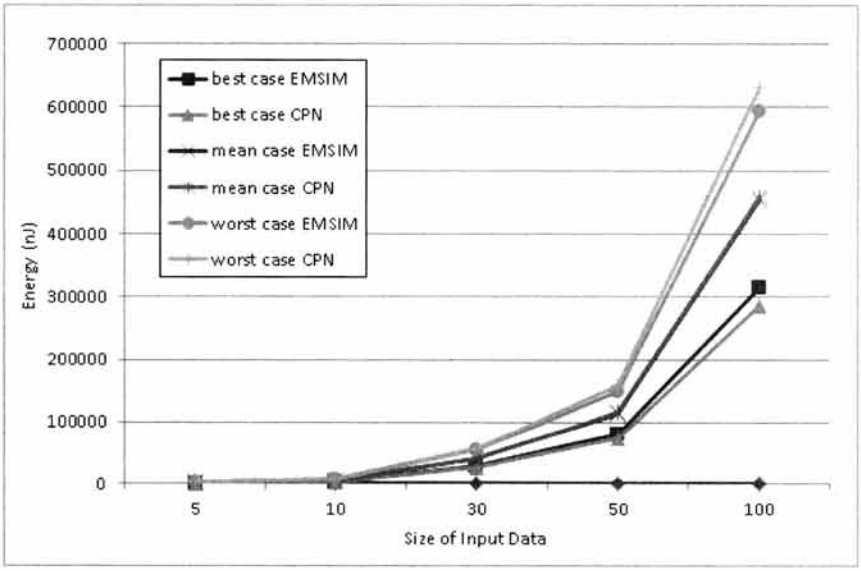


그림 9. 버블정렬에 대한 소모전력 측정 값
Fig. 9. Energy consumption graph for bubble sort

8. 결론 및 향후 연구

본 논문에서는 UML 상태 기계 다이어그램을 이용하여 소모전력 분석을 위한 기법을 제시하였다. 이를 위해 상태 기계에서 에너지 소비를 유발하는 요소인 액션을 분석하여, 에너지 라이브러리의 VIS를 이용하여 소모전력 특성을 얻었다. 또한 상태 기계 모델의 분석을 위해 모델 구성 요소들을 CPN 모델로 변환하기 위한 규칙을 제시하였다. 변환된 CPN 모델은 시뮬레이션을 통해 소모전력을 분석하였다.

본 논문의 연구 결과로, 임베디드 소프트웨어의 개발 과정에서 소모전력 분석을 수행함으로써, 전력 소모가 적은 소프트웨어 개발 가능성을 향상 시켰다. 또한 UML 상태 기계 다이어그램을 이용한 모델 기반 분석 기술을 제시함으로써, 소스 코드가 개발되기 이전에 소프트웨어의 소모전력이 어느 정도인지를 예측할 수 있도록 하였다. 비록 임베디드 시스템에 대한 소모전력의 측정은 실제 하드웨어 플랫폼에서의 실측, 코드 기반의 명령어를 분석을 통한 시뮬레이션이 보다 정확한 소모전력의 값을 제공할 수 있으나, 본 연구에서는 코드 개발 이후에 발생할 수 있는 소모전력 요구사항에 대한 미충족 문제를 사전에 근사적으로 확인함으로써 재작업의 노력을 감소시키고자 하는데 의미가 있다.

향후의 연구로는 소모전력 분석 결과의 정확성을 높이기

위하여 소프트웨어의 실측 값과상태 기계 다이어그램을 통한 모델 기반 분석결과의 오차율을 줄이기 위한 연구를 진행할 것이다. 특히 소프트웨어의 목적 코드 분석을 통해 생성된 실행코드를 확인함으로써, 상태기계 다이어그램을 CPN 모델로 변환할 때 이를 반영하고자하는 연구를 수행할 예정이다.

참 고 문 헌

[1] M. Caldari, M. Conti, M. Coppola, P. Crippa, S. Orcioni, L. Pieralisi, C. Turchetti, "System-Level Power Analysis Methodology Applied to the AMBA AHB BUS," Design, Automation and Test in Europe, pp.32-37, 2003.

[2] Reinaldo A. Bergamaschi, Yunjian W. Jiang, "State-Based Power Analysis for Systems-on-Chip," Design Automation Conference, pp.638-641, 2003.

[3] Youngjin Cho, Younghyun Kim, Sangyoung Park, Naehyuck Chang, "System-Level Power Estimation using an On-Chip Bus Performance Monitoring Unit," IEEE/ACM International Conference on Computer-Aided Design, pp.149-154, 2008.

[4] V. Tiwari, S. Malik, and A. Wolfe "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.2, Issue 4, pp.437-445, 1994.

[5] 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.

[6] Klass, B., et. al., "Modeling Inter-instruction energy effects in a digital signal processor," Proceeding of the Power Driven Microarchitecture Workshop, ISCA'98, 1998.

