ARM 기반의 실시간 내장형 소프트웨어를 위한 성능분석 도구의 설계

*황요섭°, *안성용, *이정아, **심재홍
*조선대학교 컴퓨터공학부,**조선대학교 인터넷소프트웨어 공학부
E-mail :bluewind@stmail.chosun.ac.kr°, {dis, jalee, jhshim}@chosun.ac.kr

Design of the Performance Analysis Tool for ARM-based Real Time Embedded Software

*Yo-seop Hwang°, *Seong-yong Ahn, *Jeong-a Lee, **Jae-hong Shim
*Dept. of Computer Engineering, **Dept. of Internet Software Engineering, Chosun Univ

요 약

내장형 컴퓨터 시스템은 특정 기능을 수행하기 위해 소프트웨어와 이를 구동시키기 위한 프로세서로 구성되어 있다. 이러한 시스템의 대부분은 실시간 제약들을 만족해야 한다. 실시간 제약들을 만족하는 애플리케이션을 빠른 시간안에 구현하기 위해서는 제작 전 성능을 예측하는 도구가 필수적이다. 본 논문에서는 현재 내장형 시스템 플랫폼으로 널리 활용되고 있는 ARM 기반의 내장형 애플리케이션의 극단적인(최적, 최악) 경우의 수행 시간 경계를 예측하는 문제를 연구하였다. ARM 기반의 내장형 시스템의 수행시간의 경계를 예측하기 위하여 기존의 실시간 내장형 소프트웨어의 성능 예측 도구인 "Cinderella"의 기본 프레임웍을 ARM 프로세서를 지원하도록 확장하여 성능분석도구를 설계하였다.

1. 서 론

마이크로프로세서와 집적회로의 급속한 발전으로 인해 내장형 시스템의 응용범위는 빠르게 확장되고 있다. 이러한 내장형 시스템의 시장은 매년 급속도로 성장하여 90년대 초기에는 천억 달러 규모의 시장을 갖게 되었으며 평균 10% 이상의 비율로 성장하고 있다[1]. 이 분야에는 다른 어떤 분야보다 생산성 향상을 위한 도구들에 대한 요구가 강한 실정이다. 특히 멀티미디어와 휴대용 무선 통신에 대한 관심이 높아지면서 음성이나 화상을 처리하는 디지털 신호 처리 기술이 핵심이 되는 내장형 시스템의 반도체 구현(SoC)이 필요하게 되었다. 내장형 시스템의 복잡도가 증가하고 반면에 기술 경쟁이 치열하여 가는 시점에서 이러한 시스템을 빠른 시간 안에 저렴한 개발비와 제작비를 이용하여 구현하는 것이 내장형 시스템 설계의 궁극적인 목표이다.

내장형 시스템은 특정 기능을 수행하기 위해 소프트웨어와 이를 구동시키기 위한 하드웨어로 구성되어 있다. 내장형 시스템을 설계하는데 있어서 하드웨어의 논리회로보다는 프로세서에서 구동되는 프로그램의 코드가 중요한 요소로 대두되었다. 이와 같은 변화는 2가지의 이유 때문인데. 첫째는 반도체 회사의 제작라인의 설비 비용이 증가하기 때문이고, 두 번째는 시장에 내놓을 시기를 줄이는 것뿐만 아니라 작업 계획을 조정할 수 있어야 하기 때문이다. 이처럼 내장형 시스템을 설계하는데 있어서 내장형 소프트웨어는 중요한 요소이며 소프트웨어의 성능을 평가하는 기본 척도이다. 소프트웨어의 성능을 분석하기 위해서는 프로그램의 극단적인(최적, 최악) 경우의 수행시간을 측정해야 한다. 내장형 소프트웨어의 경우에는 이의 성능을 예측하기 위해 소프트웨어 컴포넌트의 성능 수행시간을 정확히 측정하기 위한 고성능 시간 분석에 관한 연구들이 진행되고 있다[2]. 이러한 연구들은 명령어 수준에서 성능을 분석하는 경우가 많다[3]. 다중 처리가 가능한 내장형 시스템의 성능 분석을 위한 연구는 여러 개의 작업들이 각각 다른 프로세서에서 분산 처리되는 경우에 대하여도 성능분석을 가능하게 한다[4].

