중소기업정보기술융합학회논문지 제1권 제1호 pp.55~65, 2011

임베디드 소프트웨어 개발 프로세스에서의 저전력 특성의 설계지원 기법

김종필, 김두환, 홍장의*
충북대학교 전자정보대학 컴퓨터과학과

Techniques to Support Low-Power Characteristics in Embedded Software Development Process

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

요 약 모바일 통신, 센서 네트워크, 웨어러블 컴퓨터 등 IT 응용기술 분야의 급속한 발전으로 인하여 매우 다양한 영역에서 임베디드 소프트웨어의 요구가 증가하고 있다. 이러한 다양한 응용영역에서 저전력을 소모하는 임베디드 소프트웨어의 개발은 배터리를 이용한 전원공급 시스템으로 인하여 매우 중요하게 여겨지고 있다. 따라서 본 논문에서는 임베디드 소프트웨어의 개발 과정에서 소모전력 특성을 고려하는 소프트웨어 개발 기법에 대하여 제안한다. 일반적으로 소프트웨어의 소모전력은 코드 기반 분석을 통해 절감할 수 있지만, 이 경우는 소프트웨어를 다시 개발할 수 도 있는 문제점을 가지고 있다. 따라서 저전력 소모에 대한 요구사항을 기반으로 소프트웨어 개발 과정에서 소모전력을 감소시켜야 하는 방법이 필요하다. 제안하는 소모전력 절감을 위한 임베디드 소프트웨어 개발 프로세스는 다양한 응용분야에서 활용되는 모바일 시스템의 개발에 활용하여 소모전력 측면에서의 경쟁력있는 소프트웨어의 개발이 가능하게 할 것으로 판단된다.

키워드 : 임베디드 소프트웨어, 저전력 소프트웨어, 소모전력 절감기법

Abstract Due to the rapid advance of IT technologies such as mobile communication, sensor network, wearable computer, and so on, the needs of embedded software has increased. In those domain areas, the development of low-power embedded software is one of critical issues to enhance servicability of the system because almost embedded system depends on battery-based power supply system. Therefore this paper identifies the factors that can reduce the power consumption in embedded software operation, and proposes the method that how to handle the factors in software development process. Even though the existing and general studies about power reduction has been performed with code-based analysis, this analysis approach can lead reworks when the requirement for power consumption was not met. Our proposed techniques will support the power reduction in embedded software development process whenever the code was not developed. Our proposed process for lop-power embedded software development can gives the high quality in power-related serviceability.

Key Words : Embedded software, Low-power software, Power reduction technique

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보통신기술개발사업의 지원을 받아 수행된 연구임 (No. 2011-0020523).
*교신저자(jehong@chungbuk.ac.kr)
접수일(2011년 7월 4일), 심사완료일 (2011년 10월 31일)

1. 서 론

최근 스마트 폰, 태블릿 PC 등 모바일 기기의 출현과 센서 네트워크, 웨어러블 컴퓨터 등의 기술 발전 및 응용 분야의 확대로 인하여 다양한 형태의 임베디드 소프트웨어가 개발되고 있다.

특히 스마트 폰에 내장된 소프트웨어의 운영 환경을 살펴보면, 다양하고 방대한 어플의 탑재 및 설치, 과도한 스마트 폰 사용 시간, 그리고 모바일 뱅킹이나 사용자 인증, 이미지 및 동영상에 대한 지속적 사용 등이 스마트 폰 사용의 대표적인 패턴이다. 이러한 상황에서 스마트 폰의 전원 공급은 제품의 매우 중요한 요소로 부각되고 있다. 또한 센서 네트워크를 기반으로 하는 다양한 응용에서도 - 비록 센서가 초소형이며, 이를 운영하기 위하여 적은 전력이 필요하기는 하지만 - 전원 공급은 매우 중요한 요소이다. 과도하게 설치된 센서와 쉽게 접근하기 어려운 위치에 설치된 센서 등에 대한 베터리 지원은 용이하지 않기 때문이다.

이와 같은 임베디드 소프트웨어를 탑재한 시스템에 있어서 소모전력을 절감하는 일은 매우 중요하다. 일반적으로 전력 소모가 적은 하드웨어 장치를 개발하여 사용하고는 있지만, 하드웨어의 소모전력은 탑재된 소프트웨어의 동작에 의존적이다. 따라서 점점 복잡해지고 다기능화 되어가는 모바일 기기의 소모전력을 절감하기 위하여 저전력의 소프트웨어를 개발하는 것은 관심의 대상이 되고 있다[1,2].

과거 소프트웨어의 소모전력 절감을 이하여 프로그램 코드를 기반으로 하는 절감 기법이 많이 제시되었다[3,4]. 그러나 이는 절감 기법이라기보다는 복잡도가 낮은 알고리즘 개발을 통해 전력 절감 효과를 제공하는 방향으로 진행되었다[5,6]. 따라서 본 연구에서는 저전력 소모에 대한 요구사항을 만족시키는 임베디드 소프트웨어 개발을 어떻게 진행할 것인가에 대한 프로세스를 제시하고 이를 통해 에너지 효율적인 소프트웨어 개발이 가능하도록 하는 개발 프로세스를 제안한다. 이의 연구 결과는 추후 저전력 소프트웨어가 요구되는 임베디드 소프트웨어 개발에서 유용하게 사용될 수 있을 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 소프트웨어 소모전력 분석에 대한 기존의 연구들을 살펴보고, 3장에서는 임베디드 소프트웨어에서 소모전력 절감이 가능한 요소들은 무엇인지 식별하고 정의한다. 4장에서는 식별된 소모전력 요소들을 고려하는 저전력 임베디드 소프트웨어 개발 프로세스(ProLoPES: Process for Low-Power Embedded Software development)를 제시하고, 5장에서는 프로세스 각 단계별 적용 가능한 분석 설계 기법에 대하여 정의한다. 마지막으로 6장에서는 결론과 향후 연구 내용을 제시한다.

