논문 13-38B-01-02

한국통신학회논문지 '13-01 Vol.38B No.01
http://dx.doi.org/10.7840/kics.2013.38B.1.10

동적 분기 예측을 지원하는 임베디드 코어 자동 생성 시스템의 설계와 구현

이 현 철•, 황 선 영°

Design and Implementation of an Automatic Embedded Core Generation System Using Advanced Dynamic Branch Prediction

Hyun-cheol Lee•, Sun-young Hwang°

요 약

본 논문은 분기 예측을 지원하는 임베디드 코어 자동 생성 시스템을 제안한다. 제안된 시스템은 동적 분기 예측 모듈에 히스토리/분기방향 flag가 추가된 BTAC(Branch Target Address Cache)를 포함하여 타겟 어플리케이션의 수행 속도를 향상 시킬 수 있도록 하였다. 시뮬레이션으로부터 해당 어플리케이션의 분기 정보를 추출하고 이를 토대로 BHT(Branch History Table)와 BTAC의 entry를 결정한다. 제안된 분기 예측의 효율성을 검증하기 위해서 동적 분기 예측 모듈을 포함하는 ARM9TDMI 코어를 SMDL로 기술하고 코어를 생성하였다. 실험 결과는 entry의 수에 따라 면적은 60%까지 증가하였고 어플리케이션의 수행 사이클과 BTAC의 miss rate는 평균 1.7%, 9.6%씩 감소하였다.

Key Words : ASIP, MDL, Branch Prediction, BHT, BTAC

ABSTRACT

This thesis proposes an automatic embedded core generator system that supports branch prediction. The proposed system includes a dynamic branch prediction module that enhances execution speed of target applications by inserting history/direction flags into BTAC(Branch Target Address Cache). Entries of BHT(Branch History Table) and BTAC are determined based on branch informations extracted by simulation. To verify the effectiveness of the proposed branch prediction module, ARM9TDMI core including a dynamic branch predictor was described in SMDL and generated. Experimental results show that as the number of entry rises, area increase up to 60% while application execution cycle and BTAC miss rate drop by an average of 1.7% and 9.6%, respectively.

Ⅰ. 서 론

최근 반도체 기술이 빠르게 발전됨에 따라 VLSI 설계 방법은 단순히 회로 설계 수준을 넘어서 하나의 시스템을 설계하는 수준으로 그 범위가 확장되었다. 이에 따라 하나의 칩에 시스템을 집적한다는 의미를 지닌 System on Chip(SoC)이라는 용어가 학계뿐만이 아니라 업계에서도 많이 사용되고 있다. 임베디드 시스템에 사용되는 SoC는 제한된 자원으로 최고의 성능을 요구하기 때문에 이들의 trade-off를 고려하면서 Time-To-Market 요구를 만족시키는 것은 상당히 어려운 문제이다. 이러한 문제점을 해

※ 본 연구는 교육과학기술부의 재원으로 한국연구재단의 지원(#2012-0002586)에 의해 수행되었으며, IDEC에서 제공한 CAD tool을 이용해 simulation을 수행 하였습니다.
• 주저자 : 서강대학교 전자공학과 CAD & ES 연구실, hclee@sogang.ac.kr, 준회원
° 교신저자 : 서강대학교 전자공학과 CAD & ES 연구실, hwang@sogang.ac.kr, 종신회원
논문번호 : KICS2012-11-551, 접수일자 : 2012년 11월 26일, 최종논문접수일자 : 2012년 12월 14일

결하기 위해 최근에는 고정된 범용 프로세서를 사용하는 대신 설계물의 유연성과 재사용률을 높이고 어플리케이션에 최적화할 수 있는 Application Specific Instruction-set Processors (ASIP)의 사용이 증가하고 있다[1,2].

