(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년12월04일
(11) 등록번호 10-1803284
(24) 등록일자 2017년11월24일

(51) 국제특허분류(Int. Cl.)
G06F 1/32 (2006.01)
(52) CPC특허분류
G06F 1/324 (2013.01)
G06F 1/3225 (2013.01)
(21) 출원번호 10-2016-0151711
(22) 출원일자 2016년11월15일
심사청구일자 2016년11월15일
(56) 선행기술조사문헌
KR1020070061320 A*
(뒷면에 계속) 기술이전 희망 : **기술양도**

(73) 특허권자
인천대학교 산학협력단
인천광역시 연수구 아카데미로 119 (송도동)
(72) 발명자
박문주
서울특별시 관악구 남부순환로230길 111, 602호(봉천동, 코스코수아파트)
장재현
경상북도 구미시 송동로 154, 102동 1302호(도량동, 파크맨션)
(74) 대리인
특허법인충정

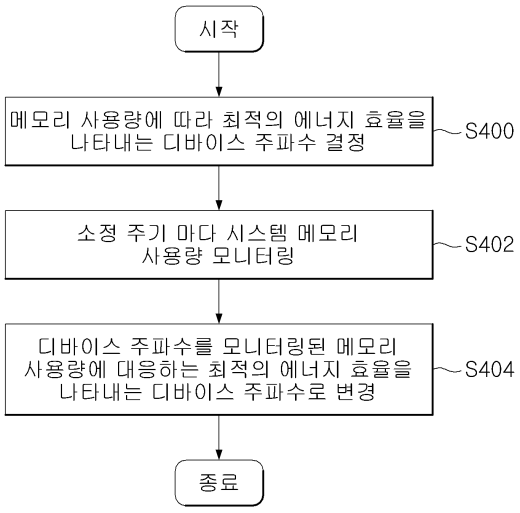
전체 청구항 수 : 총 2 항 심사관 : 손경완

(54) 발명의 명칭 **메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법 및 장치**

(57) 요 약

본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법은, (a) 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 단계; (b) 소정 주기마다 시스템 메모리 사용량을 모니터링하는 단계; 및 (c) 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 단계를 포함하여, 메모리 총 사용량에 따라 디바이스 주파수를 최적의 주파수로 변경함으로써, 임베디드 리눅스 시스템에서의 에너지 소모량을 줄일 수 있어, 시스템 전체의 에너지를 절감할 수 있다

대 표 도 - 도4a



(52) CPC특허분류
G06F 1/3275 (2013.01)
Y02B 60/1217 (2013.01)

(56) 선행기술조사문헌
KR1020130110459 A*
KR101666549 B1
JP5548775 B2
JP2010020387 A
*는 심사관에 의하여 인용된 문헌

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

(a) 제어부가, 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 단계;

(b) 상기 제어부가, 소정 주기마다 시스템 메모리 사용량을 모니터링하는 단계; 및

(c) 상기 제어부가, 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 단계를 포함하고,

상기 최적의 에너지 효율을 나타내는 디바이스 주파수는 특정 작업을 수행하는데 소모되는 소모 에너지와 수행 시간의 곱인 EDP(Energy Delay Product)가 최소인 디바이스 주파수이며,

상기 디바이스 주파수는 메모리 주파수이고,

상기 단계 (a)는,

(a-1) 중앙 처리장치(CPU) 주파수를 고정시킨 상태에서 디바이스 주파수를 변화시키면서 메모리 사용량에 따른 디바이스 주파수별 에너지 효율을 획득하는 단계; 및

(a-2) 상기 메모리 사용량에 따른 디바이스 주파수별 에너지 효율 중에서 메모리 사용량별 최고의 에너지 효율을 나타내는 디바이스 주파수를 상기 메모리 사용량에 따른 최적의 에너지 효율을 나타내는 디바이스 주파수로서 결정하는 단계를 포함하며,

상기 단계 (c)에서,

CPU를 사용하지 않는 아이들(idle) 상태에서는 디바이스 주파수를 최저 디바이스 주파수로 변경하고, 일정 수준 이상의 메모리 사용량이 사용되는 경우 디바이스 주파수를 최고 디바이스 주파수로 변경하는, 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 동작, 소정 주기마다 시스템 메모리 사용량을 모니터링하는 동작, 및 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 동작을 수행하도록 구성된, 제어부; 및

상기 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수가 저장되어 있는 메모리를 포함하고,

상기 최적의 에너지 효율을 나타내는 디바이스 주파수는 특정 작업을 수행하는데 소모되는 소모 에너지와 수행 시간의 곱인 EDP(Energy Delay Product)가 최소인 디바이스 주파수이며,

상기 디바이스 주파수는 메모리 주파수이고,

상기 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 동작은,

중앙 처리장치(CPU) 주파수를 고정시킨 상태에서 디바이스 주파수를 변화시키면서 메모리 사용량에 따른 디바이

스 주파수별 에너지 효율을 획득하는 동작; 및

상기 메모리 사용량에 따른 디바이스 주파수별 에너지 효율 중에서 메모리 사용량별 최고의 에너지 효율을 나타내는 디바이스 주파수를 상기 메모리 사용량에 따른 최적의 에너지 효율을 나타내는 디바이스 주파수로서 결정하는 동작을 포함하며,

상기 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 동작에서, 상기 제어부는,

CPU를 사용하지 않는 아이들(idle) 상태에서는 디바이스 주파수를 최저 디바이스 주파수로 변경하고, 일정 수준 이상의 메모리 사용량이 사용되는 경우 디바이스 주파수를 최고 디바이스 주파수로 변경하는, 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치.

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 최근의 임베디드 시스템들은 크기와 무게는 줄이는 방면 수행하는 작업들은 기존의 임베디드 시스템에서의 작업보다 복합적이고 복잡하게 변화하는 등 이전의 역할보다 더 많은 일을 하도록 요구되어지기 때문에 처리 능력이 뛰어난 중앙 처리장치(CPU)를 사용하게 되었다.

[0003] 하지만 이런 고사양의 중앙 처리장치는 전력을 더 많이 소모할 수밖에 없고 이는 곧 임베디드 장치의 전체 에너지 사용량의 상승이라는 결과를 가져왔다. 이러한 단점은 휴대용 임베디드 시스템에서 더 치명적이었는데 휴대용 임베디드 시스템은 대부분 시스템의 전원을 배터리로부터 가져오기 때문이다. 따라서 시스템을 한번 충전한 뒤에 사용할 수 있는 시스템의 사용시간은 임베디드 시스템에서 중요한 문제로 떠오르고 있다. 시스템의 고정적인 배터리의 용량에서 사용시간을 더 늘리기 위해 중앙 처리장치의 전압(Voltage)과 주파수(Frequency)를 최적화하는 것이 임베디드 시스템에서 높은 전력효율을 달성하는 하나의 방법이고 이것을 동적 전압 및 주파수 스케일링(DVFS: Dynamic Voltage & Frequency Scaling) 기술이라고 한다.

[0004] 리눅스 시스템에서 DVFS 기술은 거버너(Governor)를 통하여 구현이 되어있고 거버너(Governor)중에서 전력효율을 위해서 인터랙티브(Interactive)와 온-디맨드 거버너(on-demand Governor)를 가장 많이 사용한다. 하지만 임베디드 시스템에서 동작하는 작업의 특성에 따라 메모리 접근이 빈번하지 않은 작업에도 메모리 주파수가 불필요하게 높게 동작하여 불필요한 전력이 소모될 경우가 있다.

[0005] Devfreq이란 CPU처럼 디바이스 또한 여러 클록(Clock)을 가지고 있는 환경에서 디바이스가 거버너 정책(governor policy)에 의해 작동하는 주파수를 조절할 수 있도록 해주는 포괄적(generic) DVFS 프레임워크(framework)이다. 디바이스마다 여러 작동 주파수 목록을 가지고 있으며 주어진 상황에 따라 거버너(Governor)가 적절하게 주파수와 전압을 조절하여 전력효율을 달성한다. CPU만을 위한 거버너(Governor)가 따로 존재를 하고 CPU외에 메모리, 버스(Bus), I/O 장치들을 위한 DVFS가 Devfreq을 통하여 구현된다.

[0006] 메모리 주파수 조절을 위해 devfreq가 사용되며 디바이스 주파수 조절이 곧 메모리 주파수 변경을 뜻한다. CPU가 메모리의 정보를 읽어오는 과정은 다음과 같다.

