한국통신학회논문지(J-KICS) '13-12 Vol.38A No.12
http://dx.doi.org/10.7840/kics.2013.38A.12.1094

4-Way 캐쉬의 선택된 Element를 이용한 향상된 동적 분기 예측기 구현

황 인 성◆, 황 선 영°

An Improved Dynamic Branch Predictor by Selective Access of a Specific Element in 4-Way Cache

In-sung Hwang◆, Sun-young Hwang°

요 약

본 논문은 4-Way 캐쉬의 선택된 element만을 사용하여 어플리케이션 수행 사이클을 줄인 향상된 동적 분기 예측기를 제안한다. 제안된 동적 분기 예측기는 분기명령어가 페치되면 MRU 버퍼를 참조하여 4-Way 캐쉬의 선택된 element에서 타깃 주소를 얻으므로, 모든 element에 접근하는 기존의 동적 분기 예측기보다 제한된 전력하에서 BTAC entry 수를 증가시킬 수 있어 분기 예측 성공률과 어플리케이션의 수행속도가 상당히 향상된다. 제안된 동적 분기 예측기의 효율성을 SMDL 시스템에 의해 생성된 코어가 벤치마크 어플리케이션을 수행하여 검증한다. 실험결과 동적 분기 예측기가 없는 코어에 비해 생성된 코어의 어플리케이션 수행 사이클은 평균 10.1% 감소하고 어플리케이션의 전력소모는 7.4% 증가한다. 기존 동적 분기 예측기를 사용하는 코어에 비해 수행 사이클은 평균 4.1% 줄어든다.

Key Words : Embedded System, Branch Prediction, MRU Buffer, BTAC, MDL

ABSTRACT

This paper proposes an improved branch predictor that reduces the number execution cycles of applications by selectively accessing a specific element in 4-way associative cache. When a branch instruction is fetched, the proposed branch predictor acquires a branch target address from the selected element in the cache by referring to MRU buffer. Branch prediction rate and application execution speed are considerably improved by increasing the number of BTAC entries in restricted power condition, when compared with that of previous branch predictor which accesses all elements. The effectiveness of the proposed dynamic branch predictor is verified by executing benchmark applications on the core simulator. Experimental results show that number of execution cycles decreases by an average of 10.1%, while power consumption increases an average of 7.4%, when compared to that of a core without a dynamic branch predictor. Execution cycles are reduced by 4.1% in comparison with a core which employs previous dynamic branch predictor.

Ⅰ. 서 론

최근 스마트폰 시장이 확대되고 스마트 기술이 발달함에 따라 기존 임베디드 프로세서보다 깊은

※ 본 연구는 삼성전자(주)의 지원으로 수행되었으며 IDEC에서 제공한 CAD tool을 이용해 simulation을 수행 하였습니다.
◆ First Author : 서강대학교 전자공학과 CAD & ES 연구실, insung78@sogang.ac.kr, 학생회원
° Corresponding Author : 서강대학교 전자공학과 CAD & ES 연구실, hwang@sogang.ac.kr, 종신회원
논문번호 : KICS2013-11-502, 접수일자 : 2013년 11월 21일, 심사일자 : 2013년 12월 4일, 최종논문접수일자 : 2013년 12월 9일

