임베디드 코어 설계를 위해 설계 계층을 이용한 효율적인 아키텍처 탐색

준회원 김 상 우*, 정회원 황 선 영*

An Efficient Architecture Exploration for Embedded Core Design Exploiting Design Hierarchy

Sang-Woo Kim* *Associate Member*, Sun-Young Hwang* *Regular Member*

요 약

본 논문은 임베디드 코어의 설계 계층을 이용한 아키텍처 탐색 방법론을 제안한다. 제안된 방법은 다양한 설계 검증과 계층적인 설계 수준에 따른 성능 측정을 고려한 체계적인 아키텍처 탐색을 수행한다. 성능 측정 도구는 설계 모듈에 관련 있는 성능 데이터를 가진 프로파일을 생성한다. 프로파일 생성기는 설계 모듈과 성능 매개변수에 대한 연관 규칙을 얻기 위해 데이터마이닝을 수행한다. 프로파일 생성기의 추론 엔진은 다음 탐색 과정의 설계 성능을 향상시키는 새로운 연관 규칙을 얻는다.

제안된 아키텍처 탐색 방법론의 효율성을 확인하기 위해 JPEG 인코더, Chen-DCT, FFT의 어플리케이션에 대한 아키텍처 탐색을 수행하였다. 제안된 방법을 이용하여 설계된 임베디드 코어는 MIPS R3000의 초기 임베디드 코어에 비해 평균 60.8%의 수행 사이클 감소를 보인다.

Key Words : 아키텍처 탐색, Embedded System, Design Hierarchy, 성능 estimation, 프로파일

ABSTRACT

This paper proposes an architecture exploration methodology for the design of embedded cores exploiting design hierarchy. The proposed method performs systematic architecture exploration by taking different approaches for verifying designs and estimating performances depending on the hierarchy level in design process. Performance estimation tools generate profile having performance data related with design modules of an embedded core. Profile analyzer performs data-mining to acquire association rules between the design modules and performance parameters. Inference engine in the profile analyzer updates the association rules which will be used to improve the design performance at next exploration steps.

To show the efficiency of the proposed architecture explorations methodology, experiments had been performed for JPEG encoder, Chen-DCT, and FFT application functions. The embedded cores designed by taking the proposed method show performance improvement by 60.8% in terms of clock cycles on the average when compared with the initial embedded core in MIPS R3000.

Ⅰ. 서 론

한정된 용량의 배터리로 동작하는 임베디드 시스템에서 다양한 어플리케이션에 대한 짧은 time-to-market과 끊임없는 성능개선 등에 대한 요구를 충족시키기가 어렵게 되어, 핵심기능을 단일 칩으로 집약하고 전

※ 본 연구는 교육과학기술부의 재원으로 한국연구재단의 지원을 받아 수행되었습니다. (#2010-0008043).
* 서강대학교 전자공학과 CAD & ES 연구실 (hwang@sogang.ac.kr)
논문번호 : KICS2010-06-284, 접수일자 : 2010년 6월 29일, 최종논문접수일자 : 2010년 12월 7일

력 소비를 최소화한 시스템 반도체의 설계, 개발이 중요하게 되었다[1][2]. 시스템 수준의 저전력 설계를 위해 동적 전력 관리, 동적 전압 조절, 저전력 코드를 생성하는 컴파일러 등의 방법이 제안되었다. 동적 전력 관리는 시스템 구성요소가 불필요한 전력 소모를 최소화하기 위해 시스템의 동작 환경에 따라 전력 상태를 변화시키는 방법이고[3], 동적 전압 조절은 프로세서의 동작 전압과 이에 따른 동작 주파수를 조절하는 방법이다[4]. 저전력 코드를 생성하는 컴파일러는 명령어의 recoding과 스케줄링을 통해 프로세서에 저전력 동작이 가능한 명령어를 재구성한다[5][6].

특정 어플리케이션에 대한 고성능 동작과 설계 스펙의 변동에 능동적으로 대처할 수 있는 ASIP (Application Specific Instruction-set Processor)의 사용으로 최적화된 명령어 집합과 프로세서 구조를 갖는 임베디드 코어의 설계가 짧은 시간 내에 가능하게 되었으나, 저전력 ASIP 설계는 최적화된 명령어 집합과 프로세서의 구조에 알맞은 저전력 코드를 생성하는 컴파일러 설계에 많은 비용과 시간이 필요하다. 이러한 문제점을 해결하기 위해 제안된 머신 기술 언어를 이용한 ASIP 설계 방식에서는 기술된 타겟 프로세서에 적합한 컴파일러와 시뮬레이터 등을 자동 생성하여 설계의 정확성과 일관성을 검증할 수 있도록 하여야, 보다 다양하고 넓은 탐색 공간에서의 아키텍처 탐색이 가능하다[5].