[0007] 첫째, CPU가 원하는 주소를 보낸다. 둘째, 메모리의 알맞은 뱅크(Bank)를 액티브(Active) 상태로 둔다. 셋째, 뱅크(Bank)의 로우(Row) 데이터 전체를 센스 증폭기(Sense Amplifier)로 옮긴다. 넷째, 알맞은 컬럼(Column)에 해당하는 데이터를 출력한다. 다섯째, 센스 증폭기(sense amplifier)의 데이터를 원래 자리로 복귀시킨다.

[0008] 이와 같은 과정을 통해 메모리에 저장된 정보를 CPU가 읽기 위해서는 몇 가지 타이밍 정보가 필요한데 액티브(Active)된 뱅크(Bank)에서 로우(Row) 데이터를 읽는데 걸리는 시간인 RAS to CAS 지연 시간(Latency), 컬럼(Column) 정보를 읽는데 걸리는 시간인 CAS 지연 시간(Latency), 센스 증폭기(Sense Amplifier)로 복귀시키는데 걸리는 시간인 RAS 프리차지 지연 시간(Precharge Delay), 로우(Row) 주소를 받는데 걸리는 시간인 로우 어드레스 스트로브(Row Address Stobe)이다. 메모리와 관련된 이전의 연구들은 대부분 위와 같은 메모리 타이밍 정보를 바탕으로 메모리에 관한 저전력 연구를 해왔다.

[0009] 비특허문헌 [5]의 경우에는 SDRAM의 개선된 파워 모델(power model)을 제안하고 마이크론 모델(Micron model)의 에너지 소비와 비교해 최대 28%의 에너지 절감을 달성하였다. 위와 같은 메모리 타이밍에 관한 연구가 있는 반면 비특허문헌 [2]와 같이 동적 전압/주파수 스케일링(Dynamic Voltage/Frequency Scaling)을 통해 서버 시스템의 메모리의 소비전력을 절감한 연구도 있다.

[0010] 일반적인 컴퓨터 시스템에서 CPU가 메모리로부터 데이터를 받기 위해서는 시스템 버스(system bus)와 I/O 브릿지(bridge), 메모리 버스(Bus)를 거쳐야 한다. devfreq는 디바이스 주파수를 조절하게 되고 메모리 버스(bus)의 동작 클록(Clock)을 조정하여 메모리의 전력을 절감하게 된다.

[0011] 비특허문헌 [7]은 메모리의 채널과 DRAM 디바이스의 동적인 주파수 스케일링(frequency scaling)으로부터 저전력(low-power) 모드를 제안하여 낮은 하드웨어의 비용대비 높은 에너지 절감을 보였다.

[0012] 하지만 비특허문헌 [6]과 비특허문헌 [7]은 실제 구현된 것이 아닌 시뮬레이션을 통한 결과를 보였다. 운영체제 수준에서 메모리의 전력 상태(power state)를 제어한 경우도 있는데 비특허문헌 [8]에서는 스케줄러에 의해 메모리를 제어하도록 제안하였다.

[0013] 반면 비특허문헌 [9]는 DRAM의 랭크(Rank)에 저장된 데이터의 접근 빈도에 따라 핫(Hot), 웜(Warm), 콜드(Cold) 페이지로 구분하고 해당 페이지들이 같은 랭크(Rank)로 위치하도록 옮겨 비어있는 랭크(Rank)는 전력을 차단하고 웜(Warm)과 콜드(Clod) 페이지가 위치한 랭크(Rank)는 파워-다운(Power-Down) 모드보다 더 적은 전력을 소모하는 셀프-리프레시(Self-Refresh) 모드를 사용하도록 하여 비특허문헌 [9]에서는 전력소비를 30-40% 감소시켰다.

선행기술문헌

비특허문헌

[0014] (비특허문헌 0001) [1] M. Weiser, B. Welch, A.J. Demer, and S. Shenker, "Scheduling for Reduced CPU Energy," Proc. of the 1st USENIX Conf. on Operating Systems Design and Implementation, pp. 13-23, Nov. 1994. (in USA)

(비특허문헌 0002) [2] D. Brodowski. CPUFreq Governors. [Online]. Available:https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt, (downloaded 2016. March, 21)

(비특허문헌 0003) [3] A. Vorontsov. (2012, Feb 8) Interactive Governor, [Online]. Available: https://lkml.org/lkml/2012/2/7/483, (downloaded 2016. March, 21)

(비특허문헌 0004) [4] V. Pallipadi and A. Starikovskiy, "The Ondemand Governor," Proc. Ottawa Linux Symp., pp. 223-238, Jul. 2006. (in Canada)

(비특허문헌 0005) [5] B. Akesson, K. Goossens, "Improve Power Modeling of DDR SDRAMs," Proc. of the Digital System Design (DSD) 2011, pp. 99-108, Sept. 2011. (in Finland)

(비특허문헌 0006) [6] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory Power Management via Dynamic Voltage/Frequency Scaling," ICAC'11 Proc. of the 8th ACM international conference on Autonomic computing, 2011. pp. 31-40, 2011. (in USA)

(비특허문헌 0007) [7] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch and R. Bianchini, "MemScale:

Active low-power modes for main memory," Proc. of the 16th International Conference on Architectural Support for Programming Language and Operating Systems, pp. 225-238, Mar. 2011. (in USA)

(비특허문헌 0008) [8] V, Delauz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Scheduler-based DRAM energy management," Proc. of the 39th annual Design Automation Conference, pp. 697-702, 2002. (in USA)

(비특허문헌 0009) [9] H. Cho and D. Shin, "Flash-Aware DRAM Power Management By Considering Flash Memory," Proc. of the Korea IEIE Conference, pp. 567-568, Nov. 2009. (in Korea)

(비특허문헌 0010) [10] H. Kim and M. Park, "A Real-Time Analysis Tool for Applications on Multicore Embedded Linux Systems," Proc. of Korea Information Science Society Congress, pp. 80-82. Jun. 2014. (in Korea)

(비특허문헌 0011) [11] N. Nethercote and J. Seward, "Valgrind: A Program Supervision Framework," Electronic Notes in Theoretical Computer Science, Vol. 89, Issue 2, pp. 44-66, Oct. 2003. (in Netherlands)

(비특허문헌 0012) [12] F. Timme, "How To Benchmark Your System (CPU, File IO, MySQL) With sysbench," [Online] Available: https://www.howtoforge.com/how-to-benchmark-your-system-cpu-file-io-mysql-with-sysbench (downloaded 2016, March, 21)

(비특허문헌 0013) [13] R. M. Hollander and P. V. Bolotoff, "RAMspeed: A cache and memory benchmarking tool," [Online] Available: http://alasir.com/software/ramspeed/ (downloaded 2016, March, 21)

(비특허문헌 0014) [14] ab-Apache HTTP server benchmarking tool [Online] Available:https://httpd.apache.org/docs/2.4/en/programs/ab.html(downloaded 2016, March, 21)

(비특허문헌 0015) [15] W. Norcott, "IOzone Filesystem Benchmark" [Online] Available:http://www.iozone.org/ (downloaded 2016, March, 22)

(비특허문헌 0016) [16] John D, McCalpin, Stream Benchmark [Online], Available:https://www.cs.virginia.edu/stream/ref. html (downloaded 2016, March, 22)

(비특허문헌 0017) [17] R. Gonzalez and M. Horowitz, "Energy Dissipation In General Purpose Microprocessors," IEEE Journal of Solid-State Circuits, Vol. 31, Issue 9, pp. 1277-1284, Sep. 1996.

(비특허문헌 0018) [18] H. Kim, J. Jang, M. Park, "Energy Efficiency Analysison an Embedded Linux System per CPU Frequency," Proc. of Korea Information Science Society Congress, pp. 80-82, Jun. 2014. (in Korea)

(비특허문헌 0019) [19] A. Raj, "CPU hotplug Support in Linux(tm) Kernel,"[Online] Available:https://www.kernel.org/doc/Documentation/cpu-hotplug.txt(downloaded 2016, Mar. 22)

발명의 내용

해결하려는 과제

[0015] 본 발명이 해결하고자 하는 과제는 디바이스의 주파수를 조절해주는 주파수 변경 프레임워크를 통하여 메모리 총 사용량에 따라 최고의 에너지 효율을 갖는 메모리 주파수를 찾고, 메모리 총 사용량에 따라 디바이스 주파수를 최적의 주파수로 변경함으로써, 임베디드 리눅스 시스템에서의 에너지 소모량을 줄일 수 있는, 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법을 제공하는 것이다.