최근까지 Machine Description Language(MDL) 기반의 ASIP 설계 자동화 시스템에 관한 연구가 활발히 진행되어 왔다. 기존에 개발된 MDL은 Technical University of Berlin의 nML[3], Aachen University of Technology의 LISA[4], UCI의 EXPRESSION[5], 오사카 대학의 Micro Operation Description[6], MIT의 ISDL[7] 등을 대표적인 예로 들 수 있다. nML은 언어 자체에서 파이프라인을 지원하지 않고, 이를 활용하여 간단한 컨트롤러만 생성할 수 있다. 그렇기 때문에 파이프라인 기능이 있는 프로세서 생성에는 어려움이 있다. LISA는 nML의 단점을 개선하여 파이프라인 기술이 가능하며, DSP 계열 프로세서에 대한 기술이 가능하다. 그러나 코어 생성 측면에서 데이터패스와 디코딩 정보만을 추출하여 고정된 파이프라인 컨트롤러에 맵핑하는 방식을 취하고 있기 때문에 다양한 머신 구조를 생성하는 데에는 한계가 있다. 또한, 설계자가 해저드 검출 및 처리에 대한 정보를 자세히 기술하도록 하여 프로세서 개발 초기부터 타겟 프로세서에 대해 구체적이고 정확한 기술이 요구된다. EXPRESSION은 MDL을 근간으로 코어를 생성하는 기존의 ASIP 설계 자동화 시스템과는 용도와 구조가 다른 시스템이다. 기존에 존재하는 범용 프로세서 IP 정보를 이용하여 프로세서 라이브러리를 구축하고 이 라이브러리에 있는 프로세서를 EXPRESSION으로 기술하여 시뮬레이터와 컴파일러 등의 툴 셋을 자동 생성하는 방식을 취하고 있다. 이러한 특이한 시스템 구조 때문에 EXPRESSION 시스템은 새로운 프로세서를 모델링하여 자동 생성하기보다는 이미 존재하는 코어를 이용하여 타겟 시스템을 설계하는 것에 적합하다는 특징이 있다. Micro Operation Description은 기술의 추상화 수준이 낮은 RTL 수준의 기술이 필요하므로, 타겟 프로세서의 컴파일러 개발에 제약이 있고, 블록 데이터 전송을 위한 멀티사이클 인스트럭션, 파이프라인 해저드 처리가 고려되지 않았다.

지금까지 개발된 MDL 기반의 ASIP 설계 자동화 시스템은 대부분 분기 예측을 지원을 하지 않았다. MIT에서 개발한 ISDL의 경우도 하드웨어의 지원이 필요한 동적 분기 예측이 아닌 런타임 이전에 컴파일러를 통한 정적 분기 예측만을 지원한다. 본 연구에서는 설계자가 MDL 기반의 설계 방식인 SMDL (Sogang Machine Description Language)로 타겟 프로세서를 기술하여 코어를 생성하는 과정에서 어플리케이션의 수행속도를 향상 시킬 수 있도록 면적 대비 높은 정확도를 가지는 동적 분기 예측 모듈을 자동으로 생성하는 과정을 추가하였다.

논문의 구성은 다음과 같다. 2절에서는 연구의 배경이 되는 분기 예측과 구현 Platform에 대해 설명하고, 3절에서는 성능과 면적을 동시에 고려한 동적 분기 예측 모듈 생성 과정을 제시한다. 4절에서는 제안된 시스템의 검증을 위해 기존의 BTAC로 구현된 동적 분기 예측 모듈을 사용하여 생성된 코어와 제안된 BTAC로 구현된 동적 분기 예측 모듈을 사용하여 생성된 코어의 어플리케이션 수행속도와 Cache Miss Rate를 비교하였다. 5절에서는 결론 및 추후 과제를 제시한다.

Ⅱ. 연 구 배 경

본 절에서는 기존의 분기 예측에 관한 관련 연구를 소개하고 제안된 동적 분기 예측기가 구현될 SMDL 시스템에 대해 전체적인 구성요소들과 ASIP에 적합한 설계 도구 생성과정에 대해 간략하게 기술한다.

2.1. Branch Prediction 연구

분기 예측은 크게 정적 분기 예측과 동적 분기 예측 두 가지 방법으로 나뉜다. 정적 분기 예측이란 런타임 이전에 컴파일러를 통해 분기를 예측하는 방법으로서 일반적으로 모든 분기에 대해 Taken 혹은 Not Taken으로 일관되게 분기를 예측한다. 정적 분기 예측 방법은 특정한 하드웨어의 추가 없이 구현이 가능하고 추가적인 하드웨어나 전력 소모에 대한 고려가 필요 없다. 하지만 분기 예측에 대한 정확도가 다른 분기 예측 방법에 비해 낮다는 단점이 있다. 동적 분기 예측은 어플리케이션을 수행하는 과정에서 입력된 Branch 인스트럭션에 대해 이전까지의 분기 결과를 바탕으로 현재의 분기를 Taken할 것인지 혹은 Not Taken할 것인지를 결정하는 방법이다. 대표적인 동적 분기 예측 방법에는 Bimodal Predictor, Gshare Predictor, Combined Bimodal Predictor, Tournament Predictor가 있다[8]. 동적 분기 예측기는 기존의 분기 예측결과를 프로세서 내부에서 사용하기 때문에 빠른 속도와 큰 면

적을 가지는 캐쉬와 같은 전용 메모리가 필수적이다. 그러나 BHT와 BTAC같은 테이블의 증가는 테이블의 인덱싱 시간을 증가시킴과 더불어 전체적인 타겟 프로세서의 면적을 증가시킨다[9]. 이러한 이유 때문에 동적 분기 예측기의 하드웨어적 구현은 성능과 면적의 Trade-off를 신중하게 고려해야 한다. 지금까지의 임베디드 코어는 면적과 전력 소모의 제한 때문에 주로 정적 분기 예측 방법을 이용하였다. 하지만 임베디드 코어 성능이 향상되고 파이프라인의 Stage가 증가함에 따라 분기 예측 실패에 따른 페널티가 증가되면서 임베디드 코어에서도 정확성 높은 동적 분기 예측의 필요성이 강조되고 있다[10,11].

