고정우선순위 버스 프로토콜 환경에서 DMA I/O 요구의 최악 응답시간 분석

한주선* 하란† 민상렬*

서울대학교 컴퓨터공학과* 홍익대학교 컴퓨터공학과†

Analysis of Worst Case DMA Response Time in Fixed-Priority Bus Arbitration Protocol

Joosun Hahn* Rhan Ha† Sang Lyul Min*

Dept. of Computer Engineering, Seoul National University*

Dept. of Computer Engineering, Hong-Ik University†

요 약

CPU에게 최상위 우선순위가 할당된 고정우선순위 버스 프로토콜에서는 CPU와 DMA 컨트롤러의 버스 요구가 충돌할 경우 DMA 전송이 지연된다. 본 논문에서는 CPU와 다수의 DMA 컨트롤러가 시스템 버스를 공유하는 환경에서 DMA I/O 요구의 최악 응답시간을 분석하는 기법을 제안한다. 제안하는 최악 응답시간 분석 기법은 다음의 세 단계로 구성되어 있다. 첫번째 단계에서는 CPU 상에서 수행중인 각 CPU 태스크별로 최악 버스 요구 패턴을 구한다. 두번째 단계에서는 이들 CPU 태스크의 최악 버스 요구 패턴을 모두 통합해 CPU 전체의 최악 버스 요구 패턴을 구한다. 최종 세번째 단계에서는 CPU의 최악 버스 요구 패턴으로부터 DMA 컨트롤러의 버스 가용량을 구하고 DMA I/O 요구의 최악 응답시간을 산출한다. 모의 실험을 통해 제안하는 분석 기법이 일반적인 DMA 전송량에 대해 20% 오차 범위 이내에서 안전한 응답시간을 산출함을 보였다.

1. 서 론

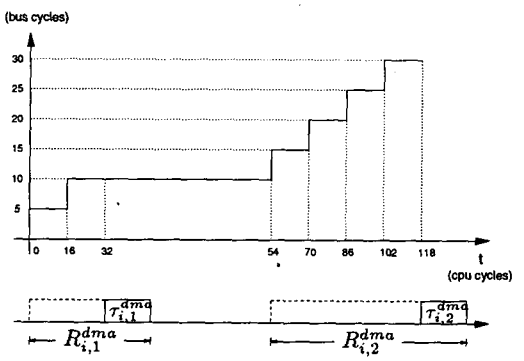
실시간 시스템(real-time system)의 가장 큰 특징은 모든 태스크의 시간 제약 조건이 만족됨을 보장해야 한다는 것이다. 이를 위해, 여러 스케줄 가능성 분석 기법들이 제시되었으나 [3, 4, 5, 6], 이들 연구를 통해 다양한 CPU 태스크 스케줄 가능성 조건이 제시된 반면, I/O 요구의 스케줄 가능성에 대한 연구는 많지 않았다 [2]. 최근 상당수의 실시간 시스템, 특히 내장형 실시간 시스템들(embedded real-time system)이 DMA(direct memory access) 방식의 I/O를 채택하고 있다. 이러한 실시간 시스템의 스케줄 가능성을 보장하기 위해서는 CPU 태스크 뿐만 아니라 I/O 요구의 스케줄 가능성도 반드시 검증되어야 한다.

본 논문에서는 CPU와 다수의 DMA 컨트롤러가 시스템 버스를 공유하는 환경에서 DMA I/O 요구의 최악 응답시간을 예측하는 기법을 제안한다. 이때, 시스템 버스는 CPU에게 최상위 우선순위가 할당된 고정우선순위 버스 프로토콜로 동작함을 가정한다. 이와 같은 버스 프로토콜에서 DMA 요구의 시간 분석을 어렵게 하는 요소는 CPU와 DMA 컨트롤러의 버스 요구가 충돌할 때마다 DMA 전송이 지연된다는 점이다. 이 문제를 해결하기 위해, CPU의 버스 요구에 대한 최대 도착 함수 $f_{cpu}(t)$와 DMA 컨트롤러의 버스 요구에 대한 최소 서비스 함수 $g_{dma}(t)$를 이용해 DMA I/O 요구의 최악 응답시간을 산출하였다.

