동적 프리퀀시 스케일링을 사용한 프로세서의 인터럽트 처리와 I/O 시스템 성능 향상 기법

유시환° 유혁
고려대학교 컴퓨터학과
{shyoo°, hxy}@os.korea.ac.kr

Interrupt Processing in Dynamic Frequency Scaling Processor System

See-hwan Yoo°, Chuck Yoo
Department of Computer Science and Engineering, Korea University

요 약

동적 전력 관리 기법을 활용한 프로세서의 등장은 고성능 임베디드 장치들의 저전력 설계에 있어서 큰 영향을 주고 있다. 특히, XSCALE과 같은 고성능 프로세서의 소비전력은 동작 클럭의 속도와 비례하여 빠르게 증가하고 있으며, 이를 극복하기 위한 다양한 기법이 제시되었다. 동적 전력 관리 기법은 크게 1) 동적 전압 관리 기법과 동적 프리퀀시 관리 기법으로 구분된다.

동적 프리퀀시 관리 기법을 사용한 프로세서는 필요에 따라 프로세서의 동작 클럭 속도를 변경한다. 이는 전체적인 프로세서 성능의 저하를 수반하게 된다. 특히, 주변 장치들의 전력 관리가 동시에 이루어지지 않을 경우에는 시스템의 전체적인 성능에 큰 영향을 끼치게 된다. I/O 장치의 인터럽트는 CPU의 현재 실행을 잠시 멈추고, 인터럽트 처리를 우선적으로 수행하도록 한다. 따라서 CPU가 처리할 수 있는 양보다 많은 인터럽트 발생은 인터럽트 처리 이후에 실제 응용 프로그램들이 동작할 시간을 줄이게 되어 CPU는 살아있으나, 인터럽트 이외의 실제 프로세스 실행을 진행할 수 없는 라이브록(livelock) 현상이 발생한다.

동적 프리퀀시 스케일링을 사용하는 경우, 프로세서의 동작 속도 저하로 인한 livelock 현상이 발생할 수 있으며 이를 막기 위하여, 인터럽트 처리를 제한하는 기법을 제시한다.

1. 서 론

최근 들어 다양한 고사양 임베디드 장치들이 등장함으로 인해, 이러한 임베디드 장치에 사용되는 프로세서 역시 높은 동작 클럭과 높은 소비전력을 가지고 있다. 인텔의 XSCALE의 경우, 동작 클럭의 프리퀀시는 400Mhz로 임베디드 장치이면서도 고속의 데이터 처리가 가능하도록 구성되어있다. 즉, 임베디드 장치의 고사양화로 인하여 임베디드 시스템이 점차 범용 시스템화하고 있으며, 따라서 범용 시스템에서 사용될만한 프로세서의 사용은 휴대용 장치에 사용되는 임베디드 프로세서 환경에서 또다른 문제를 제기하게 되었다. 즉, 최근의 고사양 임베디드 프로세서에서의 소비전력과 온도문제는 최근들어 큰 문제로 등장하고 있으며 이를 해결하기 위한 다양한 접근 방식들이 등장하고 있다[1-7].

소비전력을 낮추기 위한 프로세서 구조 연구는 지속적으로 진행되어왔으며, 크게는 활성/비활성 모드를 분리하여 실행에 따른 소비전력을 줄이는 방법이 널리 사용되고 있다. 최근 들어서는 동적 전압관리와 동적 프리퀀시 조절 기법이 주로 연구되고 있다. 프로세서의 전압과 동작 클럭의 프리퀀시는 소비전력과 직접적으로 관련이 있다. 즉, 사용되는 소비전력은 수식 1에서 제시하는 바와 같이 동작 프리퀀시와 전압의 제곱, 커패시턴스와 비례한다.

$$P \propto C \cdot V^2 \cdot f$$

수식 1. 소비전력과 전압, 동작 프리퀀시의 관계식

하지만, 이 때 실제 소비한 에너지량인 소비전력양은 수식 2와 같이 시간과 소비전력의 곱이며, 동작 프리퀀시에 따라 실행 시간은 반비례하므로, 결국 수식 3과 같이 소비에너지의 양은 커패시턴스와 전압의 제곱에 비례하는 것을 알 수 있다.

