전력 소비 최적화를 위한 지역 및 전역 최적화 기술

김성진*, 윤종희**, 고광만*
*상지대학교 컴퓨터정보공학부, **강릉원주대학교 컴퓨터공학과
e-mail: sjkim.sangji@gmail.com, jhyoun@gwnu.ac.kr, kkman@sangji.ac.kr

Local and Global Optimization Techniques for Power Consumption Optimization

Seongjin Kim*, Jonghee M. Youn**, Kwangman Ko*
*Department of Computer Data & Engineering, Sangji University, Korea
**Department of Computer Science & Engineering, Gangneung-Wonju National University, Korea

요 약

임베디드 시스템은 여러 분야에서 사용되고 있으며, 그 범위는 더욱더 다양하게 늘어나고 있다. 이러한 다양성은 임베디드 시스템이 사용되는 목적에 따라 새로운 아키텍처를 요구하게 되면서, 아키텍처 구조, 동작에 대한 변경 또는 새로운 설계에 대해 개발 시간과 비용을 줄이기 위한 재목적 컴파일러의 개발 필요성과 중요성이 강조되고 있다. 더욱이 전력이 제한적인 모바일 기기에서 동작하는 어플리케이션의 최적화와 이러한 최적화를 위한 컴파일러 개발은 매우 중요한 이슈가 되고 있으며, 특히 어플리케이션 성능에 직접적인 영향을 주는 컴파일러 후단부는 다양한 방법론들이 적용되어 있고 많은 연구가 수행되고 있다. 이 논문에서는 EXPRESSION의 재목적 컴파일러인 EXPRESS의 후단부에서 코드 최적화를 위해 적용된 기법을 분석하고, 기존 코드 스케줄링과 더불어 성능 개선을 위해서 기본 블록 스케줄링을 추가한 모델을 설계하고 성능평가 방법을 제시한다.

1. 서론

임베디드 시스템은 산업 기기, 통신 장비, 가정 제품 등 여러 분야에서 폭넓게 사용되고 있으며, 사용 목적에 따라 변화하는 아키텍처의 구조, 동작 또는 새로운 설계에 대해 개발 시간과 비용을 줄이기 위한 다양한 연구 시도가 계속되고 있다. 특히, 임베디드 시스템의 빠른 발전 속도는 Time-to-Market 제약이라는 문제에 직면하고 있으며, 새로운 임베디드 시스템을 설계 할 경우 실제 하드웨어적인 구현 전에 소프트웨어적으로 디자인을 검증할 수 있는 도구의 필요성은 매우 중요한 의미를 가지고 있다[1]. 또한 이러한 내장형 시스템은 특정 기능만을 수행할 수 있도록 고안되었으며, 하드웨어적으로 기능 수행을 위한 프로세서와 제한적인 메모리 혹은 저장장치로 이루어져 있다. 이런 제약적인 환경을 극복하기 위해서는 소프트웨어적인 최적화가 필수적이다.

더불어 최근 모바일 기기의 급속한 보급으로 인해 상시전원을 사용할 수 없는 환경에서 전력·에너지 소비 문제는 매우 중요한 이슈가 되었으며, 하드웨어는 물론 소프트웨어가 소비하는 전력소모에 대한 관심이 높아지고 있다. 특히, 임베디드 시스템에서 동작하는 어플리케이션은 프로세서와 메모리 시스템 운영에 상당한 영향을 줄 수 있는 결정적 요인이 될 수 있으며, 전체 에너지 소비에서 상당한 영향력을 가지고 있다. 따라서 소프트웨어를 보다 빠르고 안정적으로 동작할 수 있는 코드를 생성하고 아키텍처 설계, 소프트웨어 개발의 비용과 시간을 줄일 수 있는 재목적 컴파일러의 필요성이 이슈화 되었으며 어플리케이션 성능에 직접적인 영향을 주는 컴파일러 후단부는 코드 선택, 레지스터 할당, 코드 스케줄링 등의 다양한 최적화 기법들이 적용되어 있다.

본 논문에서는 재목적 컴파일러와 시뮬레이터를 생성하는 EXPRESSION[2]를 기반으로 하여 컴파일러인 EXPRESS의 후단부에 적용된 최적화 기법들을 분석하고, 이들 최적화 중 전역 코드 스케줄링인 Trailblazing percolation scheduling[3]과 더불어 기본 블록 스케줄링인 리스트-스케줄링(List Scheduling)을 추가로 적용한 모델을 설계하고 그에 따른 검증 및 성능평가를 위한 방법을 제시한다.

