온도 인지 마이크로프로세서를 위한 듀얼 레지스터 파일 구조

(A Dual Integer Register File Structure for Temperature-Aware Microprocessors)

최 진 항 † 공 준 호 † 정 의 영 †† 정 성 우 †††
(Jinhang Choi) (Joonho Kong) (Eui-Young Chung) (Sung Woo Chung)

요 약 오늘날 마이크로프로세서의 설계는 전력 소모 문제만이 아닌 온도 문제에서도 자유롭지 않다. 제조 공정의 미세화와 고밀도 회로 집적화가 칩의 전력 밀도를 높이게 되어 열섬 현상을 발생시키기 때문이다. 이를 해결하기 위해 제안된 동적 온도 제어 기술은 냉각 비용을 줄이는 동시에 칩의 온도 신뢰성을 높인다는 장점을 가지지만, 냉각을 위해 프로세서의 성능을 희생해야 하는 문제점을 가지고 있다. 본 논문에서는 프로세서의 성능 저하를 최소화하면서 온도를 제어하기 위해 듀얼 레지스터 파일 구조를 제시한다. 온도 제어를 고려하였을 때 가장 관심을 끄는 것은 레지스터 파일 유닛이다. 특히 정수형 레지스터 파일 유닛은 그 빈번한 사용으로 인하여 프로세서 내부에서 가장 높은 온도를 가진다. 듀얼 레지스터 파일 구조는 정수형 레지스터 파일에 대한 읽기 접근을 두 개의 레지스터 파일에 대한 접근으로 분할하는데, 이는 기존 레지스터 파일이 소모하는 동적 전력을 감소시켜 열섬 현상을 제거하는 효과를 가져온다. 그 결과 동적 온도 제어 기법에 의한 프로세서 성능 감소를 완화시키는데, 평균 13.35% (최대 18%)의 성능 향상을 확인할 수 있었다.

키워드 : 온도 인지 마이크로프로세서, 레지스터 파일, 동적 온도 제어

Abstract Today's microprocessor designs are not free from temperature as well as power consumption. As processor technology scales down, an on-chip circuitry increases power density, which incurs excessive temperature (hotspot) problem. To tackle thermal problems cost-effectively, Dynamic Thermal Management (DTM) has been suggested; DTM techniques have benefits of thermal reliability and cooling cost. However, they require trade-off between thermal control and performance loss. This paper proposes a dual integer register file structure to minimize the performance degradation due to DTM invocations. In on-chip thermal control, the most important functional unit is an integer register file. It is the hotspot unit because of frequent read and write data accesses. The proposed dual integer register file migrates read data accesses by adding an extra register file, thus reduces per-unit dynamic power dissipation. As a result, the proposed structure completely eliminates localized hotspots in the integer register file, resulting in much less performance degradation by average 13.35% (maximum 18%) improvement compared to the conventional DTM architecture.

Key words : Temperature-aware Microprocessor, Register File, Dynamic Thermal Management

· 본 논문은 2006년 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥 재단의 지원을 받아 연구되었음(KRF-2006-331-D00452)
· 본 논문의 시뮬레이션 환경 설정에 많은 조언을 주신 버지니아 대학 LAVA 그룹의 Karthik Sankaranarayanan에게 감사 인사를 드립니다.

† 비 회 원 : 고려대학교 컴퓨터통신공학부
cepiross@korea.ac.kr
luisfigo77@korea.ac.kr
†† 비 회 원 : 연세대학교 전기전자공학부 교수
eychung@yonsei.ac.kr
††† 종신회원 : 고려대학교 컴퓨터통신공학부 교수
swchung@korea.ac.kr

논문접수 : 2008년 3월 11일
심사완료 : 2008년 11월 3일

정보과학회논문지: 시스템 및 이론 제35권 제12호(2008.12)

1. 서 론

오늘날 고성능 마이크로프로세서의 설계는 이상 발열 문제에서 자유롭지 않다. 제조 공정이 미세화하면서 급격하게 증가한 전력 밀도가 칩 내부에 열섬(hotspot) 현상을 만들어, 기능 유닛의 오작동 및 회로의 손상을 일으키기 때문이다. 게다가, 현재 널리 사용되고 있는 열발산판(heat sink), 히트스프레더(heat spreader), 냉각팬(cooling fan)과 같은 기계적 냉각 방식은 물리적 한계와 냉각 비용 상승에 의한 제약으로 인해 점차 사용이 제한되고 있다[1]. 결국, 칩 설계 단위에서 온도를 고려해야 할 필요성이 제기되었고, 최신 프로세서는 온도 제어 기술을 내장함으로써 발열 문제를 해결하고자 노력하고 있다.

마이크로프로세서에서 수행되는 온도 제어 기술은 크게 정적인 방법과 동적인 방법으로 구분된다. 정적인 온도 제어 기법은 기능 유닛의 저전력 설계와 플로어플래닝(floorplanning)[2,3]을 조합하여 발열에 최적화된 회로를 구성하는 방식이다. 그런데, 이 방식은 칩을 직접 냉각하는 것이 아니어서 온도 관리에 제한이 있기 때문에 동적 온도 제어(Dynamic Thermal Management) 기법[4]이 함께 사용된다. 동적 온도 제어 기법은 프로세서의 주파수나 전압을 조절 하거나(DVFS: Dynamic Voltage and Frequency Scaling), 명령어의 인출을 지연(fetch throttling)시켜 발열을 조절한다. 이 방법은 칩의 발열을 효과적으로 낮춰 프로세서의 온도 안정성을 높이고 냉각 비용을 줄인다. 하지만 온도 제어를 위해 프로세서의 성능을 희생하기 때문에, 성능 감소를 최소화하는 효율적인 동적 온도 제어 기술에 대한 요구가 끊임없이 이어져 왔다.

본 논문에서는 정수 레지스터 파일을 이용한 듀얼 레지스터 파일 구조에 대하여 기술한다. 온도 제어를 고려할 때 가장 문제가 되는 유닛은 정수 레지스터 파일이다. 프로세서에서 발생하는 열섬 현상이 주로 정수 레지스터 파일에 의하여 발생하며, 그 온도는 최고 100°C 이상까지 상승하는 양상을 보인다. 본 논문에서 제안하는 구조는 마이크로프로세서가 레지스터 파일에 접근하는 횟수를 분할하여 하나의 레지스터 파일이 소모하는 동적 전력량을 낮추도록 유도한다. 그 결과 칩의 최고 온도가 100°C 아래로 완화되며 동적 온도 제어 기법이 덜 수행되고, 시스템적으로는 프로세서의 성능이 최대 18%까지 향상되는 결과를 얻을 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 온도 제어 기술의 성능 감소를 줄이기 위해 사용된 기존 연구의 특징을 알아보고, 3장에서는 정수 레지스터 파일이 프로세서의 열섬 현상에 끼치는 영향을 파악한다. 이어서, 4장에서는 제안한 듀얼 레지스터 파일의 구조와 특성에 대하여 설명하고, 5장에서는 모의 실험 환경을 기술한다. 6장에서는 발열 안전성과 성능의 측면에서 듀얼 레지스터 파일 구조가 적용된 마이크로프로세서를 평가한다. 마지막으로, 7장에서는 결론과 함께 본 연구의 의의 및 향후 과제에 대하여 기술한다.