$$E = P \cdot t$$

수식 2. 에너지량과 소비전력, 시간의 관계식

$$E \propto C \cdot V^2$$

수식 3. 에너지량과 전압의 관계식

하지만 동작 프리퀀시를 낮춤으로써 전압을 낮출 수 있기 때문에, 일반적으로 에너지와 동작 프리퀀시 사이에는 수식 4와 같은 관계식이 성립한다.

$$E \propto \left(V_{th} + \frac{f}{2k} + \sqrt{\frac{V_{th} f}{k} + \left(\frac{f}{2k}\right)^2} \right)^2$$

수식 4. 에너지와 동작 프리퀀시 사이의 관계식

결과적으로, 동적 프리퀀시 스케일링을 통한 소비전력 감소기법은 프로세서의 전력 감소에 큰 도움을 준다. 최근까지의 연구들은 언제 동작 프리퀀시를 조절할 것인가에 초점이 맞추어져 있었다. 대개, 프리퀀시 조절은 프로세서의 활용률(Utilization)과 관련이 있다. 프리퀀시를 낮추어도 무방한 활용률이 낮은 상황에서 소비전력을 낮추는 것이 일반적인 방법이다. 하지만, CPU의 실행과는 무관한 주변 장치들의 인터럽트 발생으로 인한 활용률

상승은 막을 방법이 없다.

2. 본 론

최근 임베디드 시스템에서 사용되는 구조는 I/O 버스를 분리하여 사용하지 않고, SoC 형태로 프로세싱 코어와 주변 장치들을 모두 하나의 칩 안에 넣고, 모든 장치들이 버스를 공유하여 사용하는 방식이다. 이러한 방식은 각각의 주변 장치들을 프로세서와 직접 연결하여 주므로 응답시간 측면에서는 범용 시스템과 비교하여 볼 때 유리하다고 할 수 있으나, 전체적인 성능과 주변장치의 증가에 따른 확장성 측면에서는 단점이 있다.

특히, 프로세서의 동작 클럭 속도에 따라 시스템이 공유하는 버스 전체의 동작 클럭 속도가 변화하므로, 이러한 환경에서의 동적 프리퀀시 스케일링 기법은 전체 시스템의 성능을 크게 떨어뜨린다고 볼 수 있다.

버스의 동작 속도가 프로세싱 코어의 속도와 비례하여 떨어진다고 가정하여 볼 때, 랜덤하게 발생하는 버스 처리 요청에 대하여, 리틀의 법칙이 성립함을 알 수 있다. 즉, 프로세서의 동작 속도의 감소에 따른 처리 지연 시간 증가는 데이터 요청속도의 비례적 감소를 의미한다.

$$N=\lambda \cdot T$$

수식 5. Little's Theorem: N은 평균 버퍼의 길이, λ는 데이터 요청을 나타내는 포아송 랜덤 프로세스, T는 평균 데이터 처리 시간의 관계식

따라서 버스를 통해 들어오는 데이터의 요청 속도가 크게 제한받게 되는데, 이는 주변장치들에서 랜덤하게 발생하는 인터럽트에 의하여 큰 영향을 받게 된다. 주변장치들의 경우 버스의 속도와 독립적으로 동작하는 경우가 많으며, 특별한 전력 관리 기법을 사용하지 않는 이상, 인터럽트의 발생은 기존과 동일한 방식으로 이루어지게 된다.

인터럽트가 고속으로 발생하는 네트워크 인터페이스의 경우, 인터럽트 발생의 속도가 프리퀀시 스케일링과 무관하게 동일하다면 인터럽트 발생으로 인한 livelock 현상이 발생할 수 있다. 인터럽트에 의한 livelock이란, 인터럽트 발생의 빈도가 프로세서의 처리 속도보다 높아 실제 사용자 프로세스들은 프로세서의 사용시간을 할당받지 못하고, 실행하지 못하는 현상이다[8,9]. 이러한 문제는 인터럽트 처리의 우선순위가 가장 높은 값을 가지고 있기 때문이다.

최근 들어 등장하고 있는 UWB와 같은 네트워크 인터페이스들은 근거리 고속 데이터 통신을 목적으로 하고 있고, 대개의 장치들은 인터럽트 기반의 처리를 원칙으로 하고 있기 때문에, 인터럽트 발생으로 인한 livelock 현상의 발생 가능성은 더욱 높다.