파이프라인으로 구성된 고사양 임베디드 코어에 대한 수요가 증가하고 있다[1,2]. 파이프라인이 깊어질수록 분기명령의 페널티로 인해 지연되는 시간이 길어지므로 분기명령의 페널티를 줄여야 한다. 분기의 taken 여부를 결정하고 다음 타깃 주소를 결정하기까지 걸리는 시간이 분기명령 hazards에 의한 페널티가 된다. 분기명령의 페널티 감소와 코어 성능 향상을 위해 분기 예측이 사용되며[3], 정적인 방법과 동적인 방법으로 나뉜다. 정적 분기 예측은 어플리케이션 수행이전에 컴파일러로 분기를 예측하는 방법으로 일관되게 분기의 결과를 taken 혹은 not-taken으로 결정하거나 미리 결정해 놓은 테이블의 값을 사용한다[4]. 어플리케이션 수행 전에 분기 결과가 결정되므로 추가적인 하드웨어가 필요없다는 장점이 있으나 분기 예측 성공률이 낮고, 어플리케이션의 분기 특성이나 분기명령의 정보를 반영해야 분기 예측의 효율을 높일 수 있다는 단점이 있다[5,6]. 동적 분기 예측은 기존의 분기 정보를 활용하여 런타임에 분기를 예측하므로 정적 분기 예측보다 예측률이 높으나 추가적인 하드웨어 자원이 필요하다는 단점이 있다. 대표적인 동적 분기 예측 방식으로 bimodal, tournament, Gshare가 제안되었다[3]. Bimodal은 최근의 분기 결과를 BHT (Branch History Table)에 저장하고 그 결과에 따라 다음 분기 결과를 예측하는 방식이다. 간단하고 추가되는 하드웨어도 작으나 분기 예측률이 낮다. Tournament 방식은 global 분기와 local 분기를 각각 고려하여 분기 예측률이 높으나 예측이 성공해도 모든 페널티의 제거가 불가능하다. Gshare 방식은 분기명령이 taken 될 경우의 타깃 주소를 BTAC (Branch Target Address Cache)에 미리 저장하여 예측 성공 상황의 분기명령 페널티를 제거한다. 분기 예측 정확도가 높고 타깃 주소를 저장함으로써 분기명령 페널티를 대부분 제거하나 대용량의 BTAC와 BHT가 필요하다[7]. 기존의 분기 예측기는 대부분 범용 프로세서를 기준으로 설계되므로 전력 소모가 많고 분기의 예측률을 높이기 위해 노력하였다[8-11]. 임베디드 프로세서는 범용 프로세서와는 다르게 소모 전력의 제한이 있으므로 제한된 전력 하에서 어플리케이션 수행 사이클을 감소시킬 수 있는 동적 분기 예측기가 필요하다. 본 논문은 4-Way BTAC에 MRU (Most Recently Used)버퍼를 사용하여 element를 선택적으로 동작시켜 제한된 전력하에서 어플리케이션 수행 사이클을 효과적으로 감소시킬 수 있는 향상된 동적 분기 예측기를 제안한다.

논문의 구성은 다음과 같다. 2절에서 연구의 배경이 되는 기존의 Gshare 동적 분기 예측기를 설명하고 임베디드 코어에 적합한 분기 예측기의 구현을 위해 분기명령 특성을 분석하는 과정을 기술한다. 또한 동적 분기 예측기가 구현된 시스템의 platform에 대해 설명한다. 3절에서는 합리적인 BTAC entry 수를 찾는 과정과 동적 분기 예측기의 성능 향상을 위해 사용되는 MRU 버퍼의 구조와 크기를 결정하는 과정을 기술한다. 또한 제안된 동적 분기 예측기의 전체 구조에 대해 설명한다. 4절에서는 제안된 동적 분기 예측기가 포함된 코어가 구현된 platform에 대해 설명하고, 생성된 코어의 성능 측정을 위한 벤치마크 어플리케이션 수행결과를 보인다. 측정된 어플리케이션 수행 사이클과 소모 전력 및 면적을 동적 분기 예측기가 포함되지 않은 코어와, Gshare 분기 예측기가 포함된 코어, 제안된 분기 예측기가 포함된 코어에서 비교한다. 5장에서는 결론 및 추후 과제를 제시한다.

Ⅱ. 연구배경

본 절에서는 Gshare 분기 예측기 관련 기존 연구를 소개하고 임베디드 코어에 적합한 동적 분기 예측기를 구현하기 위한 분기 특성을 분석하여 기술한다.

2.1. Gshare 분기 예측기

Gshare 분기 예측기는 GHR (Global History Register)과 BHT, BTAC로 구성된다. 분기명령이 수행되면 최근 분기들의 taken 경향이 저장된 GHR과 프로그램 카운터를 xor하여 BHT의 주소로 사용한다. BHT에는 주소에 따른 taken 여부가 저장되어 있어 분기명령이 taken 될지 혹은 not-taken 될지를 이후 전부 결정한다. 분기명령으로 인해 생기는 페널티를 모두 제거 가능한 Gshare 분기 예측기는 대용량의 BHT와 BTAC를 사용하여 기존의 동적 분기 예측기보다 분기 예측률이 높다[12]. Gshare 분기 예측기는 BTAC의 entry 수가 256개일 때 89%의 확률로 분기를 예측하며 entry가 증가함에 따라 예측률은 거의 100%에 이른다[13]. Gshare 분기 예측기를 개선하기 위해 R. Sendag는 Branch BMP (Misprediction Predictor)를 두어 conflict 상황을 제거하였고[8], Y. Maa는 BTAC의 주소를 Hedging하여 접근하였다[9]. T. Chen은 2단으로 구성된 분기

예측기를 사용하였다. 이런 방법들은 Gshare 분기 예측기의 예측률을 높이는데 치중하여 전력소모가 크므로, 제한된 전력하에서 분기 예측률이 높은 동적 분기 예측기가 필요하다[10].

