온도 인지 마이크로프로세서에서 연산 이관을 위한 유닛 선택 기법

(Active Unit Selection Method for Computation Migration in Temperature-Aware Microprocessors)

이 병 석 † 김 철 홍 ††
(Byeong Seok Lee) (Cheol Hong Kim)
이 정 아 †††
(Jeong-A Lee)

요 약 마이크로프로세서의 온도 관리를 위해 사용되는 대표적인 기술인 동적 온도 관리 기법이 적용되면 임계온도 이상의 발열 발생시 온도를 제어하기 위해 성능이 저하되는 단점이 있다. 따라서 마이크로프로세서의 발열 온도를 낮추면 동적 온도 관리 기법을 통해 온도를 제어하는 시간이 줄어들면서 성능 저하를 최소화 시킬 수 있다. 본 논문에서는 유닛의 발열 제어를 위해 사용되는 연산 이관시 유닛을 선택하는 기준에 대한 다양한 기법들을 모의 실험을 통하여 비교 분석함으로써 유닛의 발열 현상으로 인한 마이크로프로세서의 성능 저하를 최소화시킬 수 있는 방안을 도출하고자 한다. 모의 실험 결과, 동적 연산 이관 기법에서 임계 온도와 유닛 온도 사이의 차이를 기준으로 동작할 유닛을 선택하는 기법이 발열에 가장 효과적으로 대응하여 성능이 우수하다는 것을 확인할 수 있다.

키워드 : 온도 인지 마이크로프로세서, 동적 온도 제어, 연산 이관 기법

Abstract Dynamic Thermal Management (DTM) degrades the processor performance for lowering temperature. For this reason, reducing the peak temperature on microprocessors can improve the performance by reducing the performance loss due to DTM. In this study, we analyze various unit selection techniques for computation migration. According to our simulation results, dynamic computation migration based on the thermal difference between the units shows best performance among compared models.

Key words : Temperature-Aware Microprocessor, Dynamic Thermal Management (DTM), Computation Migration

1. 서 론

반도체 공정 기술의 발전으로 마이크로프로세서의 성능은 크게 향상된 반면, 높은 주파수와 고밀도 집적화로 인해 전력 소모가 크게 증가하면서 마이크로프로세서 내부의 특정 유닛이 과열되는 열섬(hotspot) 현상이 성능 향상을 제한시키는 새로운 문제로 등장하였다. 일부 유닛의 전력 밀도가 높아지면서 발생하는 열섬 현상은 시스템을 오동작 시키거나 심할 경우에는 칩의 회로가 손상되어서 신뢰성을 저하시키는 원인이 되기도 한다. 과열된 칩을 냉각하기 위해 사용하는 물리적인 방법은 냉각 성능에 한계가 있고 냉각 비용이 비싸다는 단점이 있다[1].

최근 온도 인지 마이크로프로세서(Temperature-Aware Microprocessor)는 동적 온도 관리(Dynamic Thermal Management)[2] 기술을 이용하여 발열 문제를 해결하고 있다. 동적 온도 관리 기법은 마이크로프로세서에 내장된 디지털 온도 센서에서 측정한 온도가 일정 온도 이상이 되면 칩의 발열을 제어하는 방식으로, 가변 주파수 조절(Dynamic Frequency Scaling), 가변 전압 조절(Dynamic Voltage Scaling), 명령어 인출 지연(Fetch Throttling) 기법 등을 적용하여 칩을 냉각시키기 때문에 발열 제어 시 마이크로프로세서의 성능이 저하되는 단점이 있다. 보조 유닛을 추가하여 주 유닛의 사용률을 낮춤으로써 특정 유닛의 발열을 감소시키는 연산 이관(Computation Migrating) 기법[3]은 듀얼 정수 레지스터 파일 구조[4,5], 최고 교차점 온도 제어[6]에서 이용하고 있다. 연산 이관 기법은 적절한 시점에 주 유닛에서 보조 유닛으로(또는 보조 유닛에서 주 유닛으로) 연