2. 관련 연구

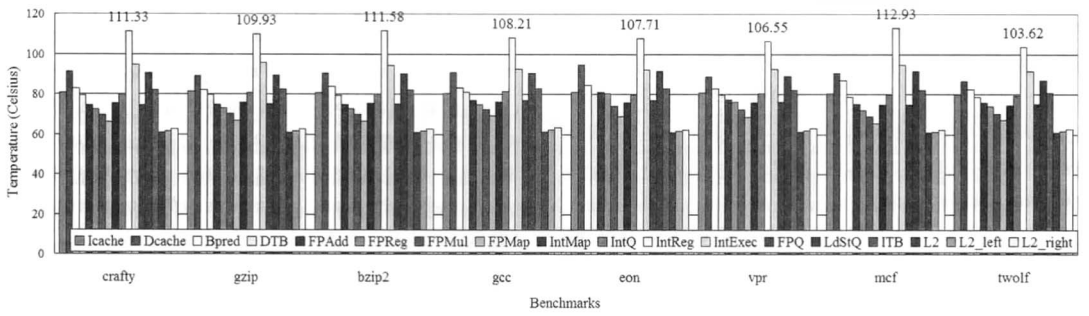
국제 반도체 기술 이정표(ITRS, International Technology Roadmap for Semiconductors)는 130nm 이하의 초미세 공정에서 칩의 최고 접합 온도(maximum junction temperature)가 90°C보다 낮아져야 한다고 발표하였는데[5], 이 조건을 만족하기 위해 표본 집단에서 최악의 경우에 해당하는 응용프로그램(worst typical application)[6]의 온도를 추적, 반영하여 동적으로 온도 제어를 수행하는 방법이 제안되었다[7]. 온도 센서가 얻어낸 정보가 일정 수준을 넘을 때만 온도 제어 기법을 수행하는 이 방식은 가변 주파수 조절(DFS)에서 2%, 가변 전압 주파수 조절(DVFS)에서 6~9%, 명령어 인출 조절(fetch toggling)에서 8%의 작은 성능 저하만으로 발열을 제어하여, 고성능 마이크로프로세서에 적합하다는 것을 입증하였다. 하지만 프로세서의 유닛 온도가 정의된 수준 이상으로 지속적으로 형성되면 끊임없이 온도 제어 기법이 수행되어 프로세서에 심각한 성능 저하를 가져오게 된다. 이 문제를 피하기 위한 해결책은 크게 두 가지이다. 첫 번째는 성능 계수기(on-chip performance counter)를 이용하여 온도를 예측한 뒤[8], 미리 온도 제어 기법을 수행하여 칩의 온도를 극적으로 낮춰서 전체적으로는 제어 기법의 수행을 적게 하는 방법이며, 두 번째는 보조 기능 유닛을 두고 주 기능 유닛의 사용률을 내려서 발열을 감소시키는 연산 이관(computation migration) 기법이다[9]. 본 논문에서는 연산 이관 기법에 주목하였다. 가장 뜨거운 온도를 보이는 정수 레지스터 파일[10,11]에 연산 이관 기법을 적용시키면, 칩에서 나타나는 열섬 현상을 줄임으로써 프로세서의 온도 제어 성능을 끌어올릴 수 있다. Skadron et al은 정수 레지스터 파일을 복제한 뒤 임계 온도에 다다를 경우 연산 이관을 수행하는 온도 제어 기법을 제안하였는데[7], 8%의 성능 희생으로 온도 제어를 수행할 수 있음을 제시하였다. 한편, Patel *et al*은 하나의 레지스터 파일을 두 영역으로 구분하여 번갈아 사용하는 연산 분할(computation division)방식으로 레지스터 파일의 온도를 낮추는 구조를 제안하였다[12]. 위의 두 기법은 레지스터 파일을 복제, 혹은 분할하여 프로세서의 온도를 효과적으로 제어할 수 있음을 보여준다. 그러나 각각 큰 단점이 존재하는데, Skadron *et al*의 레지스터 파일 구

조는 연산 이관을 수행하기 위해 파이프라인을 멈추고 데이타를 복사한 뒤 재실행하는 등의 추가 작업이 성능에 영향을 미치게 되며, Patel *et al*의 레지스터 파일 구조는 분할된 레지스터 파일의 온도가 서로 영향을 주게 되므로 레지스터 파일에서 열섬 현상이 발생할 때 프로세서의 사용을 멈춰야 하는 문제가 발생한다. 본 논문에서는 두 기법의 장점을 취합하여 각각의 문제를 해결하고자 한다. 듀얼 레지스터 파일 구조는 연산 분할을 기준으로 서로 떨어진 두 레지스터 파일에 접근한다. 이 구조 상에서는 레지스터 파일 간의 데이타 전송에 대한 추가 부담 없이 온도 제어를 수행할 수 있으며, 더불어 칩의 온도 분포에 최적화된 온도 제어를 통하여 프로세서의 성능을 효과적으로 끌어올릴 수 있다.

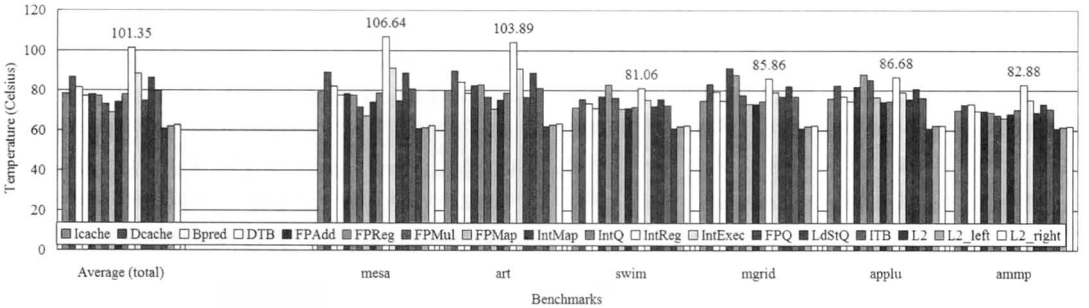
3. 사례 연구: 정수 레지스터 파일에 의한 열섬 현상

그림 1은 14개의 벤치마크 프로그램에 대한 Alpha 21264 프로세서의 각 기능 유닛 별 최고 온도를 보여주는 그림이다. 실험에서 프로세서는 동적 온도 제어 기법을 수행하지 않도록 설정했고, 그 결과 정수 레지스터 파일(IntReg)의 온도는 최고 112.9°C(*mcf*)에서 최저 81.1°C(*swim*)까지 분포한다. 그림 1의 데이타는 벤치마크를 실행하는데 있어 정수 레지스터 파일이 언제나 가장 높은 온도를 나타내는 것은 아니지만, 항상 높은 온도군에 속한다는 것을 보여준다(평균온도 IntReg: 101.4°C, IntExec: 88.4°C, Dcache: 86.7°C, LdStQ: 86.3°C). 따라서, 본 논문에서는 정수 레지스터 파일을 열섬 가능 인자(*hotspot-possible factor*)로 설정하여 듀얼 레지스터 파일의 효과에 대하여 논의한다. 가장 뜨거운 정수 레지스터 파일(IntReg)의 온도는 평균 100.0°C를 넘으며, 두 번째로 뜨거운 정수 연산 유닛(IntExec)과의 온도 차는 평균 13.0°C이다. 상대적으로 가장 차가운 기능 유닛인 L2 캐쉬의 온도가 평균 61.8°C임을 고려할 때, 정수 레지스터 파일의 온도를 낮추는 것은 마이크로프로세서의 온도 제어 측면에서 많은 이득을 가져다 준다는 것을 알 수 있다.