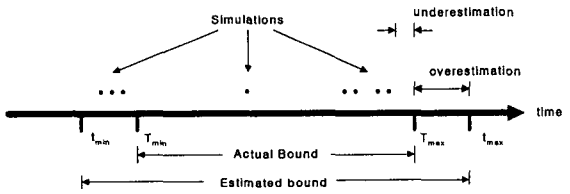
본 논문은 내장형 소프트웨어를 분석하기 위한 성능 측정 도구로서 'Cinderella'를 이용하는 방법을 제시하고자 한다[5]. 내장형 시스템은 그것을 제공하는 벤더들이 많고 하드웨어 제품도 다양해서 기존의 성능분석 도구는 어느 한가지 제품에 제한을 가지고 있다. 그러나 'Cinderella'는 다른 프로세서 하드웨어 모델을 쉽게 재구성 할 수 있다는 장점이 있어서 새로운 제품들을 추가할 수 있는 확장성이 용이하다. 뿐만 아니라 복잡한 프로그램을 분석하기 위해 사용자가 직접 조건들을 추가할 수 있다. 또한 프로세서를 모델링하여 캐쉬 메모리나 파이프라인을 지원할 수도 있다. 현재 'Cinderella' 도구는 COFF(Common object file format) 실행 파일을 지원하며 I960과 M68K 프로세서를 지원한다. 하지만 현재 사용되는 대표적인 이진 실행 파일 포맷인 ELF(Executable and Linkable Format)는 지원되지 않는다. 현재 내장형 시스템에 가장 많이 사용되는 프로세서로 ARM 프로세서가 있다. ARM 프로세서의 가장 큰 장점은 저 전력 소모와 저렴한 가격이다. 우리는 현재 내장형 시스템에서 가장 널리 사용되는 ARM 프로세서를 지원하기 위해서 'Cinderella' 도구를 확장 설계하였다. 'Cinderella'에 ARM머신을 지원하기 위해 ELF 목적파일과 ARM 명령어, ARM 머신을 위한 모듈을 설계하였다.

본 논문의 구성은 2장에서 소프트웨어 성능분석 도구인 'Cinderella'에 대해 서술하고, 3장에서 ARM을 지원하기 위한 애플리케이션의 흐름 분석을, 4장에서 ARM기반 소프트웨어의 성능 예측 방법을, 5장에서는 결론 및 향후 연구과제를 제시한다.

2. 소프트웨어 성능 분석도구(Cinderella)

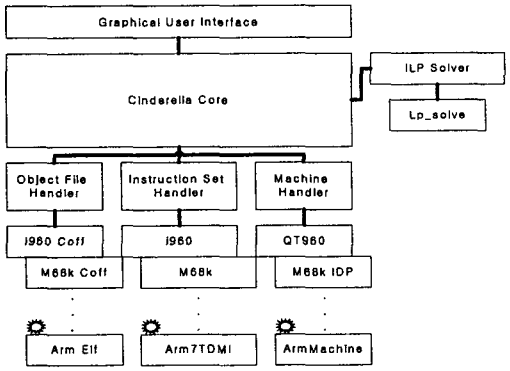
'Cinderella'는 ILP(Integer Linear Programming)기반으로 되어있는 내장형 소프트웨어 성능분석도구이다. 'Cinderella'는 정해진 프로세서에서 프로그램 수행 시간의 적합한 영역을 실제 수행시간에 최대한 근접한 영역을 찾을 수 있게 한다. 프로그램의 수행 시간의 최소값과 최대값을 T_{min}과 T_{max}라고 했을 때 프로그램의 실제 수행시간 영역(actual bound)은 [T_{min}, T_{max}]사이에 존재한다. 그리고 프로그램 수행시간을 어림할 수 있는 최적조건(best case)과 최악조건(worst case)을 t_{min}, t_{max}라고 했을 때, 프로그램의 어림 수행 시간 영역(estimated bound)은 [t_{min}, t_{max}]사이이다(그림 1).