최적화된 ASIP을 설계하기 위한 아키텍처 탐색은 전력 소모, 실행 시간, 칩 면적 등을 조합한 비용 함수의 값이 최적화된 설계 모듈을 찾아내는 과정이다[7][8]. 최적화된 설계 모듈을 얻어내는 과정은 컴파일러와 시뮬레이터 등을 생성하는 과정, 성능 측정의 결과를 획득하고 분석하는 과정에 소요되는 시간이 필요하므로, 고려해야 할 설계 모듈이 많을수록 설계 시간과 비용이 증가하게 된다[7]. 아키텍처 탐색의 효율성을 향상시키기 위해 서로 trade-off 관계에 있는 설계 검증을 위한 시뮬레이션의 수행 속도와 성능 측정의 정확성을 동시에 고려한 효율적인 시스템의 개발이 필요하다[9-11].

컴파일러 기반의 아키텍처 탐색은 타겟 프로세서와 컴파일러 설계의 일관성을 위해 머신 기술 언어 기반 방식을 이용하며, 어플리케이션 프로그램의 정적 분석 결과와 시뮬레이션을 통해 얻은 동적 분석 결과를 이용하여 어플리케이션에 최적화된 타겟 프로세서의 구조와 명령어 집합을 결정한다[8,12]. 이 방법은 다양한 컴파일러의 최적화 기법을 고려하여 고성능 ASIP을 얻을 수 있으나 아키텍처 탐색 과정에서 컴파일러의 최적화 기법과 관련이 없는 시뮬레이션과 프로파일 분석 등을 수행하므로 최종 ASIP을 얻는 과정에 많은 시간이 소요된다[11]. 이를 극복하기 위해 시뮬레이션의 수행 속도와 성능 측정의 정확성을 동시에 고려한 아키텍처 탐색 방법은 어플리케이션 프로그램의 각 basic-block의 수행 빈도와 수행 사이클 수 예측한 결과를 바탕으로 초기 ASIP을 빠르게 결정하고, 시뮬레이터를 이용하여 얻은 프로파일을 분석한 결과를 가지고 초기 ASIP을 개선하는 방법이 제안되었으나[6], 역시 설계 hierarchy에 따른 특성을 활용하며 아키텍처 탐색을 수행하지 않고 있다.

본 연구에서는 임베디드 코어 설계를 위해 머신 기술 언어 기반 방식의 아키텍처 탐색을 위한 방법론을 제시한다. 제안된 아키텍처 탐색 방법은 어플리케이션에 최적화된 임베디드 코어를 빠르게 얻기 위해, 아키텍처 탐색 과정에서 설계 단계에 따라 성능 측정할 설계 모듈을 정해놓고 불필요한 프로파일링의 과정을 최소화한다. 아키텍처 탐색 과정에서 얻은 프로파일은 다음 설계 단계에서의 아키텍처 탐색의 수행 단계에서 시뮬레이터를 수행하여 얻은 프로파일에 포함되도록 계층적으로 프로파일을 프로파일 데이터베이스에 저장된다. 아키텍처 탐색의 수행 단계를 효율적으로 결정하기 위해 제안된 프로파일 분석기는 프로파일 데이터베이스에서 현재 아키텍처 탐색의 수행 단계에 해당되는 관련 프로파일을 입력으로 분석 과정과 추정 과정을 통해 임베디드 코어의 성능을 향상시킬 수 있는 임베디드 코어의 설계 모듈에 대한 정보를 찾아낸다. 임베디드 코어의 성능을 보다 향상시킬 수 있는 설계 모듈을 중심으로 아키텍처 탐색의 다음 단계를 위한 임베디드 코어의 구조를 개선한다.

본 논문의 구성은 다음과 같다. 제 2절에서는 제안하는 방식이 적용되는 전체 framework인 RISGen (Retargetable Instruction-set Simulator Generator) 시스템의 개관을 설명하고, 제 3절에서는 제안된 아키텍처 탐색 흐름도와 프로파일 데이터베이스, 그리고 프로파일 분석기에 대해 설명한다. 제 4절에서는 제안된 아키텍처 탐색 방법을 이용하여 얻어낸 최종 임베디드 코어의 성능과 아키텍처 탐색 방법의 효율성에 대한 실험결과를 보이며, 마지막으로 제 5절에서 결론 및 추후과제를 제시한다.

Ⅱ. RISGen 시스템 개관

RISGen 시스템은 SMDL 언어[13-15]로 기술한 타겟 프로세서 모델의 구조적 정보와 행위 정보를 가진 IR

1759

(Intermediate Representation)로부터 시뮬레이션 커널과 성능 측정 모델을 참조하여 event-driven 방식의 시뮬레이터와 retargetable 컴파일러, 그리고 임베디드 코어를 자동 생성한다. 그림 1은 RISGen 시스템의 개관을 보인다.

임베디드 코어 모델은 IR을 입력으로 한 임베디드 코어 생성기에서 생성되며 임베디드 코어의 파이프라인 정보, 데이터패스 구조 정보, 기능유닛 간의 연결 정보 등을 가진다[13]. RISGen 라이브러리는 event-driven 방식의 시뮬레이션 커널, 운영체제에 동작하는 시스템 콜을 시뮬레이션하기 위한 시스템 콜 모델, trade-off 관계에 있는 시뮬레이션의 수행 속도와 성능 측정의 정확성에 대한 성능 측정 모델 등을 가진다. 시뮬레이터 생성기는 임베디드 코어 모델과 RISGen 라이브러리를 참조하여 성능 측정 모델에 해당되는 event-driven 방식의 시뮬레이터를 생성한다[15].