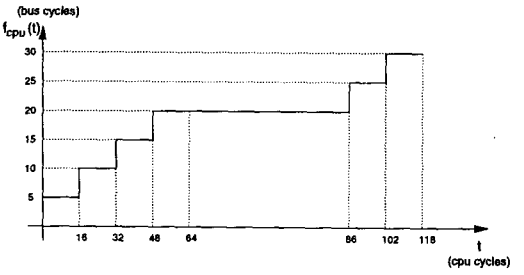
1.1. 시스템 모델과 태스크 모델

본 논문에서는 CPU와 주 메모리, 그리고 다수의 DMA 컨트롤러가 시스템 버스를 공유하는 시스템을 가정하고 있다. CPU에는 캐쉬가 장착되어 있어서 캐쉬 접근 실패가 발생하는 경우에만 시스템 버스 요구를 생성한다. 이중, 명령어 참조의 캐쉬 접근 실패로 인한 영향을 분석하기 위해 모든 데이타 참조는 캐쉬 접근에 성공한다고 가정하였다. 다수의 DMA 컨트롤러는 시스템 버스의 사용을 위해 서로간에 혹은 CPU와 경쟁한다. 이때, CPU와 각 DMA 컨트롤러에는 고정우선순위가 할당되어 있어서 가장 높은 우선순위를 가지는 장치에게 시스템 버스 사용이 허용되는데, 모든 장치중 CPU가 최상위 우선순위를 가지고 있다. 즉, DMA 컨트롤러는 CPU 혹은 상위 우선순위를 가지는 다른 DMA 컨트롤러에서 버스 요구가 없을 경우에만 자신의 I/O를 수행할 수 있다.

CPU 태스크 집합은 n_{cpu}개의 주기적 태스크 τ_1^{cpu}, τ_2^{cpu}, $\cdots$, $\tau_{n_{cpu}}^{cpu}$로 구성된다. 각 CPU 태스크 τ_i^{cpu}는 (주기 T_i^{cpu}, 최악 수행시간 C_i^{cpu}, 마감시간 D_i^{cpu})로 정의되며, 고정우선순위 스케줄링 정책에 따라 스케줄링된다. DMA 컨트롤러 i의 I/O 요구는 주기적인 DMA 태스크 τ_i^{dma}, (주기 T_i^{dma}, 마감시간 D_i^{dam}, 전송량 S_i^{dma})로 정의되며, $i < j$인 경우 DMA 컨트롤러 i의 τ_i^{dma}가 DMA 컨트롤러 j의 τ_j^{dma}보다 높은 우선순위를 가진다.



(a) CPU 버스 요구의 최악 도착 함수



(b) CPU 버스 요구의 최대 도착 함수

그림 1: CPU의 최대 도착 함수.

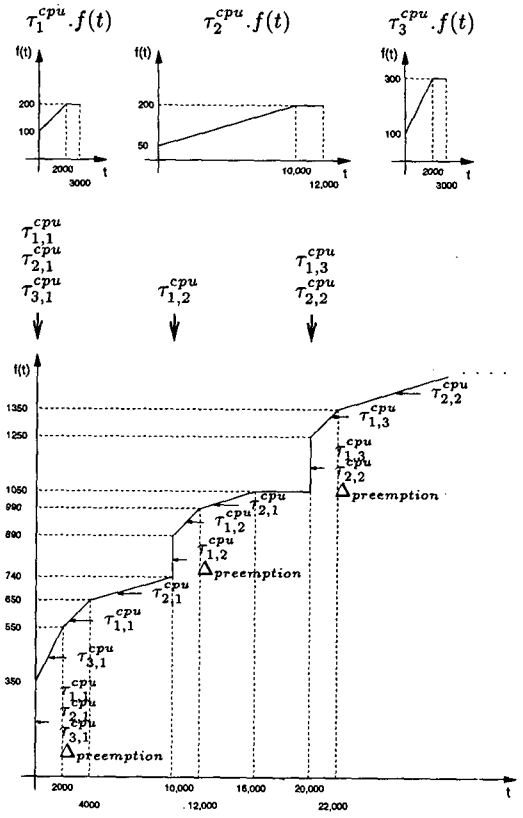


그림 2: 각 CPU 태스크의 최대 도착 함수 통합.

2. 최대 도착 함수 $f_{cpu}(t)$와 최소 서비스 함수 $g_{dma}(t)$

CPU의 최대 도착 함수(maximum arrival function) $f_{cpu}(t)$는 길이 t의 모든 수행 구간 $[x, x+t]$에 대해서 CPU로부터 생성되는 최대 버스 요구량으로 정의된다. 이 정의로부터 $f_{cpu}(t)$가 길이 t의 모든 CPU 수행 구간에서 생성되는 버스 요구량의 상한이며, 구간의 길이 t에 대해 단조증가함을 알 수 있다. 마찬가지로, DMA 컨트롤러의 최소 서비스 함수(minimum service function) $g_{dma}(t)$는 길이 t의 모든 수행 구간 $[y, y+t]$에 대해서 DMA 컨트롤러들에게 보장되는 최소 버스 가용량으로 정의된다. 이때, $g_{dma}(t)$ 값만큼의 최소 버스 가용량이 DMA 수행 구간 $[0, t]$에 제공되는 것이 DMA I/O 요구의 관점에서 최악임을 알 수 있다.