[0016] 본 발명이 해결하고자 하는 다른 과제는 디바이스의 주파수를 조절해주는 주파수 변경 프레임워크를 통하여 메모리 총 사용량에 따라 최고의 에너지 효율을 갖는 메모리 주파수를 찾고, 메모리 총 사용량에 따라 디바이스 주파수를 최적의 주파수로 변경함으로써, 임베디드 리눅스 시스템에서의 에너지 소모량을 줄일 수 있는, 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치를 제공하는 것이다.

과제의 해결 수단

[0017] 상기 과제를 해결하기 위한 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법은,

[0018] (a) 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 단계;

[0019] (b) 소정 주기마다 시스템 메모리 사용량을 모니터링하는 단계; 및

[0020] (c) 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 단계를 포함한다.

[0021] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에 있어서, 상기 단계 (a)는,

[0022] (a-1) 중앙 처리장치(CPU) 주파수를 고정시킨 상태에서 디바이스 주파수를 변화시키면서 메모리 사용량에 따른 디바이스 주파수별 에너지 효율을 획득하는 단계; 및

[0023] (a-2) 상기 메모리 사용량에 따른 디바이스 주파수별 에너지 효율 중에서 메모리 사용량별 최고의 에너지 효율을 나타내는 디바이스 주파수를 상기 메모리 사용량에 따른 최적의 에너지 효율을 나타내는 디바이스 주파수로서 결정하는 단계를 포함할 수 있다.

[0024] 또한, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에 있어서, 상기 최적의 효율적인 디바이스 주파수는 실질적으로 상기 메모리 사용량의 제곱에 비례할 수 있다.

[0025] 또한, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에 있어서, 상기 단계 (c)에서, CPU를 사용하지 않는 아이들(idle) 상태에서는 디바이스 주파수를 최저 디바이스 주파수로 변경하고, 일정 수준 이상의 메모리 사용량이 사용되는 경우 디바이스 주파수를 디바이스 주파수를 최고 디바이스 주파수로 변경할 수 있다.

[0026] 상기 다른 과제를 해결하기 위한 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치는,

[0027] 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 동작, 소정 주기마다 시스템 메모리 사용량을 모니터링하는 동작, 및 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 동작을 수행하도록 구성된, 제어부; 및

[0028] 상기 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수가 저장되어 있는 메모리를 포함한다.

[0029] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치에 있어서, 상기 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 동작은,

[0030] 중앙 처리장치(CPU) 주파수를 고정시킨 상태에서 디바이스 주파수를 변화시키면서 메모리 사용량에 따른 디바이스 주파수별 에너지 효율을 획득하는 동작; 및

[0031] 상기 메모리 사용량에 따른 디바이스 주파수별 에너지 효율 중에서 메모리 사용량별 최고의 에너지 효율을 나타내는 디바이스 주파수를 상기 메모리 사용량에 따른 최적의 에너지 효율을 나타내는 디바이스 주파수로서 결정하는 동작을 포함할 수 있다.

[0032] 또한, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치에 있어서, 상기 최적의 효율적인 디바이스 주파수는 실질적으로 상기 메모리 사용량의 제곱에 비례할 수 있다.

[0033] 또한, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치에 있어서, 상기 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 동작에서, 상기 제어부는, CPU를 사용하지 않는 아이들(idle) 상태에서는 디바이스 주파수를 최저 디바이스 주파수로 변경하고, 일정 수준 이상의 메모리 사용량이 사용되는 경우 디바이스 주파수를 최고 디바이스 주파수로 변경할 수 있다.

발명의 효과

[0034] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법 및 장치에 의하면, 디바이스의 주파수를 조절해주는 주파수 변경 프레임워크를 통하여 최고의 에너지 효율을 갖는 메모리 주파수를 찾고, 메모리 총 사용량에 따라 디바이스 주파수를 최적의 주파수로 변경함으로써, 임베디드 리눅스 시스템에서의 에너지 소모량을 줄일 수 있어, 시스템 전체의 에너지를 절감할 수 있다.

도면의 간단한 설명

[0035] 도 1은 아이들(idle) 상태에서 각 디바이스 주파수별 전력 소모량을 도시한 도면.

도 2는 각 벤치마크별 메모리 사용량과 소모전력을 도시한 도면.

도 3은 메모리 사용량과 L2 캐시미스율의 관계 및 추세선을 도시한 도면.

도 4a는 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법의 흐름도.

도 4b는 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치의 블록도.

도 5는 디바이스 주파수에 따른 벤치마크 에너지 효율을 도시한 도면.

도 6은 메모리 사용량과 그에 따른 효율적인 디바이스 주파수를 도시한 도면.

도 7은 퍼포먼스 CPU 거버너에서 디바이스 거버너별 각 벤치마크 에너지 소모량을 도시한 도면.

도 8은 인터랙티브 CPU 거버너에서 디바이스 거버너별 각 벤치마크 에너지 소모량을 도시한 도면.

도 9는 온-디맨드 CPU 거버너에서 디바이스 거버너별 각 벤치마크 에너지 소모량을 도시한 도면.

발명을 실시하기 위한 구체적인 내용

[0036] 본 발명의 목적, 특정한 장점들 및 신규한 특징들은 첨부된 도면들과 연관되어지는 이하의 상세한 설명과 바람직한 실시예들로부터 더욱 명백해질 것이다.

[0037] 이에 앞서 본 명세서 및 청구범위에 사용된 용어나 단어는 통상적이고 사전적인 의미로 해석되어서는 아니되며, 발명자가 그 자신의 발명을 가장 최선의 방법으로 설명하기 위해 용어의 개념을 적절하게 정의할 수 있는 원칙에 입각하여 본 발명의 기술적 사상에 부합되는 의미와 개념으로 해석되어야 한다.

[0038] 본 명세서에서 각 도면의 구성요소들에 참조번호를 부가함에 있어서, 동일한 구성 요소들에 한해서는 비록 다른 도면상에 표시되더라도 가능한 한 동일한 번호를 가지도록 하고 있음에 유의하여야 한다.

[0039] 또한, "제1", "제2", "일면", "타면" 등의 용어는, 하나의 구성요소를 다른 구성요소로부터 구별하기 위해 사용되는 것으로, 구성요소가 상기 용어들에 의해 제한되는 것은 아니다.

[0040] 이하, 본 발명을 설명함에 있어, 본 발명의 요지를 불필요하게 흐릴 수 있는 관련된 공지 기술에 대한 상세한 설명은 생략한다.

[0041] 이하, 첨부된 도면을 참조하여 본 발명의 바람직한 실시형태를 상세히 설명하기로 한다.

[0042] IoT 기기가 대중들에게 보급화됨에 따라서 임베디드 시스템의 성능뿐만 아니라 소모하는 에너지에 관한 문제가 중요하게 대두되고 있다. 이는 특히 휴대용 임베디드 기기에서 더 중요한데, 휴대용 임베디드 기기들은 각 디바이스들이 가지고 있는 배터리를 통하여 전원을 얻기 때문에 얼마나 효율적으로 잘 사용하는지가 곧 시스템의 사용시간으로 귀결되기 때문이다.

[0043] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서는, 임베디드 리눅스 시스템에서의 에너지 소모량을 줄이기 위하여 디바이스의 주파수를 조절해주는 주파수 변경 프레임워크를 통하여 최고의 에너지 효율을 갖는 메모리 주파수를 찾고, 메모리 총 사용량을 통한 새로운 디바이스 주파수 변경 기법을 제안한다.

[0044] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서는 메모리 주파수 조절을 위해 devfreq가 사용되며 디바이스 주파수 조절이 곧 메모리 주파수 변경을 뜻한다.

[0045] 실험을 통해 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에

서 제안하는 정책은 기존의 정책과 비교하여 최악의 경우에도 나쁜 에너지 효율을 보이지 않으며 최대 18%의 에너지 감소율을 보였다.

[0047] 시험 및 분석

[0048] 동기부여(MOTIVATION)

[0049] 메모리의 주파수/전압 스케일링(frequency/voltage scaling)을 통하여 에너지 절감 효과를 얻기 위해서는 devfreq이 지원하는 주파수 목록들(frequency list) 간의 소비 전력차이가 있어야한다.