아키텍처 탐색 과정을 효율적으로 수행하기 위해설계 단계에 알맞은 성능 측정의 속도와 정확도를 가진 성능 측정 모델이 필요하다[9][11]. 그림 2는 RISGen 시스템에서 수행 속도와 정확도 관계를 보인다.

인스트럭션 스케줄을 이용하는 방법은 컴파일러 인스트럭션 스케줄러가 어플리케이션 프로그램을 정적으로 분석한 결과인 basic-block 수준의 프로파일을 빠르게 얻어낸다[17]. 사이클 수준 시뮬레이터를 이용하는 방법은 타겟 프로세서의 파이프라인 구조를 총체적으로 반영한 시뮬레이션을 수행하여 정확한 성능 측정을 구할 수 있으나 시뮬레이션 속도가 느리다. 명령어 수준 시뮬레이터를 이용하는 방법은 각 명령어가 타겟 프로세서의 파이프라인에 동작되는 사이클 값을 이용하여 시뮬레이션을 수행하며 파이프라인 구조의 세부적인 정보를 고려하지 않으므로, 사이클 수

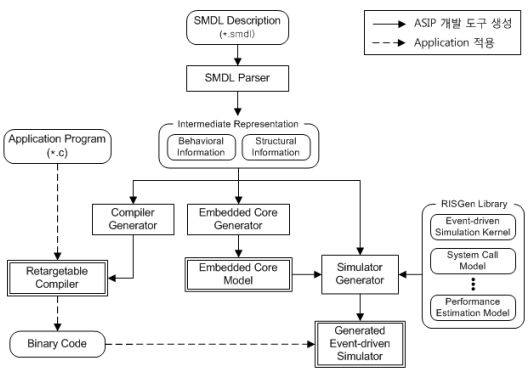

그림 1. RISGen 시스템의 개관

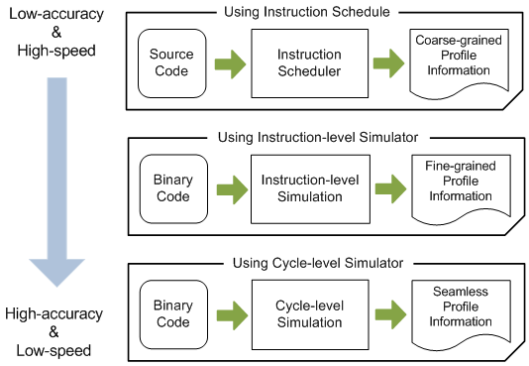

그림 2. RISGen 시스템에서 수행 속도와 정확도 관계

준 시뮬레이터를 이용한 방법보다 성능 측정의 정확도가 낮으나 빠른 시뮬레이션 속도를 가진다.

Ⅲ. 제안된 아키텍처 탐색 시스템

본 절에서는 제안된 임베디드 코어의 설계 계층을 이용한 아키텍처 탐색 흐름과 프로파일 분석기를 제시한다.

3.1 임베디드 코어의 설계 계층을 이용한 아키텍처 탐색 흐름

아키텍처 탐색은 일반적으로 매우 긴 시간을 요구할 과정으로 최적화된 프로세서 모델을 얻어내는 과정에서 불필요한 성능 측정 등을 최소화하는 것이 중요하다[5,10,11]. 본 논문에서 제안된 아키텍처 탐색 시스템은 임베디드 코어의 설계 단계에서 서로 다른 설계 계층에서의 특성을 이용하여 체계적으로 탐색 시간을 줄일 수 있도록 한다. 그림 3은 제안된 아키텍처 탐색 흐름도를 보인다.

머신 기술 언어의 파싱에 의해 생성된 프로세서 모델은 임베디드 코어 생성기, SRCC (Sogang Retargetable Compiler Compiler) 시스템과 시뮬레이터 생성기에서 임베디드 코어 모델, retargetable 컴파일러와 시뮬레이터 생성에 필요한 타겟 프로세서의 구조적 정보와 행위 정보를 가진다[13][14][15]. SRCC 시스템에서 생성된 retargetable 컴파일러는 초기 임베디드 코어를 얻을 때까지 어플리케이션 프로그램을 정적으로 분석한 결과인 basic-block 수준의 프로파일을 생성하고, 초기 임베디드 코어가 확정되면 바이너리 코드로 변환한다. 본 논문의 아키텍처 탐색 시스템에 이용되는 컴파일드 코드 방식의 시뮬레이터는 시뮬레이션 속도를 향상시키기 위해 제안되었다[16]. 시뮬레이터 생성기는 임베디드 코어 모델과 RISGen 라이브