2. 분기 특성 연구

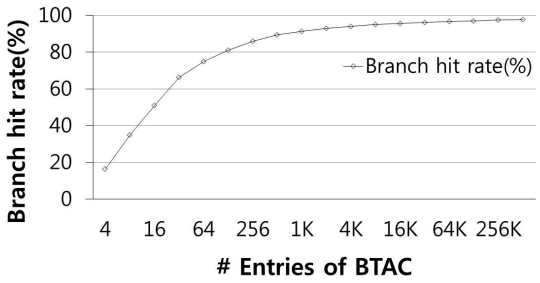
제한된 전력하에서 동적 분기 예측기의 분기 예측률을 높이기 위해 어플리케이션의 분기 특성을 분석 한다. 루프 명령은 분기 명령의 56%를 차지하나 루프문의 종류가 평균 5개로 적어 주로 사용되는 루프명령이 높은 비율로 반복된다[3,14]. 또한 어플리케이션의 명령어 중 84%가 루프 내부에 위치하므로 루프가 아닌 분기명령은 루프 내부에서 주로 수행된다. 전체 루프의 평균 76.0%가 단일루프이고, 루프로 건너뛰는 인스트럭션은 100개 미만인 경우가 많아 한 루프 내의 분기명령이 반복적으로 사용된다[15]. 따라서 분기명령은 temporal locality가 높으며, 분기명령의 높은 temporal locality를 활용함으로써 동적 분기 예측기의 분기 예측률을 증대시킬 수 있다.

Ⅲ. 제안된 동적 분기 예측기

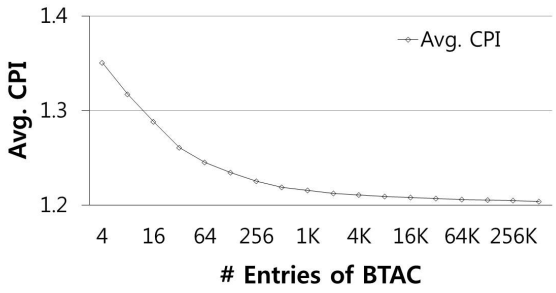
본 절에서는 제안된 임베디드 프로세서용 동적 분기 예측기를 제시한다. 먼저 최대의 효율을 내는 BTAC entry 수를 결정하는 과정을 보이고, 전력소모를 줄이기 위해 동적 분기 예측기 안에 포함된 MRU 버퍼의 구조와 합리적인 MRU 버퍼 entry 수를 정하는 방법을 제시한다. 또한 BTAC와 MRU 버퍼가 포함된 전체 동적 분기 예측기의 구조를 보이고 그 동작을 설명한다.

3.1. BTAC entry 수 결정

제안된 동적 분기 예측기의 BTAC entry 수가 많을수록 분기 예측률이 높으나 전력소모가 많고 면적이 크다. 임베디드 프로세서는 제한된 조건에서 최적의 성능을 내야 하므로 합리적인 BTAC entry 수를 결정하기 위해 BTAC entry 수를 변경하며 평균 분기 예측률을 측정한다. 분기 예측 확률이 어플리케이션 수행 속도에 미치는 영향을 알기 위해 벤치마크 어플리케이션의 분기명령 비율과 분기명령어의 CPI를 이용하여 벤치마크 어플리케이션의 평균 CPI를 계산한다[16,17]. 그림 1. (a)는 BTAC entry 수에 따른 분기 예측률을 보이며, 그림 1. (b)는 분기명령의 비율이 10%인 어플리케이션의 BTAC



(a)



(b)

그림 1. BTAC entry 수에 따른 분기 예측률과 CPI. (a) 분기 예측률 (b) CPI.
Fig. 1. Branch hit rate and CPI vs. number of BTAC entry. (a) Branch hit rate (b) CPI.

entry 수에 따른 평균 CPI의 변화를 계산한 결과를 보인다. BTAC entry 수가 64일 때 분기 예측률은 75%, 평균 CPI는 1.23이다. entry 수가 64보다 적으면 예측률과 CPI가 크게 변하나 반대 경우는 분기 예측률의 변화폭이 높지 않고 CPI 변화도 작다. 임베디드 프로세서는 파이프라인이 얕아 예측 실패에 대한 페널티가 작고, 면적이 작아야 유리하므로 BTAC entry 수를 64로 정하는 것이 합리적이다.

3.2. MRU 버퍼

동적 분기 예측기의 동적 전력소모 감소를 위해 MRU 버퍼를 추가한다. 제안된 동적 분기 예측기는 MRU 버퍼를 참조하여 타깃 주소가 저장되어 있는 BTAC의 element를 선택적으로 동작시킨다. 동적 전력소모가 줄어 BTAC entry 수를 증가시킬 수 있어 제한된 전력하에서 Gshare 분기 예측기에 비해 분기 예측률이 높다. Temporal locality가 높은 분기의 특성에 활용하고 MRU 버퍼 추가에 의한 페널티를 최소화하기 위해 MRU 버퍼 entry 수를 줄이고 비교기를 추가하였다.