마이크로프로세서에서 나타나는 온도 양상은 크게 정수형과 부동소수점형으로 분류된다. 정수형 온도군에서는 정수 기능 유닛(IntReg, IntExec)과 데이타 접근에 사용되는 유닛(Dcache, LdStQ)이 다른 기능 유닛보다 뜨겁다. 모든 정수 벤치마크는 정수형 온도군에 속하며, 위에서 언급한 뜨거운 유닛 집단의 온도는 평균 85.0°C를 넘는다. 특히 모든 경우의 정수 레지스터 파일의 온도가 100.0°C를 넘으며, 이는 앞에서 정의한 열섬 가능



정수형 벤치마크 실험 결과



부동소수점형 벤치마크 실험 결과 및 전체 평균

그림 1 Alpha 21264의 각 기능 유닛에 대한 최고 온도

인자에 정확히 적용된다. 한편, 부동소수점형 온도군의 경우에는 부동소수점 기능 유닛들(FPReg, FPAdd, FPMul)의 온도가 높게 형성된다. 가장 뜨거운 유닛과 가장 차가운 유닛의 온도 차이가 25.0°C 아래이고(정수형 온도군의 경우 약 40.0°C의 차이를 보임), 정수형 온도군의 정수 레지스터 파일처럼 다른 기능 유닛과 온도 차이가 현격하게 나는 유닛이 없다는 것이 특징이다. 이 결과는 상대적으로 특정 기능 유닛이 과도하게 뜨거운 열섬 현상의 발생 가능성이 적음을 의미한다. 그러나 여전히 정수 레지스터 파일은 높은 온도군에 속하며 열섬 가능 인자로 분류 가능하다.

정리하면 정수 레지스터 파일은 마이크로프로세서 내에서 발생 가능한 열섬 현상의 주원인이며, 이 유닛의 온도를 제어하는 것만으로도 온도 제어 기법의 효율성을 향상시키는 것이 가능하다.

4. 듀얼 정수 레지스터 파일의 구조

정수 레지스터 파일은 정수 연산 명령어, 적재/저장 명령어, 분기 명령어가 수행될 때마다 레지스터 값을 읽거나 쓰기 위해 접근된다. 만약, 레지스터 파일에 대한 접근 횟수가 줄어든다면 자연히 전력 소모가 줄어들게 되고, 따라서 해당 기능 유닛의 발열도 감소하게 될 것이다. 위의 연산 이관 기법을 기반으로 하여, 기존 정수 레지스터 파일과 크기가 일치하는 추가 정수 레지스터 파일을 마이크로프로세서에 추가한다.

그러나, 기존의 연산 이관 방법은 레지스터 파일간 데이타를 전송해야 하는 시스템 부담이 존재하기 때문에 그대로 사용하는 것은 무리가 있다. 본 논문에서는 연산 분할을 이용하여 두 정수 레지스터 파일을 접근한다. 마이크로프로세서가 정수 레지스터 파일에 데이타를 쓸 때는 두 개의 정수 레지스터 파일에 동시에 값을 기록한다. 반대로 프로세서가 레지스터 파일에서 데이타를 읽어 들일 때는, 미리 정해진 비율에 따라 첫 번째 레지스터 파일과 두 번째 레지스터 파일을 번갈아 접근한다. 이 접근 방법은 두 정수 레지스터 파일간에 데이타를 주고받기 위한 데이타패스를 필요로 하지 않는다. 게다가 위 구조는 데이타 접근에 단순히 레지스터 파일 선택기(selector)만을 추가로 요구하기 때문에, 데이타 접근 시간을 기존의 레지스터와 같다고 가정할 수 있다. 하지만 다음과 같은 시스템 부담이 발생한다.

먼저, 레지스터 파일 유닛과 다른 연산 유닛 간의 데이타 패스가 복잡해진다. 하나가 아닌 두 개의 정수 레지스터 파일 유닛이 각 연산 유닛과 연결되어야 하므로 칩 내부의 기능 유닛 배치에 따라 데이타 전송 경로가 복잡하게 구성될 가능성이 있다. 이 문제를 경감시키기 위해서는 레지스터 파일 유닛이 연산 유닛으로부터 거리가 먼 곳에 배치되지 않는 설계상의 제한이 필요하다. 본 논문에서 사용된 프로세서의 경우, 정수 레지스터 파일이 직접적으로 연결되는 유닛은 명령어 큐(IntQ)이다. 다른 연산 유닛은 명령어 큐를 거쳐 정수 레지스터 파일에 접근하게 되므로 상대적으로 듀얼 레지스터의 배치는 자유롭다. 그러나 앞에서 제시한 거리 제약에 의하여 첫 번째 정수 레지스터 파일은 정수형 명령어 큐 바로 옆에 위치하며, 두 번째 정수 레지스터 파일은 정수형 명령어 큐로부터 두 블록 떨어진 곳에 위치한다.

다음으로, 추가 정수 레지스터 파일이 들어갈 공간이 필요하다. 마이크로프로세서 설계에서 회로 면적의 제한은 큰 제약이다. 본 논문에서는 L2 캐시 크기를 희생하여 레지스터 파일을 추가하는 정책을 사용하였다. 표 1을 참고하면 정수 레지스터 파일의 면적은 L2 캐쉬 면적의 0.5%에 해당하는데, 설정된 L2 캐쉬 크기는 512KB이다. 다시 말하면 약 3KB에 해당하는 L2 캐쉬 공간을 희생하고 추가적인 정수 레지스터 파일을 삽입하는 것인데, 이 경우 캐쉬 크기의 변화에 의한 성능 저하 문제가 제기될 수 있다. 그러나 [13]에 따르면 일정 크기 이상의 캐쉬에서는 크기의 변화에 따른 성능 영향

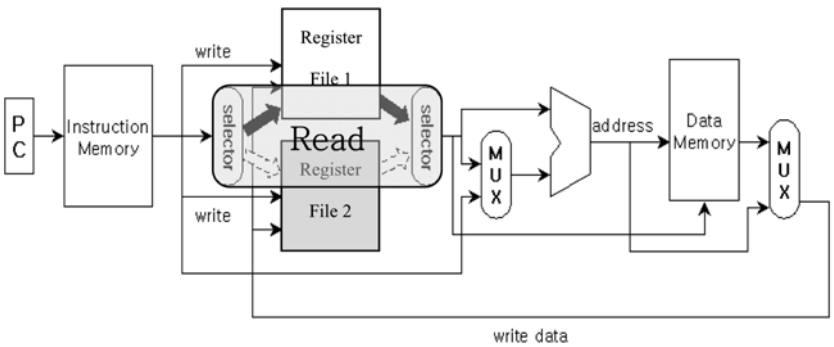


그림 2 듀얼 정수 레지스터 파일의 구조

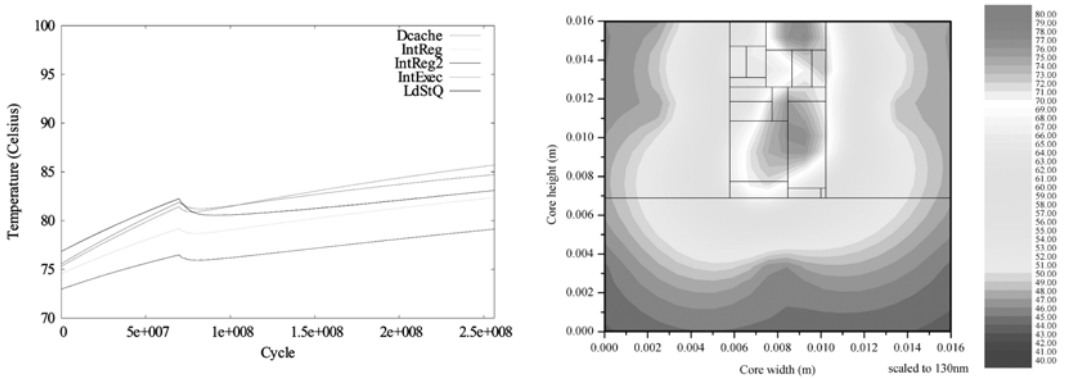
표 1 130nm 공정 Alpha 21264의 면적 명세 ($\times 10^{-6}$ m^2)

기능 유닛	면적	기능 유닛	면적
FPAdd	1.940	FPQ	1.445
FPReg	0.835	LdStQ	1.252
FPMul	2.076	ITB	0.782
FPMap	1.445	Bpred	2.269
IntMap	1.283	DTB	2.295
IntQueue	1.758	Icache	8.346
IntReg	1.226	Dcache	7.824
IntExec	3.985	L2	214.317

이 미미하다고 알려져 있으므로, 현재 사용하고 있는 L2 캐쉬가 충분히 큰 점을 감안하여 프로세서의 성능에는 영향이 없다고 가정하였다.