1760

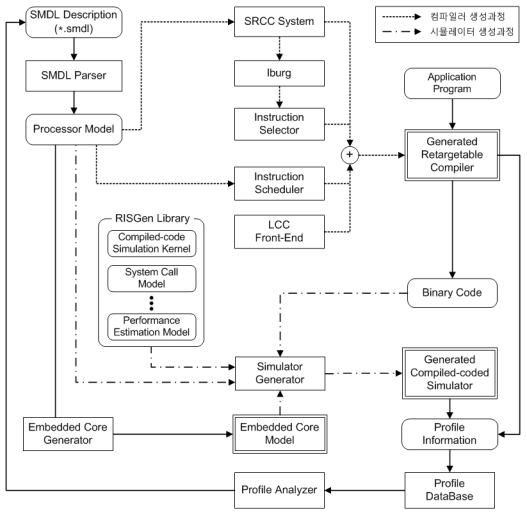

그림 3. 제안된 아키텍처 탐색 흐름도

러리, 그리고 바이너리 코드를 참조하여 시뮬레이션의 명령어 디코드 과정을 정적으로 결정하고 사이클 수준과 명령어 수준 컴파일드 코드 방식의 시뮬레이터를 생성한다. 성능 측정으로 획득한 프로파일 정보는 다음 설계 단계에서의 아키텍처 탐색의 수행 단계에서 얻어낼 프로파일에 포함되도록 계층적 구조를 갖는 프로파일 데이터베이스에 저장된다. 프로파일 분석기는 현재 아키텍처 탐색의 수행 단계에 해당되는 프로파일 데이터베이스의 관련 프로파일을 이용하여 임베디드 코어의 성능을 향상시킬 수 있는 설계 모듈을 구하고, 얻어낸 설계 모듈을 바탕으로 임베디드 코어의 구조를 개선한다. 아키텍처 탐색의 수행 단계에서 목표하는 임베디드 코어를 얻을 때까지 위 과정을 반복한다.

제안된 아키텍처 탐색은 프로파일 정보를 얻어내는 성능 측정 도구가 가진 성능 측정의 수행 속도와 정확도를 고려하여 어플리케이션에 최적화된 임베디드 코어를 빠르게 얻어내는 3단계 아키텍처 탐색 방식을 수행한다. 아키텍처 탐색의 1단계는 초기 임베디드 코어의 구조를 빠르게 결정하기 위해 basic-block 수준의 프로파일 정보를 얻어내는 컴파일러의 인스트럭션 스케줄링 결과를 이용한다. 이 과정은 넓은 설계 공간을 가지며 프로세서 리소스, 프로세서 구조 등의 관련 설계 모듈을 고려하여 초기 임베디드 코어에 결정하고 최적화된 컴파일러의 명령어 선택기를 얻는다. 개선된 임베디드 코어의 각 설계 모듈은 임베디드 코어의 성능 기여도 값이 설계자가 정해놓은 기준 값보다 작으면 개선할 설계 모듈의 목록에 제외하며, 고려할 설계 모듈이 없을 경우 초기 임베디드 코어의 구조로 정하고 컴파일러의 명령어 선택기를 생성한다. 아키텍처 탐색의 2단계는 1단계보다 축소된 설계 공간에서 임베디드 코어의 각 설계 모듈에 대해 정확한 프로파일 정보를 얻어내기 위해 명령어 수준 컴파일드 코드 방식의 시뮬레이션을 수행한다. 이 과정에서 레지스터 파일, 기능 유닛, 명령어 집합 아키텍처 구조, 프로세서 리소스간의 연결 구조, 데이터패스 구조 등의 설계 모듈을 고려하여 향상된 임베디드 코어를 결정하고 최적화된 컴파일러의 인스트럭션 스케줄러를 생성한다. 아키텍처 탐색의 3단계는 데이터 포워딩 등과 같은 최적화된 임베디드 하드웨어 기법을 이용한 임베디드 코어 설계를 위해 2단계보다 더욱 정확한 임베디드 코어의 각 설계 모듈에 대한 프로파일을 생성하는 사이클 수준 컴파일드 코드 방식의 시뮬레이션을 수행한다. 이 과정에서 임베디드 코어의 세부적인 파이프라인 구조를 중심으로 레지스터 파일, 기능 유닛, 명령어 집합 아키텍처 구조, 데이터패스 구조 등의 설계 모듈을 개선하여 임베디드 코어의 성능을 극대화시킬 수 있는 결과를 생성한다.

3.2 프로파일 분석기

제안된 아키텍처 탐색을 효율적으로 수행하기 위해서는 임베디드 코어의 설계 모듈에 관련 있는 성능 데이터를 올바르게 분석하는 프로파일 분석기가 필요하다. 그림 4는 제안된 프로파일 데이터베이스를 보인다.

제안된 프로파일 데이터베이스에서는 아키텍처 탐색의 수행 단계에 따른 성능 측정이 요구되는 임베디드 코어의 설계 모듈을 고려한 프로파일 분석을 위해 현재 아키텍처 탐색의 수행 단계에서 얻은 프로파일이 다음 설계 단계의 아키텍처 탐색의 수행 과정에서 얻은 프로파일에 포함되도록 계층적으로 프로파일을 구성된다[18]. 아키텍처 탐색의 1단계는 임베디드 코어