2.2. 구현 Platform

SMDL 시스템은 MDL을 이용한 ASIP 설계 자동화 시스템으로서, 합성 가능한 타겟 프로세서를 자동 생성하는 ECGen(Embedded Core Generator), 생성된 프로세서에서 수행 가능한 최적화된 어셈블리 코드를 생성하는 컴파일러를 생성해주는 SRCC(Sogang Retargetable C Compiler), 그리고 타겟 프로세서의 인스트럭션 셋 시뮬레이터를 자동 생성하는 RISGen (Retargetalbe Instruction-set Simulator Generator)로 구성된다. 타겟 프로세서의 SMDL 기술은 파싱을 통해 각 하위 기능 블록들에 적합한 중간 형태로 변환된다. 생성된 중간 형태는 SMDL Description에 기술한 내부/외부 구조 정보 및 인스트럭션에 대한 정보를 그대로 유지하며 하위 기능 블록인 SRCC, RISGen, ECGen의 입력으로 이용된다. SRCC를 통해 생성되는 C 컴파일러에 C 언어로 기술된 어플리케이션을 입력하여 어플리케이션의 바이너리 코드를 얻는다. 변환된 바이너리 코드는 RISGen을 통해 생성되는 인스트럭션 셋 시뮬레이터를 통해 검증되며, ECGen을 통해 생성되는 임베디드 코어 모델은 시뮬레이션 결과와 함께 HDL 코드 생성기에 입력으로 이용된다. 생성기들을 통해 생성된 HDL 모델과 시뮬레이터를 이용하여 타겟 ASIP 및 어플리케이션에 대한 검증을 수행한다[12,13]. 제안된 시스템의 평가를 위해 상용화된 임베디드 코어 중 ARM9TDMI를 SMDL로 기술하고 구축된 시스템을 통해 프로세서 모델을 생성한 뒤, 각각의 모델을 실제 프로세서의 동작과 비교하여 생성된 프로세서 모델의 정확성을 검증하였으며 어플리케이션을 실행하여 동적 분기 예측 모듈을 추가하지 않은 SMDL 생성 프로세서 모델과 수행 속도 및 면적을 비교하였다.

Ⅲ. 성능과 면적을 동시에 고려한 동적 분기 예측 모듈 생성 과정

제안된 동적 분기 예측 모듈 생성 과정은 어플리케이션에 대한 분석 과정, BTAC(Branch Target Address Cache) 생성과정, 생성된 BTAC를 이용한 동적 분기 예측 모듈 생성 과정으로 나뉘어진다. 어플리케이션에 대한 분석과정을 통해 어플리케이션의 분기 명령어의 수, 분기 명령어의 종류에 대한 정보를 추출한다. BTAC 생성과정은 칩의 면적과 분기 예측을 통한 수행 속도 향상을 고려하여 BTAC 모듈을 생성하고 생성된 BTAC 모듈과 선택된 BHT(Branch History Table)를 이용해 동적 분기 예측 모듈을 구성한다. 생성된 분기 예측 모듈은 타겟 프로세서에 포함되어 ARM9TDMI 코어로 구현된다.

3.1. 어플리케이션 분석 과정

ASIP 설계에서 가장 중요한 부분은 프로세서 생성 이전에 어플리케이션에 대해 분석하고 코어 설계 이전에 성능 향상을 고려해야 한다는 점이다. 생성된 ASIP에서 수행될 어플리케이션의 분기 명령어 수, 분기 명령어의 종류에 대한 정보를 시뮬레이션을 통해 추출한다. 그림 1은 Profile을 생성하고 분기 정보를 추출하는 일련의 과정을 나타낸다. C언어로 기술된 어플리케이션을 SRCC로 생성된 Retargetable 컴파일러를 이용해 컴파일하고 생성된 바이너리 코드를 RISGen을 통해 생성된 Retargetable 시뮬레이터에 입력하여 Profile을 얻는다. 생성된 Profile 중에서 분기 예측 모듈 생성과정에 필요한 정보를 임베디드 코어 생성기에 입력하는 과정을 확인할 수 있다.