2. SW 소모전력 분석 및 절차

소프트웨어 소모전력 분석에 대한 기존의 연구는 크게 3가지 접근방법에서 진행되어 왔다.

2.1 명령어 기반 소모전력분석

명령어 기반으로 하는 소모전력분석에 대한 연구는 컴파일된 저수준의 명령어(instruction) 코드를 기반으로 소모전력을 감소시키는 방법이다[3,6,7,8].

이 방법에서는 명령어에 대한 인스트럭션 사이클을 분석하여 명령어 사이클을 감소시키기 위한 방법으로 연구가 진행되었으나, 이를 위해서는 개발 된 코드에 대한 하위 수준의 명령어를 모두 분석해야 하는 노력과 전문성을 요구한다.

2.2 소스코드 기반 소모전력 분석

이는 C언어 등과 같은 고급언어에서의 소모전력을 분석하는 접근 방법이다[9,10]. 비록 입력되는 정보는 고급언어 기반의 소스코드 이기는 하지만, 분석을 위한 접근 방법은 명령어 기반의 분석 기법과 동일하다고 볼 수 있다[11,12]. 소스코드 기반의 소모전력 분석은 함수 단위의 분석이 가능하다는 장점을 제공하며, 함수에 전달되는 파라메터의 크기에 따라 처리 또는 연산량의 변화가 발생하기 때문에 소모전력의 값은 일반적인 스칼라(scalar) 값과 파라메터 값에 의해 결정되는 매크로(macro) 함수로 결정된다.

2.3 모델 기반의 소모전력분석 기법

소프트웨어 코드보다 추상화된 모델을 이용하여 소모전력을 분석하는 기법이다[14]. 이 방법에서는 추상화된 모델을 입력받아 행동 패턴을 분석하고, 그래프 축소(graph reduction)와 같은 기법을 이용하여 동등한 모델로 변환하는 것이다. 변환된 모델로부터 소모전력이 적

은 소프트웨어를 생성할 수 있다. 또한 UML 2.0의 순차 다이어그램(sequence diagram), 상태 기계(state machine) 등을 이용하여 소모전력을 분석하는 연구들도 진행되어 왔다[14,16]. 이 방법은 소스코드가 개발되기 이전에 모델 수준에서 선행적으로 소모전력 분석을 시행해 볼 수 있다는 장점을 제공한다. 특히 UML 다이어그램[17]과 같은 모델을 기반으로 분석하는 경우에는 소프트웨어 개발 절차와 쉽게 통합하여 소모전력이 분석될 수 있다는 장점을 제공한다.

2.4 소모전력 절감 기반 개발 프로세스

임베디드 소프트웨어의 개발에서 소모전력 특성을 고려하는 임베디드 소프트웨어의 개발 절차를 제시한 연구는 쉽게 찾아볼 수 없다. 일반적으로 제시된 소프트웨어 개발 프로세스는 성능, 신뢰성, 안전성 등의 품질 요소들을 만족하기 위한 방법들로 제시되어 왔다[18,19].

소모전력을 고려한 개발 활동은 UML 2.x를 기반으로 하는 소프트웨어 모델링 기법에 대하여 제시한 연구를 찾아볼 수 있으나[15,16], 이들 연구는 전체 개발 과정을 포함하는 프로세스 모델로 제시되지 못하였다.

3. 임베디드 소프트웨어의 소모전력 절감 요소

임베디드 소프트웨어의 개발을 위한 개념 정립 단계에서 요구사항 명세, 설계, 개발, 그리고 테스트 단계를 거쳐 필드 운영에 이르기 까지 전체 개발 수명주기 단계에서 소모 전력을 절감할 수 있는 요소들은 무엇인지 정의한다.

3.1 요구사항 정의 단계에서

최근 소프트웨어에 대한 요구사항을 정의하는 과정은 신 RFP 체계를 근간으로 보다 구체적인 10개 분류 - 시스템요구사항, 기능요구사항, 성능요구사항, 보안요구사항, 품질요구사항, 인터페이스 요구사항, 데이터요구사항, 시스템운영요구사항, 개발 제약사항, 프로젝트지원요구사항 - 의 요구사항을 기술한다. 이러한 요구사항 기술 체계는 대상 소프트웨어에 대한 보다 명확하고 모호성이 없는 기능 및 성능 등에 대한 요구사항을 정의할 수 있게 한다. 표 1은 신 RFP 체계를 기반으로 하는 요구사항 정의 양식의 예를 보여준다.

요구사항에 대한 정확한 정의는 추후 소프트웨어의 분석 및 설계 단계에서 보다 체계적인 그리고 변경을 필요로 하지 않는 강건한 모델 개발이 가능해 진다. 따라서 임베디드 소프트웨어에 대한 정확한 요구사항 개발은 소모전력에 대한 절감을 가져다 줄 수 있는 근간의 정보를 제공하게 된다.