[0050] 높은 주파수(frequency)와 낮은 주파수(frequency)간의 전력차이가 없다면 DVFS의 의미가 없기 때문이다. 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서 타깃으로 삼은 임베디드 보드는 하드커널(Hardkernel)사의 ODROID-XU3이며 하드웨어 주요 사양은 다음 표 1과 같다.

표 1

[0051]

Processor	Samsung Exynos5422 CortexTM-A15 2.0Ghz quad core and CortexTM-A7 quad core CPUs
Memory	2Gbyte LPDDR3 RAM at 933MHz
3D Accelerator	Mali-T628 MP6
Storage	SanDisk Extreme 16GB
Display	HDMI, DisplayPort
USB Host	USB 3.0 × 1, USB 2.0 × 4
Ethernet	10/100Mbps Ethernet

[0052] 전력측정은 메모리가 소비하는 전력뿐만 아니라 디바이스 전체에 들어가는 소비전력을 측정하기 위하여 하드커널(Hardkernel)사의 스마트 파워(Smart Power)를 사용하였다. 스마트 파워(Smart Power)는 연결된 디바이스에서 사용하는 전압(Voltage), 전류(Current), 전력(Power)을 측정할 수 있게 해주는 파워 서플라이(Power supply)이다.

[0053] 타깃이 지원하는 디바이스 주파수(device frequency)는 가장 낮은 주파수로부터 138㎒, 165㎒, 206㎒, 275㎒, 415㎒, 543㎒, 633㎒, 728㎒, 825㎒ 순이며 시스템 부팅이 완료된 후 아이들(idle) 상태일 때 각 주파수별로 소비하는 전력(W)은 도 1과 같다.

[0054] 최고 주파수부터 최저 주파수의 차이가 약 0.22W이며 시스템의 상태에 따라 주파수를 조절한다면 충분한 전력 절감 효과를 얻을 수 있다.

[0056] **모니터링 도구**

[0057] 컴퓨터 시스템에서 CPU가 필요한 데이터가 있으면 가장 먼저 캐시 메모리(Cache Memory)를 조사하게 된다. 캐시 메모리에 원하는 데이터가 없으면 캐시 미스(Cache miss)가 발생하여 그때 메모리에 접근해 데이터를 캐시 메모리로 가져와 수행하게 되는데 이때 디바이스 주파수가 낮다면 느린 속도로 데이터를 가져가게 될 것이고 반대의 상황에서는 빠르게 원하는 데이터에 접근하게 된다.

[0058] 이전에 잠깐 언급하였듯 CPU-집중적인 태스크는 메모리 접근이 거의 일어나지 않기에 메모리 주파수를 낮추어주면 에너지 절감 효과를 기대할 수 있는 것이다. 따라서 CPU가 얼마나 빈번하게 메모리에 접근하는지 아는 것이 필요하고 이는 곧 캐시 적중률(Cache hit ratio)에 따라서 구할 수 있다.

[0059] 하지만 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법의 타깃 보드인 ODROID-XU3에서 PMU(Performance Monitoring Unit)이 라스트 레벨 캐시(LLC: Last-Level Cache) 적중률(Hit Ratio)을 구할 수 있도록 지원하지 않기에 LLC 적중률을 직접적으로 하드웨어로부터 구할 수 없다.

[0060] 캐시 미스와 연관된 어떠한 지표를 사용하여 간접적으로 캐시 미스를 구하여 모니터링하도록 하여야 하는데 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서는 그 지표를 총 메모리 사용량으로 한다.

[0061] 리눅스는 유저레벨에서 커널의 데이터를 얻을 수 있도록 proc 가상파일시스템을 제공하지만 이러한 proc 파일시스템을 이용한 모니터링은 커널레벨과 유저레벨 사이의 데이터 교환에 따른 오버헤드가 심각하고 proc 파일 시스템을 통해 얻은 데이터를 근거로 거버너(governor)를 수행하면 오버헤드 때문에 최신의 시스템 상태가 아닌 일정시간이 지난 이전의 데이터를 근거로 판단하기 때문에 정확하지 않을 가능성이 있다.

[0062] 따라서 직접 <linux/sched.h>에 정의된 task_struct 구조체에 접근하여 오버헤드를 거의 없게 하고 모든 테스크의 메모리 사용량을 측정한다. 초기 태스크부터 시작하여 태스크 리스트를 순차적으로 이동하며 사용하는 크기를 얻어 더하게 되고 태스크의 끝까지 이동하는 루프를 통하여 모든 태스크를 순회한다.

[0064] **메모리 사용량과 캐시 적중률**

[0065] 타깃 보드에서는 캐시 적중률을 구할 수 있도록 PMU가 지원되지 않기에 총 메모리 사용량으로 기준을 잡는다고 하였는데 그것이 유효한지 검증하는 과정이 필요하다. 그 과정은 valgrind라는 리눅스 기반의 오픈소스 DBI(Dynamic Binary Instrumentation) 도구를 통하여 테스트하였다.

[0066] DBI는 실행 정보를 수집하기 위해 프로그램에 임의의 코드를 삽입하는 기술로서 기존의 실행시간보다 길어진다는 단점이 있지만 여러 자세한 디버깅 정보를 제공한다. valgrind는 이러한 DBI도구 중 하나이며 메모리관리 문제를 위한 Memcheck detects, CPU 캐시 프로파일러(cache profiler)인 Cachegrind, 프로그램의 힙(Heap) 영역 관리를 위해 시간별로 힙 영역의 사용량을 그래프로 보여주는 Massif 등 여러 가지 디버깅 툴(tool)을 제공한다.

[0067] 그 중 valgrind의 Cachegrind tool을 이용하여 메모리 총 사용량과 캐시 적중률의 관계를 파악하고자 한다. 도 2는 표 2에서 사용한 벤치마크 툴별로 수행시간동안 메모리 총 사용량과 전력을 나타내는데 각 워크로드별로도 메모리 사용량의 변화에 따라 전력 사용량이 유사하게 변하는 것을 보여준다. 사용한 워크로드는 CPU, 메모리, 디스크 등에서 sysbench, ramsmp, apache benchmark, iozone, stream benchmark이며 각 벤치마크가 수행하는 동안의 메모리 총 사용량(KB)과 캐시 미스율(cache miss ratio)은 표 2와 같다.

표 2

[0068]

	Memory usage	Cache Miss Ratio
sysbench	430664KB	0.03%
apache bench	436568KB	0.01%
iozone	448352KB	0.8%
ramsmp	544240KB	8%
stream	666848KB	13.7%

[0069] 표 2를 보면 서로 다른 다섯 가지 벤치마크 툴(tool)별로 수행 중 총 메모리 사용량과 그에 따른 cachegrind의 캐시 미스율의 결과가 나타나있다. 그 결과로 총 메모리 사용량이 높아질수록 캐시 미스율이 따라서 증가함을 알 수 있다. 표 2의 메모리 사용량과 캐시 미스율의 관계를 나타낸 그래프가 도 3이다. 도 3의 실선은 실제 측정된 값들을 연결한 선이고 점선은 메모리 사용량과 L2 캐시 미스율의 추세선이다.

[0070] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서는 추세선과 실제 측정값이 얼마나 일치하는지 알아보기 위하여 결정계수(R squared) 값을 이용한다.

[0071] y_i 가 실제 데이터들이고 f_i 가 추세선의 예측 값이라고 하면, $SS_{tot} = \sum_i (y_i - \bar{y})^2$, $SS_{res} = \sum_i (y_i - f_i)^2$ 라고 할 수 있고 R squared는 $1 - SS_{res}/SS_{tot}$ 으로 구할 수 있으며 1에 가까울수록 실제 값과 예측 값이 일치한다고 볼 수 있다.

[0072] 도 3의 R squared 값은 0.9792로 높은 일치율을 보인다. 따라서 메모리 사용량과 캐시 미스율의 관계를 추세선

이 대표하며 y를 캐시 미스율, x를 메모리 사용량이라 할 때 수학식 1과 같은 관계를 보인다.

수학식 1

[0073] $$y = 0.0128x^2 - 0.0412x + 0.0283$$

[0074] 메모리 사용량이 늘어날수록 캐시 미스율이 수학식 1에 따라 제곱에 비례한다고 할 수 있고 결과적으로 메모리 접근 횟수가 늘어나게 되어 총 전력 소모량이 증가한다고 볼 수 있다.

[0075] 도 4a는 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법의 흐름도이다.