마지막으로, 칩의 전력 소모가 증가한다. 본 논문에서 사용한 Alpha 21264 프로세서의 경우, 명령어를 인출한 뒤 정수 레지스터 파일로부터 데이타를 읽은 뒤 연산을 수행하는데 드는 전력이 전체의 10%를 차지한다[14]. 따라서, 보수적 관점으로 접근하면 듀얼 레지스터 파일에 의해 예상되는 추가 전력 소모는 10%에 이른다. 그러나 제안된 구조는 읽기에 대한 동적 전력 소모를 두 레지스터 파일이 공유하므로, 보수적 접근보다 적은 전력 소모를 기대할 수 있다. 실제로, 실험 결과 듀얼 레지스터 파일을 사용하여 늘어난 프로세서의 전력 소모량은 기존 대비 2~3%에 불과한 작은 수치였다. 이는 칩의 온도 제어를 통해 얻을 수 있는 이득에 비하면 미미한 수준이다.

듀얼 정수 레지스터 파일은 정해진 비율에 따라 레지스터 파일에 대한 읽기 접근이 번갈아 수행된다. 단지 각 레지스터 파일의 전력 소모만을 고려한다면, 듀얼 레지스터 파일의 읽기 접근 비율은 1:1이 되어야 한다. 하지만 이 경우 두 레지스터 파일의 온도가 달라지는 문제가 발생한다. 그림 3(a)를 보면 듀얼 레지스터 파일 구조는 85°C 이하의 온도를 유지하고 있으나, 두 레지스터 파일의 온도가 5°C 이상 차이를 보이고 있으며 시간이 갈수록 그 차이가 점점 커지는 것을 확인할 수 있다. 이 온도 차는 듀얼 정수 레지스터 파일의 주변 유닛에서 발생하는 열이 각 정수 레지스터 파일에 영향을 주기 때문에 발생한다. 그림 3(b)는 듀얼 레지스터가 적용된 Alpha 21264 프로세서(그림 4(b))의 온도 분포를 나타내는데, 뜨거운 유닛인 정수 연산(IntExec), 적재/저장 큐(LdStQ), 데이타 캐쉬(Dcache) 유닛이 두 번째 정수 레지스터 파일(IntReg2)보다 첫 번째 정수 레지스터 파일(IntReg1)에 가까이 있는 것을 확인할 수 있다. 이런 기능 유닛의 배치가 고려되지 않고 읽기 접근 비율 1:1을 사용한다면, 프로세서는 듀얼 레지스터 파일 구조의 잠재적 이득인 냉각 비용의 감소, 회로 안전성, 성능 향상을 잃게 된다. 예를 들어, 듀얼 레지스터 파일의 한 레지스터 파일이 높은 온도에 의해 사용할 수 없게 되면 반대쪽 레지스터 파일이 전체 레지스터 파일에 대한 읽기/쓰기 접근을 책임져야 한다. 이는 곧 듀얼 레지스터 파일의 실패를 의미하며, 따라서 레지스터 파일에 의한 이상 발열 현상이 다시 발생하게 되는 것이다. 본 논문에서 레지스터 파일의 분할이 아닌 복제가 사용된 이유도 이와 같다. [12]에서 제안된 레지스터 파일 구조는 실제 프로세서 내에 존재하는 물리적 레지스터 파일의 이용률이 명령어 윈도우 크기(instruction window size)와 명령어간 종속성에 의해 40%에서 60%로 제한된 것을 이용하여 레지스터 파일을 반으로 분할한 뒤 데이타 접근을 번갈아 하는 방식을 사용하였다. 이 구조를 이용하면 전력이나 공간 비용을 효율적으로 조정하는 것이 가능하다. 그러나 레지스



(a) 온도 변화 (b) 온도 분포

그림 3 읽기 접근 비율 1:1인 듀얼 레지스터 파일이 적용된 Alpha 21264 플로어플랜에서 DTM 기법이 수행되지 않았을 때 나타나는 발열 양상

표 2 듀얼 레지스터 파일의 최적 접근 비율

벤치마크	정수 SPEC 벤치마크							
응용프로그램	Crafty	Gzip	Bzip2	Gcc	Eon	Vpr	Mcf	Twolf
접근비율 (IntReg1 : IntReg2)	1:2	1:2	1:2	1:2	1:2	1:2	1:2	1:2
벤치마크	부동소수점 SPEC 벤치마크							
응용프로그램	Mesa	Art	Swim	Mgrid	Applu	Ammp		
접근비율 (IntReg1 : IntReg2)	1:2	1:1	2:1	2:1	3:1	1:1		

표 3 프로세서 Alpha 21264의 특성

프로세서 핵심부(Processor Core)	
Instruction Window Physical Registers Memory Order Queue	RUU 80, LSQ 64 Fetch queue 8 IntQ: 20, FPQ: 15 (only value)
Fetch width Decode width Issue width Commit width	4 4 4 (INT: 4 FP: 2 LSQ: 2) 4
Functional Unit	4 Int ALU / 1 Int Mul/Div 2 FP ALU / 1 FP Mul/Div 2 Memport
분기 예측(Branch Prediction)	
Local History Table Local Predict Global History Register Global Predict Choice Predict	Combined, Bimodal 4K table 2-Level (GAg) 1 table, 12bit history 4K chooser
BTB Return-address stack	2K entry, 2-way 32 entry
메모리 구조(Memory Hierarchy)	
L1 D-cache Size L1 D-cache Associativity	64K, 2-way (LRU) 64B blocks, 2cycle latency
L1 I-cache Size L1 I-cache Associativity	64K, 2-way (LRU) 64B blocks, 2cycle latency
TLB Size (full associativity)	128 entry fully assoc, 4K pages 30 cycle miss latency Memory access latency (255 / 2)

터 파일의 분할은 응용프로그램의 특성을 반영하여 읽기 접근을 수행하는 듀얼 레지스터 파일 구조에 제한을 가하게 되어, 본 논문에서 얻고자 하는 열섬 현상의 제거를 얻어내지 못한다.

발열 간섭 문제를 해결하려면 기능 유닛의 배치를 반영하여 레지스터 파일의 읽기 접근이 이루어져야 한다. 본 논문의 듀얼 레지스터 파일을 적용한 Alpha 21264 플로어플랜(그림 4(b))은 자주 접근하도록 설정하는 레지스터 파일이 두 번째 정수 레지스터 파일임을 보여준다. 표 2는 각 벤치마크에 대한 최적의 읽기 접근 비율을 도출한 결과인데, 모든 정수 벤치마크 응용프로그램은 1:2의 비율로 두 레지스터 파일을 접근한다. 이는 듀얼 레지스터 구조의 읽기 접근 비율을 고정적으로 설정하여 사용할 수 있음을 암시한다. 하지만 부동소수점 벤치마크 응용프로그램에서는 읽기 접근 비율이 불규칙하게 나타난다. 이렇게 읽기 비율이 차이가 나는 것은, 부동소수점 벤치마크의 경우 부동소수점 기능 유닛을 자주 사용하는데 이 유닛이 첫 번째 정수 레지스터 파일보다 두 번째 레지스터 파일에 더 가까이 있는 데에서 비롯한다. 따라서 부동소수점 벤치마크의 경우에는 상대적으로 부동소수점 기능 유닛과 떨어져 있는 첫 번째 정수 레지스터 파일이 읽기 접근을 더 자주 수행하여야 한다. (하지만, *mesa*처럼 최적의 읽기 접근 비율이 정수 벤치마크와 같은 경우도 있다. 자세한 내용은 6장에서 다룬다.)

