Timing-C 언어에서의 시간 분석 도구 설계

○최 영 준*, 서 진 철*, 이 준 동**, 원 유 헌*
홍익대학교 컴퓨터공학과*
원주대학 전산정보처리과**

Design of Timing Analysis Tool for Timing-C Language

○Young-Jun Choi*, Jin-Cheol Seo*, Jun-Dong Lee**, Yoo-Hun Won*
Department of Computer Engineering, Hongik University*
Department of Computer Information Processing, Wonju National College**

요 약

실시간 시스템에서 프로그램의 실행시간을 예측하는 것은 중요한 일이다. 기존의 언어에서는 실행시간은 예측하기에 힘든 요소들이 있다. Timing-C는 이러한 요소를 제거하고 사용자로부터 시간 제약을 입력받을 수 있도록 하였다. Timing-C언어를 이용하여 실시간 프로그래밍을 하기 위해 작성한 프로그램이 시간 제약을 준수하고 있는지 알기 위해 시간 분석 도구가 필요하다 시간 분석 도구는 작성된 프로그램의 실행시간을 계산하여 사용자에게 예측된 결과를 알려주는 도구이다 개발자는 이러한 도구를 이용하여 작성하고있는 프로그램의 수행시간을 더욱 정확하게 예측할 수 있다.

1. 서론

실시간 시스템이란 계산의 논리적인 결과와 그 결과가 산출된 시간을 모두 중시하는 시스템이다. 실시간 응용프로그램이란 실시간 시스템의 개념을 가지고있는 응용프로그램으로서 사용자의 기능적 요구사항을 만족함과 동시에 시간적 요구사항을 만족시키는 프로그램이다 기능적 요구사항이란 주어진 입력을 오류 없이 수행하여 정확한 결과를 출력하는 것을 의미하고, 시간적 요구사항은 프로그램의 실행시간이 사용자가 기대하는 시간 이내에 속하는 것을 의미한다 사용자가 원하는 이러한 시간적 요구 사항을 시간 제약이라 한다

실시간 응용 프로그램을 작성할 때에 프로그램이 시간제약을 지키는 가를 알기 위해서는 프로그램의 실행시간을 알아야 한다. 프로그램의 실행시간을 알아내는 방법은 크게 세 가지로 직접 실행시간을 측정하는 법, 시뮬레이션 하는 법, 소스 코드를 분석하는 법 등이 있다. 직접 실행 시간을 측정할 때는 시스템의 시계를 이용한다 실제 측정하기 위해서는 프로그램 개발단계가 아닌 완료 단계에서 측정해야 하기 때문에 개발 싸이클이 길어져 개발 시간이 길어진다. 또 프로그램의 실행시간이 길 경우 측정이 힘들며 프로그램에 가능한 모든 경로를 전부 수행해 보아야 한다.

시뮬레이션은 명세나 모델링 언어를 이용한다. 시뮬레이션을 이용할 때는 테스트 데이터를 정확하게 작성해 주어야 한다 소스 코드 분석법은 소스코드를 분석하여 각 코드마다 동작할 시간을 예측하는 방법이다.

소스코드 분석법을 이용할 때 가장 큰 문제는 분석할 코드는 고급언어이고 실제로 기계에서 이용하는 언어는 기계어라는 점이다. 프로그램의 동작시간은 기계어 코드를 분석해서 얻을 수 있지만 실제로 프로그램이 작성되고 시간 제약을 명세하는 곳은 고급 언어에서이다. 따라서 고급언어와 기계어 사이의 관계를 알 수 있게 해주는 도구가 필요하다. 이런 일을 하여 소스코드 분석을 하도록 하는 도구가 시간 분석 도구(timing analysis tool)이다. 시간 분석 도구는 고급 언어에서 사용자가 실행시간을 알고자하는 코드 부분에 해당하는 기계어 부분을 찾아 사용자 - 이 경우 프로그램의 개발자 - 에게 해당 기계어 코드와 예측된 실행시간 등을 알려 주는 역할을 한다.