[0076] 도 4a를 참조하면, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법은, 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 단계(단계 S400), 소정 주기마다 시스템 메모리 사용량을 모니터링하는 단계(단계 S402), 및 디바이스 주파수를 메모리 사용량에 근거한 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 단계(단계 S404)를 포함한다.

[0077] 상기 단계 S400은, CPU 주파수를 고정시킨 상태에서 디바이스 주파수를 변화시키면서 메모리 사용량에 따른 디바이스 주파수별 에너지 효율을 획득하는 단계, 및 메모리 사용량별 최고의 에너지 효율을 나타내는 디바이스 주파수를 상기 메모리 사용량에 따른 최적의 에너지 효율을 나타내는 디바이스 주파수로서 결정하는 단계를 포함한다.

[0078] 한편, 도 4b는 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치의 블록도를 도시한 것이다.

[0079] 도 4b를 참조하면, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 장치는, 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 동작, 소정 주기마다 시스템 메모리 사용량을 모니터링하는 동작, 및 디바이스 주파수를 상기 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 동작을 수행하도록 구성된, 제어부(410), 및 상기 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수가 저장되어 있는 메모리(412)를 포함한다.

[0080] 하기에, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에 대해 상세히 설명하기로 한다.

[0082] **에너지 효율적인 적정 디바이스 주파수**

[0083] 각 메모리 사용량을 기준으로 가장 효율적인 디바이스 주파수를 찾기 위해 CPU 주파수를 고정시킨 상태에서 메모리 주파수를 변화시켜가며 각 주파수별 에너지 효율을 표 2에 나열된 벤치마크별로 측정해보았다.

[0084] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서는 에너지 효율을 나타내기 위하여 소모 에너지와 수행시간을 곱한 EDP(Energy Delay Product)를 사용한다.

[0085] 디바이스 주파수를 낮추면 전력(W)이 낮아지지만 수행시간이 길어져 에너지 효율이 떨어지게 되므로 적정수준의 디바이스 주파수를 찾는 것이 필요하다.

[0086] 실험에서는 CPU 주파수를 조절하지 않고 고정하였기 때문에 CPU 주파수와 그에 따른 캐시 접근속도는 고정적이고 메모리 사용량과 디바이스 주파수만이 가변적이다.

[0087] 타깃 보드는 CPU의 코어들을 0번부터 7번까지 각각 온(on), 오프(off)시킬 수 있는데 도 5의 결과는 CPU를 0번 코어 하나만 온라인(online)으로 둔 상태에서 표 2에 나열된 벤치마크 툴들을 타깃 보드가 지원하는 디바이스 주파수마다 수행하면서 소모한 에너지 효율을 나타낸다. 도 5를 보면 메모리 총 사용량에 따라서 최고의 에너지 효율을 보이는 디바이스 주파수가 각 벤치마크의 특성에 따라 다르다는 것을 볼 수 있다.

[0089] **기존 거버너(governor)와 제안 거버너(governor)**

[0090] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법의 타깃인 ODROID-XU3 보드는 리눅스 커널 3.10.96이며 최신 커널은 4.5버전까지 있다.

[0091] 4.5에서 지원하는 devfreq 거너버(governor)는 총 4가지로 퍼포먼스(Performance)와 파워세이브(Powersave)는 각각 최대 주파수와 최저 주파수로 설정하는 거버너(governor)이고 유저스페이스(Userspace)는 사용자의 입력에 따라 가변적이다. CPU 거버너(governor)와 비교해 비교적 거버너(governor) 종류가 적다.

[0092] CPU의 온-디맨드 거버너(on-demand governor)와 비슷한 거버너(governor)가 심플_온디맨드(simple_ondemand)인데 해당 거버너(governor)는 CPU의 사이클(Cycle) 수를 통한 busy/total 카운터 값에 의해 디바이스의 주파수가 정해진다. 그렇기 때문에 메모리의 접근이 거의 없는 작업에서도 CPU의 로드(load)가 높다면 필요 없이 전력을 낭비해가면서 메모리의 주파수를 높게 설정하게 되고 반대인 상황에서는 메모리의 접근이 자주 일어나지만 주파수가 낮아 처리속도가 늦어져 총 소비 전력이 커지는 결과를 낳을 수 있다.

[0093] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서는 이러한 문제를 해결하고자 메모리 사용량을 근거로 한 기법을 제안하며 시스템 메모리 사용량 모니터링을 위한 과정을 통해 매 일정 주기마다 시스템의 상태를 읽어 가장 효율적인 디바이스 주파수로 변경하여 주는 새로운 에너지 효율적인 기법을 제안한다.

[0094] 효율적인 디바이스 주파수는 다음 도 6을 통해서 정할 수 있다. 도 6은 도 5를 근거로 메모리 사용량과 해당 메모리 사용량에서 가장 에너지 효율이 좋은 디바이스 주파수를 나타낸다.

[0095] 도 3에서 캐시 미스율은 메모리 사용량의 제곱에 비례함을 보였고 에너지 소모량은 캐시 미스율과 비례하므로 효율적인 디바이스 주파수도 메모리 사용량의 제곱에 비례하는 형태를 보인다. CPU를 사용하지 않는 아이들(idle) 상태에서는 도 1에서 보인 바와 같이 가장 낮은 디바이스 주파수로 유지하여주고 반대로 일정 수준 메모리 사용량을 넘어서게 되면 디바이스가 지원하는 가장 높은 주파수로 설정해주게 된다.

[0096] 두 가지 경우를 제외한 상황에서는 도 6을 참고하여 타깃 보드가 지원하는 가장 가까운 디바이스 주파수로 변경하여 준다.

[0098] **실험 및 성능 평가**

[0099] **실험 환경**

[0100] 실험은 표 1에 나타난 ODROID-XU3 보드를 사용하였으며 각 벤치마크는 표 2와 같다. 각 벤치마크는 표 3과 같은 특성을 가지며 CPU 거버너(governor)는 퍼포먼스(performance), 인터랙티브(interactive), 온-디맨드(on-demand) 3종류이고 디바이스 거버너(governor)는 퍼포먼스(performance), 심플_디맨드(simple_ondemand), 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 에너지 디바이스 주파수 변경 방법에서 제안하는 정책을 반영한 거버너(governor)인 뉴_거버너(new_governor) 3가지, 총 9개의 CPU와 디바이스 거버너(governor) 조합으로 수행하였다.

[0101] CPU 거버너(governor)를 다르게 하여 수행하는 이유는 각 CPU 거버너(governor)마다 특성이 다르기에 모든 경우에서 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서 제안하는 디바이스 거버너(governor)의 효율을 보이기 위함이며 각 벤치마크가 수행하기 전에 이전에 수행하였던 벤치마크가 남겨놓은 캐시 데이터를 삭제하고 벤치마크가 소모하는 전력을 측정하였다. 스마트 파워(Smart Power)를 통해 얻은 전력 값은 단위를 줄(J)로 통일하였다.

[0103] **실험 결과 및 분석**

[0104] 실험 환경에서 벤치마크들을 수행한 결과가 다음 도 7, 도 8 및 도 9에 나타나있다. 도 7은 CPU 거버너(governor)가 퍼포먼스(performance), 도 8은 인터랙티브(interactive), 도 9는 온-디맨드(on-demand)에서 수행한 결과이다. 도 7에서 각 벤치마크는 디바이스 거버너(governor)마다 소모하는 에너지가 다르고, 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서 제안하는 뉴_거버너(new_governor)는 퍼포먼스(performance)와 심플_온디맨드 거버너(simple_ondemand governor) 중 에너지 소모량이 적은 것과 비교하였을 때 sysbench에서 심플-온디맨드(simple_ondemand) 대비 약 5%, apache benchmark와 stream에서는 같은 에너지 효율을 보였으며, ramsmp에서 약 7%, iozone에서 11%의 에너지 소모량 감소를 보였다.

[0105] 다음 도 8과 도 9에서는 도 7에서 감소한 에너지 소모량이 퍼포먼스(performance) CPU 거버너(governor)일 경우

에만 해당되는지 알아보기 위하여 수행하였으며 도 8의 인터랙티브(interactive) CPU 거버너(governor)에서는 기존 대비 최저 5%, 최대 18%의 에너지 감소량을 보였다. 온-디맨드(On-demand) CPU 거버너(governor)에서는 기존 거버너(governor) 대비 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서 제안한 디바이스 거버너(governor)가 에너지 소모량이 같거나 최대 9%의 에너지 감소량을 보였다.