이 논문의 구성은 제 2장에서 본 논문의 기반이 되는 EXPRESSION과 컴파일러인 EXPRESS의 구조 및 후단부에 적용된 최적화 기법을 설명하고, 제 3장에서는 컴파일러의 성능 개선을 위한 모델을 설계하고 성능 평가 방법을 소개한다.

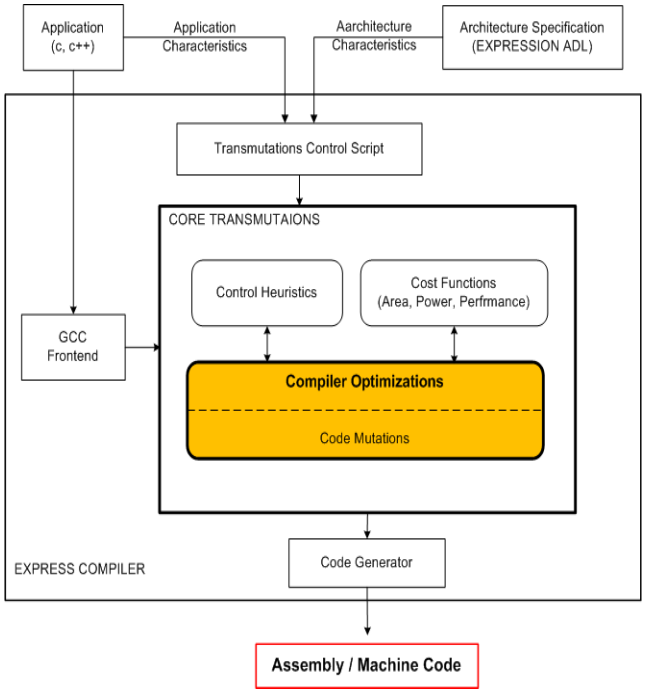
2. 관련연구

임베디드 프로세서 아키텍처에 대한 효율성이 높은 컴파일러를 위해서는 다양한 종류의 최적화를 적용시키는

것이 매우 중요하다. 이러한 최적화는 조건부 명령어, 어플리케이션 혹은 SIMD(Single Instruction Multiple Data)와 같은 특정 목적을 가진 아키텍처에 매우 의존적이다.

EXPRESSION은 구조/행위 정보를 가진 LISP-like한 ADL로부터 컴파일러(EXPRESS)와 시뮬레이터(SIMPRESS)를 생성한다. EXPRESS 컴파일러는 프로그램의 특성과 이용 가능한 자원들을 기반으로 최적화를 위한 각 단계들의 동적인 순서조합을 제공하기 위한 핵심 변형(Core transmutations)을 제공하며, ADL을 통해 자동적으로 예약 테이블(RT)을 생성할 수 있는 강력한 기능을 제공한다[4]. 또한 EXPRESS 컴파일러는 하나의 최적화한 메모리-인식 ILP 컴파일러 기능을 수행하는데, EXPRESSION ADL로부터 다양한 프로세서 아키텍처와 메모리 시스템을 목적에 맞게 바꾸기 위해 사용된다.

[그림 1]은 EXPRESSION ADL로부터 생성되는 EXPRESS 컴파일러 프레임워크를 보여준다. 우선 EXPRESS 컴파일러의 입력으로 C/C++로 기술된 어플리케이션과 EXPRESSION ADL에 기술된 아키텍처 정보가 사용된다. 어플리케이션을 분석하는 전단부(frontend)는 GCC 기반이며 일반적인 최적화가 적용된 형태로 실행된다.



[그림 1] EXPRESS 컴파일러 프레임워크

EXPRESS 컴파일러 후단부(backend)인 핵심 변형 부분은 코드 최적화를 위한 다양한 컴파일러 최적화 기법들이 적용되어 있는데, 루프 파이프라이닝 기술 중 하나인 Resource-directed loop pipelining(RDLP)[5]와 전역 코드 스케줄링 기술 중 하나인 Trailblazing percolation scheduling(TiPS), 그리고 명령어 선택, 레지스터 할당을 위한 Mutation Scheduling(MS)[6]과 if-conversion 최적화 기법이 사용되었다. 이러한 최적화 기법들이 적용된 후단부는 코드 생성기를 거쳐 MPIS 어셈블리 코드와 기계코드(중간코드)를 생성하게 된다.

후단부의 핵심 변형이 적용된 EXPRESS 컴파일러는 리소스 활용성과 프로그램 지역성을 기반으로 한 프로그램 코드와 변형 순서를 동적으로 적용할 수 있게 만드는 특징을 가진다.