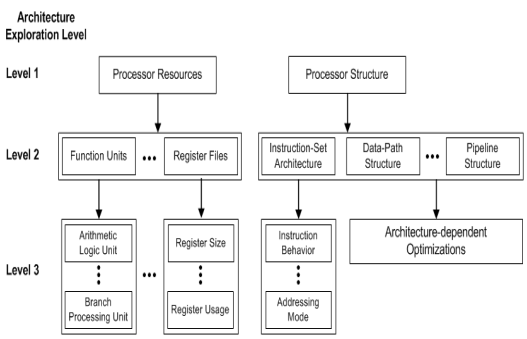

그림 4. 제안된 프로파일 데이터베이스

1761

를 빠르게 결정하기 위해 컴파일러의 인스트럭션 스케줄링 결과를 이용하여 프로세서 리소스, 프로세서 구조 등에 대한 프로파일 정보를 얻는다. 아키텍처 탐색의 2단계는 초기 임베디드 코어의 성능을 향상시키기 위해 명령어 수준 컴파일드 코드 방식의 시뮬레이터를 이용하여 기능 유닛, 레지스터 파일, 명령어 집합 아키텍처 등과 데이터패스 구조, 파이프라인 구조 등에 대한 프로파일 정보를 얻는다. 아키텍처 탐색의 3단계는 임베디드 코어의 성능을 극대화시키기 위해 사이클 수준 컴파일드 코드 방식의 시뮬레이터를 이용하여 산술 논리 유닛, 분기 처리 유닛, 레지스터 크기, 레지스터 사용정보, 명령어 행위, 어드레싱 모드 등에 대한 프로파일 정보를 얻는다. 아키텍처 탐색의 수행 단계가 증가할수록 더욱 구체적인 프로파일 정보를 얻어야 하므로 성능 측정의 높은 정확도를 가진 성능 측정 도구를 사용한다.

그림 5는 프로파일 데이터베이스에서 현재 아키텍처 탐색의 수행 단계에 해당되는 관련 프로파일을 분석하여 개선된 임베디드 코어의 설계 정보를 생성하는 프로파일 분석기를 보인다.

제안된 프로파일 분석기는 프로파일 분석을 통해 연관규칙을 찾아내는 데이터마이닝 기법을 적용한다[18]. 데이터마이닝 블록은 임베디드 코어의 설계 모듈에 관련 있는 성능 데이터를 가진 프로파일을 분석 과정에 의해 설계 모듈과 성능 매개변수에 대한 연관 규칙을 구하여 연관 규칙 저장소에 저장한다. 프로파일 분석기의 라이브러리는 임베디드 코어의 설계에서 고려해야 할 설계 모듈에 대한 우선순위를 구성한다. 추론 엔진은 데이터마이닝 블록의 연관 규칙 저장소와 라이브러리를 참조하여 패턴 매칭을 수행하며 추론할 수 없는 연관 규칙을 데이터마이닝 블록의 연관 규칙 저장소에 업데이트한다. 추론 엔진에서 임베디드 코어의 성능을 향상시킬 수 있는 새로운 연관 규칙을 더 이상을 구할 수 없으면 본 과정을 완료하고 다음 탐색 과정에서 임베디드 코어의 성능을 향상시킬 수 있는 설계 모듈의 정보를 가진 새로운 연관 규칙을 얻는다. 얻어낸 연관 규칙을 바탕으로 SMDL 언어로 기술된 타겟 프로세서를 개선한다.

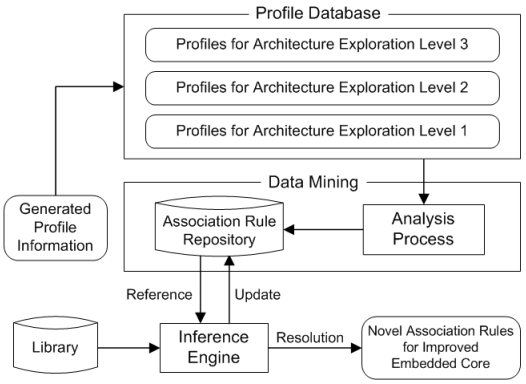

그림 5. 제안된 프로파일 분석기

Ⅳ. 실험 결과

제안된 아키텍처 탐색 시스템의 효율성 검증을 위해 JPEG 인코더, Chen-DCT, FFT에 최적화된 임베디드 코어를 설계하였다. MIPS R3000 프로세서를 SMDL 언어로 기술하여 초기 프로세서로 사용하였다. MIPS R3000 프로세서는 IF, ID/OF, ALU, MEM, WB 스테이지로 5단 파이프라인 구조를 가지며 ALU, load, store, jump branch, condition branch 명령어 종류로 구성된 명령어 집합을 가진다.

JPEG 인코더에 최적화된 초기 임베디드 코어를 얻기 위해 아키텍처 탐색 1단계를 수행하였다. 아키텍처 탐색 1단계는 산술 논리 유닛, 분기 처리 유닛 등의 구조에 따른 컴파일러의 명령어 선택기를 자동 생성하기 위해 명령어 종류에 따른 수행 빈도를 가진 basic-block 수준의 프로파일을 얻어낸다. 생성된 프로파일은 컴파일러 인스트럭션 스케줄러를 이용하여 JPEG 인코더 프로그램을 정적으로 분석한 결과인 각 basic-block 수행 빈도, basic-block의 명령어 종류별 개수와 스케줄 지연시간, 조건 분기의 taken과 non-taken 횟수, 각 basic-block의 수행 빈도와 스케줄 지연시간을 곱하여 총합한 전체 수행 사이클의 값 등을 가진다. 그림 6은 MIPS R3000의 초기 임베디드 코어에서 명령어 종류에 따른 수행 빈도를 보인다.