· 본 연구는 한국연구재단 핵심연구(구 한국과학재단 특정기초연구, R01-2007-000-20750-0)의 지원과 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터(NIPA-2009-(C1090-0903-0008)) 지원사업의 연구결과로 수행되었음
· 이 논문은 2009 한국컴퓨터종합학술대회에서 '온도 인지 마이크로 프로세서에서 동적 연산 이관을 위한 유닛 선택 기법에 관한 비교 분석'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 조선대학교 컴퓨터학과
novrain@chosun.ac.kr
†† 종신회원 : 전남대학교 전자컴퓨터공학부 교수
chkim22@chonnam.ac.kr
††† 종신회원 : 조선대학교 컴퓨터공학부 교수
jalee@chosun.ac.kr

논문접수 : 2009년 8월 14일
심사완료 : 2009년 11월 10일

산이 이관되어야만 유닛의 온도를 효과적으로 낮출 수 있다[5]. 동적 온도 관리 기법은 마이크로프로세서 내부의 특정 유닛 온도가 미리 설정한 임계 온도보다 높을 시점에서만 동작되므로[2], 내부 유닛들의 온도가 낮을수록 마이크로프로세서의 성능 저하를 최소화하면서 발열로 인한 신뢰성 저하 현상을 방지할 수 있다. 본 논문에서는 연산 이관 기법에서 사용될 수 있는 다양한 유닛 선택 기법들에 대하여 연구하고, 실험 도구를 이용하여 각 기법 적용으로 인한 발열 온도와 이에 따른 마이크로프로세서의 성능을 비교 분석한다.

2. 연산 이관을 위한 유닛 선택 기준

연산 이관을 수행하기 위해서는 동작할 유닛을 선택하는 기준이 필요하다. 연산 이관 기법은 미리 정해진 시간을 기준으로 동작할 유닛을 선택하는 정적 연산 이관 기법과, 측정된 유닛의 온도를 기준으로 동작할 유닛을 선택하는 동적 연산 이관 기법으로 분류할 수 있다. 정적 연산 이관 기법은 부가적인 하드웨어가 필요 없고 동작할 유닛을 선택하는 방법이 단순하다는 장점이 있지만, 예상치 못한 원인으로 인하여 유닛이 과열되더라도 동적으로 대처 할 수 없다는 단점이 있다. 동적 연산 이관 기법은 유닛의 발열 문제를 즉각적으로 대처할 수 있다는 장점이 있지만, 온도 센서와 비교기(Comparator) 추가로 인한 하드웨어 복잡도 증가와 유닛을 선택하는 기준이 잘못될 경우에는 한 유닛만이 과열 될 수 있다는 문제점이 있다. 따라서 연산 이관 기법은 동작할 유닛을 선택하는 기준에 따라 성능이 결정된다고 할 수 있다. 본 논문에서는 효율적인 연산 이관 기법 구현을 위해 동작할 유닛을 선택하는 다양한 기준들을 비교, 분석하고자 한다.

2.1 클록 카운터

클록 카운터 연산 이관 기법에서는 클록 카운터를 기준으로 동작할 유닛을 선택한다. 클록 카운터 값을 저장하는 레지스터를 R_{COUNT}, 하나의 유닛이 사용되는 시간을 $C_{COMPARE}$, 주 유닛을 $UNIT_{PRIMARY}$, 보조 유닛을 $UNIT_{SENCONDARY}$라고 할 때, 유닛을 선택하는 과정은 그림 1에서 보이는 바와 같다. 클록 주기에 의해 클록 카운터 값이 업데이트 되면(1) 미리 설정한 유닛 사용 시간 값과 클록 카운터 값을 비교한다(2). 비교한 결과를 이용하여 주 유닛(3)이나 보조 유닛(4)을 선택하도록 한다.

```
1:  R_COUNT = R_COUNT + 1
2:  IF (R_COUNT & C_COMPARE) THEN
3:      SELECT UNIT_SENCONDARY
    ELSE
4:      SELECT UNIT_PRIMARY
```

그림 1 클록 카운터를 기준으로 유닛을 선택하는 과정

클록 카운터 기법은 클록 카운터 값을 이용하여 간단하게 동작할 유닛을 선택할 수 있다는 장점이 있지만, 외부의 영향에 따른 온도 변화에는 동적으로 대처할 수 없다는 단점이 있다.