$f_{cpu}(t)$와 $g_{dma}(t)$의 관계를 설명하기 위해, CPU의 수행중 생성되는 버스 요구의 최악 도착 시나리오의 한 예를 그림 1 (a)에 나타내었다. 이 최악 도착 함수는 CPU의 최단 수행 시간 내에 최대로 발생 가능한 캐쉬 접근 실패를 분석해 구할 수 있다. 즉, 캐쉬 접근 실패가 발생하지 않는 한 파이프라이닝의 매 CPU cycle마다 명령어가 패치(fetch)되고, 캐쉬 접근 실패가 발생할 경우 시스템 버스 요구를 생성한다. 그림 1의 예에서는 한번의 캐쉬 접근 실패를 서비스하기 위해 5 bus cycles 혹은 15 CPU cycles가 필요하다고 가정하였다.

이때, 임의의 DMA 태스크 τ_i^{dma}가 각각 시점 0와 시점 54에 도착하는 경우를 살펴보자. 그림에서 보듯이, DMA 태스크는 CPU의 버스 요구가 폭주할수록 최악의 응답시간을 보임을 알 수 있다. 따라서, 길이 t의 모든 수행 구간을 고려해 CPU의 최대 버스 요구량, 즉 최대 도착 함수 $f_{cpu}(t)$를 결정해야, 최악 응답 시간을 보이는 시점(이 예에서는 시점 54)에 도착한 DMA 태스크에게 제공되는 최소 버스 가용량, 즉 최소 서비스 함수 $g_{dma}(t)$를 구할 수 있다. 그림 1 (b)에 이 예에 대한 최대 도착 함수 $f_{cpu}(t)$를 나타내었다.

3. 최악 응답시간 분석 기법

제안하는 최악 응답시간 분석 기법은 다음의 세 단계로 구성된다. 첫번째 단계에서는 우선 CPU 상에서 수행중인 각 CPU 태스크 τ_i^{cpu}의 버스 요구에 대한 최대 도착 함수 $\tau_i^{cpu}.f(t)$를 구한다. 두번째 단계에서는 이들 CPU 태스크의 최대 도착 함수를 모두 통합해 CPU의 버스 요구에 대한 최대 도착 함수 $f_{cpu}(t)$를 구한다. 이렇게 산출된 $f_{cpu}(t)$는 길이 t의 구간에서 CPU에 의해 DMA 전송이 지연되는 시간의 상한이 된다. 따라서, 세번째 단계에서는 DMA 컨트롤러의 버스 요구에 대한 최소 서비스 함수 $g_{dma}(t) = t - f_{cpu}(t)$를 구하고 DMA 태스크 τ_i^{dma}의 최악 응답시간을 산출한다. 최악 응답시간 분석 기법에 대한 보다 상세한 설명을 [1]에 기술하였다.

CPU 태스크 τ_i^{cpu}의 최대 도착 함수 $\tau_i^{cpu}.f(t)$: 앞에서 설명한 바와 같이, $\tau_i^{cpu}.f(t)$는 CPU 태스크 τ_i^{cpu}의 수행 중 발생하는 최대 캐쉬 접근 실패를 분석함으로써 구할 수 있다. 제안하는 분석 기법에서 $\tau_i^{cpu}.f(t)$는 2-구간 선형 함수(2-piecewise linear function)의 형태로 표현되며(그림 2), 그 형성 과정은 다음과 같다. (1) 우선 각 기본 블록(basic block)에 대해, 모든 명령어 참조를 캐쉬 접근 실패로 간주하고 최대 도착 함수를 정의한다. (2) 이들을 구문 구조에 따라 인접한 블록끼리 계층적으로 접속해 가면서 보다 넓은 수행 구간에 대해 최대 도착 함수를 구한다. 이때, 캐쉬 접근 실패로 잠정 분석되었던 참조들의 개선된 참조 정보가 반영된다. (3) 최종 결과로 τ_i^{cpu}의 전체 수행 구간에 대해 최대 도착 함수를 얻을 수 있다. 접속 과정에서 형성되는 중간 블록들에 대한 최대 도착 함수를 유지하기 위해 MAFA(maximum arrival function abstraction) 자료구조을 두고, 두 인접한 블록의 MAFA 자료구조간 접속 연산(concatenation operation)을 정의하였다.