[0106] 실험에서는 또한 CPU의 거버너(governor) 효율에 관해서도 알 수 있다. CPU의 사용량에 비례하여 동작하는 클록을 조절하여 에너지 절감 효과를 보기 위한 거버너(governor)가 도 8과 도 9의 인터랙티브(interactive)와 온-디맨드(on-demand) CPU 거버너(governor)인데 도 7의 성능보다 에너지 효율 측면에서 좋지 않은 결과를 보인다.

[0107] 그 이유는 타겟 보드의 동적 소비전력 대비 정적 소비 전력이 너무 높아 DVFS를 통해 얻는 에너지 절감 효과보다 일정한 주기마다 시스템의 상태를 읽어 판단하는 오버헤드가 크기 때문인데 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서와 같이 정적 소비전력이 큰 타깃 보드에서는 CPU 핫-플러깅(Hot-Plugging)이나 추가적인 논리 회로를 통해 특정 회로에 클록을 공급하지 않는 클록 게이팅(Clock Gating) 기법을 통한 전력 절감기술이 더 유효할 수 있다고 보고 있다.

[0109] **향후 연구 및 결론**

[0110] 본 발명의 일 실시예에 의한 메모리 사용량에 근거한 에너지 효율적 디바이스 주파수 변경 방법에서 제안하는 거버너(governor)는 메모리 사용량과 캐시 미스율의 연관성을 실험을 통해 밝히고 실험을 통해 퍼포먼스(performance) CPU 거버너(governor) 뿐만 아니라 가장 많이 사용하는 인터랙티브(interactive)와 온-디맨드(on-demand) CPU 거버너(governor)에서도 CPU, 메모리, 디스크(Disk), 네트워크 벤치마크(Network benchmark) 도구 5가지를 통해 제안하는 정책의 에너지 감소를 비교하여 결과적으로 메모리 사용량이 많을수록 더 높은 디바이스 주파수로 변경해 주어야 총 에너지 사용량이 줄어들고 낮은 메모리 사용량에서는 디바이스 주파수를 최적의 에너지 효율을 가지도록 낮추어 시스템 전체의 에너지 절감 효과를 보았다.

[0111] 이상 본 발명을 구체적인 실시예를 통하여 상세하게 설명하였으나, 이는 본 발명을 구체적으로 설명하기 위한 것으로, 본 발명은 이에 한정되지 않으며, 본 발명의 기술적 사상 내에서 당 분야의 통상의 지식을 가진 자에 의해 그 변형이나 개량이 가능함은 명백하다고 할 것이다.

[0112] 본 발명의 단순한 변형 내지 변경은 모두 본 발명의 영역에 속하는 것으로, 본 발명의 구체적인 보호 범위는 첨부된 청구범위에 의하여 명확해질 것이다.

부호의 설명

[0113] S400 : 메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수를 결정하는 단계

S402 : 소정 주기마다 시스템 메모리 사용량을 모니터링하는 단계

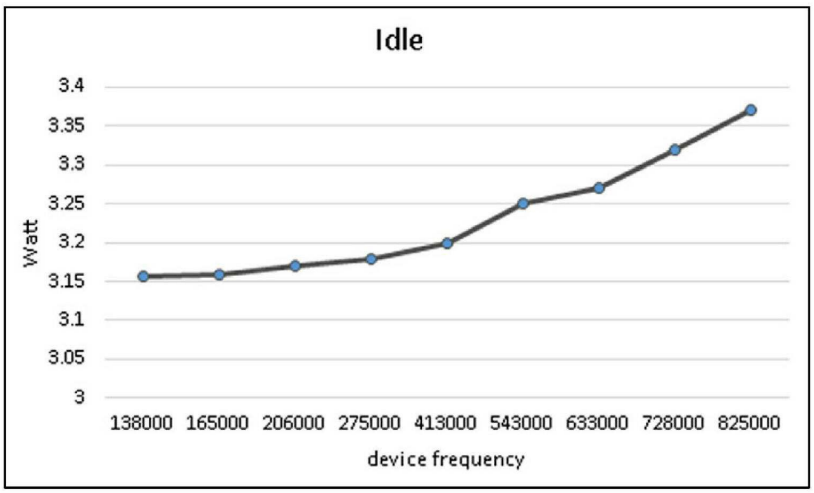
S404 : 디바이스 주파수를 모니터링 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경하는 단계

410 : 제어부

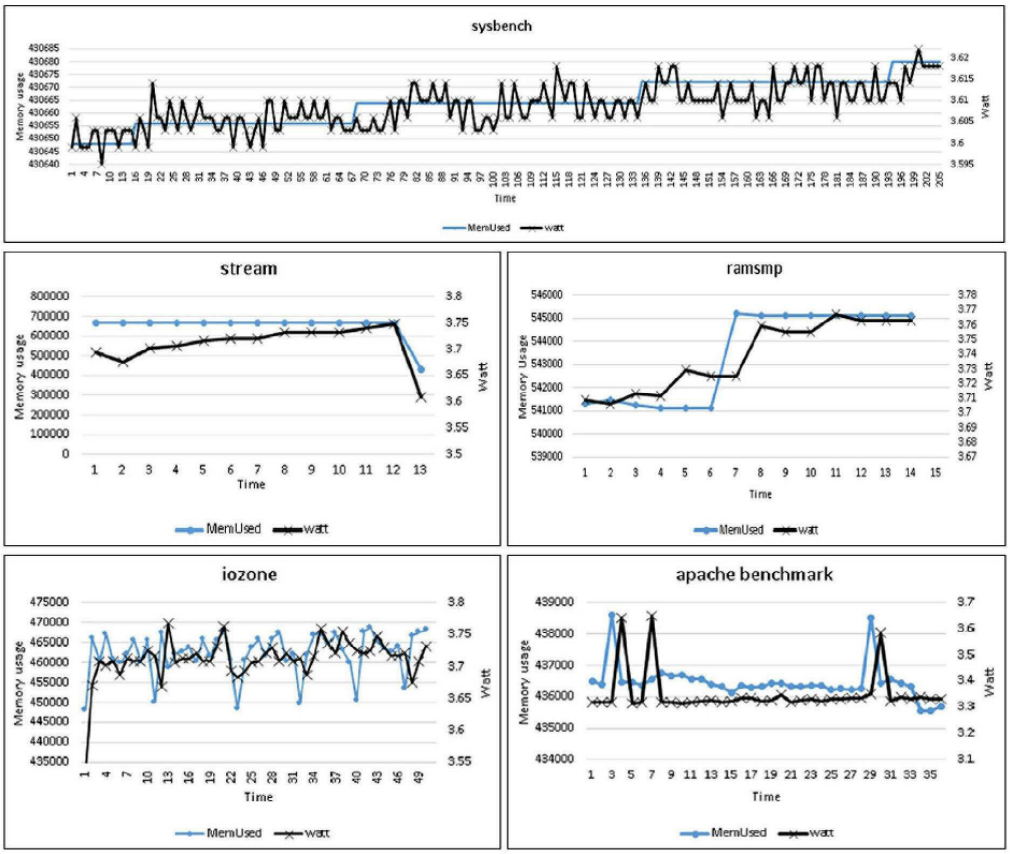
412 : 메모리

도면

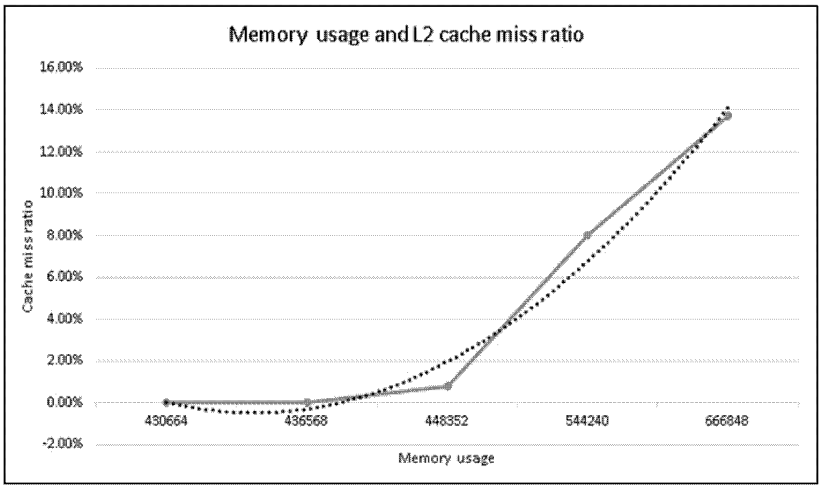
도면1


Idle
Watt
3.4
3.35
3.3
3.25
3.2
3.15
3.1
3.05
3
138000 165000 206000 275000 413000 543000 633000 728000 825000
device frequency