2.2 주 유닛의 온도

온도 센서를 이용한 동적 연산 이관 기법 구현 방식 중 하나로 주 유닛의 온도가 임계 온도(Threshold Temperature)보다 높을 때 보조 유닛을 선택하도록 한다. 주 유닛의 온도를 $T_{PRIMARY}$, 유닛 전환을 결정하는 임계 온도를 $T_{THRESHOLD}$라고 할 때, 유닛을 선택하는 과정은 그림 2와 같다. 주 유닛의 온도가 미리 설정한 임계 온도보다 높은 경우에는(1) 보조 유닛을 선택하도록(2) 동작한다.

```
1:  IF (T_PRIMARY >= T_THRESHOLD) THEN
2:      SELECT UNIT_SENCONDARY
    ELSE
3:      SELECT UNIT_PRIMARY
```

그림 2 주 유닛 온도를 기준으로 유닛을 선택하는 과정

이 기법은 유닛을 선택하는 방법이 단순하다는 장점이 있지만, 주 유닛의 온도가 임계 온도보다 높을 경우에는 보조 유닛만 지속적으로 동작하면서 오히려 보조 유닛이 과열 되는 문제점이 발생할 수 있다.

2.3 동작중인 유닛의 온도

동작중인 유닛의 온도가 임계 온도보다 높을 때 동작하지 않고 있는 유닛을 선택하는 기법이다. 보조 유닛의 온도 값을 $T_{SECONDARY}$라고 할 때, 유닛을 선택하는 과정은 그림 3과 같다. 동작 과정은 크게 주 유닛이 동작할 때(1~3)와 보조 유닛이 동작할 때(4~6)로 구분할 수 있다. 주 유닛이 동작하면서(1) 발열 온도가 임계 온도보다 높을 경우에는(2) 보조 유닛을 선택한다(3). 보조 유닛이 동작할 때 역시 발열 온도가 임계 온도보다 높으면 다시 주 유닛을 선택하는 과정으로 동작한다.

이 기법은 주 유닛의 온도가 임계 온도보다 높을 경우에 보조 유닛이 지속적으로 동작하는 문제점을 해결할 수 있지만, 두 유닛 모두 임계 온도보다 높을 경우에는 상호 번갈아 선택되면서 모두 온도가 상승하는 문제점이 발생할 수 있다.

```
1:  IF (UNIT_PRIMARY) THEN
2:      IF (T_PRIMARY >= T_THRESHOLD) THEN
3:          SELECT UNIT_SENCONDARY
4:  ELSE IF (UNIT_SENCONDARY) THEN
5:      IF (T_SENCONDARY >= T_THRESHOLD) THEN
6:          SELECT UNIT_PRIMARY
```

그림 3 동작중인 유닛 온도를 기준으로 유닛을 선택하는 과정

2.4 유닛의 온도와 냉각 임계 온도

동작중인 유닛의 발열 온도가 임계 온도보다 높을지라도, 선택할 유닛의 온도가 냉각 임계 온도(Cooling Threshold Temperature)보다 낮을 때까지 기다린 후 선택한다. 냉각의 기준이 되는 냉각 임계 온도의 값을 $T_{COOLING}$이라고 할 때, 유닛을 선택하는 과정은 그림 4와 같다. 주 유닛이 동작하면서(1) 발열 온도가 임계 온도보다 높다 하더라도(2) 보조 유닛의 발열 온도가 설정한 냉각 임계 온도보다 낮을 경우에만(3) 보조 유닛을 선택하는(4) 과정으로 동작한다.

```
1:  IF (UNIT_PRIMARY) THEN
2:      IF (T_PRIMARY >= T_THRESHOLD
3:      and T_SENCONDARY <= T_COLLING) THEN
4:          SELECT UNIT_SENCONDARY
5:  ELSE IF (UNIT_SENCONDARY) THEN
6:      IF (T_SENCONDARY >= T_THRESHOLD
7:      and T_PRIMARY <= T_COLLING) THEN
8:          SELECT UNIT_PRIMARY
```

그림 4 동작 유닛의 온도와 냉각 임계 온도를 기준으로 유닛을 선택하는 과정

이 기법은 두 유닛의 발열 온도가 모두 임계 온도보다 높은 경우에 선택할 유닛이 냉각할 수 있는 시간을 주어서 두 유닛의 온도가 같이 올라가는 문제를 해결할 수 있지만, 선택할 유닛이 인접한 주변 유닛의 발열로 인하여 냉각 임계 온도보다 냉각되지 않는 경우에는 대체 유닛을 선택할 수 없는 문제점이 발생할 수 있다.