5. 실험 환경

본 논문은 모의 실험 도구인 *sim-wattch*를 이용하여 실험을 수행하였다. *sim-wattch*는 *Wattch*[14]와 *Sim-*

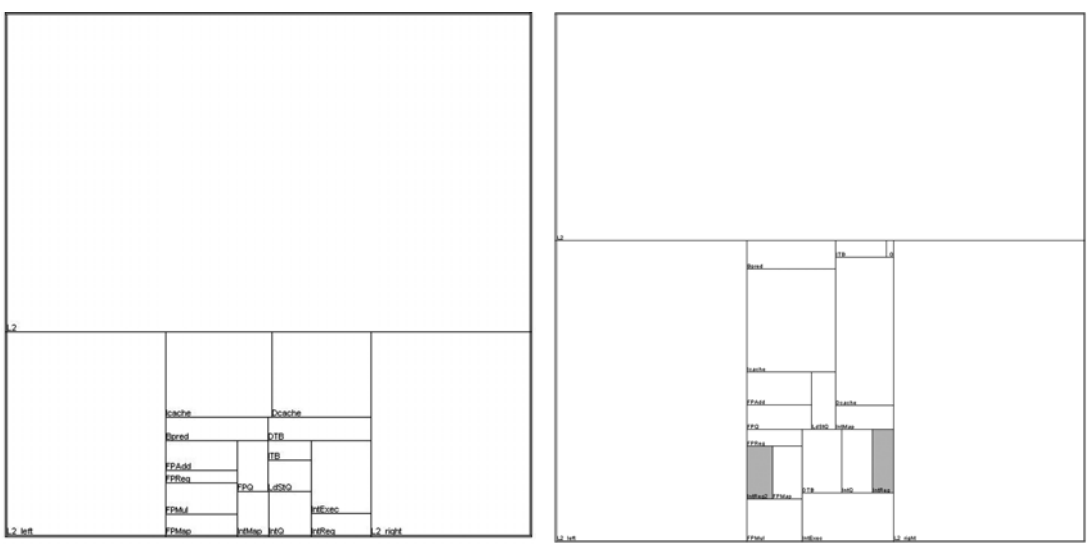
plescalar[15]가 통합된 프로그램으로써, 싸이클 단위로 프로세서의 성능과 전력 소모를 측정하는 것을 가능하게 한다. 본 논문에서는 *sim-wattch*의 프로세서가 듀얼 레지스터 파일 구조를 가질 수 있도록 정수 레지스터 파일을 추가 삽입하였다. 두 정수 레지스터 파일의 접근 시간 및 크기는 동일하게 설정하였다. 모의 실험에 사용된 마이크로프로세서는 Alpha 명령어 집합 구조 기반의 Alpha 21264 프로세서를 사용하였으며, 이에 대한 특성 값 설정은 [7]을 따랐다. 프로세서는 130nm 공정의 3GHz 4-wide 슈퍼스칼라 구조라고 가정하였다.

모의 실험에서는 동적 온도 제어 기법으로 명령어 인출 지연 기법과 가변 전압 주파수 조절 기법을 사용한다. 마이크로프로세서의 온도가 일정 수위에 도달하면 동적 온도 제어가 수행되는데, 단계는 두 단위로 나뉜다. 첫 번째는 구동 단계(*trigger threshold*)로, 프로세서의 온도가 구동 단계를 넘어서면 명령어 인출 지연 기법을 수행하며 온도를 조절한다. 두 번째는 온도를 급격하게 낮춰야 하는 긴급 단계(*emergency threshold*)로, 이 단계에서는 가변 전압 주파수 조절 기법을 이용하여 칩의 온도가 최대 한계를 넘지 않도록 조절한다. 위의 2단계 정책은 고성능 프로세서에서 성능 손실을 적게 하며 온도를 조절하는데 도움을 준다. 본 논문에서 동적 온도 제어의 구동 단계 온도는 긴급 단계 온도보다 2.0°C 낮게 설정되었다.

본 논문은 세분화된 기능 유닛 단위로 온도를 측정하기 위하여 *Hotspot*[7] 모의 실험 도구를 사용한다. *HotSpot*은 전력 데이타를 입력 값으로 받아들여, 아키텍처의 온도 분포를 출력한다(온도 분포는 *sim-wattch*에 전달되어 동적 온도 제어에 쓰인다). *HotSpot*의 설정은 [7]의 명세를 따른다. 본 논문은 정밀한 실험 결과를 얻기 위해 60.0°C를 시작점으로 한 뒤, 3억 개의 명령어를 프로세서의 예열(warm-up) 기간으로 소모한다. 예열 기간이 끝나면, *HotSpot*이 출력하는 안정 상태 온도(steady state temperature)를 프로세서의 초기 온도로 설정하여 5장의 실험 결과에 해당하는 온도와 성능 변화를 기록한다. 모의 실험의 경우 시간이 많이 걸리므로, 실험 결과를 위한 명령어 수행의 개수는 5억 개로 제한했다.

본 논문의 실험은 SPEC CPU2000[16]의 26개 응용 프로그램 중 14개의 벤치마크를 선정하여 수행하였다. 선정된 응용프로그램은 12개의 정수 벤치마크 중 8개인 *crafty*, *gzip*, *bzip2*, *gcc*, *eon*, *vpr*, *mcf*, *twolf*와 14개의 부동소수점 벤치마크 중 6개인 *mesa*, *art*, *swim*, *mgrid*, *applu*, *ammp*이다. 모든 응용프로그램은 Alpha 명령어 집합에 기반한 실행파일이며, *ref* 데이타를 입력 값으로 사용하였다.

본 논문은 플로어플래닝 도구인 *hotfloorplan*[2]을 이용하여 듀얼 레지스터 파일이 적용된 Alpha 21264 프로세서의 플로어플랜(그림 4(b))을 설계하였다(음영 처리된 부분이 듀얼 정수 레지스터 파일이다). *hotfloorplan*도구 및 Alpha 21264 플로어플랜(그림 4(a))의 구성과 명세는 *Hotspot* 도구 모음[17]에 의하여 제공된다. 그림 4(a), 4(b)의 플로어플랜 모두 130nm 공정에 맞게 조정되었다. 그림 4는 QUILT[18]를 이용하여 출력하였



(a) 기존 프로세서의 플로어플랜 (b) 듀얼 레지스터 파일을 적용한 프로세서의 플로어플랜

그림 4 Alpha 21264(EV68)의 플로어플랜

는데, 그림 3(b)의 플로어플랜과는 달리 왼쪽 윗부분이 기준좌표가 되어 그림이 반전되어 있음을 유의해야 한다(그림 3(b)의 플로어플랜은 왼쪽 아랫부분이 기준좌표이다).