그림 1에서 보는 것처럼 실제 수행시간은 어림 수행시간 영역에 반드시 들어가야 한다.



(그림 1) 실제 수행 시간 영역과 어림 수행 시간 영역

어림 수행 시간 영역을 결정하기 위해 즉, WCET(Worst Case Execution Time)를 구하는 문제를 해결하기 위해서 'Cinderella'는 2가지를 고려하였다. 첫 번째는 프로그램 흐름을 분석하는 것이고 두 번째는 마이크로 아키텍처를 모델링하는 것이다.



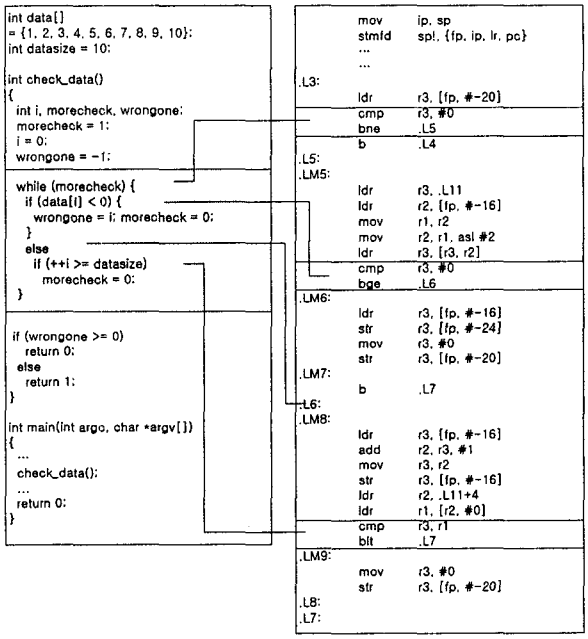
(그림 2) ARM 머신을 지원하기 위한 프레임웍

그림 2에서 보는 것처럼 ARM 머신을 지원하기 위해 ARM-ELF Object file 모듈과 ARM 프로세서를 지원하기 위해 ARM7TDMI Instruction set 모듈, ARM을 지원하기 위한 테스트 보드로 ARM-Machine을 모듈 부분을 추가함으로써 가능하다.

3. ARM 기반 애플리케이션의 흐름 분석

프로그램 흐름 분석은 구조적 제약(Structural Constraints)과 기능적 제약(Functionality Constraints)이라고 정의되는 두 가지 형태의 선형 제약(linear Constraints)을 이용하여 프로그램 흐름을 표현한다.

구조적 제약은 각각의 노드들에 대한 제어 흐름 제약을 기반으로 구성되어지는 CFG(Control Flow Graph)로부터 자동적으로 추출되어질 수 있다. 기능적 제약은 반복 구간의 수행 빈도나 프로그램의 경로 정보를 사용자가 직접 입력함으로써 명시할 수 있다. 그림 3은 C 소스 코드와 이를 ARM으로 컴파일 했을 때의 어셈블러 코드를 보여주고 있다.



```
int data[]
= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int datasize = 10;

int check_data()
{
  int i, morecheck, wrongone;
  morecheck = 1;
  i = 0;
  wrongone = -1;

  while (morecheck) {
    if (data[i] < 0) {
      wrongone = i; morecheck = 0;
    }
    else
      if (++i >= datasize)
        morecheck = 0;
  }

  if (wrongone >= 0)
    return 0;
  else
    return 1;
}

int main(int argc, char *argv[])
{
  ...
  check_data();
  ...
  return 0;
}
```

```
                mov     ip, sp
                stmfd   sp!, {fp, ip, lr, pc}
                ...
                ...
.L3:
                ldr     r3, [fp, #-20]
                cmp     r3, #0
                bne     .L5
                b       .L4
.L5:
.LM5:
                ldr     r3, .L11
                ldr     r2, [fp, #-16]
                mov     r1, r2
                mov     r2, r1, asl #2
                ldr     r3, [r3, r2]
                cmp     r3, #0
                bge     .L6
.LM6:
                ldr     r3, [fp, #-16]
                str     r3, [fp, #-24]
                mov     r3, #0
                str     r3, [fp, #-20]
.LM7:
                b       .L7
.L6:
.LM8:
                ldr     r3, [fp, #-16]
                add     r2, r3, #1
                mov     r3, r2
                str     r3, [fp, #-16]
                ldr     r2, .L11+4
                ldr     r1, [r2, #0]
                cmp     r3, r1
                blt     .L7
.LM9:
                mov     r3, #0
                str     r3, [fp, #-20]
.L8:
.L7:
```