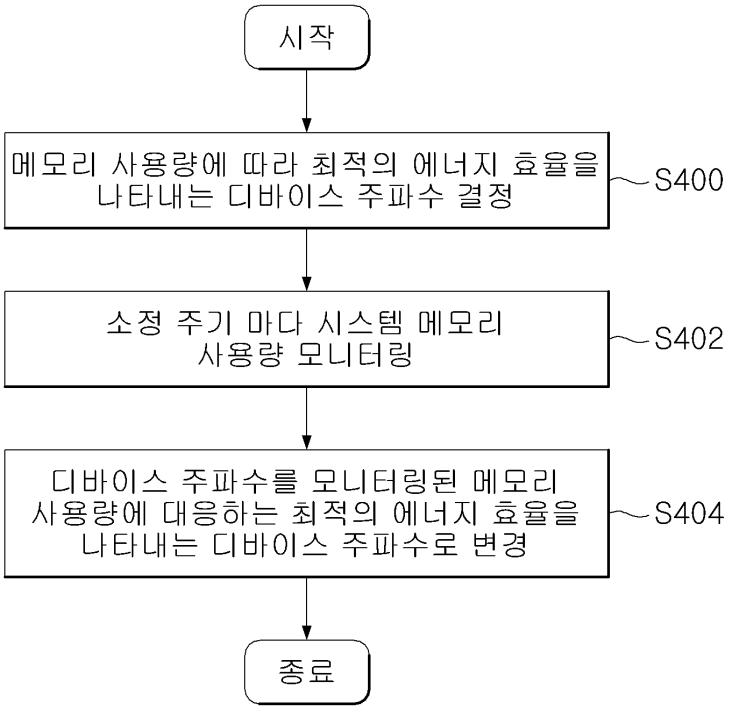
도면2


sysbench
stream
ramsmp
iozone
apache benchmark
Memory usage
Watt
Time
MemUsed
watt

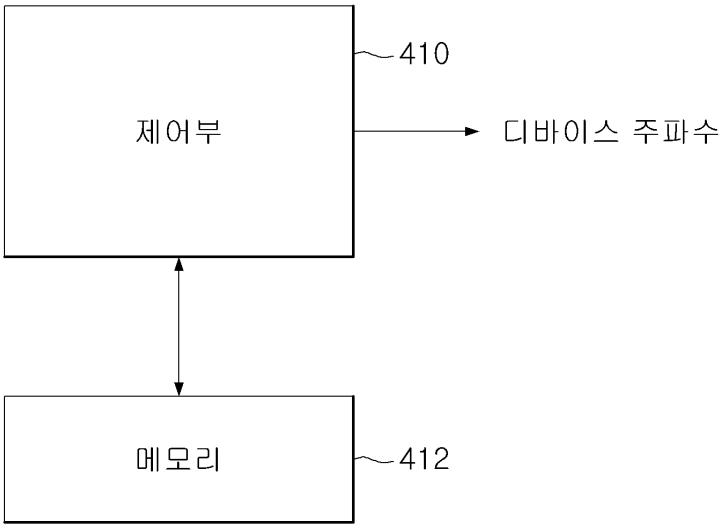
도면3


Memory usage and L2 cache miss ratio
16.00%
14.00%
12.00%
10.00%
8.00%
6.00%
4.00%
2.00%
0.00%
-2.00%
Cache miss ratio
430664
436568
448352
544240
666848
Memory usage

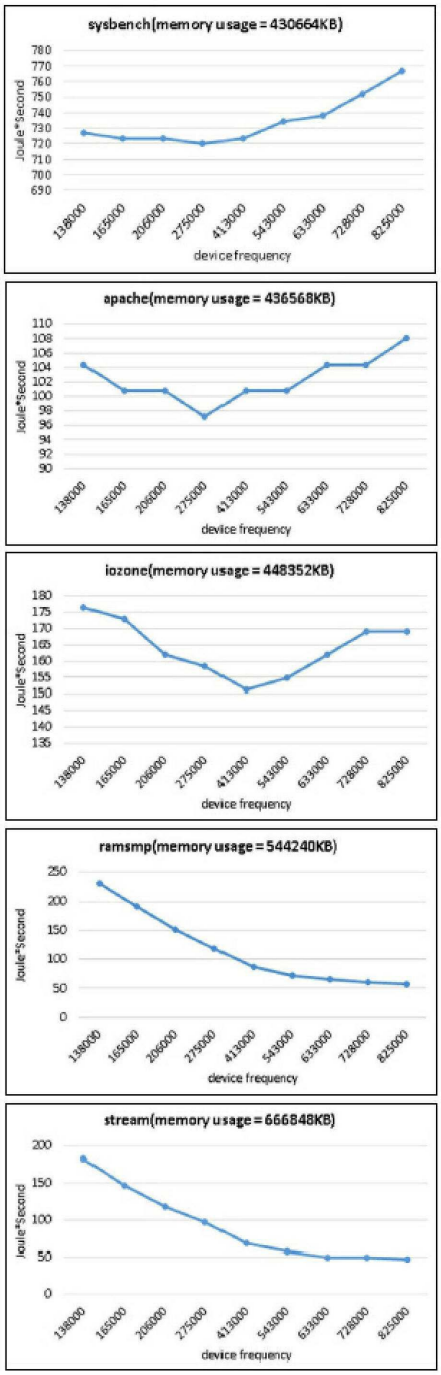
도면4a


시작
메모리 사용량에 따라 최적의 에너지 효율을 나타내는 디바이스 주파수 결정
S400
소정 주기 마다 시스템 메모리 사용량 모니터링
S402
디바이스 주파수를 모니터링된 메모리 사용량에 대응하는 최적의 에너지 효율을 나타내는 디바이스 주파수로 변경
S404
종료