2.5 동작 유닛과 유휴 유닛의 온도차

동작중인 유닛의 발열 온도가 임계 온도보다 높을지라도, 선택할 유닛의 온도가 일정한 차이만큼 냉각이 될 때까지 기다린 후 선택한다. 주 유닛과 보조 유닛의 온도 차이 값을 $T_{DIFFERENCE}$, 냉각 기준이 되는 온도 차이 값을 $S_{DIFFERENCE}$라고 할 때, 유닛을 선택하는 과정은 그림 5와 같다. 주 유닛과 보조 유닛의 온도 차이 값은 아날로그 비교기(Analog Comparator)를 이용하여 구할 수 있다(1). 주 유닛이 동작하면서(2) 발열 온도가 임계 온도보다 높다 하더라도(3) 보조 유닛과의 온도 차이가 설정한 온도 차이 값만큼 발생하는 경우에만(4) 보조 유닛을 선택하는(5) 과정으로 동작한다.

```
1:  T_DIFFERENCE = |T_PRIMARY - T_SENCONDARY|
2:  IF (UNIT_PRIMARY) THEN
3:      IF (T_PRIMARY >= T_THRESHOLD
4:      and T_DIFFERENCE >= S_DIFFERENCE) THEN
5:          SELECT UNIT_SENCONDARY
6:  ELSE IF (UNIT_SENCONDARY) THEN
7:      IF (T_SENCONDARY >= T_THRESHOLD
8:      and T_DIFFERENCE >= S_DIFFERENCE) THEN
9:          SELECT UNIT_PRIMARY
```

그림 5 동작 유닛의 온도와 두 유닛의 온도 차이를 기준으로 유닛을 선택하는 과정

이 기법은 선택할 유닛이 냉각할 수 있는 시간을 주는 것과 동시에 냉각 임계 온도가 동적으로 변하므로 주변 유닛의 상황이 변하더라도 연산 이관을 수행할 수 있다는 장점이 있다.

3. 모의실험 및 분석

3.1 실험 대상 유닛 구조

본 논문은 실험 대상으로 마이크로프로세서의 성능 저하 없이 연산 이관을 수행하기 위하여 그림 6과 같은 구조로 유닛을 구성하였다. 이 구조는 주 유닛($UNIT_{PRIMARY}$)과 보조 유닛($UNIT_{SENCONDARY}$), 유닛 선택기(UNIT Selector), 멀티플렉서 등으로 구성된다. 데이터는 주 유닛과 보조 유닛에 같이 입력되지만, 유닛 선택기가 선택한 유닛만 동작하면서 전력 소비와 함께 열이 발생한다. 선택되지 않는 유닛은 전력 소비가 거의 없기 때문에 냉각을 하게 된다. 마지막으로 멀티플렉서를 이용하여 선택한 유닛에서 데이터가 출력되도록 한다. 이러한 연산 이관 구조는 유닛이 전환하면서 발생하는 지연이 없기 때문에, 지연으로 인한 성능 저하는 발생하지 않는다.