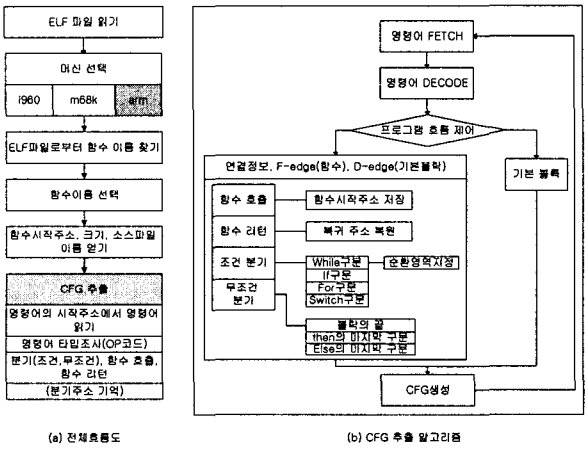
(그림 3) C 소스 코드와 ARM 어셈블러

CFG는 명령어를 읽어서 파싱했을 때 어셈블러 코드에서 조건 분기(blt, bge, bne, etc.)와 무조건 분기(b) 명령어를 파싱함으로써 분기 명령어가 나오기 전까지의 명령어를 기본 블록으로 구분하며, 분기하는 주소를 연결정보를 표시하여 프로그램의 제어흐름을 표현한다. 각 기본 블록의 수행 시간은 ARM 어셈블러 명령어들의 수행 사이클 테이블을 참고하여 구할 수 있다.

4. ARM기반 소프트웨어의 성능 예측 방법

내장형 시스템 설계에 있어서 소프트웨어의 성능 예측은 극단적인(최적, 최악) 경우의 수행 시간을 찾기 위한 프로그램의 정적인 분석을 통해 이루어진다.

분석을 위한 기본 정보로서 프로그램의 제어 흐름, 프로그램의 경로 정보, 반복 구간 정보, 각 명령어의 수행 시간, 마이크로 아키텍쳐에 관한 정보(캐쉬 분석을 위한 시스템 캐쉬 구조나 명령어 주소 등에 관한 정보)들이 필요하다. 이러한 정보들은 프로그램으로부터 제어 흐름과 경로정보를 알 수 있으며 하드웨어 모델로부터 명령어 시간과 캐쉬 구조 등에 관한 정보들을 얻을 수 있다. 또한 제어 흐름 정보는 컴파일러에 의해 코드 변환을 통해 추출할 수 있으며 경로 정보는 소스 프로그램에서 구할 수 있다. 시간 분석은 실행 코드 단계에서 수행된다. 이 단계는 어셈블리 명령어와 명령어 주소가 고정되어 있기 때문이다. 이와 같은 정보들을 이용하여 프로그램은 기본블록과 그것들이 연결된 컨트롤 플로우 그래프로(CFG)로 모델링 된다. 각 기본 블록의 수행 시간이 주어지면 프로그램의 가능한 경로중 가장 긴 경로가 바로 최악시간이며 가장 짧은 경로가 최적시간이다. 그렇지만 각 블록간에 제어 흐름이 영향을 주기 때문에 가능한 경로를 찾는 것은 매우 어렵다. 이와 같은 문제점을 해결하기 위해 정수 선형 프로그래밍(ILP) 방법을 이용하여 해를 구한다. 프로그램의 전체 수행시간은 선형표현으로 표시되어질 수 있다. 또한 제어 흐름과 논리 흐름의 모든 제약사항들은 선형 표현식으로 사용할 수 있으며 이 문제는 정수 선형 프로그래밍(ILP)으로 표현된다. ILP에서의 최적의 해는 프로그램의 어림수행시간과 같다.