Livelock 현상을 막기 위한 기법으로는 인터럽트의 발생 빈도를 제한하는 방법이 있다. 인터럽트의 발생 빈도를 제한하기 위해서는 1) 인터럽트를 모아서 처리하거나, 2) 인터럽트의 우선순위를 낮추어 처리하는 방법이 있다. 첫 번째 방법은 고속 네트워크 환경에서 널리 사용되는 방법으로 Interrupt Coalescence 라고 한다. 이 방법은 네트워크 카드가 들어오는 데이터 패킷들을 일정 시간 동안 버퍼링하고 있다가, 정해진 타이머에 의해 한꺼번에 트리거 하는 방법이다. 이러한 방법은 고속 네트워크의 등장으로 시스템 속도가 네트워크 속도보다 느린 현상이 발생하여, receive livelock이 리포트 되고, 이 때 해결책으로 제시된 기법이다. Interrupt coalescence를 사용하면, 발생하는 인터럽트의 빈도를 조절함으로써, 네트워크 처리량을 크게 향상 시킬 수 있었다. 하지만, 이 방법을 사용하기 위해서는 네트워크 카드와 같은 하드웨어 장치에 추가적인 버퍼를 유지하고, 타이머를 관리해야 하는 오버헤드가 있다.

두 번째 방법은 인터럽트의 유실을 강제적으로 발생시키는 방법이다. 즉, 인터럽트의 최대 발생 빈도를 정해두고, 그 이상의 인터럽트에 대해서는 처리하지 않도록 하는 방법이다. 이를 위해서는 타이머를 이용한 카운터를 활용하는 방법이 있다. 즉, 일정한 간격으로 발생하는 타이머마다 허용 가능한 최대 인터럽트 개수를 지정해 두고, 인터럽트 발생시마다 값을 깎아 내려감으로써, 최대 인터럽트 발생 횟수를 제한할 수 있다. 인터럽트 발생 횟수가 최대값을 넘으면, 인터럽트를 마스킹해두어 다음 타이머 틱까지 발생하지 않도록 제한할 수 있다.

인터럽트의 발생은 버스의 동작과 동기화되어있으므로, 버스의 동작 속도와 관련이 있다. 또한, 버스의 동작 속도는 프로세서의 동작 속도에 대하여 일정한 비율을 유지하므로, 프로세서의 클럭 프리퀀시 스케일링을 사용할 때, 주어진 버스 동작 속도에 대하여 최대 인터럽트 발생 빈도를 구하는 작업이 필요하다.

인터럽트의 발생이 포아송 과정 PP(λ) 을 따른다고 하면, 확률변수 X(t)는 t 시간 동안 발생한 인터럽트의 횟수로 가정할 수 있다. Chebyshev 부등식에 의하면,

$$P[|X(t)-m|\geq a]\leq \frac{\sigma^2}{a^2}$$

수식 6. Chebyshev 부등식: m은 평균, σ^2은 분산

, $m=\lambda t$, $\sigma^2=\lambda t$이다. 이 때 $a=\sqrt{\lambda}$로 놓으면, $P[|X(t)-\lambda t|\geq\sqrt{\lambda}]\leq\frac{\lambda t}{\lambda}=t$ 이다. 여기에서, 측정 시간 t를 운영체제에서 측정하는 타이머의 1틱 값으로 제한하면, ($t=tick$), 다음 수식 7 을 얻는다.

$$P[\lambda\cdot tick-\sqrt{\lambda}\leq X(tick)\leq\lambda\cdot tick+\sqrt{\lambda}]\geq tick$$

수식 7. 인터럽트 발생의 확률 과정

위 식에 따르면, 인터럽트가 매 틱 안에 평균 $\lambda\cdot tick$ 개 발생할 때, $\sqrt{\lambda}$개 이상이나 이하의 인터럽트가 발생할 확률은 tick보다 작거나 같다.