표 1. 신 RFP 체계에서의 요구사항 정의
Table 1. Req. def. by new RFP system

<table>
<tr><td>요구사항 ID</td><td colspan="3">PER-001</td><td>응락수준</td><td colspan="3">필수</td></tr>
<tr><td>요구사항명</td><td colspan="6">사용자 인증 응답시간</td></tr>
<tr><td>기 능 명</td><td colspan="6">모바일 상거래 사용자 인증</td></tr>
<tr><td>기능입력정보</td><td colspan="3">- 보안인증서 식별번호
- 보안인증서 비밀번호</td><td>기능출력 정보</td><td colspan="2">- 인증 결과 (메시지)</td></tr>
<tr><td>입출력 유형</td><td colspan="3">- 사용자 입력</td><td>관련 파일</td><td colspan="2">- 외부파일 (보안인증 DB)</td></tr>
<tr><td>기 능 내 용</td><td colspan="6">다음과 같은 행위의 수행을 통하여 사용자 인증을 위한 응답시간은 최대 3초 이내에 완료되어야 한다.
1) 사용자는 모바일 구매를 통한 대금 결재를 위하여 결재 요청을 시작한다.
2) 시스템은 인증을 위하여 보안 인증서가 실행되고, 비밀번호를 요청한다.
3) 사용자는 인증서의 비밀번호를 입력한다.
4) 시스템은 입력된 비밀번호를 체크한다. 이때 3번 이상의 비밀번호 오류에 대하여 별도의 처리 과정을 거친다.
5) 비밀번호 확인후 사용자 인증 결과를 표시한다.</td></tr>
<tr><td>기 능 검 증 (테스트 요건)</td><td colspan="6">- 비밀번호를 올바르게 입력한 경우의 응답 시간을 체크한다.
- 비밀번호를 3번 잘못 입력하는 경우, 응답시간을 체크한다.
- 비밀번호가 암호화되어 전송되는지 점검한다.</td></tr>
<tr><td rowspan="2">관련 요구 사항 ID</td><td colspan="3">비기능 요구사항</td><td colspan="2">인터페이스</td><td rowspan="2">데이터</td><td rowspan="2">제약사항</td></tr>
<tr><td>성능</td><td>보안</td><td>품질</td><td>사용자</td><td>시스템</td></tr>
<tr><td></td><td></td><td>SER-001</td><td></td><td>UIR-007</td><td>SIR-015</td><td></td><td>COR-001</td></tr>
</table>

3.2 요구사항 분석 단계에서

요구사항 분석은 정의된 요구사항을 충분히 이해하고, 사용자에게 제공할 서비스 중심의 기능을 상세히 분할하고 구체적인 특성 정보를 정의해 가는 과정으로써, 소프트웨어 시스템에 대한 기능적 모델, 정적 모델, 그리고 동작 모델을 개발한다.

UML을 이용하는 객체지향 기반의 소프트웨어 개발 방법에서 소프트웨어의 요구사항은 Use case diagram, Class diagram, Sequence diagram, 그리고 State machine 등으로 모델링된다.

- 클래스 다이어그램에서의 클래스는 독립된 서비스를 제공하기 위하여 단일 엔티티(entity)에 의해 제공 가능한 기능들을 캡슐화한다. 클래스의 응집력(cohesion)을 높이는 것은 클래스가 다른 클래스와의 상호 작용을 감소시키는 것을 의미하기 때문에

소모전력 감소 효과를 가져 온다.

- Sequence diagram은 실행 중에 나타나는 객체간의 상호작용을 가시적으로 나타낸 모델이다. 따라서 이들은 추후 구현과정에서 메소드(혹은 함수) 호출의 순서 및 제어 구조를 담고 있다. 따라서 다른 객체와의 상호 작용 - 함수 호출의 횟수 -을 줄이는 방향으로 표현되어야 소모전력 절감효과를 가져올 수 있다.

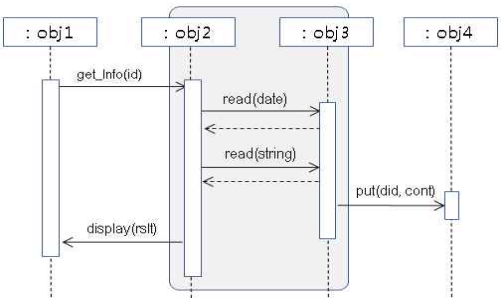

그림 1. Seq. Diagram의 간소화 예시
Fig 1. Example of SD simplification

그림 1과 같이 본래 4개의 객체(obj)가 갖는 7개의 메시지 전송에서 객체 obj2와 obj3을 통합하여 하나의 객체로 정의한다면 총 3개의 메시지 전달만으로 해당 기능의 구현이 가능해 질수 있을 것이다.

3.3 소프트웨어 설계 단계에서

소프트웨어의 설계 단계는 소모전력 절감을 가져올 수 있는 최적의 단계로 인식되고 있다. 모듈(혹은 클래스) 통합이나 모듈 분할 등을 통해 보다 적합한 소프트웨어 구조를 개발할 수 있기 때문이다.

- 패키지 다이어그램은 동일한(혹은 유사한) 기능을 갖는 클래스 또는 모듈을 통합하여 하나의 서브시스템으로 구성하고자 하는 것이다. 따라서 패키지 간의 상호 작용을 최소화 하는 것은 소모전력 절감에 있어서 매우 중요하다.
- Deployment diagram은 소프트웨어 컴포넌트와 하드웨어 구성요소와의 할당을 위한 매핑관계를 표현한다. 하드웨어 리소스간의 상호 작용은 매우 많은 전력 소모를 초래한다. 따라서 소프트웨어 컴포넌트와 하드웨어 자원, 즉 프로세서와 메모리 사용의 효율성을 고려하는 자원 매핑이 요구된다.

그림 2와 같이 초기에는 패키지, pkg A, pkg B, pkg C의 3개 패키지가 식별되었고, 이들을 각각 서로 다른 서브시스템으로 고려할 수 있으나, pkg A와 pkg B의 상관관계로부터 두 클래스 “employee”와 “history”가 합성 관계에 놓여 있음을 알 수 있다. 이는 구현 과정에서 많은 정보 접근을 유발하는 형태로 나타나기 때문에 pkg A와 pkg B를 통합함으로써, 서브시스템의 상호 작용을 줄일 수 있다.

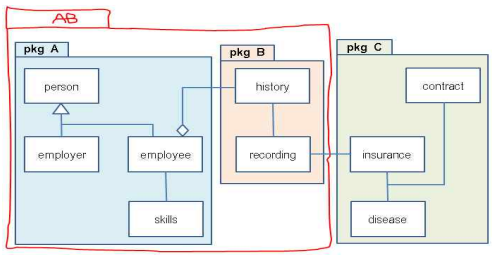

그림 2. Package Diagram의 간소화 예시
Fig 2. Example of PD simplification

3.4 소프트웨어 구현단계에서

구현단계는 설계 산출물을 기반으로 특정 프로그래밍 언어를 이용한 소스 코드 개발 및 시험 단계이다. 코드의 구성에 있어서 동일한 제어 구조를 어떠한 명령문을 사용하는 가에 명령어 처리수와 이에 따른 소모 전력이 달라진다. 예를 들면, 조건 분기문에서 동일한 로직에 대하여 if 문과 switch문을 이용하여 구현 가능하나 switch 문이 실행속도가 빠르다는 것을 알 수 있다.