6. 발열 안정성 및 성능 평가

본 장에서는 동적 온도 제어 기법이 사용되지 않은 상황에서 듀얼 정수 레지스터 파일을 적용하면 칩의 온도에 어떤 변화가 있는지를 확인한다. 그리하여, 동적 온도 기법이 사용되는 프로세서에서 성능에 어떤 이득이 있는가를 평가한다.

6.1 듀얼 레지스터 파일 구조에 의한 발열 감소

정수 레지스터 파일과 다른 기능 유닛의 온도 차이가 클수록, 듀얼 레지스터 파일을 사용하였을 때의 이득을 크게 기대할 수 있다. 그림 5는 동적 온도 제어 기법을 사용하지 않은 Alpha 21264 프로세서에서 기존의 레지스터 파일과 듀얼 레지스터 파일이 사용되었을 때 나타나는 정수 레지스터 파일의 최고 온도를 나타낸다. 여기에서 사용된 듀얼 레지스터 파일의 읽기 접근 비율은 표 2의 결과를 적용하였다. 본 논문의 첫 번째 목표는 열섬 가능 인자인 정수 레지스터 파일의 온도를 낮추는 것이며, 위 목표가 듀얼 레지스터 파일에 의해 성공적으로 이뤄진 것을 그림 5에서 확인할 수 있다. 정수 벤치마크의 경우 최소 22.2°C(*eon*)에서 최대 28.0°C(*mcf*)까지 정수 레지스터 파일의 온도가 떨어졌으며, 부동소수점 벤치마크는 최소 3.7°C(*applu*)에서 최대 22.8°C(*mesa*)까지 다양한 분포를 보이며 온도가 떨어졌다.

정수 레지스터 파일의 온도 하강 만을 놓고 봤을 때, 듀얼 레지스터 파일의 효과는 그다지 성공적이지 않다는 비판이 있을 수 있다. 부동소수점 벤치마크의 경우 정수형과 같은 성공적인 결과를 보장하지 않기 때문이다. 그러나, 모든 벤치마크의 정수 레지스터 파일의 온도는 80°C 근처에 있거나 혹은 그 밑에서 형성되고 있다(정수 레지스터 파일의 최고 온도는 85.5°C(*eon*)이며, 최저 온도는 72.2°C(*ammp*)이다). 그림 1에서 뜨거운 온도 집단에 속하는 기능 유닛의 최고 온도가 평균 85.0°C를 넘는다는 것을 상기해 볼 때, 위 결과는 정수 레지스터 파일이 열섬 현상 인자로부터 제외되었다는 것을 의미한다. 한편, 듀얼 레지스터 파일의 두 레지스터 파일간 온도 차이는 최대 5.5°C(*mcf*)이며, 평균 3.0°C의 차이를 보이고 있다. 제안된 구조에서 두 레지스터 파일 간의 온도 차이가 작아야 한다는 것을 상기하면, 평균 3.0°C의 온도 차이는 만족할 만한 결과이다.

위의 결과를 요약하면, 듀얼 레지스터 파일이 적용된 마이크로프로세서는 85.0°C 이상의 온도에서 수행되는 동적 온도 제어 기법의 수행을 효과적으로 줄일 수 있다. 그림 6의 온도 변화 양상이 이를 뒷받침한다. 그림 6(b), 6(d)를 보면 듀얼 레지스터 파일을 적용한 프로세서가 정수 레지스터 파일이 아닌 다른 기능 유닛에 의해 동적 온도 제어가 수행되는데, *eon*의 경우 데이타 캐쉬와 적재/저장 큐에 의하여, *mgrid*의 경우 부동소수점 연산 유닛에 의하여 동적 온도 제어가 수행된다(그림 6의 동적 온도 제어 기법은 90.0°C를 긴급 단계 온도로 사용한다).

6.2 듀얼 레지스터 파일에 의한 성능 향상

듀얼 레지스터에 의해 동적 온도 제어 기법이 받은 영향을 평가하기 위해, 본 장에서는 동적 온도 제어의 시작부터 종료까지의 시간을 측정한 값을 동적 온도 제어 기간으로 정하여 비교해보았다. 그림 7은 해당 응용 프로그램의 전체 실행 싸이클을 기준으로 정규화된 동적 온도 제어 기법 호출 기간이다. 정수 벤치마크의 경우, 기존 프로세서를 보면 긴급 단계 온도가 90°C일 때 동적 온도 제어 기간은 전체 프로그램 실행시간 대비 41.5%에서 54.3%를 차지하며, 긴급 단계 온도가 100°C

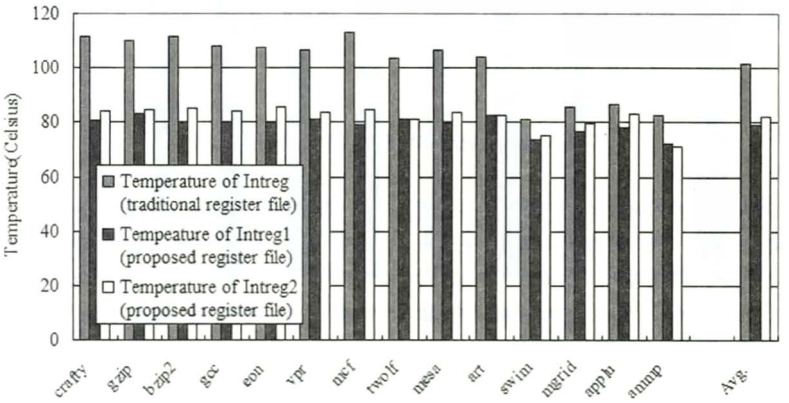
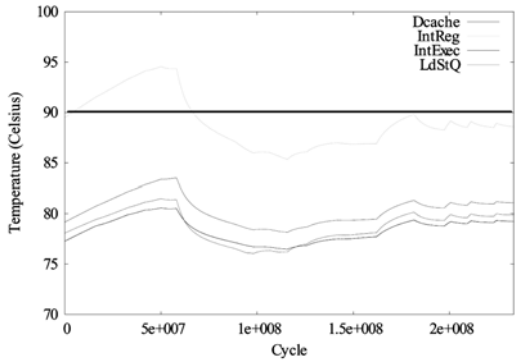
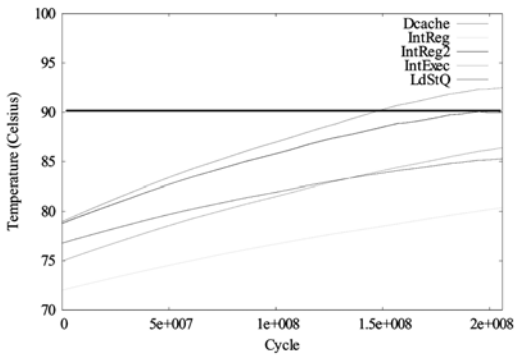


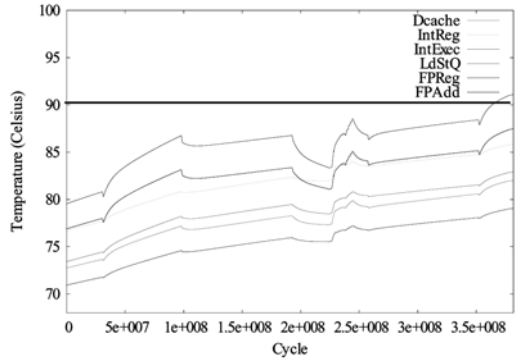
그림 5 기존 프로세서와 듀얼 레지스터 파일을 적용한 프로세서의 최고 온도 비교