[7] E. Senn, J. Laurent, N. Julien, and E. Martin, "Softexplorer: estimating and optimizing the power and energy consumption of a C program for DSP applications," EURASIP Journal on Applied Signal Progressing, 2005.

[8] A. Muttreja, A. Raghunathan, S. Ravi, N. K. Jha, "Hybrid Simulation for Energy Estimation of Embedded Software," IEEE Transaction on Computer-AIDED Design of Integrated Circuits and Systems, Vol.26, No.10, pp.1843-1854, 2007.

[9] T. K. Tan, A. Raghunathan, et al., "Energy Macromodeling of Embedded Operating Systems," ACM Transaction on Embedded Computing Systems, Vol.4, Issue 1, pp.231-254, 2005.

[10] X. Yue, Z. Xuehai, L. Xi, G. Yuchang, "OOEM: Object-Oriented Energy Model for Embedded Software Reuse," IEEE International Conference on Information Reuse and Integration, pp.551-558, 2003.

[11] T. K. Tan, A. Raghunathan, et al., "Software Architectural Transformations: A New Approach to Low Energy Embedded Software," Proceeding of Design, Automation & Test in Europe, pp.1046-1051, 2003.

[12] Kim, D.H., Kim, J.P., and Hong, J.E., "A Power Consumption Analysis Technique Using UML-Based Design Models in Embedded Software Development," LNCS Vol.6543, pp.320-331, 2011.

[13] H. Jun, L. Xuandong, Z. Guoliang, W. Chenghua, "Modelling and Analysis of Power Consumption for Component-Based Embedded Software," Proc. Embedded Ubiquitous Computing Workshops 2006, pp.795-804, 2006.

[14] B. Nogueira, P. Maciel, E, Tavares, E. Andrade, R. Massa, G. Callou, R. Ferraz1, "A Formal Model for Performance and Energy Evaluation of Embedded Systems," EURASIP Journal on Embedded Systems, 2011.

[15] E. Carneiro, P. Maciel, G. Callou, E. Tavares, B. Nogueira, "Mapping SysML State Machine Diagram to Time Petri net for Analysis and Verification of Embedded Real-Time Systems with Energy Constraints," International Conference on Advances in Electronics and Micro-electronics, pp.1-6, 2008.

[16] E. Kerkouche, A. A. Chaoui, El Bay Bourennane, O. Labbani "A UML and Colored Petri Nets Integrated Modeling and Analysis Approach using Graph Transformation," Journal of Object Technology, Vol.9, No.4, pp.25-43, 2010.

[17] J. Trowitzsch and A. Zimmermann, "Using UML State Machines and Perti Nets for the Quantitative Investigation of ETCS," ACM Value Tools'06, pp.1-8, 2006.

[18] H. Jung and S. Joo, "Transformation of an Activity Model into a Colored Petri Net Model," Int'l conf. on Trends in info. sci. and computing, pp.32-27, 2010.

[19] Kurt Jensen, Coloured Pert Nets, Vol.1 and Vol.2, Spring-Verlag, 1992.

[20] Bammi, J.R., et al., "Software Performance Estimation Strategies in a System-Level Design Tool," Proceedings of CODES, pp.82-86, 2000.

[21] Kim, D.H. and Hong, J.E., "Energy Component Library for Power Consumption Analysis of Embedded Software," The KIPS Transactions: Part D, Vol.16-D, pp.871-880, Dec., 2009.

[22] OMG, Unified Modeling Language Superstructure, Version 2.3, Doc #: 2010-05-05, May, 2010.

[23] H.S. Min, Embedded System Programming with UML: Rhapsody in C++ (Ver 7.X), Bogdoo Publisher, 2009.

[24] Meta Software, Design/CPN Reference Manual: Part 3, CPN ML Reference, 1993.

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

이 재 욱

e-mail : jwlee@selab.cbnu.ac.kr
2010년 충북대학교 컴퓨터공학부(학사)
2011년 충북대학교 컴퓨터과학과(공학석사)
2011년~현 재 충북대학교 컴퓨터과학과 박사과정
관심분야 : 소프트웨어 품질공학, 정형 기법, 저전력 소프트웨어, 소프트웨어 모델링 등

홍 장 의

e-mail : jehong@chungbuk.ac.kr
1988년 충북대학교 전자계산학과(학사)
1990년 중앙대학교 컴퓨터공학과(공학석사)
2001년 KAIST 전산학과(공학박사)
1990년~1995년 국방과학연구소 선임연구원
2002년~2004년 (주) 솔루션링크 기술연구소장
2004년~현 재 충북대학교 소프트웨어학과 교수
관심분야 : 소프트웨어 품질공학, 모델기반 검증분석, 소프트웨어 아키텍처, 저전력 소프트웨어, 소프트웨어 프로세스 개선 등