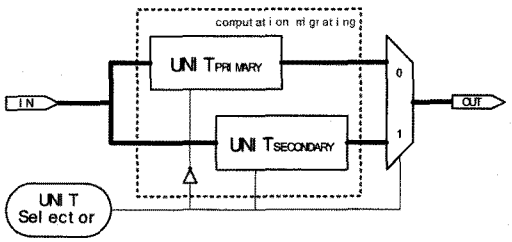


그림 6 동적 연산 이관을 위한 유닛 구조

3.2 실험 환경

모의 실험 대상 프로세서는 AlphaEV6(21264) 프로세서로 가정하였으며, Wattch[7], hotfloorplan[8], HotSpot[9] 도구들을 활용하여 모의 실험을 수행하였다. 기본적인 모의 실험 환경은 [4]와 동일하게 설정하였으며, 부동소수점 가산기(FPAdd)에 보조 부동소수점 가산기(FPAdd2)를 추가하여 연산 이관 기법을 수행할 수 있도록 실험 환경을 구성하였다. 여기서 보조 부동소수점 가산기는 데이터 전송 경로를 고려하여 부동소수점 레지스터 파일(FPReg)과 인접하게 배치하였다. 또한 정밀한 실험 결과를 얻기 위하여 [5]와 같이 3억 개의 명령어로 프로세서 예열(warm-up)한 후, HotSpot이 출력하는 안정 상태 온도(Steady State Temperature)로 프로세서의 초기 온도를 설정하였다. 연산 이관 기법에 대한 유닛의

최고 온도 변화를 모의실험 하기 위해 동적 온도 관리 기법은 사용하지 않는다. 벤치마크 프로그램은 SPEC CPU2000[10] 중에서 부동소수점 가산기의 최고 온도가 가장 높은 lucas 부동소수점 벤치마크를 사용하였다. 정적 연산 이관 기법에서 선택 주기는 228 클록 주기로 설정한다. 동적 연산 이관 기법에서 필요한 임계 온도는 80°C, 냉각 임계 온도는 78°C, 두 유닛의 온도 차이는 2°C로 설정한다.

3.3 유닛 선택 기법 분석

모의 실험을 통한 부동소수점 가산기 유닛에 대한 온도 변화 결과는 그림 7과 같다. 그림 7(a)는 한 개의 부동소수점 가산기 유닛의 온도를, 그림 7(b)~(f)는 연산 이관 기법을 적용하여 두 개의 부동소수점 가산기 유닛의 온도 변화를 보여준다.

그림 7(a)는 연산 이관 기법을 사용하지 않는 경우에 대한 부동소수점 가산기 유닛의 온도 변화를 나타내는데, 시간이 경과됨에 따라 계속 뜨거워지는 것을 볼 수 있다.

그림 7(b)에서는 정적 연산 이관 기법으로 일정한 주기로 주 유닛과 부 유닛을 선택하는 경우에 대한 두 유닛의 온도 변화를 나타내고 있다. 두 유닛이 발열과 냉각을 반복하면서 유닛의 전체 온도가 연산 이관을 사용하지 않는 그림 7(a)에 비해서는 낮은 것을 확인할 수 있다.

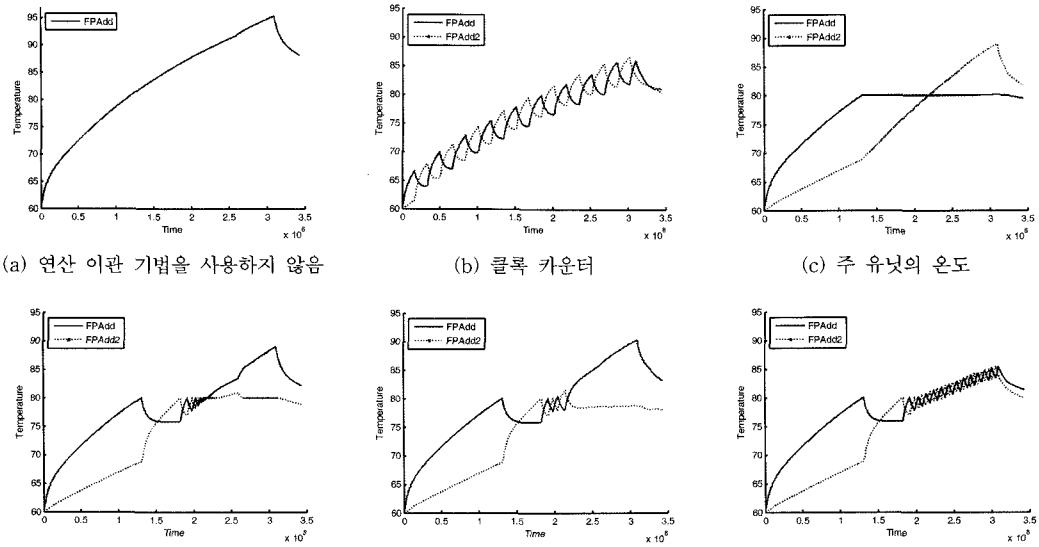
그림 7(c)에서는 주 유닛의 온도가 임계 온도보다 높아지는 경우에는 보조 유닛만을 선택하게 되므로, 이 때부터 보조 유닛의 온도가 급격하게 상승하는 것을 볼 수 있다. 주 유닛이 임계 온도 이하로 냉각되면 다시 동작하다가 바로 임계 온도를 넘으면 보조 유닛을 선택하기 때문에 주 유닛의 온도는 임계 온도로 유지되지만, 보조 유닛의 온도는 계속 상승하는 문제점을 확인할 수 있다.