위와 같이 요구사항 분석 단계에서부터 구현단계까지 많은 설계 결정 요소들이 소모전력 절감을 위한 고려사항이 될 수 있다. 이와 같은 고려사항을 기반으로 임베디드 소프트웨어의 개발 절차를 제시한다.

4. ProLoPES 개발 프로세스

3장에서 제시한 소모전력 관련 설계 요소를 고려하는 임베디드 소프트웨어의 개발절차를 제시한다. 본 연구에서 제시하는 ProLoPES는 다음과 같은 몇 가지 전제사항을 가정하여 설계 되었다.

1) 본 논문에서 제시하는 개발 프로세스는 객체지향

개념을 근간으로 제시한다.

2) 개발 과정에서의 수정이나 변경을 위한 피드백은 발생 가능하다.

3) 오픈 소스나 재사용 컴포넌트를 활용한 개발 지원을 위해 개발 절차의 변형을 제시한다.

4.1 프로세스 모델

본 논문에서 제시하는 ProLoPES는 그림 3과 같은 단계로 정의한다. 그림 3과 같이 ProLoPES 모델은 크게 요구사항에서 구현까지 5개의 영역에서 8개의 세부 단계로 구성된다.

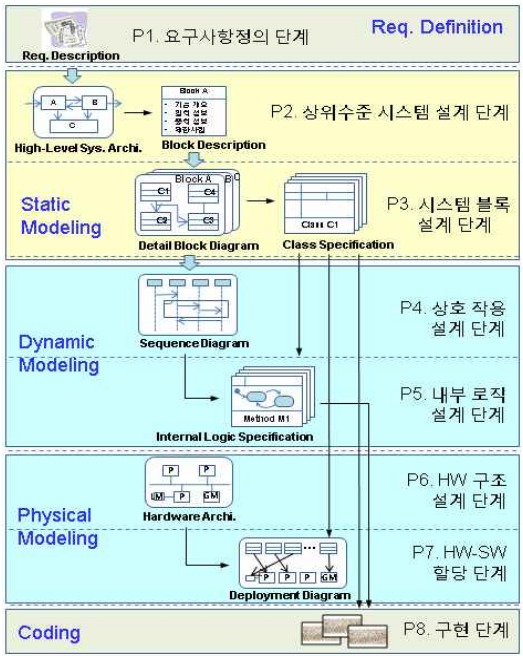

그림 3. ProLOPES 프로세스 모델
Fig 3. ProLOPES Process Model

정적 모델링 (Static modeling) 영역은 임베디드 시스템의 전체덕인 기능 중심의 정적 구조를 개발하는 활동으로 정의된 요구사항으로부터 상위 수준의 아키텍처를 생성하고, 아키텍처 구성요소들에 대한 분할 과정을 거쳐 소프트웨어시스템의 상세구조를 도출한다.

동적 모델링 (Dynamic modeling) 영역은 정적 모델을 구성하는 요소간의 상호 작용을 도출하는 단계로 구성되며, 상호작용에 근간한 세부 구성요소의 내부 행위 및 로직을 정의한다. 그리고 물리적 모델링 (Physical modeling) 영역은 하드웨어 구조설계와 하드웨어 구성요소로의 소프트웨어 구성요소 할당 과정을 수행한다.

이와 같이 정적 모델링, 동적 모델링, 그리고 물리적 모델링 과정을 거쳐 구현단계로 전이된다.

4.2 단계별 활동 및 산출물

(1) 요구사항정의단계 : 요구사항에 대한 정의는 표 1에서 보여주는 것과 같은 신 RFP 체계의 기본 양식을 준수하여 작성한다.

- 작성주체 : 개발자와 사용자
- 산출물 : 요구사항기술서
- 고려사항 : 불명확한 요구사항은 추후 보완될 수 있도록 한다.

(2) 상위수준 시스템 설계단계 : 요구사항에 주어진 기능을 중심으로 시스템 수준에서의 기능 블록을 정의하고 이들 간의 관계를 설정한다.

- 작성주체 : 개발자
- 산출물 : 상위수준 아키텍처, 블록설명서
- 고려사항 : 상위수준의 시스템 아키텍처는 시스템 컨텍스트(context) 차원에서 기능간의 독립성을 고려하여 식별한다.

상위수준 설계단계에서 나타난 기능블록은 재사용 가능한 컴포넌트에 의해 대체될 수 있다. 재사용 블록을 이용하여 개발한다면, 해당 블록에 대한 블록 설명서가 명확히 제시되어야 한다. 이후 해당 블록은 인터페이스 기반의 상호 작용이 모델링의 대상이 된다.

(3) 시스템 블록 설계단계 : 이전 단계의 블록 설명서를 기반으로 각 블록에 대한 세부 구성요소 식별과 이들 간의 관계를 도출한다.

- 작성 주체 : 개발자
- 산출물 : 상세블록 다어이그램, 클래스명세서
- 고려사항 : 블록 구성요소의 식별은 요구사항 정의를 근간으로 수행한다. 요구사항과 블록설명서의 매핑을 통해 각 블록의 상세 구성요소가 요구사항을 충분히 반영되는지 확인한다. 클래스명세서는 상세블록단위로 클래스를 식별하고, 필요한 행위를 정의한다.

상세블록 차원에서도 재사용 컴포넌트를 이용하여 개발 가능한데, 이 경우에도 상위 블록과 마찬가지로 정형화된 인터페이스 명세를 기반으로 모델링 과정을 진행한다.