3.2.1. MRU 버퍼 구조

그림 2는 제안된 MRU 버퍼 구조를 나타낸다. 기존의 MRU 버퍼는 주로 데이터 캐쉬의 element를 예측하므로 entry 수가 많고 예측을 수행하는데 1사이클이 소모된다[18,19]. 분기명령은 종류가 적고 temporal locality가 커 MRU 버퍼 entry의 수가 적어도 element 예측률이 높다. MRU 버퍼 entry 수를 줄이고 기존의 direct-mapped 구조 대신 각각의 entry에 비교기를 포함하여 예측 수행시간을 최소화한다. 분기명령이 수행되면 각 entry의 태그와 분기명령의 태그를 비교하여 타깃주소가 저장된 BTAC element를 예측한다. 예측에 성공하면 MUX에 예측된 element가 출력되며 element hit는 1을 출력한다. 예측에 실패하면 element hit는 0을 출력하고 NRU (Not Recently Used) 알고리즘을 적용하여 선택된 MRU 버퍼 entry를 갱신한다[20].

3.2.2. MRU 버퍼 의 크기 결정

기존의 BTAC에 MRU 버퍼를 추가하므로 MRU 버퍼의 면적이 동적 분기 예측기의 페널티로 작용한다. 페널티를 최소화하기 위해 MRU 버퍼 entry 수를 변경해가며 MiBench 어플리케이션을 반복적으로 수행하여 element 예측률과 BTAC 전력소모율을 측정한다. 그림3. (a)는 MRU 버퍼 entry 수에 따른 평균 예측률 변화이며, 그림 3. (b)는 BTAC의 전력소모율이다. 분기 명령의 종류가 적고 temporal locality가 크므로, entry 수가 1 ~ 4까지 증가하는 구간에서 element 예측률이 급격하게 증가하고

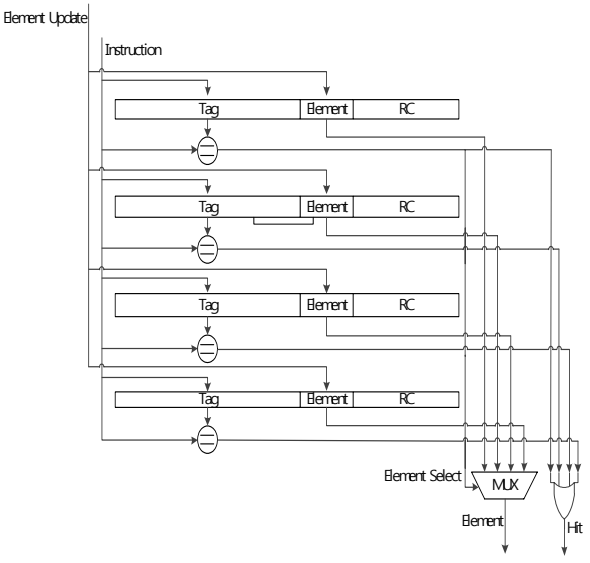


그림 2. MRU 버퍼 구조.
Fig. 2. Construction of MRU buffer.

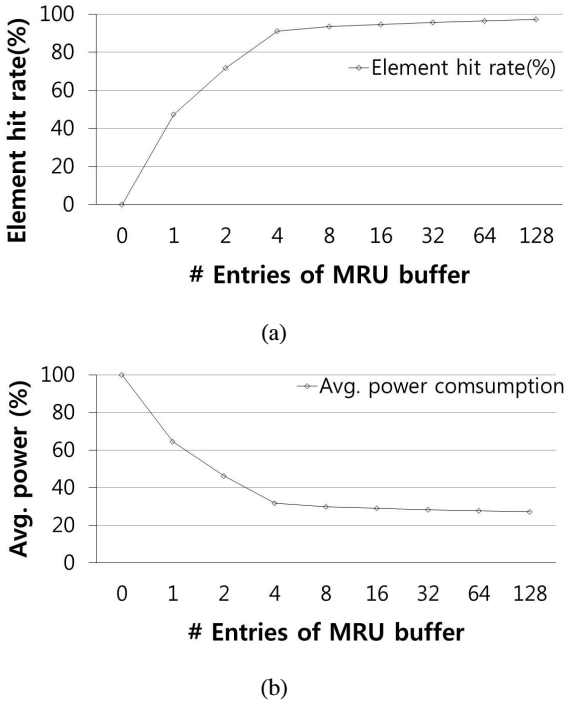


그림 3. MRU 버퍼 entry 수에 따른 element 예측 확률과 BTAC평균 전력소모. (a) Element 예측률 (b) BTAC 전력소모율.
Fig. 3. Element hit rate and BTAC power consumption vs. Number of MRU buffer entries. (a) Element hit rate (b) BTAC power consumption.