Basic-block 수준의 프로파일을 입력으로 한 프로파일 분석기는 ALU와 load 명령어 종류의 수행 빈도가 전체 75.4% 이고 조건 분기 명령어 종류에서 총

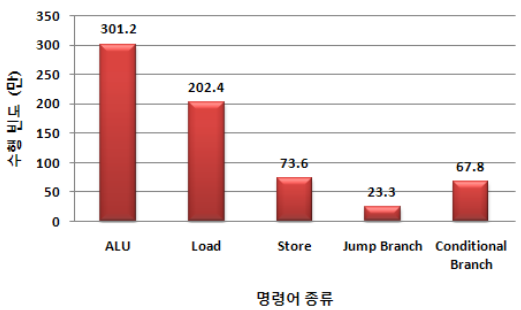

그림 6. MIPS R3000의 초기 임베디드 코어에서 명령어 종류에 따른 수행 빈도

1762

454,264번 not-taken과 총 50,307번 taken의 횟수 등의 결과를 얻었다. JPEG 인코더 프로그램의 이진코드 크기와 전체 수행 사이클의 값을 감소하기 위해 ALU와 load 명령어 종류에서 많이 수행하는 명령어 중심으로 합성된 명령어 집합을 구성하고 산술 유닛 구조를 개선하였다[14]. PC 블록을 포함한 분기 처리 유닛은 non-taken 횟수가 전체 90%를 차지하므로 not-taken 중심으로 분기 타겟 주소를 미리 계산하는 구조를 가진다. 이러한 과정을 반복하여 프로파일 분석기에서 더 이상 최적화할 필요가 없다는 분석 결과가 나오면 아키텍처 탐색 2단계로 수행하였다. 2단계에 사용되는 명령어 수준 컴파일드 코드 방식의 시뮬레이터는 파이프라인 스테이지의 구체적인 구조적 정보와 행위 정보를 고려하지 않고 빠른 시뮬레이션의 수행 속도를 가지므로 레지스터 데이터의 의존성 등을 최소화하여 개선된 레지스터 파일, 데이터패스 구조 등을 빠르게 설계할 수 있다. 레지스터 파일 구조의 설계는 가장 많이 수행되는 명령어 패턴을 중심으로 컴파일러 생성 시스템에서 사용되는 레지스터 용도에 따른 레지스터 크기를 결정하였다[8][14]. JPEG 인코더 프로그램을 고려한 레지스터 파일 구조는 전체 수행 사이클의 값을 일정 비율로 감소할 때까지 MIPS R3000 프로세서의 레지스터 파일 구조에서 temporary 레지스터 7개와 argument 레지스터 2개로 정하여 불필요한 레지스터의 개수를 최소화하였다. Temporary 레지스터는 전체 수행 사이클의 값을 감소시키기 위해 common sub-expression elimination 등의 코드 최적화를 수행하여 연산 과정의 임시 데이터를 저장하기 위한 레지스터로 사용한다[14]. Argument 레지스터는 함수 호출의 매개변수 전달을 위한 레지스터이며, JPEG 인코더 프로그램의 함수들이 3개 이상의 argument 레지스터를 필요하지 않다. 합성된 명령어 집합과 레지스터 파일 구조를 바탕으로 데이터패스 구조를 재구성하고 최적화된 컴파일러의 인스트럭션 스케줄러를 자동 생성하였다. 마지막으로 아키텍처 탐색 3단계는 전체 수행 사이클의 값을 감소시킬 수 있도록 파이프라인 스테이지별 구조를 개선한다. 개선된 파이프라인 구조는 ID/OF 스테이지에서 분기 처리 유닛을 배치하여 non-taken 분기를 수행하고 ALU 스테이지에서 자주 수행하는 명령어 패턴을 고려한 산술 유닛과 재구성된 데이터패스 등의 구조를 배치하였다. 또한 데이터 포워딩 등과 같은 최적화된 임베디드 하드웨어 기법을 이용하여 파이프라인 세부구조를 개선하고 최종 임베디드 코어를 생성하였다. 최종 임베디드 코어는 특정 어플리케이션에 최적화된 하드웨어 자원과 수행 사이클 등을 가지며 제안된 아키텍처 탐색 방법에서 생성된 컴파일러를 이용하면 다른 응용 프로그램의 소스코드를 최종 임베디드 코어의 구조에 적합한 머신 코드로 변환하므로 보다 많은 어플리케이션 적용이 가능하다.