(4) 상호작용 설계단계 : 상세블록 다이어그램의 구성요소, 즉 클래스를 단위로 객체간의 상호 호출 관계를 정의한다.

- 작성주체 : 개발자
- 산출물 : Sequence diagram
- 고려사항 : 상세 블록 구성요소간의 호출관계 및 제어구조를 표현할 수 있도록 작성한다.

(5) 내부로직 설계단계 : 요구사항 정의서를 기반으로 수도코드 형태의 제어로직을 개발하여 메소드 수준에서의 처리 로직을 개발한다.

- 작성주체 : 개발자
- 산출물 : 로직 명세서
- 고려사항 : 내부로직에 대한 개발은 모든 메소드에 대하여 수행하지 않고, 복잡하거나 도메인 로직을 포함하는 메소드만 작성한다. 또한 상태 기반의 제어가 포함된 메소드는 상태 기계 (State machine)을 이용하여 작성할 수 있다.

(6) HW 구조 설계단계 : 하드웨어 플랫폼에 대하여 구성요소들을 식별하고 이들 간의 연결 관계와 필요한 하드웨어 성능을 정의 한다.

- 작성주체 : 개발자
- 산출물 : 하드웨어 아키텍처
- 고려사항 : 프로세서와 메모리 등의 특성을 반영하는 별도의 하드웨어 장치 명세서를 활용할 수 있다.

(7) HW-SW 할당단계: 상세블록 다이어그램에서 도출된 구성요소들을 하드웨어 플랫폼의 구성요소에 매핑한다.

- 작성주체 : 개발자
- 산출물 : Deployment diagram
- 고려사항 : 하드웨어 구성요소의 성능과 SW 구성요소간의 상호작용이 고려된 할당이 이루어져야 한다.

(8) 구현단계 : 특정 프로그래밍 언어를 기반으로 소스코드를 생성한다.

5. 단계별 소모전력 절감 기법

그림 3에서 제시한 8단계에 대하여 개발 방법론 측면에서의 소모전력 절감이 가능한 단계는 P3. 시스템 블록 설계단계, P4. 상호작용 설계단계, P7. HW-SW 할당 단계, 그리고 P8. 구현단계이다. 각 단계별로 CCS (Cruise Control System) 예제를 이용하여 어떻게 소모전력 절감이 이루어질 수 있는지에 대하여 설명 한다.

CCS는 차량 운전자의 편의를 위하여 도로의 상태와 무관하게 정속을 유지시켜주는 차량 탑재용 임베디드 소프트웨어로써, 그림 4와 같은 상위수준의 아키텍처를 갖는다.

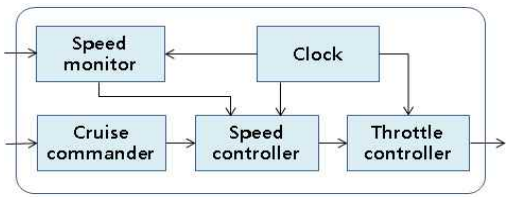

그림 4. CCS의 상위수준 아키텍처
Fig 4. High-level Architecture of CCS

그림 4의 아키텍처 상에 나타나는 기능 블록에 대하여 기능의 간단한 설명과 기능 수행을 위한 입출력 내용과 특별한 제한사항을 포함하는 블록 설명서를 작성한 후, 이를 기반으로 각 기능 블록의 내부 구성요소를 도출한다.

5.1 시스템블록 설계에서의 전력절감 기법

그림 4에 나타난 “Speed monitor” 블록에 대한 분할 (decomposition)을 통해 상세 블록 다이어그램을 작성할 수 있다. 차량의 속도를 결정하는 인자는 브레이크, 가속기, 그리고 엔진의 상태에 의해 결정된다. “Speed monitor” 블록의 분할에 의한 상세블록 다이어그램은 그림 5와 같다.

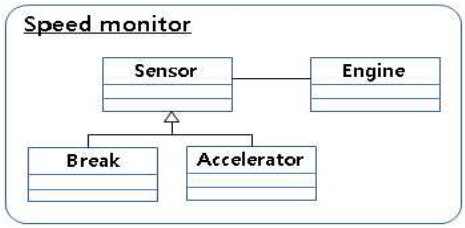

그림 5. Speed monitor 블록의 상세블록도
Fig 5. Detail BD of speed monitor block

그림 5와 같은 상세블록도에서 소모전력 감소를 위해 다음과 같은 요소를 고려할 수 있다.

(1) 불필요한 상세 블록의 제거

CCS 시스템에서 브레이크는 정속 주행을 멈추기 위한 용도로 사용할 수 있다. 그러나 가속기는 CCS 시스템에서 속도제어를 위한 역할을 수행하지 못한다. 정속 주행은 바퀴의 회전수를 기반으로 엔진을 직접 제어하는 메카니즘으로 구현되기 때문이다. 따라서 가속기 블록은 상세 블록에서 제외되어도 CCS 구현에 문제가 되지 않는다. 이와 같은 불필요한 블록의 존재는 단순히 해당 블록의 행위로 제어가 전이되었다가, 특정 조건을 만족하지 않은 상태로 다시 제어를 반환하는 알고리즘을 만들어 낸다. 이는 시스템의 성능이나 전력 소모 차원에서 부정적인 효과를 유발하게 된다.

(2) 블록간의 관계정의에서의 오버헤드 제거

블록간의 관계를 정의하는 것은 추후 블록간의 메시지 전송이나 제어 흐름과 같은 상호 작용이 있음을 의미한다. 따라서 블록간의 관계가 올바르게 정의된다는 것은 그 만큼 상호작용을 줄임으로써, 소모전력을 감소할 수 있는 기회를 갖는다.