전력소모도 크게 감소한다. MRU 버퍼 entry 수가 4인 지점부터 예측률의 증가율이 줄어들고 전력 감소율도 73%에 수렴한다. 따라서 entry가 4보다 커지면 element 예측률 향상에서 얻는 이점보다 entry 수의 증가로 인한 페널티가 더 커지므로 MRU 버퍼 entry 수는 4로 고정한다.

3.2.3. 제안된 동적 분기 예측기 구조

임베디드 코어용 동적 분기 예측기는 제한된 전력하에서 예측률이 높아야 한다. 임베디드 코어는 범용 코어에 비해 예측률이 낮아도 상대적으로 페널티가 낮고, 전력소모와 면적이 치명적인 제한조건이므로 BTAC의 크기를 64로, MRU 버퍼 entry의 수를 4로 정한다. 동적 분기 예측기의 예측률을 높이기 위해 4-Way 캐쉬를 사용한다. 4-Way 캐쉬는 direct-mapped 방식의 캐쉬에 비해 동적 전력소모가 크므로 BTAC 내부에 MRU 버퍼를 추가하여 BTAC의 동적 전력소모를 줄인다[18,19]. MRU 버퍼에 저장된 최근 분기명령의 태그와 element를 사용하면 분기명령을 수행할 때 BTAC의 모든 element에 접근하지 않고 지정된 element에만 접근하여 BTAC의 동적 전력소모를 줄인다.

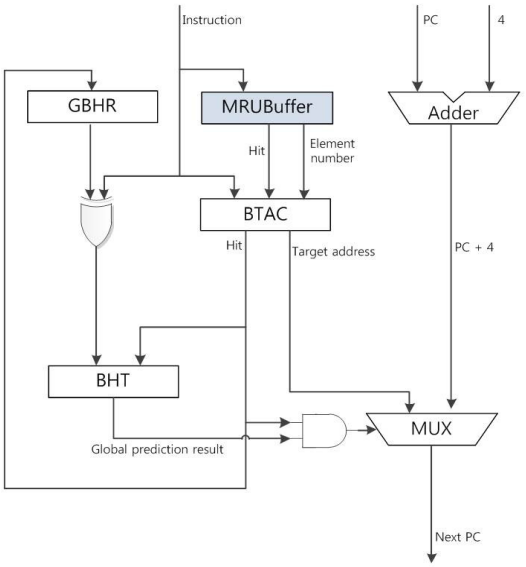


그림 4. MRU 버퍼가 추가된 제안된 동적 분기 예측기 구조.
Fig. 4. Construction of proposed branch predictor with MRU buffer.

그림 4는 MRU 버퍼가 추가된 제안된 동적 분기 예측기의 전체 구조를 나타낸다. 분기명령이 페치되면 분기 예측 기록이 저장된 GBHR과 프로그램 카운터를 xor 연산하여 BHT에 접근한다. BHT 출력결과가 taken 이면 global prediction result가 1이 되고, MRU 버퍼를 참조하여 타깃 주소가 저장된 BTAC element를 예측한다. 예측에 성공하면 MRU 버퍼에서 1과 타깃 주소가 저장된 element의 위치가 출력된다. BTAC는 출력된 결과를 받아 예측된 element만 동작시켜 타깃 주소를 출력한다. 예측에 실패하면 element hit가 0이 되고 BTAC의 모든 element에 접근해야 다음 프로그램 카운터로 사용할 타깃 주소가 출력된다. 만약 BTAC에 타깃 주소가 없거나 BHT 출력결과가 not-taken 이면 다음 프로그램 카운터는 정상적으로 증가시킨다.

Ⅳ. 실험 결과

본 절에서는 제안된 동적 분기 예측기의 검증을 위해 동적 분기 예측기가 포함된 코어가 구현된 platform을 기술하고 동적 분기 예측기가 포함되지 않은 코어와 Gshare 분기 예측기가 포함된 코어, 제안된 동적 분기 예측기가 포함된 코어에서 Mibench 어플리케이션 수행 사이클과 소모전력을 비교하였다.