표 1은 JPEG 인코더, Chen-DCT, FFT 어플리케이션에 따른 MIPS 3000의 임베디드 코어와 제안된 아키텍처 탐색 방법을 수행하여 얻은 최종 임베디드 코어의 성능 비교를 보인다. 측정된 임베디드 코어의 수행 사이클은 사이클 수준 컴파일드 코드 방식의 시뮬레이터로 시뮬레이션을 수행한 결과이다. 제안된 아키텍처 탐색 방법을 수행하여 얻은 최종 임베디드 코어는 MIPS R3000의 초기 임베디드 코어에 불필요한 하드웨어 자원을 최소화하기 위해 특정 어플리케이션에 최적화된 산술 연산 유닛과 분기 처리 유닛의 구조, 레지스터 파일 구조, 데이터패스 구조, 파이프라인 구조 등을 개선되었으며, MIPS R3000의 초기 임베디드 코어에 비해 평균 60.8% 수행 사이클이 감소된 결과를 보인다. 또한 컴파일러 기반의 아키텍처 탐색 방법[8]을 수행하여 얻은 JPEG 인코더 어플리케이션에 최적화된 임베디드 코어와 비교하면 평균 11.7% 수행 사이클의 값이 감소되었다. 설계 계층을 이용한 아키텍처 탐색 방법은 설계 모듈별로 적합한 컴파일러의 최적화 기법을 적용하여 향상된 수행 사이클을 가진 임베디드 코어를 설계할 수 있다.

그림 7은 세 가지 어플리케이션에 대해 제안된 아키텍처 탐색 방법으로 최종 임베디드 코어를 얻을 때까지 각 아키텍처 탐색의 수행 단계에서 총 반복 횟수 비교를 보인다.

아키텍처 탐색 1단계의 총 반복 횟수를 1로 놓고 아키텍처 탐색 2단계와 3단계의 상대적인 총 반복 횟수를 구하였다. 아키텍처 탐색 3단계는 1단계에 비해 평균 11.3배 총 반복 횟수를 가지며 3단계는 2단계에 비해 평균 3.9배 총 반복 횟수를 가진다. 아키텍처 탐색의 수행 단계가 증가할수록 임베디드 코어, 컴파일러와 컴파일드 코드 방식의 시뮬레이터의 생성과 프

표 1. 어플리케이션별 임베디드 코어의 성능 비교

어플리케이션	MIPS R3000의 초기 임베디드 코어 (# cycles)	개선된 임베디드 코어 (# cycles)	비고
JPEG 인코더	6,635,496	2,592,952	-60.9%
Chen-DCT	191,742	81,345	-57.6%
FFT	53,178	22,439	-57.8%
Average	2,293,472	898,912	-60.8%

1763

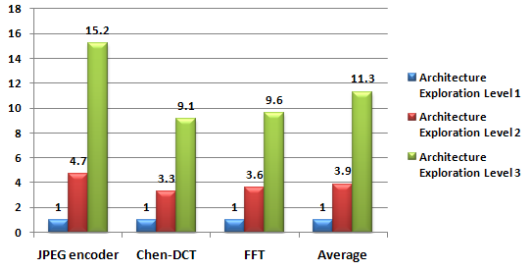

그림 7. 아키텍처 탐색의 수행 단계에 따른 총 반복 횟수 비교

로파일 분석 과정의 총 반복 횟수가 증가하였다. 이 결과를 통해 제안된 아키텍처 탐색 방법은 보다 짧은 time-to-market을 위해 아키텍처 탐색 2단계와 3단계의 반복 횟수를 줄어야하나 빠른 시뮬레이션 속도를 가진 컴파일드 코드 방식의 시뮬레이터로 체계적으로 설계 공간을 축소시키면서 컴파일러의 최적화된 명령어 선택기와 인스트럭션 스케줄러를 생성할 수 있다는 점에서 의의가 있다.

V. 결론 및 추후과제

본 논문은 설계 과정에서 설계 hierarchy에 따라 다른 탐색 방법을 갖도록 하여 효율적으로 탐색 공간을 축소시키고 불필요한 성능 측정과정을 최소화하는 아키텍처 탐색 방법을 제안하였다. 아키텍처 탐색 1단계는 컴파일러의 인스트럭션 스케줄을 이용하여 빠르게 성능 측정된 프로파일을 바탕으로 컴파일러의 명령어 선택기를 결정하고 초기 임베디드 코어를 얻었다. 아키텍처 탐색 2단계는 컴파일러의 인스트럭션 스케줄러를 결정하기 위해 1단계보다 높은 성능 측정의 정확성을 가진 명령어 수준 컴파일드 코드 방식의 시뮬레이터를 수행하였고, 3단계는 임베디드 코어의 파이프라인 구조를 중심으로 사이클 수준 컴파일드 코드 방식의 시뮬레이터를 수행하여 프로파일링하고 최적화된 임베디드 코어를 생성하였다. 초기 임베디드 코어를 빠르게 결정할 수 있으며, 임베디드 코어의 설계 모듈에 관련 있는 성능 데이터를 분석하는 프로파일 분석기를 제안하였다. 프로파일 분석기는 프로파일 데이터베이스에서 현재 아키텍처 탐색의 수행 단계에 해당되는 관련 프로파일을 분석 과정과 추정 과정을 통해 임베디드 코어의 성능을 향상시킬 수 있는 설계 모듈의 정보를 가진 새로운 연관 규칙을 찾아낸다. 제안된 아키텍처 탐색 방법을 이용하여 JPEG 인코더, Chen-DCT, FFT 어플리케이션에 최적화된 임베디드 코어를 설계하였으며, 그 결과는 MIPS R3000의 초기 임베디드 코어에 비해 평균 60.8% 수행 사이클이 감소되었다.