(3) 저전력 재사용 컴포넌트 선정

기능 블록이 재사용 컴포넌트를 이용하여 개발되도록 정의되었다면, 저전력을 소모하는 컴포넌트를 선정하는 것이 중요하다. 일반적으로 재사용 컴포넌트에 대한 명세는 기능, 성능이 대부분이며, 소모 전력에 대한 명세는 쉽게 찾기 어렵다. 따라서 소모전력에 대한 명세를 갖는 재사용 컴포넌트 저장소 구축이 요구될 수 있다.

5.2 상호작용 설계에서의 전력절감 기법

그림 6에 나타난 블록 “Speed controller”는 바퀴의 회전수를 카운트하여 엔진 출력을 높일 것인가 낮출 것인가를 결정하는 역할을 수행한다. “Speed controller”의 Sequence diagram은 그림 6과 같다.

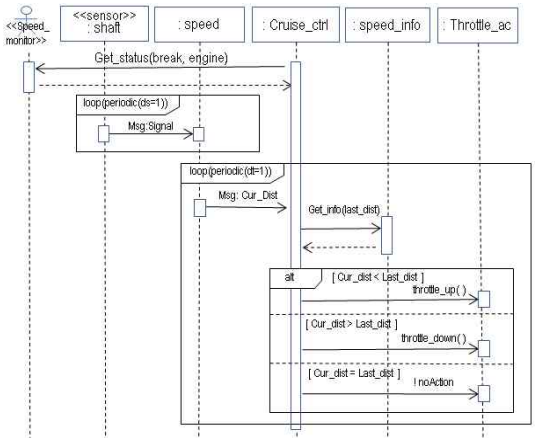

그림 6. Speed controller의 Sequence diagram
Fig 6. SD of Speed controller

그림 6의 상호작용도에서는 객체 “cruise_ctrl”은 “speed” 객체로부터 현재 바퀴의 이동거리를 기반으로 계산된 운행거리를 얻어오고 과거 단위시간당 운행거리를 객체 “speed_info”로부터 얻어와 상호 비교를 하게 된다. 비교 결과에 따라 Throttle의 밸브를 더 열어줄 지, 닫아줄 지를 결정하게 된다.

위와 같은 동작과정에서 두 객체 “crusie_ctrl”과 “speed_info”는 $\Delta t = 1$ 단위의 주기로 운행 거리를 비교하게 되며, 이는 매우 많은 객체간의 상호작용을 유발하게 된다. 따라서 이 두 객체는 소모전력과 성능 향상 측면에서 병합되어야 한다.

그러나 “cruise_ctrl”과 “speed” 객체도 주기적인 동작에 포함되어 있으나, 객체 “speed”는 객체 “shaft”와 시간을 기준으로 비동기적으로 동작하고 있기 때문에 상호 통합될 수 없다. 따라서 그림 6은 그림 7과 같이 재작성될 수 있다.

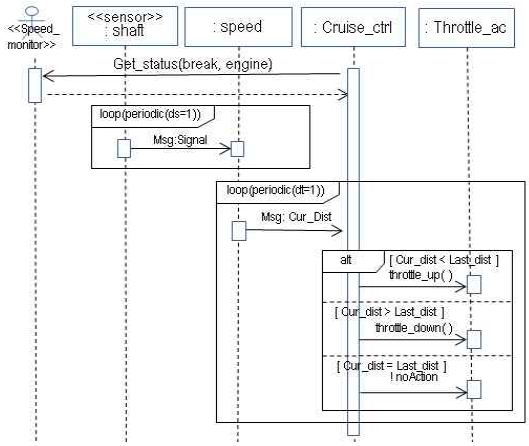

그림 7. 수정된 Speed controller의 Sequence diagram
Fig 7. Refined SD of Speed Controller

5.3 HW-SW 할당단계에서의 전력절감 기법

소프트웨어의 소모전력이 하드웨어의 동작에 의해 발생하기 때문에 하드웨어의 동작을 제어하는 소프트웨어가 어떻게 배치되고 처리되는가에 따라 소모전력 량은 매우 달라지게 된다.

하드웨어 플랫폼이 다수의 프로세서를 갖는 경우는 프로세스의 상호작용이 반드시 고려된 소프트웨어 할당이 이루어져야 한다.

일반적으로 병렬성을 높이면 실행시간이 빨라서 소모전력량이 줄어들 것으로 예측하기 쉬우나 두 개의 프로세스가 각각 피크 부하(peak load)를 가지게 되어 오히려 단일 프로세스에서 연속으로 수행하는 것보다 소모전력이 많아질 수 있다. 따라서 그림 8에서 보여주는 것과 같이 적정수준의 병렬성을 유지하고 소모전력이 낮은 방향으로 프로세스가 할당되어야 한다.

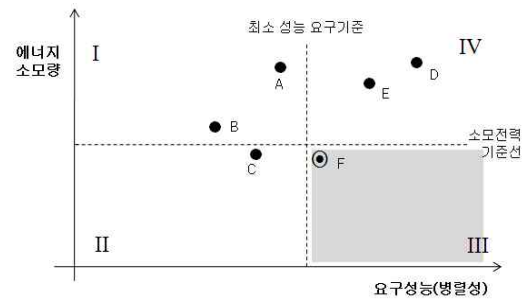

그림 8. 소모전력과 병렬성과의 관계
Fig 8. Relation of Power and Parallelism

위와 같은 특성을 고려하여 소프트웨어 블록을 하드웨어 구성요소에 할당할 때, 다음과 같은 기준을 만족하도록 할당한다.

(1) 블록간 상호작용이 많으면 동일 프로세서로

블록간 상호작용이 많다면 IPC(Interprocess communication) 수행에 의한 통신 부하가 많음을 의미한다. 즉 에너지 소모량 E에 대하여 다음의 수식이 성립된다.

$E(P1, P2) < E(P1 // P2)$

이는 프로세스 블록 P1과 P2가 병렬 수행하는 E(P1 || P2)이 소모전력이 더 많음을 의미하며, 이는 상호 작용을 통한 공유 메모리 접근이나 메시지 송수신에 많은 소모전력을 사용하기 때문이다. 따라서 두 블록을 다른 프로세스에 할당한다면 많은 성능저하와 함께 소모전력량이 증가하게 된다.