3.2. 동적 분기 예측 모듈 생성 과정

기존의 SMDL 시스템의 코어 생성 과정은 SMDL 기술만을 이용하여 코어를 생성하기 때문에 BTAC 모듈 생성 과정이 포함되어 있지 않다. 본 논문에서는 생성된 ASIP의 수행 속도 향상과 분기 예측 모듈의 면적을 감소시키기 위한 분기 예측 모듈 생성과정을 추가하였다. 그림 2는 이러한 분기 예측 모듈 생성 부분을 추가한 코어 생성기의 개관을 보인다. 동적 분기 예측 모듈은 분기를 정확하게 예측하기 위하여 캐쉬와 같은 빠른 메모리를 코어

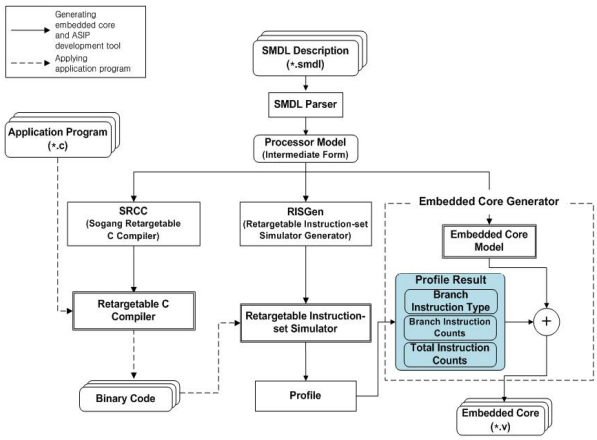

그림 1. Profiling을 통한 어플리케이션 정보 추출 과정
Fig. 1. Process of extracting application information through profiling

내부에 추가한다. 캐쉬는 빠른 속도로 데이터를 처리한다는 이점을 가지고 있지만, 상대적으로 많은 공간을 차지하기 때문에 동적 분기 예측의 정확도를 높이기 위해서 단순히 큰 테이블을 사용할 수는 없다. 제안하는 동적 분기 예측 모듈은 BHT에 비해 상대적으로 많은 공간을 차지하는 BTAC의 면적을 줄이면서도 어플리케이션의 수행 속도를 향상을 시킬 수 있는 방법을 제시한다.

3.2.1. BTAC의 생성과정

대부분의 동적 분기 예측기는 BHT와 BTAC의 entry를 적절히 조율하여 분기 예측기의 정확도를 높인다. General Purpose Computer의 경우에는 면적에 대한 제약이 크지 않기 때문에 테이블의 크기를 증가시켜 BTAC를 사용함으로써 공간의 제한으로부터 임베디드 코어보다 자유롭다는 이점이 있다. 하지만 임베디드 코어는 면적과 전력 소모의 증가에 상당히 민감한 특성을 가지고 있기 때문에 적절한 Trade-off 지점을 찾기가 쉽지 않다. 이러한 문제를 해결하기 위해서 분기 명령어를 다음과 같이 세분화된 종류로 분류한다. 분기 명령어는 크게 분기 명령어의 80% 정도를 차지하는 조건 분기와 나머지 20%를 차지하는 무조건 분기가 있다. 조건 분기는 다시 Forward Branch와 Backward Branch로 나뉘는데 Forward Branch는 조건 분기 명령어의 70%를 차지하지만, Not taken되는 경우가 대부분이고 실질적으로는 30% 정도만이 Taken된다. Backward Branch의 경우는 조건 분기 명령어의 30%를 차지하지만, 대부분의 분기가 Taken되는 특징을 가지고 있다[14]. 이러한 분기 명령어들의 특징들을 이용해 기존보다 적은 entry로 높은 캐쉬 히트

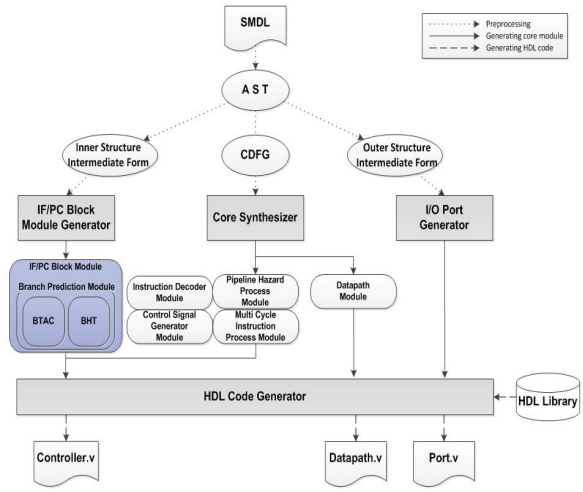

그림 2. 동적 분기 예측기를 포함한 코어 생성기 시스템의 개관
Fig. 2. Overview of core generation system including dynamic branch prediction module

율을 보이는 BTAC를 구성하면 그림 3과 같다.