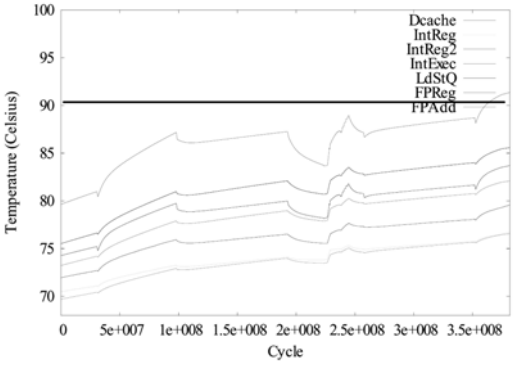
(a) eon, DTM



(b) eon, DTM+Dual

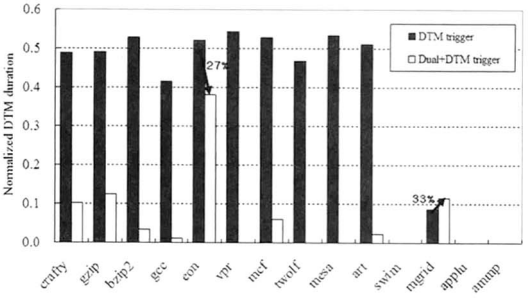


(c) mgrid, DTM

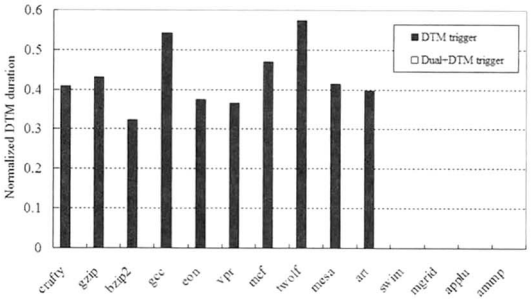


(d) mgrid, DTM+Dual

그림 6 Alpha 21264 프로세서의 온도 변화 양상



(a) 긴급 단계 온도: 90.0°C



(b) 긴급 단계 온도: 100.0°C

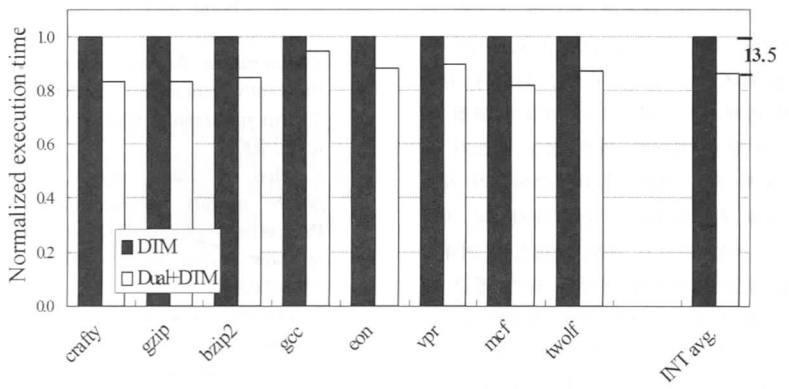
그림 7 정규화된 동적 온도 제어 기간 (cycle)

일 때 32.3%에서 57.3%를 차지하는 것을 확인할 수 있다. 이와 같은 호출 기간은 듀얼 레지스터 파일 구조를 사용하였을 때 극적으로 줄어들어, 긴급 단계 온도 100°C의 경우 동적 온도 제어 기법이 전혀 수행되지 않았다. 물론 긴급 단계 온도가 90°C일 때에는 *eon*에서 27%의 호출 기간이 감소하는 저조한 결과를 보여주지만, 적어도 모든 정수 벤치마크의 동적 온도 제어 기간이 줄어들게 된 것을 확인할 수 있다. 동적 온도 제어의 긴급 단계 온도가 90°C인 상황에서 정수 벤치마크의 결과가 그다지 좋지 않은 이유는, 듀얼 레지스터 구조에 의해 정수 레지스터 파일의 온도가 줄어들었더라도 다른 기능 유닛들의 온도가 높아 동적 온도 제어 기법이 수행되었기 때문이다. (그림 2를 보면 데이타 캐쉬, 정수연산, 데이타 적재/저장 큐 유닛의 온도가 80°C를 넘

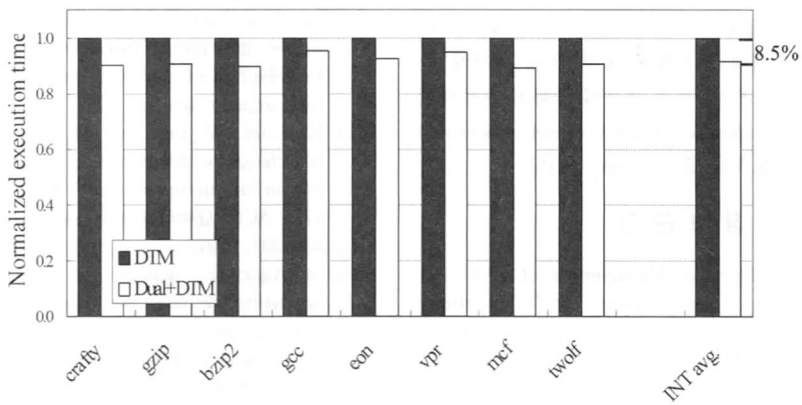
는 것을 볼 수 있다.) 앞 장의 그림 6(b)가 대표적인 예이다. 데이타 캐쉬는 긴급 단계 온도인 90°C를 넘어서면서 동적 온도 제어의 호출을 유지하고 있고, 이 경우에는 정수형 벤치마크에서 동적 온도 제어 기간의 감소를 기대할 수 없다.

반면에, 부동소수점 벤치마크의 경우에는 동적 온도 제어 기간이 기존의 프로세서나 듀얼 레지스터 파일을 적용한 프로세서 모두에서 나타나지 않는 경향을 보인다. 긴급 단계 온도 100°C의 *swim*, *mgrid*, *applu*, *ammp*와 긴급 단계 온도 90°C의 *swim*, *applu*, *ammp*가 이에 해당한다. 3장에서 설명했듯이 부동소수점 벤치마크는 정수 기능 유닛보다 부동소수점 기능 유닛을 더 사용하지만 그에 따른 온도가 다른 기능 유닛과 크게 차이가 나지 않으며, 동적 온도 제어를 수행할 만큼 온도가 높게 올라가지도 않는다. 하지만 긴급 단계 온도가 낮게 설정된 경우에는 부동소수점 기능 유닛의 온도가 구동 단계 온도를 넘게 되어, 동적 온도 제어 기간을 증가시킬 수 있다. (그림 6(d)의 부동 소수점 덧셈 유닛(FPAdd)에 의하여. 긴급 단계 온도 90°C의 *mgrid*는 동적 온도 제어 기간이 33%나 더 증가했다.) 정리하면, 부동소수점 벤치마크에서는 듀얼 레지스터 파일을 사용하였을 때 큰 이득을 기대할 수 없다. 하지만, 부동소수점 벤치마크의 *mesa*와 *art*는 동적 온도 제어 기간이 큰 폭으로 줄어드는 것을 볼 수 있다. (*mesa*는 3차원 그래픽 라이브러리를 주로 사용하고, *art*는 신경망을 이용한 이미지 인식을 수행한다.) 이는, 부동소수점 벤치마크가 정수 연산(정수 계산, 적재/저장, 분기)을 많이 사용하게 될 경우 동적 온도 제어 기간의 감소 효과를 기대할 수 있음을 의미한다.