(2) 블록간 데이터 의존성이 없는 경우 병렬처리

두 블록 P1과 P2 사이에 데이터를 주고 받거나 공유하여 사용하는 데이터가 없다면 병렬처리하는 것이 효율적이다.

$E(P1 // P2)\ iff\ (D_{p1} \in P1) \cap (D_{p2} \in P2) = \varnothing$

즉 프로세스 P1의 데이터 집합 D_{p1}와 P2의 D_{p2} 간에 교집합이 공집합일 때, 이의 처리를 병렬 수행하는 것이 효율적이다. 즉 신속한 처리를 통한 수행시간의 절감을 유도하고, 이로 인하여 소모전력이 낮아질 수 있다.

(3) 데이터를 공유하는 경우 동일 프로세서로

두 프로세스 블록 P1과 P2가 데이터를 공유하는 경우, 공유 데이터에 접근하기 위한 제어 처리 시간이 증가할 수 있다. 따라서 다음과 같은 수식이 성립한다면 P1과 P2는 동일 프로세서로 할당하는 것이 유리하다.

$E(P1, P2) + E(A(LM)) < E(P1 // P2) + E(A(SM))$

즉, 지역 메모리 접근을 위해 소모하는 에너지, E(A(LM))이 공유 메모리를 접근하기 위해 소모되는 에너지 E(A(SM))보다 현저히 작다면 동일한 프로세스로

할당하여 순차적으로 처리한다.

(4) 단일 데이터 처리를 갖는 경우 병렬처리

다른 데이터와 무관하게 지역 변수만을 기반으로 수행하는 프로세스가 있다면, 해당 데이터는 로컬 메모리에 저장되게 된다. 로컬 메모리에 데이터를 저장하여 처리하는 경우 매우 빠른 수행 시간을 보장 받을 수 있다. 따라서 이러한 형태를 갖는 다수의 프로세스들은 별도의 프로세서에 할당하여 병렬 처리를 수행하도록 한다.

(5) 동일한 타입의 서로 다른 데이터에 대한 반복처리가 발생하는 경우 병렬처리

동일한 타입의 서로 다른 데이터에 대한 반복 처리라 함은 데이터 파일을 읽어 들이거나, 데이터를 압축하여 버퍼에 넣는 경우처럼 동일한 처리를 반복적으로 수행하는 루프(loop) 연산을 의미한다. 이러한 경우 반복적으로 수행하는 단위 처리에 대한 의존성이 없기 때문에 전체 반복회수를 등분하여 서로 다른 프로세서에 할당하는 경우, 신속한 처리가 가능해진다.

$$E(P1 \oplus D_n) \rightarrow E(P1 \oplus D_{1..n/2} \;//\; P2 \oplus D_{n/2..n})$$
$$iff\ D_{1..n/2} \cap D_{n/2..n} = \varnothing$$

여기서 기호 ⊕는 “Process with”의 의미이다. 즉 n개의 데이터 집합 D_n을 이등분하여 각각 서로 다른 프로세서에서 처리하도록 할당한다.

5.4 구현단계에서이 전력절감 기법

구현단계에서 소모전력을 감소하기 위해서는 알고리즘에 나타난 처리 로직을 프로그래밍 언어의 적합한 문장으로 표현하는 것이 좋다. 이러한 경우의 대표적인 예가 다중 선택문에 대한 프로그래밍 스타일 인데, if-else 문 보다는 switch - case 문을 사용하는 것이 훨씬 빠른 수행시간을 보장받을 수 있다.

또한 임베디드 소프트웨어 개발에서 많이 사용되는 포인터 기반의 연산 및 다중 포인터 사용은 프로그래밍 오류를 만들기도 쉽고, 수행시간도 올래 걸리는 단점이 있다. 따라서 포인터 연산을 수행한 후의 결과를 직접적으로 접근하도록 프로그래밍 하거나, 이중 포인터대신 단일 포인터를 두 번 사용하는 것이 더 효율적일 수 있다.

6. 절감 기법의 적용

앞의 5장에서 제시한 ProLoPES 절차를 이용하여 임베디드 소프트웨어를 개발하는 경우에 있어서의 소모전력 절감 효과에 대하여 살펴본다. 기존에 소모전력은 코드 기반의 분석이 대부분이기 때문에 개발 과정상에서 소모전력 분석 기법을 제시하는 경우는 거의 찾아보기 어렵다. 따라서 개발 기법을 기반으로 하는 소모전력 효과는 본 연구에서 제시하는 기법을 적용하는 경우, 긍정적인 결과를 얻을 수 있다. 그렇다면 그 효과는 어느 정도가 될 것인가? 예제 시스템, FIPS-STD-197, SFT(Secure File Transfer) 알고리즘의 구현을 통해 에너지 절감 효과에 대하여 실험하였다.

6.1 실험 환경 및 방법

실험을 위하여 SFT 알고리즘을 C 언어로 구현한 모듈과 SFT 스펙을 기반으로 생성한 설계 모델 개발에서 소모전력 절감 기법을 적용한 후 구현한 모듈을 상호 비교하였다.

구현된 두 모듈에 대한 컴파일러나 실행환경은 GCC 컴파일러와 리눅스 커널 2.6.2.1에서 수행하여 하였으며, C 언어로 구현된 코드의 소모전력을 분석하기 위하여 EMSIM 2.0을 사용하였다.

6.2 실험 결과

실험결과는 두 가지 측면에서 분석하였는데, 첫 번째는 실행시간의 비교이다. 실행시간의 비교는 일반적으로 빠른 실행 시간을 갖는 모듈이 소모전력을 적게 사용할 수 있는 확률이 높기 때문이다.

표 2. 태스크 실행 시간(second) 비교
Table 2. Comparison of Task Execution (in sec.)