제안된 BTAC는 기본적으로 4 way set-associative 형태의 Cache를 사용한다. 하지만 기존의 BTAC와는 다르게 분기 명령어의 Type을 지정하는 T bit와 해당 entry가 참조된 횟수를 나타내는 Reference Count(RC) bit가 존재한다. T bit는 각각 entry에 저장된 분기 명령어가 Backward Branch이면 1, Forward Branch이면 0을 저장하고 RC bit는 각각의 entry에 저장된 Tag가 인덱싱되어 Target Address가 참조될 때마다 1씩 증가한다. 이러한 T, RC bit는 평상시엔 단순히 Flag의 역할을 한다. 하지만 모든 entry에 데이터가 저장되어 기존의 Address Tag를 지우고 새로운 Tag를 추가하는 과정에서는 T, RC bit는 priority를 나타내는 용도로 사용된다. 이러한 방법을 사용함으로써 한정적인 entry의 활용도를 높이고 Target Address의 연산 시간을 줄일 수 있다. Backward Branch이면서 7번 이상 참조된 entry의 경우는 1111로 가장 높은 priority로 지정되고 Backward Branch이면서 한 번도 참조되지 못한 entry의 경우는 1000, Forward Branch이면서 7번 이상 참조된 entry의 경우는 0111, Forward Branch이면서 한 번도 참조되지 못한 entry의 경우엔 0000 priority를 가지게 된다. 각 entry의 우선순위를 참조하여 가장 낮은 우선순위의 entry에 Address Tag, T Flag와 Target Address를 저장함으로써 자주 사용되지 않는 entry를 삭제하고 BTAC의 활용도를 높일 수 있다. 위와 같은 priority 기반의 entry 교체 방법은 분기 명령어가 많은 어플리케이션에 대해서 모든 entry의 priority

가 1111이 되는 경우가 발생할 수 있다. 이러한 경우엔 모든 entry를 비우거나 RC bit를 더욱 확장하여 priority를 늘리는 방법으로 문제를 해결할 수 있지만, 앞에서 언급한 방법들은 전체 프로세서의 면적이나 소모 전력의 증가로 이어지기 때문에 제안된 BTAC에서는 모든 entry의 RC bit를 000으로 리셋시키는 Not Recently Used(NRU) 알고리즘을 적용하였다[15]. NRU 알고리즘을 적용함으로써 모든 entry의 priority를 낮춤으로써 모든 T, RC bit가 1111이 되는 경우의 문제를 해결할 수 있었다. 제안된 방법은 임베디드 코어 환경에서 별도의 하드웨어를 추가하지 않고도 문제를 해결할 수 있다는 이점을 가지고 있다.