프로세서의 프리퀀시 스케일링에 의하여 감소된 버스의 동작 프리퀀시를 $Freq_{\min-bus}$라하면, 인터럽트 처리기에 의해 실행되는 시간 $T_{interrupt}$을 가정할 때, 최대 $Freq_{\min-bus}/T_{interrupt}$개의 인터럽트를 처리할 수 있다. tick 값은 대개 10ms=0.01s 이므로, 1%의 손실율을 유지하기 위해서는 tick 시간 동안 $\lambda\cdot tick+\sqrt{\lambda}$ 개의 인터럽트를 처리해 주어야 한다. 이 값은 최대 $Freq_{\min-bus}/T_{interrupt}$가 되므로, 수식 8을 얻는다.

$$\frac{0.01\lambda+\sqrt{\lambda}}{0.01}\leq\frac{Freq_{\min-bus}}{T_{interrupt}}$$

수식 8. 평균 인터럽트 발생률과 최대값의 관계식

위 식을 계산하면, 다음의 수식 9 같은 λ의 범위를 얻는다.

$$Min(0,\frac{-1-\sqrt{1+0.0004\cdot\frac{Freq}{T_{interrupt}}}}{0.02})\leq\sqrt{\lambda}\leq\frac{-1+\sqrt{1+0.0004\cdot\frac{Freq}{T_{interrupt}}}}{0.02}$$

수식 9. 평균 인터럽트 발생 범위

위 식으로부터, 평균 인터럽트의 발생 빈도는 최대 인터럽트 발생 가능 빈도에 근사하여 비례하도록 유지해야 함을 알 수 있다.

3. 결 론 및 추후 연구

최근 들어 등장하고 있는 고사양의 휴대용 임베디드 시스템들은 다양한 동적 전력 관리기법을 사용하여 소비전력을 효율적으로 사용하도록 한다. 동적 전력 관리기법 중에서 널리 사용되는 전압 조절 및 동작 프리퀀시 조절 기법은 프로세서의 동작 프리퀀시를 조절하여 낮은 전압에서 동작할 수 있도록 함으로써 소비전력를 줄일 수 있다. 하지만, 프로세서의 동작 프리퀀시는 임베디드 시스템의 버스 속도와 직접적으로 연관되어있으므로, 버스에 연결된 장치들 역시 이러한 점을 고려하여 인터럽트의 생성 빈도를 조절할 수 있는 기법이 필요하다. 이 논문에서는 인터럽트의 빈도가 프리퀀시 스케일링 비율과 유사하게 감소하여야 함을 보였다.

본 연구의 내용을 뒷받침 해 줄 수 있는 시뮬레이션과 실험 그리고, 보다 구체적인 시스템의 파라미터 분석이 추가적으로 필요하다.

Reference

1. Jacob Rubin Lorch, "Operating Systems Technologies for Reducing Processor Energy Consumption", http://research.microsoft.com/~lorch/papers/thesis.pdf
2. Bishop Brock and Karthic Rajamani, "Dynamic Power Management for Embedded systems", IEEE SOCC 2003, 17-20 September, Portland, OR, USA
3. Ana Luiza de Almeida Pereira Zuquim, Antônio Alfredo F. Loureiro, Claudionor N. Coelho. Jr.,Marcos Augusto M. Vieira, Luiz Filipe Menezes Vieira, Alex Borges Vieira, Antônio O. Fernandes, Diógenes Cecílio da Silva Junior, José M. da Mata, José Augusto Nacif, Hervaldo Sampaio Carvalho, "Efficient Power Management in Real-Time Embedded Systems", ETFA2003, 16-19 September 2003, Lisbon, Portugal
4. Konduri, G. , J. Goodman, A. Chandrakasan, "Energy Efficient Software Through Dynamic Voltage Scheduling," IEEE ISCAS, 358-361, May 1999.
5. Vahdat, A., Lebeck, A., and Ellis, C. S. 2000. Every joule is precious: the case for revisiting operating system design for energy efficiency. In Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the Pc: New Challenges For the Operating System (Kolding, Denmark, September 17 - 20, 2000). EW 9. ACM Press, New York, NY, 31-36.
6. Acquaviva, A., Benini, L., and Riccó, B. 2001. Energy characterization of embedded real-time operating systems. SIGARCH Comput. Archit. News vol. 29, No. 5 (Dec. 2001), pp. 13-18.
7. P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for lowpower embedded operating systems. In proc. of Symp. Operating Systems Principles (SOSP2001), pp. 89-102, 2001.
8. Mogul, J. C. and Ramakrishnan, K. K. 1997. Eliminating receive livelock in an interrupt-driven kernel. ACM Trans. Comput. Syst. vol. 15, No. 3 (Aug. 1997), pp. 217-252.
9. Dovrolis, C., Thayer, B., and Ramanathan, P. 2001. HIP: hybrid interrupt-polling for the network interface. SIGOPS Oper. Syst. Rev. 35, 4 (Oct. 2001), 50-60.