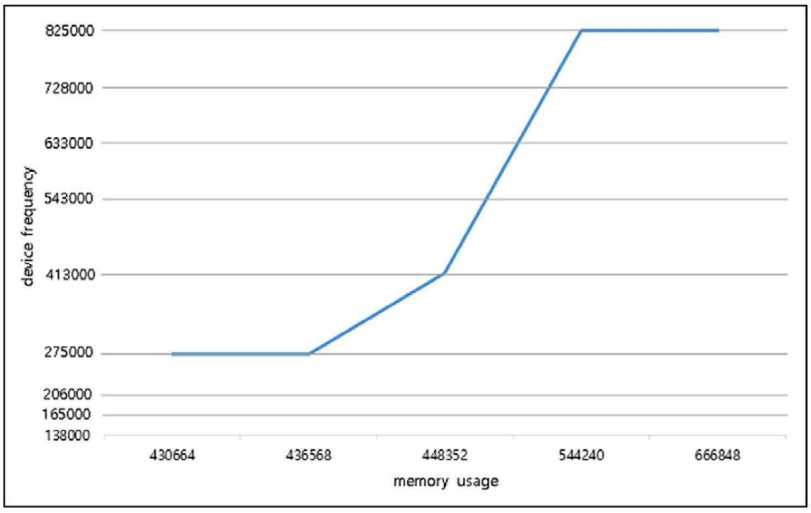
도면4b



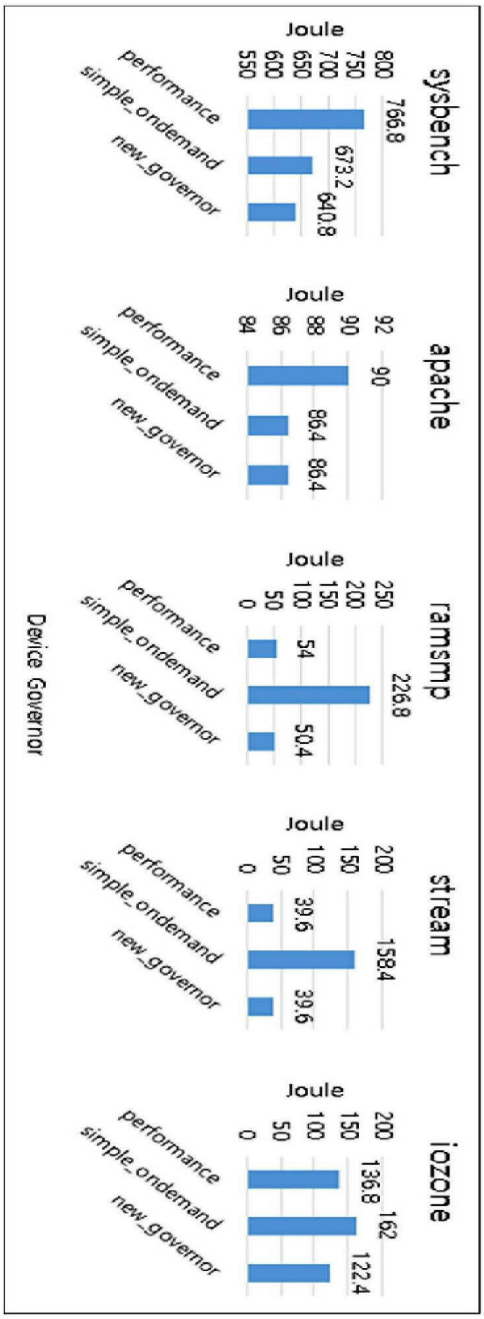
도면5


sysbench(memory usage = 430664KB)
Joule*Second
780
770
760
750
740
730
720
710
700
690
138000
165000
206000
275000
413000
543000
633000
728000
825000
device frequency
apache(memory usage = 436568KB)
Joule*Second
110
108
106
104
102
100
98
96
94
92
90
138000
165000
206000
275000
413000
543000
633000
728000
825000
device frequency
iozone(memory usage = 448352KB)
Joule*Second
180
175
170
165
160
155
150
145
140
135
138000
165000
206000
275000
413000
543000
633000
728000
825000
device frequency
ramsmp(memory usage = 544240KB)
Joule*Second
250
200
150
100
50
0
138000
165000
206000
275000
413000
543000
633000
728000
825000
device frequency
stream(memory usage = 666848KB)
Joule*Second
200
150
100
50
0
138000
165000
206000
275000
413000
543000
633000
728000
825000
device frequency

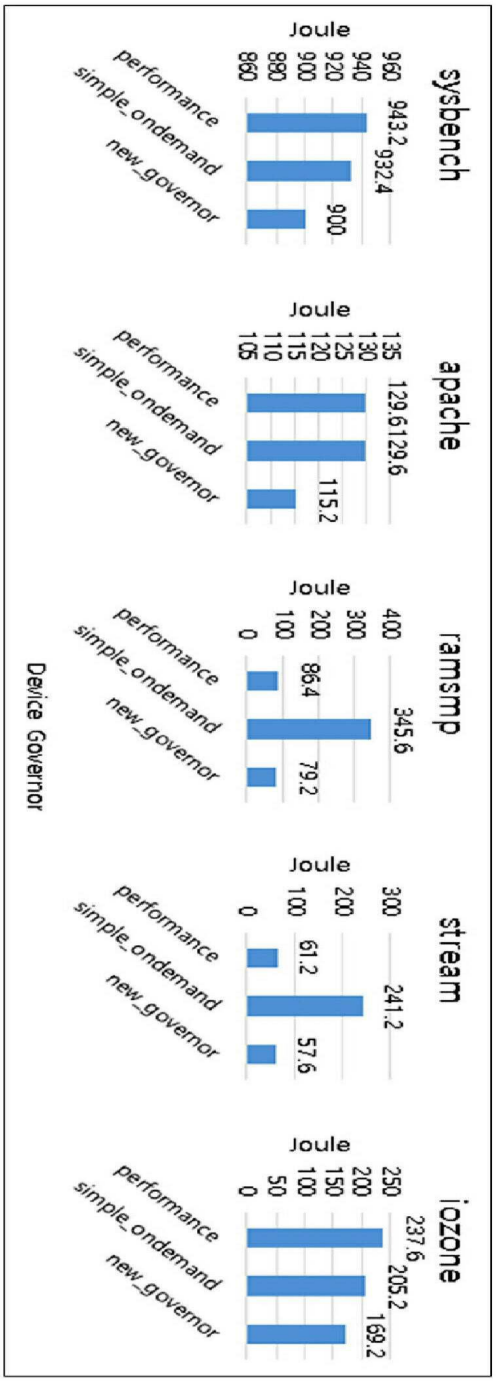
도면6


825000
728000
633000
543000
413000
275000
206000
165000
138000
device frequency
430664
436568
448352
544240
666848
memory usage

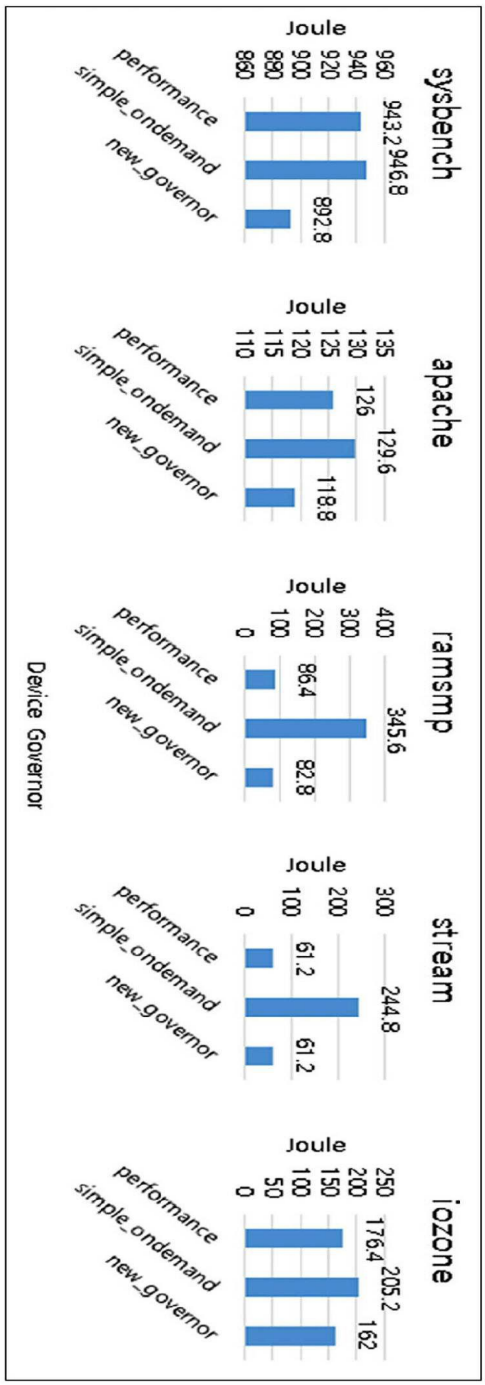
도면7


sysbench
Joule
800
750
700
650
600
550
766.8
673.2
640.8
performance
simple_ondemand
new_governor
apache
Joule
92
90
88
86
94
90
86.4
86.4
performance
simple_ondemand
new_governor
ramsmp
Joule
250
200
150
100
50
0
54
226.8
50.4
performance
simple_ondemand
new_governor
stream
Joule
200
150
100
50
0
39.6
158.4
39.6
performance
simple_ondemand
new_governor
iozone
Joule
200
150
100
50
0
136.8
162
122.4
performance
simple_ondemand
new_governor
Device Governor

도면8


sysbench
Joule
960
940
920
900
880
860
943.2
932.4
900
performance
simple_ondemand
new_governor
apache
Joule
135
130
125
120
115
110
105
129.6
129.6
115.2
performance
simple_ondemand
new_governor
ramsmp
Joule
400
300
200
100
0
345.6
86.4
79.2
performance
simple_ondemand
new_governor
stream
Joule
300
200
100
0
241.2
61.2
57.6
performance
simple_ondemand
new_governor
iozone
Joule
250
200
150
100
50
0
237.6
205.2
169.2
performance
simple_ondemand
new_governor
Device Governor

도면9


sysbench
Joule
960
940
920
900
880
860
943.2
946.8
892.8
performance
simple_ondemand
new_governor
apache
Joule
135
130
125
120
115
110
126
129.6
118.8
performance
simple_ondemand
new_governor
ramsmp
Joule
400
300
200
100
0
86.4
345.6
82.8
performance
simple_ondemand
new_governor
Device Governor
stream
Joule
300
200
100
0
61.2
244.8
61.2
performance
simple_ondemand
new_governor
iozone
Joule
250
200
150
100
50
0
176.4
205.2
162
performance
simple_ondemand
new_governor