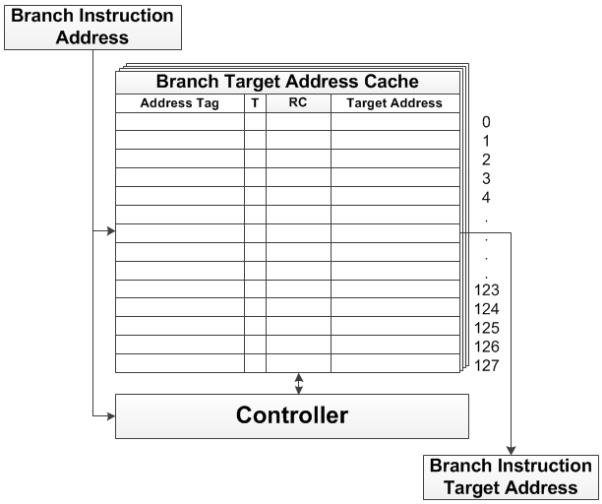

그림 3. 제안된 BTAC의 개관
Fig. 3. Overview of proposed BTAC

3.2.2. 생성된 BTAC를 이용한 동적 분기 예측 모듈 생성 과정

BTAC와 더불어 동적 분기 예측 모듈을 구성하기 위해서는 BHT가 필요하다. 분기 예측기의 특성에 따라 분기 예측 정확도가 다르기 때문에 동적 분기 예측 모듈에 사용될 BHT의 선택을 진행한다. BHT를 선택하는 과정은 크게 두 가지 단계로 나뉜다. BHT의 종류를 선택하고, 본인이 선택한 BHT를 이용해 원하는 정확도를 맞추기 위해 entry의 수를 선택하는 두 가지 단계를 수행한다. entry 수의 증가는 분기 예측의 정확도를 높여주는 대신 전력 소모와 면적도 함께 증가시키는 특성이 있다. 그중에서도 면적은 참조되는 분기 주소가 1bit가 추가될 때마다 현재 entry 수의 2배로 증가하는 특징이 있기 때문에 어떤 BHT를 선택하더라도 대부분의 경우 13bit 즉 8192 entry 이상을 선택하는 경우는 매우 드물다. 본 연구는 BHT로 인한 성능 향상이 주 목적이 아니므로 기존에 사용되던 Gshare Predictor를 동적 분기 예측 모듈에 적용한다. Gshare Predictor는 64개의 entry를 가지는 6bit의 경우에도 90% 이상의 분기 예측 정확도를 보증하기 때문에 BHT entry의 최솟값을 64로 설정하고 분기 예측의 정확도가 더는 크게 증가하지 않는 8192의 entry를 가지는 13bit를 BHT entry의 최댓값으로 설정한다. 그림 4는 ARM9TDMI 코어에 동적 분기 예측 모듈을 포함한 블록다이어그램을 보인다.

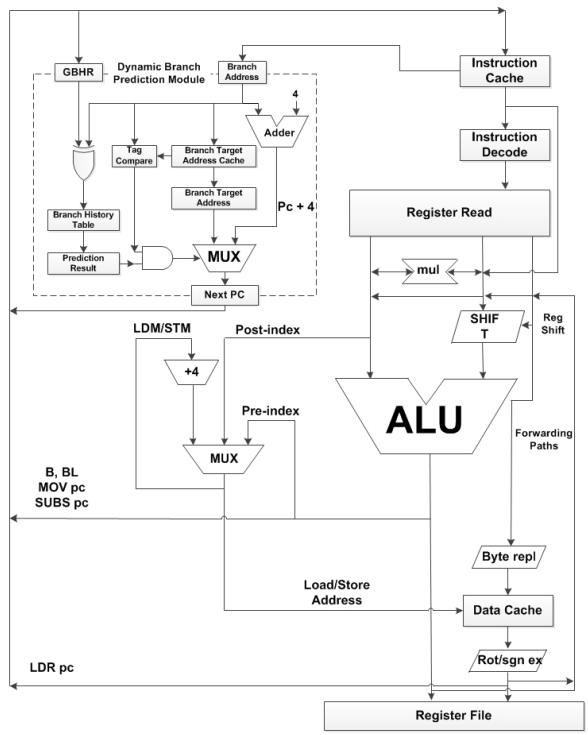

그림 4. 동적 분기 예측 모듈을 포함한 ARM9TDMI의 블록 다이어그램
Fig. 4. Block diagram of ARM9TDMI including dynamic branch prediction module

Ⅳ. 실험 결과

제안된 시스템에서 동적 분기 예측 모듈을 포함한 타겟 프로세서의 생성을 검증하기 위해 상용화된 임베디드 코어인 ARM9TDMI[16][17]를 타겟 프로세서로 정하고, 이를 SMDL 언어를 사용하여 기술하였다. 기술된 SMDL을 제안된 코어 자동 생성기에 입력하여 해당 코어를 생성하였다. 생성된 타겟 코어의 성능을 측정하기 위해 임베디드 코어용 벤치마크인 MIBENCH를 이용하였고, 면적의 감소

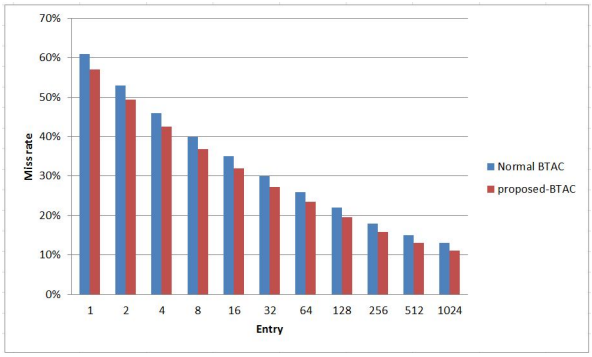

그림 5. Entry에 따른 기존 BTAC와 제안된 BTAC의 Cache Miss rate 비교
Fig. 5. Comparison of cache miss rate between original BTAC and proposed BTAC by entry alteration

와 수행속도 향상을 측정하기 위해 BTAC를 이용한 동적 분기 예측 모듈을 추가하여 결과를 비교하였다. 또한, 각각의 벤치마크는 Retargetable 컴파일러를 사용해 코드를 변환하여 사용하였다. 제안된 시스템은 SUN-Sparc 워크스테이션에서 수행되었으며, Synopsys사의 Design Vision을 이용해 합성하였다. D-Cache와 I-Cache는 오프칩 메모리를 사용한다고 가정하였고 소모 전력과 면적을 측정하기 위해 TSMC 90nm를 라이브러리를 사용하였으며, 동작 주파수는 70MHz, 동작 온도는 25℃, 동작 전압은 3.3V로 세팅하였다.