그림 7(d)에서는 그림 7(c)와는 다르게 보조 유닛의 온도가 임계 온도보다 높은 경우에는 주 유닛을 선택하면서 보조 유닛이 냉각되는 것을 볼 수 있다. 그렇지만 두 유닛 모두 임계 온도보다 뜨거울 경우에는 냉각이 제대로 이루어지지 않는다는 것을 확인할 수 있다. 특이하게도 *lucas* 벤치마크의 경우에는 부동소수점 가산기의 주 유닛(FPAdd)과 인접한 주변 유닛이 보조 유닛(FPAdd2)과 인접한 주변 유닛보다 더 뜨겁기 때문에 주 유닛의 온도가 더 급격하게 증가한다. 또한, 보조 유닛의 발열 온도가 임계 온도보다 아래로 내려가더라도 보조 유닛이 선택되면 바로 발열 온도가 임계 온도를 넘기 때문에 주 유닛이 냉각할 수 있는 시간이 부족하다는 문제점을 확인할 수 있다.

그림 7(e)에서는 냉각 임계 온도를 설정하더라도, 그림 7(d)와 비슷하게 보조 유닛이 주변 유닛의 발열에 따른 열 전달 문제로 냉각 임계 온도만큼 냉각되지 않아서 보조 유닛을 선택하지 못하는 문제점이 발생하는 것을 확인할 수 있다.

그림 7(f)는 주 유닛의 발열 온도가 임계 온도를 넘더라도 두 유닛의 온도가 미리 설정한 온도 차이만큼 발생해야만 보조 유닛을 선택한다. 그림 7(e)와는 다르게 냉각 임계 온도가 동적으로 변하기 때문에 유닛의 선택이 그림 7(b)처럼 일정한 주기로 발열과 냉각을 반복하는 것을 확인할 수 있다.

그림 8에서는 그림 7의 (a)~(f)에 대한 각 유닛의 최



(a) 연산 이관 기법을 사용하지 않음 (b) 클록 카운터 (c) 주 유닛의 온도

(d) 동작중인 유닛의 온도 (e) 유닛의 온도와 냉각 임계 온도 (f) 동작 유닛과 유휴 유닛의 온도차

그림 7 *lucas* 에 대한 부동소수점 가산기(FPAdd)의 온도 변화 양상

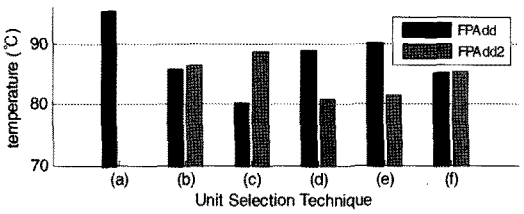


그림 8 주 부동소수점 가산기(FPAdd)와 보조 부동소수점 가산기(FPAdd2)의 최고 온도 비교

표 1 동적 온도 관리 기법 적용시 각 유닛 선택 기법에 대한 성능 비교(긴급 단계 온도: 75°C)

종류	(a)	(b)	(c)	(d)	(e)	(f)
FPAdd들의 최고 온도(°C)	76.92	75.21	75.13	75.04	76.33	74.39
CPI	0.68	0.50	0.50	0.50	0.53	0.50