4.1. 구현 platform

SMDL (Sogang Machine Description Language) 시스템은 time-to-market요구가 증대함에 따라 설계물의 재사용률을 높이기 위한 MDL (Machine Description Language)기반의 ASIP 설계 자동화 시스템의 한 종류로 다중 사이클 인스트럭션을 지원해 기존의 시스템보다 메모리 사용이 효율적이다[21,22]. SMDL 시스템은 SMDL 문법으로 기술된 문서를 중간 형태로 변환하여 retargetable 컴파일러의 입력으로 사용한다. retargetable 컴파일러는 C 언어로 기술된 어플리케이션을 추가적으로 입력받아 어플리케이션의 이진 코드를 생성하며, 임베디드 코어 생성기는 중간 형태를 기반으로 타깃 코어의 HDL 코드를 생성하고, retargetable 인스트럭션 셋 시뮬레이터 생성기는 인스트럭션 셋 시뮬레이터를 생성한다. 생성된 시뮬레이터를 사용하여 임베디드 코어 생성기에 의해 생성된 코어와 컴파일러에서 출력된 이진 코드를 검증한다[23,24]. 제안된 동적 분기 예측기의 성능 평가를 위해 상용화된 임베디드 코어 중 ARM9 코어 모델을 타깃 코어로 정하여 구현하였다[16,25,26]. TSMC 90nm 라이브러리를 기반으로 생성된 코어는 Synopsys사의 Design Vision을 이용하여 합성된 후 SUN-Sparc 워크스테이션에서 수행되었다. 또한 100MHz의 주파수로 동작하고 5단 파이프라인으로 구성되며, 2-phase non-overlapping clock을 사용한다. 동작 온도는 25℃, 동작 전압은 3.3V이다.

4.2. 실험 결과

표 1은 동적 분기 예측기가 없는 코어와 제안된 동적 분기 예측기가 포함된 코어의 어플리케이션 수행 소모 전력 비교 결과이다. 동적 분기 예측기가 포함되므로 어플리케이션 수행 전력소모량이 평균 7.4% 증가한다. 제안된 동적 분기 예측기는 BTAC의 특정 element만을 참조하여 분기 예측에 필요한 동적 전력소모를 줄이므로 분기 명령의 비율과 특정 element를 참조하는 비율이 높을수록 전력소모량 증가비율이 작다. 어플리케이션 수행 사이클 비교를 위해 전력소모를 제한하여 Gshare 분기 예측기의 BHT와 BTAC의 크기를 결정하고 Gshare 분기 예측기가 포함된 코어를 생성한다. 제안된 동적 분기 예측기는 BTAC의 특정 element를 선택하여 접근하므로 동적 전력 소모가 줄어들어 제한된 전력하에서 BTAC의 entry 수가 Gshare 분기 예측기

표 1. 벤치마크 어플리케이션 전력 소모 비교 결과
Table 1. Comparison results of benchmark applications power consumption

MiBench	Branch ratio (%)	Element hit rate (%)	without dynamic branch predictor (mW)	with proposed dynamic branch predictor (mW)	Comparison (%)
tiff2rgba	3.3	99.9	165.6	181.0	8.6
tiffmedian	4.7	90.7	276.5	291.9	3.7
blowfish	12.3	92.3	178.7	194.1	5.7
bitcount	10.3	88.0	176.9	192.3	6.5
ADPCM enc	3.3	91.3	166.5	181.9	8.6
CRC32	15.3	77.2	179.0	194.4	5.6
FFT	13.0	89.2	178.9	194.3	5.6
IFFT	12.3	99.9	191.6	207.0	4.4
ADPCM dec	5.0	93.2	158.0	173.4	8.9
GSM enc	4.3	99.9	184.9	200.3	7.2
GSM dec	9.7	99.9	150.3	165.7	8.9
stringsearch	19.3	80.8	126.2	141.6	12.2
ispell	17.7	81.1	134.2	149.6	10.7
평균	10.1	91.0	174.4	189.8	7.4