ARM9TDMI 모델을 생성하는 실험에서는 실제 코어와 마찬가지로 5단 파이프라인으로 구성되고, 클럭은 2-phase non-overlapping 클럭을 사용하였다. 기존의 타겟 프로세서는 31,260게이트로 구성되며, BHT와 BTAC의 구성에 따라 10%에서 60%까지 게이트가 증가한다. 그림 5는 4 way set-associative BTAC의 entry 수에 따른 기존의 BTAC와 제안된 BTAC의 Cache Miss rate를 나타낸다. 실험은 BTAC의 entry가 1부터 1024까지의 결과를 나타냈으며 각 entry의 Cache Miss rate는 수행된 테스트 벤치의 평균값을 나타낸다. 제안된 BTAC는 기존의 BTAC에 비해 평균 9.6% 정도 Cache Miss rate가 감소한 것을 확인할 수 있다.

표 1은 ARM9TDMI 코어에 12bit entry GShare BHT와 128 entry, 4 way set-associative BTAC로 분기 예측 모듈을 구성하였을 때, T, RC bit가 없는 기존의 BTAC와 T, RC bit가 추가된 제안된 BTAC에 대한 각각의 벤치마크 수행 사이클을 나타낸다. 제안된 BTAC는 entry의 priority를 참조함으로써 다시 참조될 확률이 높은 Address Tag와 Target Address를 중심으로 entry를 구성함으로써

표 1. ARM9TDMI에서의 벤치마크 실행 결과
Table 1. Results from executing benchmarks on ARM9TDMI

	branch ratio	Previous BTAC (# of Cycle)	Proposed BTAC (# of Cycle)	Reduction of Cycle
tiff2rgba	3.3%	55,820,480	55,506,544	0.6%
tiffmedian	4.7%	214,125,877	212,444,706	0.8%
blowfish	12.3%	72,990,635	71,500,398	2.0%
bitcount	10.3%	76,161,313	74,853,020	1.7%
ADPCM enc	3.3%	56,943,133	56,622,882	0.6%
CRC32	15.3%	81,871,923	79,806,734	2.5%
FFT	13.0%	81,144,500	79,400,669	2.1%
IFFT	12.3%	101,111,578	99,047,200	2.0%
ADPCM dec	5.0%	45,724,941	45,340,569	0.8%
GSM enc	4.3%	83,815,384	83,203,894	0.7%
GSM dec	9.7%	36,546,411	35,958,298	1.6%
stringsearch	19.3%	247,857	240,039	3.2%
ispell	17.7%	13,045,477	12,668,166	2.9%
평균	10.1%	70,734,578	69,737,932	1.7%

BTAC의 cache miss rate를 줄일 수 있으며 결과적으로 cache miss가 발생하였을 때 Target Address를 다시 연산하는데 걸리는 시간을 단축시킬 수 있다. 생성된 코어에서 벤치마크를 수행함으로써 수행 사이클이 감소되는 결과를 확인하였으며, 각 벤치마크의 분기 명령어 수가 많을수록 수행 사이클 감소량이 크다는 점 또한 확인하였다.

V. 결론 및 추후과제

본 논문은 SMDL을 이용하여 임베디드 코어를 자동 생성하는 과정에서 어플리케이션의 시뮬레이션 정보를 토대로 동적 분기 예측 모듈을 생성함으로써 BTAC의 Cache Miss rate를 감소시킬 수 있는 코어 생성 방법을 제안하였다. 제안된 시스템은 기존의 코어 생성 과정에 어플리케이션의 분석 과정, BTAC 생성과정, 생성된 BTAC를 이용한 동적 분기 예측 모듈 생성 과정을 추가하여 타겟 프로세서의 코어를 생성한다. 제안된 시스템으로부터 생성된 ARM9TDMI 프로세서에 SMDL 시스템의 Retargetable 컴파일러에 의해 변환된 어플리케이션을 실행했을 때 기존의 BTAC를 사용한 프로세서보다 제안된 BTAC를 사용한 프로세서의 인스트럭션 수행 사이클이 0.6에서 3.2%까지 감소하는 결과를 확인할 수 있었다. 분기 예측의 특성상 프로세서의 파이프라인 Stage가 증가할수록 미스 페널티는

커지지만, 분기 예측으로 인한 이득 또한 커진다는 특징을 가지기 때문에 파이프라인 Stage가 증가한 프로세서에서 제안된 분기 예측 모듈을 적용한다면 더욱 향상된 결과를 확인할 수 있을 것으로 예측된다.

추후 과제로는 매 사이클마다 동작해야 하는 동적 분기 예측기의 소모 전력을 줄이고 분기 예측의 정확도를 높일 수 있는 분기 예측기의 연구 및 개발이 필요하다.