3. 코드 최적화 성능 향상을 위한 기본 블록 스케쥴링 적용 및 평가방법

3.1 DDG 기반 리스트-스케줄링

EXPRESS 컴파일러의 후단부에 적용된 최적화 중 대표적인 코드 스케줄링인 Trailblazing percolation scheduling은 CFG(control flow graph) 기반의 전역 코드 스케줄링이다. 전역 코드 스케줄링은 기본적으로 명령어들이 하나의 기본 블록(basic Block)으로부터 다른 기본 블록으로 이동하는 코드 생성을 고려한다. 하지만 명령어 파이프 라인(instruction pipeline)을 사용하는 대부분의 프로세서에 대해서 전역 코드 스케줄링과 더불어 기본 블록내의 명령어들을 축약시키고 최적화하여 유휴 자원을 만드는 기본 블록 스케줄링을 추가로 적용함으로써 아키텍처가 제공하는 자원을 보다 더 효율적으로 사용할 있도록 하는 것은 매우 중요한 문제이다. 따라서 이를 위한 기본 블록 스케줄링 기법은 여러 종류가 있으며 대표적인 스케줄링 기법을 살펴보면 다음과 같다.

우선 어플리케이션의 실행 속도, 메모리 사용량과 같은 성능 향상을 위한 인스트럭션 스케줄링 기법은 기본 블록인 코드 세그먼트 단위로 인스트럭션의 실행 순서를 재조정하는 가장 핵심적인 기법으로 사용되고 있다[7]. 인스트럭션 스케줄링이 적용되는 코드 세그먼트의 범위가 커질수록 인스트럭션의 순서가 재조정되기 위해 움직이는 범위가 커짐에 따라 최적화의 기회와 융통성이 증가되는 장점을 가지고 있지만 상대적으로 인스트럭션 알고리즘이 복잡해지고 궁극적으로는 컴파일 시간이 증가되는 단점을 가지고 있다.

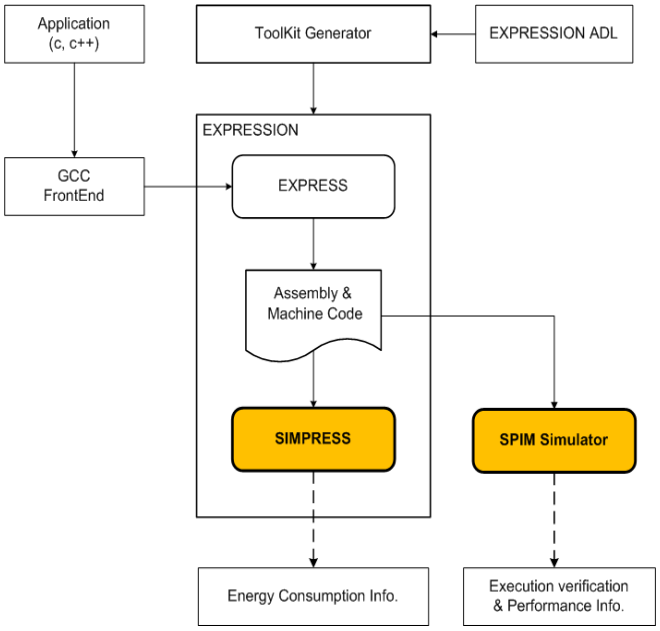
리스트-스케줄링 기법은 기본 블록 단위로 수행 순서를 조정하는 기법으로 가장 널리 적용되고 있으며 소프트웨어 파이프라인, 분기 스케줄링, 추적 스케줄링 기법 등을 포함하고 있다. 또한 기본 블록은 인스트럭션을 나타내는 노드와 인스트럭션 최적화를 위해 명령어 순서 재조정에서 필요한 데이터 의존성을 표현하는 data-dependency graph(DDG)로 표현된다. 따라서 DDG 노드로 표현되는 기본 블록에 대해 데이터 의존성을 유지하면서 최소한의 실행 시간을 유지할 수 있도록 위상 정렬을 구축하는 것이 핵심이다.

3.2 실행 검증 및 성능 평가

EXPRESS 컴파일러에서 생성되는 MIPS 어셈블리 코드는 전역 코드 스케줄링과 함께 본 논문에서 제시하는

기본 블록 스케줄링에서 DDG 기반의 리스트-스케줄링 기법이 새롭게 적용된 코드이다. 따라서 이렇게 생성된 어셈블리 코드가 어플리케이션이 의도한 결과를 정확히 표현하고 실행되는지에 대한 검증은 필수적이며 매우 중요하다. 새롭게 생성된 어셈블리 코드에 대해서 실행 및 성능 검증을 위한 전체 흐름은 [그림 2]와 같다.