(그림 4) ARM 기반의 소프트웨어 성능분석을 위한 전체흐름도와 CFG추출 알고리즘

그림 4에서 보는바와 같이 이진 실행 파일을 읽을 때 파일 포맷과 함수 이름들에 관한 정보를 읽어온다. 또한 하드웨어 플랫폼을 선택할 때 ARM을 지원하기 위한 명령어 셋 구조를 결정하고 하드웨어 모듈을 검색하게 된다. 하드웨어 모듈은 각 명령어에 대한 수행시간과 마이크로 아키텍쳐에 관한 구조를 가지고 있다. 수행시간을 측정하기 위한 함수 이름을 선택하면 CFG를 생성하며 for구문이나 while구문과 같은 반복 구간이 존재하는 지를 찾게 된다. 만약 반복 구간이 있다면 사용자가 반복 구간의 횟수를 지정해줌으로써 수행시간을 계산할 수 있도록 한다. 각 기본 블록들의 시간은 명령어 셋과 머신을 참고한다. 또한 사용자의 입력을 통해 여러 가지 제약 조건들을 생성한다. 각 기본 블록과 제약조건은 ILP(Integer Linear Programming)문제로 변환되어 프로그램의 어림수행시간 즉, WCET(Worst Case Execution Time)을 구할 수 있다.

5. 결론 및 향후 연구방향

본 논문에서는 내장형 소프트웨어의 성능분석을 위하여 현재 가장 널리 사용되고 있는 내장형 프로세서인 ARM머신을 지원하기 위해, 주어진 프로세서에서 수행되는 프로그램의 극단적인(최적, 최악) 경우의 수행 시간을 측정할 수 있는 'Cinderella'를 ARM 머신에서도 사용할 수 있도록 확장 설계하였다.

구현된 도구는 ARM 이진 실행 파일을 읽어 들여 CFG를 생성하기 위해 이진 파일을 파싱하여 각 명령어의 제어 흐름에 따라 기본 블록과 연결정보를 사용하여 표시한다. 또한 각 기본 블록들은 ARM 명령어의 수행 사이클 테이블을 참고하여 계산되어질 수 있다. 프로그램의 여러 가지 제약 사항들은 정수형 선형 공식들(ILP)로 변환되고 ILP 기술을 이용하여 전체 수행 시간을 계산하는 함수를 최대화함으로써 극단적인(최적, 최악) 경우의 수행 시간의 정적인 분석을 통하여 실제 수행시간 영역에 최대한 근접하게 추정할 수 있게 한다.

더욱 자세한 시간 분석을 위해서는 세부적인 마이크로 아키텍처를 모델링하여 캐쉬나 파이프라인까지도 지원할 수 있도록 해야 한다. 본 논문에서 설계된 ARM 머신을 지원하는 'Cinderella'는 ARM 프로세서를 지원하는 머신들의 내장형 소프트웨어의 성능을 미리 예측 및 측정함으로써 ARM 기반 시스템 제약사항들을 만족하는 하드웨어 및 소프트웨어 구성을 시스템 제작전에 파악함으로써 전체적인 내장형 시스템의 설계시간 및 비용을 현저하게 줄여줄 것으로 기대된다.

6. 참고문헌

[1] Rajesh Kumar Gupta, "Co-Synthesis of Hardware and Software for Digital Embedded Systems", PH.D dissertion, 1993, Stanford University.

[2] Kaiyu Chen, Sharad Malik, David I. August, "Retargetable Static Timing Analysis for Embedded Software," Proceedings of the International Symposium on System Synthesis (ISSS), October, 2001

[3] George Hadjiyiannis, Pierro Russo, Srinivas Devadas, "A Methodology for Accurate Performance Evaluation in Architecture Exploration" Design Automation Conference 1999

[4] Ti-Yen Yen, Wayne Wolf, "Performance Estimation for Real-Time Distributed Embedded Systems," IEEE Transactions onn Parallel and Distributed Systems, Vol. 9, 11, November 1998

[5] Yau-Tsun Steven Li, Sharad Malik, "Performance analysis of Real-Time Embedded Software," Kluwer Academic Publishers, 1999