이러한 동적 온도 제어 기간의 감소는 마이크로프로세서의 성능 향상과 직결된다. 그림 8은 동적 온도 제어 기법을 사용하는 프로세서에서 기존의 레지스터 파일 구조와 본 논문의 듀얼 레지스터 파일 구조를 사용한 경우를 비교한 정규화된 정수 벤치마크의 실행 시간이



(a) 긴급 단계 온도 90.0°C



(b) 긴급 단계 온도 100.0°C

그림 8 동적 온도 제어 기법이 수행되는 프로세서에서 실행된 정수 벤치마크의 실행시간

다. 벤치마크의 수행 시간이 긴급 단계 온도 90.0°C 에서 기존 실행시간 대비 평균 86.5%, 98.0°C에서 평균 91.5%에 해당하는 시간으로 축소되었다. 가장 두드러진 성능 향상을 보인 응용프로그램은 *mcf*로, 긴급 단계 온도 90.0°C에서 18.0%, 100.0°C에서 11.0%의 수행 시간이 단축되었다. 한편, 부동소수점 벤치마크의 경우에는 동적 온도 제어 기간의 차이가 없기 때문에 벤치마크 수행 시간에 변화가 없어 결과를 보여주는 것이 무의미하다. 하지만, *mesa*와 *art*의 경우에는 긴급 단계 온도 90.0°C에서 각각 12.3%와 8.7%, 98.0°C에서 각각 5.0%와 1.5%의 실행시간 단축이 발생하였다. 따라서, 본 논문은 듀얼 레지스터 파일이 정수형 응용프로그램을 실행할 경우에 탁월한 효과를 보이며, 일부 부동소수점형 응용프로그램을 실행할 때에도 효과를 기대할 수 있다고 예상한다.

7. 결론 및 향후 과제

본 논문은 아키텍처 수준의 온도 관리에 효과적인 듀얼 정수 레지스터 파일과 이 구조에 필요한 최적의 읽기 접근을 제안하였다. 기존의 온도 인지 프로세서와 비교하였을 때, 듀얼 레지스터 파일 구조는 다음과 같은 장점을 가진다. 첫 번째, 정수 레지스터 파일의 열섬 현상이 완화되어, 동적 온도 제어에 의한 성능 감소가 최소화 된다. 또한, 동적 온도 제어 기간이 줄어들기 때문에, 어떤 제어 기법과 조합하더라도 듀얼 레지스터 파일에 의한 프로세서의 성능 향상과 온도 안정성을 기대할 수 있다. 두 번째, 듀얼 레지스터 파일은 읽기 접근 비율을 조정하는 방법을 통해, 본 논문에서 제시된 실험과 다른 플로어플랜을 가진 어떤 아키텍처에도 적용 가능하다. 마지막으로, 본 논문에서는 다루지 않았지만 프로세서의 열섬 현상이 제거된 것은 온도에 따라 변하는 누설 전력(leakage power)에 영향을 주기 때문에, 전체 칩의 전력 소모를 줄일 수 있다. 또한, 듀얼 레지스터 파일은 응용프로그램에 최적화된 동적 온도 제어의 수행을 기대할 수 있으므로, 특정 응용에 특화된 프로세서(ASIP, application specific instruction-set processor)의 성능 향상에도 도움을 줄 수 있을 것이다.

참 고 문 헌

[1] R. Mahajan, *"Thermal Management of CPUs: A Perspective on Trends, Needs, and Opportunities,"* In Proceedings of the 8th International Workshop on THERMal INvestigations of ICs and Systems, 2002.

[2] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, *"A Case for Thermal-Aware Floorplanning at the Microarchitectural Level,"* Instruction-Level Parallelism, Vol. 8, pp. 1-16, 2005.

[3] K. Puttaswamy and G. H. Loh, *"Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D-Integrated Processors,"* In Proceedings of the 13th IEEE International Symposium on High Performance Computer Architecture, pp. 193-204, 2007.

[4] D. Brooks and M. Martonosi, *"Dynamic Thermal Management for High-Performance Microprocessors,"* In Proceedings of the 7th International Symposium on High-Performance Computer Architecture, 2001.

[5] SIA, *"The International Technology Roadmap for Semiconductors,"* 2005

[6] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, *"Managing the Impact of Increasing Microprocessor Power Consumption,"* Intel Tech Journal, Vol. Q1, 2001.

[7] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, *"Temperature-Aware Microarchitecture,"* In Proceedings of the 30th International Symposium on Computer Architecture, pp. 2-13, 2003.

[8] S. W. Chung and K. Skadron, *"Using On-Chip Event Counters for High-Resolution, Real-Time Temperature Measurements,"* In Proceedings of the tenth intersociety conference on Thermal and Thermomechanical Phenomena in Electronics Systems, 2006.

[9] S. Heo, K. Barr, and K. Asanovic, *"Reducing power density through activity migration,"* In Proceedings of the 2003 International Symposium on Low Power Electronics and Design, pp. 217-222, 2003.

[10] J. Deeney, *"Thermal modeling and measurement of large high power silicon devices with asymmetric power distribution,"* In Proceedings of the international symposium on microelectronics, pp. 300-305, 2002.

[11] F. J. Mesa-Martinez, M. Brown, J. Nayfach-Battilana, and J. Renau, *"Measuring performance, power, and temperature from real processors,"* In Proceedings of the 2007 workshop on Experimental computer science, 2007.

[12] K. Patel, W. Lee, and M. Pedram, *"Active bank switching for temperature control of the register file in a microprocessor,"* In Proceedings of the 17th ACM Gread Lakes symposium on VLSI, pp. 231-234, 2007.

[13] A. Agarwal, *"Analysis of cache performance for operating systems and multiprogramming,"* Ph.D. Thesis, Stanford University, 1987

[14] D. Brooks, V. Tiwari, and M. Martonosi, *"Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,"* In Proceedings of the 27th Annual Symposium on Computer Architecture, pp. 83-94, 2000.

[15] T. Austin, E. Larson, and D. Ernst, *"SimpleScalar: An Infrastructure for computer system modeling,"* IEEE Computer, Vol. 35, No.2, pp. 59-67, 2002.
[16] *SPEC, Standard Performance Evaluation Corporation.* http://www.spec.org/cpu2000/.
[17] *Hotspot Tool Set v3.1.* http://lava.cs.virginia.edu/HotSpot/download_form.html.
[18] G. J. Briggs, E. J. Tan, N. A. Nelson, and D. H. Albonesi, *"QUILT: A GUI-based Integrated Circuit Floorplanning Environment for Computer Architecture Research and Education,"* In Proceedings of the Computer Architecture Education, 2005.

최 진 항
2008년 2월 고려대학교 정보통신대학 컴퓨터과학 학사. 2008년 3월~현재 고려대학교 컴퓨터통신공학부 석사과정 재학중. 관심분야는 컴퓨터 구조, 고성능 컴퓨터, 실시간 데이타 처리

공 준 호
2007년 8월 고려대학교 정보통신대학 컴퓨터과학 학사. 2007년 9월~현재 고려대학교 컴퓨터통신공학부 석사과정 재학중. 관심분야는 컴퓨터 구조, 고성능 컴퓨터, 임베디드 프로세서

정 의 영
2002년 6월 Stanford University 전기공학 박사. 1990년 1월~2005년 8월 삼성전자 반도체총괄 수석 연구원. 2005년 9월~현재 연세대학교 전기전자 공학부 부교수. 관심분야는 System on Chip 아키텍처 및 디자인, 저전력 설계, VLSI CAD

정 성 우
2003년 2월 서울대학교 전기 컴퓨터공학 박사. 2003년 1월~2005년 2월 삼성전자 반도체총괄, Senior engineer. 2005년 3월~2006년 1월 방문 연구원, Department of Computer Science, University of Virginia. 2006년 3월~현재 고려대학교 컴퓨터통신공학부 조교수. 관심분야는 컴퓨터 구조, 프로세서 온도 관리, 플래쉬 메모리, System on Chip