고 온도를 보여준다. 그림 8에서 부동 소수점 가산기 유닛의 최고 온도를 비교해 보면 전체적으로 연산 이관 기법을 사용하는 것이 최소 5°C, 최고 10°C까지 온도가 낮아지는 것을 확인할 수 있다. 동적 연산 이관 기법에서는 유닛의 온도와 임계 온도와의 온도 차이를 기준으로 선택하는 (f)의 최고 온도가 85.39°C로 가장 낮은 것을 확인할 수 있다. 또한 두 유닛(주 유닛과 보조 유닛)의 최고 온도 차이가 0.07°C로 각 유닛에 대한 냉각 효과가 매우 좋은 것을 확인할 수 있다. 정적인 클록 카운터 기법인 (b) 역시 최고 온도가 86.61°C로 좋은 성능을 보여주고 있지만, 두 유닛의 최고 온도 차이는 0.65°C로 (f) 보다는 떨어지는 것을 확인할 수 있다. 표 1은 긴급 단계 온도가 75°C인 상황에서 동적 온도 제어 기법을 적용할 때 각 유닛 선택 기법에 따른 부동소수점 가산기들의 최고 온도와 CPI(Cycle Per Instruction)로, 동적 연산 이관 기법에서 유닛의 온도와 임계 온도와의 차이를 기준으로 동작 유닛을 선택하는 기법이 발열에 대한 안전성과 성능이 가장 우수하다는 것을 확인할 수 있다.

4. 결 론

연산 이관 기법은 선택할 유닛이 충분하게 냉각되지 않는 경우에는 효율성이 떨어질 수 있다. 유닛은 동작 수행 시 전력을 소비하면서 뜨거워지기도 하지만, 인접한 주변 유닛으로부터 전달되는 열로 인하여 온도가 상승할 수도 있다. 즉, 유닛이 동작을 통해 전력을 소비하지 않더라도 주변 유닛의 영향으로 인해 온도가 상승하는 현상이 발생한다. 따라서 유닛 자체의 발열뿐만 아니라 주변 유닛의 열 전달 문제도 고려하면서 동작할 유닛을 선택할 필요가 있다.

본 논문에서는 연산 이관 기법에서 동작할 유닛을 선택하는 기준에 대한 다양한 기법들을 분석하였고, 모의 실험을 통해 각 기법들을 적용하는 경우 유닛의 온도 변화를 측정하였다. 모의 실험 결과, 연산 이관 기법을 사용하지 않는 경우와 비교하면 동적 연산 이관 기법 중에서 유닛의 온도와 임계 온도와의 차이를 기준으로 동작할 유닛을 선택하는 기법의 최고 온도가 10°C 가까이 낮아지는 것을 확인할 수 있었다. 이 기법을 적용하면 온도 인지 마이크로프로세서에서 발열에 대한 신뢰성이 향상되고, 동적 온도 제어 기법으로 인한 성능 저하를 크게 완화할 수 있을 것이다.

참 고 문 헌

[1] R. Mahajan, "Thermal management of CPUs: A perspective on trends, needs and opportunities," in *Int'l Workshop on THERMal INvestigations of ICs and Systems*, 2002.

[2] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proceedings of the 17th International Symposium on High-Performance Computer Architecture*, pp.171-182, 2001.

[3] H. Seongmoo, 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.

[4] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol.1, pp.94-125, 2004.

[5] J.S. Choi, J.H. Kong, E.Y. Chung and S.W. Chung, "A Dual Integer Register File Structure for Temperature-Aware Microprocessor," *Journal of KIISE: Computer System and Theory*, vol.35, no.12, pp.540-551, 2008 (in Korean)

[6] B. Amelifard and M. Pedram, "Optimal Design of the Power-Delivery Network for Multiple Voltage-Island System-on-Chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.28, no.6, pp.888-900, 2009.

[7] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of the 27th International Symposium on Computer Architecture*, pp.83-94, 2000.

[8] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *Journal of Instruction-Level Parallelism*, vol.7, pp.1-16, 2005.

[9] Hotspot Tool Set, http://lava.cs.virginia.edu/HotSpot/

[10] SPEC(Standard Performance Evaluation Corporation) CPU2000, http://www.spec.org/cpu2000/