에 비해 많아 분기 예측률이 높다. 표 2는 동적 분기 예측기가 없는 코어와 Gshare 분기 예측기가 포함된 코어 및 제안된 동적 분기 예측기가 포함된 코어의 어플리케이션 수행 사이클의 비교 결과이다. 어플리케이션 수행 사이클은 동적 분기 예측기를 사용하지 않았을 때보다 Ghare 분기 예측기가 포함된 코어의 경우 평균 6.3%, 제안된 동적 분기 예측기를 사용한 경우 평균 10.1% 감소한다. 제안된 동적 분기 예측기가 포함된 코어는 같은 전력을 사용하는 Gshare 분기 예측기가 포함된 코어와 비교하여 어플리케이션 평균 수행 사이클이 4.1% 감소한다. 동적 분기 예측기는 분기명령의 페널티를 줄이므로 분기 비율이 높을수록 코어의 수행속도를 향상시킨다. 임베디드 프로세서에 알맞은 구조를 위해 BTAC가 합리적인 entry를 갖도록 했고, 효율적인 element 예측률을 갖도록 MRU 버퍼 entry 수를 결정하여 면적 증가를 최소화 하였다. 기존의 타깃 프로세서는 826,223 게이트로 구성된다. 전력소모가 제한된 Gshare 분기 예측기는 5,045 게이트이므로 Gshare 분기 예측기를 포함한 임베디드 프로세서는 총 831,268 게이트로 구성된다. MRU 버퍼는 311 게이트이고, Gshare와 같은 전력을 소모할 때 제안된 동적 분기 예측기는 BTAC의 크기를 키워 8,257 게이트로 구성된다. 제안된 동적 분기 예측기가 포함된 코어는 총 834,480 게이트이다. 임베디드 프로세서의 특성에 맞게 MRU 버퍼와 BTAC entry 수

표 2. 벤치마크 어플리케이션 수행 사이클 비교 결과
Table2. Comparison results of benchmark applications executing cycle

Mi-Bench	Bran - ch ratio (%)	Without dynamic branch predictor # of cycles	Gshare dynamic branch predictor # of cycles /comparison (%)	Proposed dynamic branch predictor # of cycles /comparison (%)	Comparison with Gshare dynamic branch predictor (%)
tiff2rgba	3.3	46,533,494	45,436,110 / -2.4	44,777,680 / -3.8	-1.4
tiffmedian	4.7	181,556,378	175,577,992 / -3.3	171,990,960 / -5.3	-2.0
blowfish	12.3	74,481,371	68,680,690 / -7.8	65,200,282 / -12.5	-5.1
bitcount	10.3	68,814,263	64,209,757 / -6.7	61,447,054 / -10.7	-4.3
ADPCM enc	3.3	47,469,368	46,349,914 / -2.4	45,678,242 / -3.8	-1.4
CRC32	15.3	77,959,980	70,683,926 / -9.3	66,318,294 / -14.9	-6.2
FFT	13.0	75,465,566	69,308,334 / -8.2	65,613,995 / -13.1	-5.3
IFFT	12.3	93,339,095	86,069,757 / -7.8	81,708,153 / -12.5	-5.1
ADPCM dec	5.0	38,905,353	37,548,189 / -3.5	36,733,891 / -5.6	-2.2
GSM enc	4.3	70,718,535	68,576,052 / -3.0	67,290,563 / -4.8	-1.9
GSM dec	9.7	32,809,463	30,725,754 / -6.4	29,475,529 / -10.2	-4.1
stringsearch	19.3	245,489	217,932 / -11.2	201,398 / -18.0	-7.6
ispell	17.7	12,724,094	11,389,346 / -10.5	10,588,498 / -16.8	-7.0
평균	10.1	63155573.0	59597981.0 / -6.3	57463426.1 / -10.1	-4.1

를 최소화하여 제안된 분기 예측기가 포함된 코어는 분기 예측기가 포함되지 않은 코어에 비해 면적이 1%미만 증가하였다.

V. 결 론 및 추후과제

분기명령의 페널티를 모두 제거하는 기존의 Gshare 분기 예측기는 분기 예측률이 높으나 대용량의 BTAC와 BHT가 필요하여 전력소모가 크다는 단점이 있다. 본 논문은 분기의 특성을 분석하여 동적 분기 예측기 내부의 BTAC의 크기를 줄이고 적은 entry 수를 갖는 MRU 버퍼를 추가하여 제한된 전력하에서 분기 예측률이 높은 동적 분기 예측기를 제안한다. 제안된 동적 분기 예측기는 MRU 버퍼를 참조하여 타깃 주소가 저장된 BTAC element

를 선택하여 동작시키므로 BTAC의 동적 전력소모를 줄여 같은 전력을 사용하는 기존의 분기 예측기보다 분기 예측률이 높다. 제안된 분기 예측기가 포함된 ARM9코어를 기반으로 벤치마크 어플리케이션을 수행하여 분기 예측기의 효율성을 검증한 결과 동적 분기 예측기를 포함하지 않은 코어에 비해 어플리케이션 수행 사이클이 평균 10.1% 감소하고 전력소모가 평균 7.4% 증가하였다. 같은 전력을 소모하는 Gshare 분기 예측기가 포함된 코어와 비교하면 어플리케이션의 수행 사이클이 평균 4.1% 감소하였다. 추후 어플리케이션 분기 특성을 기반으로 MRU 버퍼 entry의 수와 BTAC의 크기를 변경하여 동적 분기 예측기의 효율을 높여 분기 예측 확률을 높이고 전력소모를 줄여야 한다. 또한 파이프라인이 깊을수록 분기명령의 페널티가 증가하므로 파이프라인에 따라 다른 구조를 갖는 분기 예측기에 대한 추가 연구가 필요하다.