2장에서는 본 논문에서 제안한 시간분석 도구가 이용하는 실시간 언어인 Timing-C에 대하여 살펴보고, 3장에서는 Timing-C 컴파일러와 실행시간 분석기를 설계한다 마지막으로 4장에서 향후 연구 방법을 언급하고 결론을 맺는다

2. Timing-C언어의 정의

현재 사용중인 대부분의 언어는 시간 분석을 고려하여 설계한 것이 아니다. 그렇기 때문에 goto문이나, 재귀호출, 동적 기억장소 할당같은 시간 분석을 불가능하게 만드는 요소들을 많이 포함하고 있다 그 외에 시간 분석을 고려하여 설계한 언어로 Real-Time Euclid[Kli86], RTC[Wol91] TCEL[Hon93],

본 연구는 학술진흥재단의 자유공모과제(과제번호: 972030106)로 수행중

RTL/MCS[백97] 등이 있다

이러한 언어들이 널리 사용되고 있지 않은 이유는 아직 실시간 응용들의 요구사항을 모두 충족시키지 못하는 것도 한 요인이지만, 기존의 언어 사용자들이 새로운 언어를 배워서 이용해야 한다는 문제점도 있다 한가지 해결책은 기존의 언어에 약간의 수정으로 실시간 응용을 충족시키는 것이다 더욱이 본 논문의 목적은 실시간 응용의 만족보다는 최악실행시간의 계산이므로 이를 위하여 Timing-C를 정의한다

Timing-C의 설계 목적은 C프로그램의 최악실행시간을 분석 가능하게 하는 것이다 기존 C의 사용자들이 쉽게 이용하기 위해서는 가능하면 C의 구문의 변화 없이 제약조건을 첨가하는 것이다.

Timing-C는 C의 부분집합으로 GOTO문이 없고, 재귀적 호출과 동적 기억장소 할당이 금지된다 또한, 프로그램 단위는 하나의 입구와 출구만을 가진다 프로그램의 각 영역의 시작에서는 주석 형태로 제약조건을 명시할 수 있다 제약조건은 시간과 반복횟수 2가지가 제공되며 형태는 이래와 같다

- 반복에 관한 제약조건
  → /** Count(Max_Count) **/
- 시간에 관한 제약조건
  → /** Time(Max_Time) **/

※ Max_Count 최대 반복 횟수,
Max_Time 최대 반복 시간

제약조건이 사용된 예는 [그림1]과 같다

제약조건은 while 문장의 최대 반복 수가 5번임을 의미한다 [그림 1]의 예제에서와 같이 제약조건은 1줄로 나타나고, 이 때 이 제약조건의 범위는 while 문장의 범위를 의미한다. 즉, 3)~7)번 줄까지의 영역을 의미한다. 이와 같이 같은 Count()명령이라도 나타나는 위치에 따라 다른 영역 범위를 갖게 된다 Time()도 같은 형식으로 정의되어지는데 이 때 시간의 단위를 명시하여 주어야 한다 시간의 단위는 초를 의미하는 s와 밀리 초를 의미하는 m이 사용된다 즉, 위의 3)과 같은 경우 /** Time(15m) **/으로 표현 가능하다.

```
1)  int gcd(int x, int y)
2)  {
3)        while ( x != y)   /** Count(5) **/
4)          if (x > y)
5)            x = x - y;
6)          else
7)       y = y - x;
8)        return(x);
9)  }
```

[그림 1] Timing-C의 예제

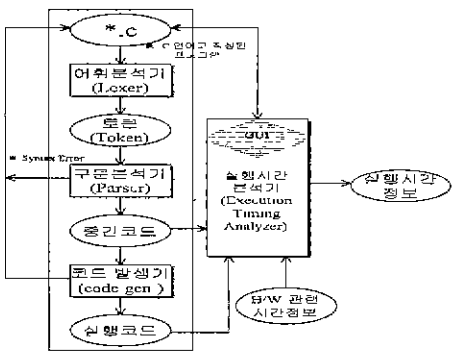
Timing-C의 목적은 가능한 한 기존의 C문법을 그대로 수용하여 프로그래머로 하여금 부담을 느끼지 않게 하면서 최악실행시간 분석을 가능하게 하는데 있다