우선 EXPRESS 컴파일러에서 생성하는 어셈블리 코드의 정확성 및 실행 검증을 위해서 SPIM Simulator[8]를 이용하여 코드 검증을 수행할 것이다. SPIM Simulator는 MIPS 어셈블리 코드의 실행 검증에 많이 사용되는 도구로써 MIPS Processor에서 동작하는 어셈블리 코드를 읽고 실행결과를 보여주며 Debugging이 가능한 특징을 가지고 있다.



[그림 2] 어셈블리 코드에 대한 실행 및 성능 검증

또한 SIMPRESS는 EXPRESSION ADL로부터 생성되는 인터프리터 시뮬레이션 기반의 사이클-정확성(cycle-accurate) 시뮬레이터로써 EXPRESS 컴파일러에서 생성하는 중간코드를 입력으로 받아 전체 사이클 수 및 캐시(cache)의 read/write 횟수, SRAM/DRAM의 load/stores 횟수와 함께 캐시와 RAM이 소비한 에너지 수치가 저장된 결과파일을 제공한다. 이렇게 제공되는 결과파일은 새롭게 적용한 최적화 방법의 성능 평가에 유용하게 이용될 것이다.

4. 결론 및 향후연구

최근 임베디드 시스템 분야에서 모바일 장치의 급속한 보급으로 인해 그에 적합한 어플리케이션의 개발이 폭발적으로 증가되고 있으며, 배터리를 사용하는 모바일 장치가 갖는 전력 공급의 한계로 인해 하드웨어적인 관리와 더불어 소프트웨어적인 노력이 매우 중요하게 평가받고 있다. 특히, 모바일에서 동작하는 어플리케이션은 프로세서와 메모리 시스템 운영에 상당한 영향을 줄 수 있는 결정적 요인이 될 수 있으므로 빠르고 안정적으로 동작할 수 있는 코드를 생성하고 아키텍처 설계 및 소프트웨어 개발 비용과 시간을 줄일 수 있는 재목적 컴파일러의 필요성이 대두되었다.

본 논문에서는 EXPRESSION ADL로부터 생성되는 재목적 컴파일러인 EXPRESS를 기반으로 후단부의 코드 스케줄링인 TiPS와 함께 기본 블록 스케줄링인 DDG 기반의 리스트-스케줄링을 적용하여 양질의 코드 생성과 함께 성능 개선을 위한 모델을 설계하고 결과물인 MIPS 어셈블리 코드에 대한 실행 검증 및 성능 평가 방법을 제시하였다. 향후 실제 구현을 통하여 성능결과를 비교 분석할 것이며, 더 나아가 어플리케이션이 소비하는 전력·에너지를 줄이기 위한 코드 스케줄링 기법을 적용한 결과를 비교 분석할 것이다.

5. Acknowledgement

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2011-0012522)

참고문헌

[1] Y. N. Srikant, "Energy-Aware Compiler Optimizations", in The Compiler Design Handbook: Optimizations and Machine Code Generation 2nd Edition(Editor: Y. N. Shikan, K. A. Vardhan), CRC Press, pages 235~265, 2008.

[2] A. Halambi, P. Grün, V. Ganesh, A. Khare, N. Dutt and A. Nicolau, "EXPRESSION: A Language for Architectural Exploration through Compiler/Simulator Retargetability," Proc. of the 1999 Design, Automation and Test in Europe Conference, March 1999.

[3] A. Nicolau and S. Novack. "Trailblazing: A hierarchical approach to percolation scheduling". In International Conference on Parallel Processing, 1993.

[4] Peter Grun , Ashok Halambi , Nikil Dutt , Alex Nicolau, "RTGEN: An Algorithm for Automatic Generation of Reservation Tables from Architectural Descriptions", Proceedings of the 12th international symposium on System synthesis, p.44, November 01-04, 1999

[5] S. Novack, A. Nicolau, and N. Dutt. "Resource Directed loop pipelining : Exposing just enough parallelism". The Computer Journal, 40(6):311-321, 1997

[6] Steven Novack , Alexandru Nicolau, "Mutation Scheduling: A Unified Approach to Compiling for Fine-Grain Parallelism", Proceedings of the 7th International Workshop on Languages and Compilers for Parallel Computing, p.16-30, August 08-10, 1994

[7] Chingren Lee, Jenq Kuen Lee, Tingting Hwang, and Shi-Chun Tsai, "Compiler Optimization on VLIW Instruction Scheduling for Low Power", ACM Trans. on Design Automation of Electronic System, Vol. 8, No. 2, pages 252~268, 2003.
[8] Larus, J., "SPIM: A MIPS32 simulator", http://pages.cs.wisc.edu/~larus/spim.html, retrieved June 10, 2005.