References

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

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

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

[4] T. Ball and J. Laurs, "Branch prediction for free," in *Proc. ACM SIGPLAN Conf. Programming Language Design Implementation*, pp. 300-313, New York, U.S.A., Aug. 1993.

[5] J. Patterson, "Accurate static branch prediction by value range propagation," in *Proc. ACM SIGPLAN Conf. Programming Language Design Implementation*, pp. 67-78, New York, U.S.A., June 1995.

[6] B. Calder, D. Grunwald, M. Jones, D. Lindsay, J. Martin, M. Mozer, and B. Zorn, "Evidence-based static branch prediction using machine learning," *ACM Trans. Programming Languages Syst.*, vol. 19, no. 1, pp. 1-43, Sep. 1996.

[7] C. Cheng, *The Schemes and Performances of Dynamic Branch Predictors*, Technical Report, Berkeley Wireless Research Center, 2000.

[8] R. Sendag, J. Yi, P. Chuang, and D. Lilja, "Low power/area branch prediction using complementary branch predictors," in *Proc. IEEE Int. Parallel Distributed Process. Symp.*, pp. 1-12, Miami, U.S.A., Apr. 2008.

[9] Y. Maa, M. Yen, S. Kuo, and G. Lee, "Cost-effective branch prediction by combining hedging and filtering," in *Proc Int. Comput. Symp.*, pp. 648-655, Tainan, Taiwan, Dec. 2010.

[10] T. Chen, P. Pan, G. Jiang, and M. Ye, "Record branch prediction : An optimized scheme for two-level branch predictors," in *Proc. IEEE 14th Int. Conf. High Performance Comput. Commun.*, pp. 1526-1533, Liverpool, U.K., June 2012.

[11] D. Parikh, K. Skadron, Y. Zhang, and M. Stan, "Power-aware branch prediction: Characterization and design," *IEEE Trans. Comput.*, vol. 53, no. 2, pp. 168-186, Feb. 2004.

[12] L. Nadav and W. Shlomo, "Low power branch prediction for embedded application processors," in *Proc. Low Power Electron. Design*, pp. 67-72, Austin, U.S.A., Aug. 2010.

[13] S. McFarling, *Combining branch predictors*, Technical Report, Western Research Laboratory, Dec. 1993.

[14] Y. Ding and W. Zhang, "Loop-based instruction prefetching to reduce the worst-case execution time," *IEEE Trans. Comput.*, vol. 59, no. 6, pp. 855-864, June 2010.

[15] M. Kobayashi, "Dynamic characteristics of loops," *IEEE Trans. Comput.*, vol. 33, no. 2, pp. 125-132, Feb. 1984.

[16] S. Segars, "The ARM9 family－High performance microprocessors for embedded applications," in *Proc. Int. Conf. Comput. Design*, pp. 230-235, Austin, U.S.A., Oct.

1998.

[17 M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Int. Workshops Workload Characterization*, pp. 3-14, Austin, U.S.A., Dec. 2001.

[18] K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in *Proc. Int. Symp. Low Power Electron. Design*, pp. 273-275, San Diego, U.S.A., Aug. 1999.

[19] M. Calagos and Y. Chu, "Hybrid scheme for low-power set associative caches," *Electron. Lett.*, vol. 48, no. 14, pp. 819-821, July 2012.

[20] K. Kedzierski, M. Moreto, F. Cazorla, and M. Valero, "Adapting cache partitioning algorithms to pseudo-LRU replacement policies," in *Proc. Parallel Distributed Process*, pp. 1-12, Atlanta, U.S.A., Apr. 2010.

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

[22] K. Choi and Y. Cho, "Recent trends in the SoC design methodology," *Inst. Electron. Eng. Korea (IEEK) Mag.*, vol. 30, no. 9, pp. 17-27, Sep. 2003.

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

[24] J. Cho, Y. Yoo, and S. Hwang, "Construction of an automatic generation system of embedded processor cores," *J. Korean Inst. Commun. Inform. Sci, (KICS)*, vol. 30, no. 6A, pp. 526-534, June 2005.

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

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

황 인 성 (In-sung Hwang)

2012년 2월 서강대학교 전자공학과 졸업

2012년~현재 서강대학교 전자공학과 CAD & ES 연구실 석사과정

<관심분야> Embedded System Design, ASIP Design

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