References

[1] N. Dutt and K. Choi, "Configurable processor for embedded computing," *IEEE Comp.*, vol. 36, no. 1, pp. 120-123, Jan. 2003.

[2] K. Choi and Y. Cho, "Recent trends in the SoC design methodology," *Mag. of the Institute of Electronics Engineers of Korea (IEEK),* vol. 30, no. 9, pp. 17-27, Sep. 2003.

[3] A. Fauth, M. Fredericks, and A. Knoll, "Generation of hardware machine models from instruction set descriptions," in *Proc. IEEE Workshop VLSI Signal Proces.*, Veldhoven, Netherlands, Oct. 1993.

[4] A. Hoffmann, T. Kogel, A. Nohl, G. Braun, O. Schliebusch, O. Wahlen, A. Wieferink, and H. Meyr, "A novel methodology for the design of application-specific instruction- set processors (ASIPs) using a machine description language," *IEEE Trans. CAD of Int. Circuits and Systems*, vol. 20, no. 11, pp. 1338-1354, Nov. 2001.

[5] P. Mishra, A. Kejariwal, and N. Dutt, "Rapid exploration of pipelined processors through automatic generation of synthesizable RTL model," in *Proc. IEEE Int. Workshop on Rapid Syst. Prototyping*, pp. 226-232, San Diego, CA, Jun. 2003.

[6] M. Itoh, Y, TAKEUCHI, M. IMAI, and A. SHIOMI, "Synthesizable HDL generation for pipelined processors from a micro-operation description," *IEICE Trans.*, vol. E83-A, no. 3, pp. 394-400, Mar. 2000.

[7] G. Hadjiyiannis, S. Hanono, and S. Devadas, "ISDL: an instruction set description language for retargetability," in *Proc. Design Automation Conf.* pp. 299-302, Anaheim, CA, Jun. 1997.

[8] J. Hennessy and D. Patterson, *Computer Architecture : A Quantitative Approach*, Morgan Kaufmann Publishers Inc, 1990.

[9] L. Nadav and W. Shlomo, "Low power branch prediction for embedded application processors," in *Proc. Low Power Electronics and Design*, pp. 67-72, Austin, Texas, Aug. 2010.

[10] T. Juan, S. Sanjeevan, and J. Navarro, "Dynamic history-length fitting: a third level of adaptivity for branch prediction," in *Proc. Computer Architecture*, pp. 155-166, Barcelona, Spain, Jul. 1998.

[11] J. Lee and A. Smith, "Branch prediction strategies and branch target buffer design," *Computer*, vol. 17, no. 1, pp. 6-22, Jan. 1984.

[12] H. Lee and S. Hwang, "Design of a high-level synthesis system for automatic generation of pipelined datapath," *J. of The Institute of Electronics Engineers of Korea (IEEK)*, vol. 31-A, no. 4, pp. 53-67, Mar. 1994.

[13] J. Cho, Y. Yoo, and S. Hwang, "Construction of an automatic generation system of embedded processor cores," *J. KICS*, vol. 30, no. 6A, pp. 526-534, Jun. 2005.

[14] J. Dongarra, *High Performance Computing: Technology, Methods and Applications,* North-Holland, 1995.

[15] K. Kedzierski, M. Moreto, F. Cazorla, and M. Valero, "Adapting cache partitioning algorithms to pseudo-LRU replacement policies," in *Proc. Parallel & Distributed Processing*, pp. 1-12, Atlanta, GA, Apr. 2010.

[16] ARM, "*ARM922T Technical Reference Manual (rev 0)*," 2001.

[17] ARM, "*ARM Architecture Reference Manual (rev 0)*," 2005.

이 현 철 (Hyun-cheol Lee)

2011년 2월 서울산업대학교 전자정보공학과 학사

2011년~현재 서강대학교 전자공학과 CAD & ES 연구실 석사과정

<관심분야> Embedded System 설계, ASIP 설계

황 선 영 (Sun-young Hwang)

1976년 2월 서울대학교 전자공학과 학사

1978년 2월 한국과학원 전기 및 전자공학과 공학석사

1986년 10월 미국 Stanford 대학 전자공학 박사

1976~1981년 삼성반도체(주) 연구원, 팀장

1986~1989년 Stanford 대학 Center for Intergrated System 연구소 책임연구원 및 Fairchild Semiconductor Palo Alto Research Center 기술 자문

1989~1992년 삼성전자(주) 반도체 기술 자문

1989년 3월~현재 서강대학교 전자공학과 교수

<관심분야> SoC 설계 및 framework 구성, CAD 시스템, Embedded System 설계, Computer Architecture 및 DSP System Design 등