3. 시간 분석 도구 설계

3.1 Timing-C 컴파일러의 설계

Timing-C 언어는 시간제약 사항을 추가한 C언어의 부분집합으로 간주할 수 있다. 실행시간 정보를 구하기 위한 Timing-C 컴파일러는 [그림 2]와 같이 구성된다

[그림 2]에서 강조된 사각형 안의 구조는 일반적인 컴파일러의 구조와 같다. 주석 형태로 제공된 시간 제한 구조는 구문 트리에 반영되어진다. 구문 트리에는 원시코드에서의 라인 번호와 시간 제한 구문들, 그리고 나중에 실행시간 분석기의 결과를 표시한다 구문 트리가 시간에 관한 정보를 가지고 있게 되면 이를 시간 트리라고 부른다 본 논문에서는 구문 트리라는 용어와 시간 트리라는 용어를 같은 것으로 간주한다



[그림 2] Timing-C 컴파일러의 구조

실행시간 분석기는 컴파일러가 생성하는 여러 가지 정보를 이용하여 최악실행시간을 계산한다. 실행시간 분석기가 필요로 하는 정보에는 시간 트리, 제어 흐름 그래프, 어셈블리 명령어, 그리고 사용자의 제약 조건 등이 있다

하드웨어 관련 시간 정보는 프로세서 의존적인 정보기 제공된다 본 Timing -C의 구현에는 캐쉬와 파이프라인이 없는 8051프로세서를 이용하였으므로 프로세서 의존적인 정보로는 매뉴얼 상의 어셈블리 명령어의 실행시간만이 테이블로 제공된다

또한, 실행시간 분석기는 사용자에게 프로그램의 최악실행시간을 제공하기 위한 인터페이스를 제공한다. 사용자는 인터페이스를 통하여 원시언어 상에서 함수나 프로그램의 일부 영역의 실행시간을 얻을 수 있다

3.2 실행시간 분석기

실행시간 분석 시에 고려할 사항은 크게 하드웨이 종속적인 부분과 하드웨어 독립적인 부분으로 나뉜다.

하드웨어 종속적인 부분은 중앙처리 장치의 주기[Rod95], 메모리 접근시간, 자원의 사용 방법, 캐쉬와 파이프라인 등이 속하며, 캐쉬와 파이프라인 등의 저급 언어 단계의 분석(low level analysis)이다. 캐쉬와 파이프라인이 최악실행시간에 미치는 영향과 과대 평가를 해결 방법은 다음의 논문들에 걸 나와있다 [Mue94][Ste95] [Whi94][Lim94]

하드웨어 독립적인 부분으로는 ILP(Integer Linear Programming)[Pus91], 기호실행(symbolic execution)방법

[Alt97]과 가상번역(Abstract Interpretation)방법[Bou93]등의 프로그램 실행 경로와 연관된 부분을 고려해야한다.

실행시간 분석기는 컴파일러에서 생성된 제어흐름 그래프와 하드웨어 관련 시간정보를 이용하여 프로그램의 최악 실행 시간을 계산한다. 하드웨어 관련 시간정보는 사용자 매뉴얼에 나와있는 명령어의 실행시간을 참조하여 만들어진다. 이와 같은 시간정보를 제어흐름 그래프와 연관시켜 각 기본블록의 실행시간을 다음의 수식에 의해 구하게 된다

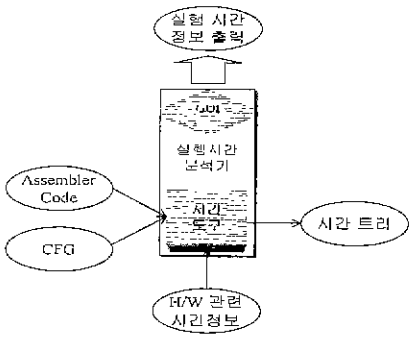
기본 블록의 실행시간은 기본 블록 안의 명령어의 클록을 더하는 것이다[Rod95]

$$WCET_{BB} = \sum_{1}^{n} WCIET_i$$

$WCET_{BB}$ 기본 블록에 대한 최악의 경우 실행시간

$WCIET_i$: 명령어 i에 대한 최악의 경우 명령어 실행시간

[그림 3]에 실행시간 분석기의 구조를 나타내었다.