추후 과제는 제안된 아키텍처 탐색 방법으로 얻은 임베디드 코어와 특정 어플리케이션에 최적화된 IP (Intellectual Property) 등을 고려하여 버스 구조 등을 결정하는 확장된 아키텍처 탐색 연구가 필요하다.

참 고 문 헌

〔1〕 J. Rabaey and M. Pedram, Eds., *Low Power Design Methodologies*, Kluwer Academic Pub., 1996.

〔2〕 T. Makimoto and Y. Sakai, "Evolution of Low Power Electronics and Its Future Applications", *in Proc. ISLPED,* pp.2-5, Aug. 2003.

〔3〕 T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System", *IEEE J. Solid-State Circuits*, Vol.35, No.11, pp.1571-1580, Nov. 2000.

〔4〕 L. Benini, A. Bogliolo, and G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management", *IEEE Trans. VLSI Systems*, Vol.8, No.3, pp.299-316, June 2000.

〔5〕 M. Jain, M. Balakrishnan, and A. Kumar, "ASIP Design Methodologies : Survey and Issues", *in Proc. IEEE/ACM Int. Conf. VLSI Design. (VLSI 2001)*, pp.76-81, Jan. 2001.

〔6〕 S. Lee, S. Lee, and S. Hwang, "A Concurrent Instruction Scheduling and Recoding Algorithm for Power Minimization in Embedded Systems", *IEICE Transactions on Information and Systems*, Vol.93-D, No.8, Aug 2010.

〔7〕 M. Gries and K. Keutzer, Eds., Building ASIPs: The Mescal Methodology, Springer, 2005.

〔8〕 이성래, 황선영, "Application에 최적의 ASIP 설계를 위한 효율적인 Architecture Exploration 방법", 한국통신학회논문지, 32권, 9호, pp.913-921, 2007년 9월.

〔9〕 K. Karuri, Al Faruque, S. Kraemer, R. Leupers, G. Ascheid, and H. Meyr, "Fine-grained Application Source Code Profiling for ASIP Design", *in Proc. Design Automation*

1764

Conference, pp.329-334, June 2005.

〔10〕 L. Cai, A. Gerstlauer, and D. Gajski, “Retargetable Profiling for Rapid, Early System-level Design Space Exploration”, *in Proc. Design Automation Conference*, pp.281-286, July 2004.

〔11〕 T. Kempf, K. Karuri, S. Wallentowitz, G. Ascheid, R. Leupers and H. Meyr, “A SW Performance Estimation Framework for Early System-level -design using Fine-grained Instrumentation”, *in Proc. Conf. Design Automation and Test in Europe*, pp.468-473, March 2006.

〔12〕 A. Shrivastava, S. Park, E. Earlie, N. Dutt, A. Nicolau and Y. Paek, “Automatic Design Space Exploration of Register Bypasses in Embedded Processors”, *IEEE Trans. Computer-Aided Design*, Vol.26, No.12, pp.2102-2115, Dec. 2007.

〔13〕 조재범, 유용호, 황선영, “임베디드 프로세서 코어 자동생성 시스템의 구축”, 한국통신학회논문지, 30권 6A호, pp.526-534, 2005년 6월.

〔14〕 이성래, 황선영, “머쉰 행위기술로부터 Retargetable 컴파일러 생성시스템 구축”, 한국통신학회논문지, 32권 5호, pp.286-294, 2007년 5월.

〔15〕 홍성민, 박창수, 황선영, “DSP 프로세서용 인스트럭션 셋 시뮬레이터 자동생성기의 설계에 관한 연구”, 한국통신학회논문지, 제32권. 9호, pp. 931-939, 2007년 9월.

〔16〕 M. Reshiadi, N. Bansal, P. Mishra, and N. Dutt, “An Efficient Retargetable Framework for Instruction-Set Simulation”, *in Proc. IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign & System Synthesis*, pp.13-18, Oct. 2003.

〔17〕 T. Conte and C. Gimarc, Fast Simulation Of Computer Architectures, Kluwer Academic Pub., 1995.

〔18〕 J. Han, M. Kamber and J. Pei, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2005.

김 상 우 (Sang-Woo Kim) 준회원

2009년 2월 서강대학교 전자공학과

2009년 3월~현재 서강대학교 전자공학과 석사과정

<관심분야> ASIP Design, Retargetable Compiler for Embedded System

황 선 영 (Sun-Young Hwang) 정회원

1976년 2월 서울대학교 전자공학과

1976년 2월 한국과학원 전기 및 전자공학과 공학석사 취득

1986년 10월 미국 Stanford대학교 전자공학 박사학위 취득

1976년~1981년 삼성 반도체(주) 연구원, 팀장

1986년~1989년 Stanford 대학 Center for Integrated Systems 연구소 책임 연구원 및 Fairchild Semiconductor, Palo Alto Research Center 기술자문

1989년~1992년 삼성전자(주) 반도체 기술자문

2002년 4월~2004년 3월 서강대학교 정보통신대학원장

1989년 3월~현재 서강대학교 전자공학과 교수

<관심분야> SoC 설계 및 framework 구성, CAD 시스템, Embedded 시스템, DSP 시스템 설계 등