소모전력 절감기법	입력 파일의 크기		
	200K	400K	600K
적용전	0.415	0.743	1.083
적용후	0.381	0.704	0.889
편차(%)	8.2	5.3	17.9

표 2로부터 대체적으로 수행시간이 소모전력 절감기법의 적용 전에 비하여 적용 후가 최대 18%정도 절감하는 결과를 얻었다.

표 3. SFT 시스템의 태스크 소모전력 비교
Table 3. Comparison of Power Consumption

기법 적용	입력 크기에 따른 소모전력 (mJ)	
	400K	600K
적용전	12150.1	18246.4
적용후	12147.1	18220.4
편차	3.0	26

표 3에서와 같이 소모전력 분석의 실험 결과를 보면 절감 기법의 적용으로부터, 입력 데이터의 크기가 600K에 해당되는 경우 26%정도의 소모전력 절감 효과가 있는 것으로 확인 되었다.

7. 결론 및 향후 연구내용

본 논문에서는 임베디드 소프트웨어 개발을 위한 소모전력 감소 기법에 대하여 제시하였다. 특히 임베디드 소프트웨어 개발을 위한 실용적 개발 모델로 ProLoPES를 제시하였고, 제시한 모델의 각 단계별로 소모전력 절감 기법의 무엇이 적용될 수 있는지를 설명하였다.

본 논문이 지속적인 연구를 통해 많은 보완사항을 제시해야 하지만, 소모전력 절감이 코드가 생성된 이후에 이루어질 수 있다는 근본적인 사고방식에서 벗어나 개발 과정, 다시 말해서 분석과 설계 과정에서도 소모 전력을 절감할 수 있는 기법을 사전에 적용한다는 것은 매우 의미 있는 연구라고 판단된다. 즉 코드가 생성된 후에 소모전력이 만족되지 않는다면, 프로그램을 수정하여 다시 로직을 정립해야 하지만, 설계 단계에 불필요한 소모 전력을 감소시킨다면 추후 코드 생성시에도 소모전력에 대한 요구사항을 만족시킬 확률이 높아지기 때문이다.

본 연구에 이어 추후 연구되어야 할 사항은 제시된 소모전력 절감 기반의 프로세스가 보다 상세하게 정의되고, 또한 이를 지원하기 위한 개발 방법론에 대한 프레임워크가 지원 도구를 포함하여 정립되어야 할 것이다.

참 고 문 헌

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

[2] Talarico, C., et. al., “A New Framework for Power Estimation of Embedded Systems,” IEEE Computer, Feb., 71-78, 2005

[2] Brandolese, C., et al., “An Instruction-level Functionality-based Energy Estimation Model for 32-bits Microprocessors,” Proceedings of ACM/ IEEE DAC 2000, pp.346-351, 2000

[4] Chang, N., Kim, K.H., and Lee, H.G., “Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI,” Proceedings of the International Symposium on Low Power Electronics and Design, pp.185-190. 2000

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

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

[7] Lee, M.T., et. al., “Power Analysis and Minimization Techniques for Embedded DSP Software,” IEEE Transactions on VLSI Systems, Vol.5, No.1, 123-135, 1997

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

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

[10] Brooks, D., Tiwari, V., and Martonosi, M., “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,” Proceedings of International Symposium on Computer Architecture, pp.83-94, 2000

[11] Jun, H., et. al., “Modeling and Analysis of Power Consumption for Component-Based Embedded Software,” Proceedings of EUC Workshop 2006, pp.795-804, 2006

[12] Kawabe, N., et al., "Function-level Power Estimation Methodology for Microprocessors," Proceedings of the DAC 2000, pp. 810-813, 2000

[13] Tan, T.K., et. al., “High-Level Energy Macromodeling of Embedded Software,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 9, 605-610, 2003

[14] Muttreja, A., et. al., “Hybrid Simulation for Energy Estimation of Embedded Software,” IEEE Transaction on

Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No. 10, 1843-1854, 2007

[15] Kim,J.P., Kim, D.H., and Hong, J.E., "Estimating Power Consumption of Mobile Embedded Software Based on Behavioral Model," Proceedings of ICCE 2010, pp. 105-106, 2010

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

[17] OMG Unified Modeling Language: Superstructure, V2.1.2. < http://www.omg.org>, 2007

[18] ESUML, ESUML: Embedded Software Modeling using UML 2.x". <http://selab.cbnu.ac.kr/ projects/esuml/index. html/>, 2009

[19] Jeon, S., Hong, J., Song, I., and Bae, D., "Developing platform specific model for MPSoC architecture from UML-based embedded software models," Journal of Systems and Software, Vol. 82, 1695-1708, 2009

저 자 소 개

김 종 필(Jong-Phil Kim) [학생회원]

- 2006년 2월 : 충북대학교 컴퓨터과학과 (이학사)
- 2008년 2월 : 충북대학교 컴퓨터과학과 (공학석사)
- 2008년 3월 ~ 현재 : 충북대학교 컴퓨터과학과 박사과정

<관심분야> : 소프트웨어공학, 서비스지향 아키텍처, 소프트웨어 테스트

김 두 환(Doo-Hwan Kim) [학생회원]

- 2007년 8월 : 충북대학교 컴퓨터과학과 (이학사)
- 2009년 8월 : 충북대학교 컴퓨터과학과 (공학석사)
- 2010년 3월 ~ 현재 : 충북대학교 컴퓨터과학과 박사과정

<관심분야> : 소프트웨어공학, 소프트웨어 소모전력분석, 소프트웨어 재사용

홍 장 의(Jang-Eui Hong) [정회원]

- 2001년 2월 : 한국과학기술원 전산학과 (공학박사)
- 2001년 1월 ~ 2002년 9월 : 국방과학연구소 선임연구원
- 2002년 10월 ~ 2004년 8월 : (주) 솔루션링크 기술연구소장
- 2004년 9월 ~ 현재 : 충북대학교 소프트웨어학과 교수

<관심분야> : 소프트웨어공학, 소프트웨어 품질, 소프트웨어 프로세스