[그림 3] 실행시간 분석기

시간도구는 하드웨어 관련 시간정보와 어셈블리 코드, 그리고 제어흐름 그래프를 사용하여 각 기본 블록의 실행시간을 계산하게 된다

이와 같이 연산된 기본 블록의 실행시간은 시간 트리에 기록되어 한번 계산 후에는 그 결과를 이용하게 된다 또한, 시간 트리는 원시언어의 라인번호를 가지고 있기 때문에 어셈블리 언어에서 구한 시간정보를 원시 언어와 연결시켜주는 역할을 하게 된다

4. 결론 및 향후 연구 과제

본 논문에서는 기존의 C언어의 최악실행시간을 예측하기 위한 시간 도구를 제안하였다 기존의 C언어는 시간 분석을 염두에 두고 설계되어진 언어가 아니므로, 시간분석을 불가능하게 만드는 많은 요소들이 포함되어져 있다 본 논문에서는 이러한 특성들의 제약사항을 설명하고, 시간과 반복 수에 대한 제약조건을 묘사하는 방법을 제안하였다 본 논문에서 Timing-C언어로 기술된 프로그램의 최악실행시간을 계산하기 위한 시간 도구를 설계한다. 본 논문에서 설계한 시간 도구를 이용하면, 태스크의 실행 시간 예측이 가능하며, 프로그램 코드 일부의 실행시간도 알 수 있다. 이와 더불어 정확한 최악실행시간을 계산하기 위하여 경로에 관한 의미 분석이 필요하다

본 논문의 향후 연구과제로써 경로의 의미 분석에 관한 연구가 뒷받침되어야 할 것이다

참 고 문 헌

[Alt97] P Altenbend, CHaRy The C-LAB Hard Real-Time System to Support Mechatronical Design, International Conference and Workshop on Engineering of Computer Based System(ECBS), Monterey, 1997

[Bou93] F Bourdoncle, Effective chaotic iteration strategies with widening, Procedings of the International Conference on Formal Methods in Programming and Their Applications, 1993

[Hon93] S. Hong and R. Gerber Compiling real-time programs into schedulable code In Proceedings of the ACM SIGPLAN '93 Conference on Programming Language Design and Implementation ACM Press, June 1993 SIGPLAN Notices, 28(6):166-176

[Kli86] E. Kligerman and A . D Stoyenko Real-time Euclid. A language for reliable real-time systems IEEE Transactions on Software Engineering, 12 941--949, September 1986

[Lim94] S Lim, Y.Bae,G Jang,B Rhee, S Min,C Park,H Shin,K Park,and C Kim An Accurate Worst Case Timing Analysis for RISC Processors Proceeding of 15th Real-Time Systems Symposium, 1994

[Mue94] F Mueller, Static cache Simulation and Its Applications, PhD Dissertation, Florida State Univ , 1994

[Pus91] P.Puschner and A Schedl Computing Maximum Task Execution Times - A Graph-based Approach 1991

[Rod95] Roderick Chapman, Static Timing Analysis and Program Proof, Computer Science, University of York, PhD thesis, 1995

[Ste95] Yau-Tsun Steven Li, S.Malik,A Wolfe Cache Modeling for Real-Time Software'Beyond Direct Mapped Instruction Caches, Princeton Univ , 1995.

[Wol91] V Wolfe, S Davidson, and I. Lee RTC Language support for real- time concurrency In Proceedings of the 12th IEEE Real-time Systems Symposium, pages 43--52, San Antonio, Texas, December 1991

[Whi94] Randall T. White, Frank Mueller, C A Healy D B Whalley, M.G. Harmon, Timing Analysis for Data Caches and Set-Associative Cache, 1994

[백97] 백정현, 원유헌, 프로그램형 자동화기기를 위한 실시간 메카니즘제어 언어의 구현기법, 대한전자공학회 논문지, 제 34권 11호, 1997