CPU의 버스 요구에 대한 최대 도착 함수 $f_{cpu}(t)$: 그림 2은 세 CPU 태스크 τ_1^{cpu}, τ_2^{cpu}, τ_3^{cpu}의 최대 도착 함수를 통합해 $f_{cpu}(t)$를 형성하는 예를 보여주고 있다. 그림에서 보듯이, 각 태스크의 도착 주기마다 $\tau_i^{cpu}.f(t)$의 기울기가 급한 순서대로 선택해 연접한다. 이때, 태스크간의 최악 선점 상황을 고려해 선점 비용으로 생성되는 버스 요구량 $\Delta_{preemption}(t)$도 부가시킨다. 부가적인 선점 비용이란, 태스크가 선점형 스케줄링 방식에 의해 스케줄링될 때 태스크간 캐쉬 간섭에 의해 부가적으로 발생하는 캐쉬 접근 실패를 서비스하기 위해 필요한 버스 요구량을 의미한다. $\Delta_{preemption}(t)$는 선형 계획법(linear programming)을 이용해 구할 수 있다.

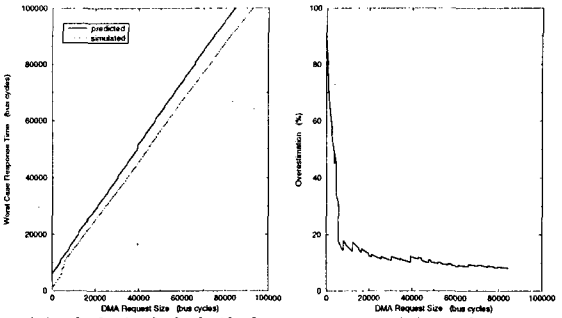
DMA 태스크 τ_i^{dma}의 최악 응답시간 R_i^{dma}: CPU의 $f_{cpu}(t)$는 길이 t의 구간에서 CPU에 의해 DMA 전송이 지연되는 시간의 상한이 된다. 따라서, DMA 컨트롤러의 버스 요구에 대한 최소 서비스 함수는 $g_{dma}(t) = t - f_{cpu}(t)$로 구할 수 있다. 그리고, $g_{dma}(t)$의 역함수 $g_{dma}^{-1}(t)$를 이용해 DMA 태스크 τ_i^{dma}의 최악 응답시간을 산출한다.

$$R_i^{dma} = g_{dma}^{-1}\left(S_i^{dma} + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{dma}}{T_j^{dma}} \right\rceil S_j^{dma}\right)$$

여기서 $g_{dma}^{-1} = \min\{t \mid s = g_{dma}(t)\}$이며, $hp(i)$는 τ_i^{dma}보다 상위 우선순위를 가지는 DMA 태스크들의 집합을 의미한다. 위의 재귀식을 반복적으로 풀어서 수렴하는 R_i^{dma}를 DMA 태스크 τ_i^{dma}의 최악 응답시간으로 산출한다. 이와 같이 산출된 최악 응답시간에는 CPU 및 상위 우선순위 DMA 컨트롤러에 의해 τ_i^{dma}의 DMA 전송이 지연되는 시간이 모두 포함되어 있다.

4. 성능 평가

제안하는 분석 기법의 성능 평가를 위해, 주어진 CPU 태스크 집합에 대해 최악 응답시간 분석으로 예측한 DMA 태스크의 최악 응답시간과 모의 실험으로 얻은 응답시간을 비교하였다. 그림 3 (a)는 DMA 태스크의 전송량을 100000 bus cycles까지 증가시켜 가면서 응답시간을 산출한 결과이다. 이 결과로부터 전송량이 5000 bus cycles 이상인 경우 20% 오차 범위 이내에서 안전한 응답시간이 산출되며, 전송량이 증가할수록 오차가 점차 감소함을 알 수 있다(그림 3 (b)). 이와 같은 오차는 (1) 각 CPU 태스크의 캐쉬 접근 실패 분석 및 최대 도착 함수 통합 과정에서 가정했던 최악 수행과 실제 수행간의 차, (2) 분석시 고려하지 않았던, 이전 주기의 수행에서 적재된 캐쉬 내용의 재사용 등에서 비롯된 것이다.



(a) 최악 응답시간 분석 (b) 오차

그림 3: DMA 태스크의 응답시간 분석과 오차.

5. 결론 및 향후과제

본 논문에서는 CPU와 다수의 DMA 컨트롤러가 시스템 버스를 공유하는 환경에서 DMA I/O 요구의 최악 응답시간을 예측하는 기법을 제안하였다. 제안하는 분석 기법에서는 CPU에게 최상위 우선순위가 주어진 고정우선순위 버스 프로토콜을 가정하고, 먼저 CPU의 최대 도착 함수 $f_{cpu}(t)$를 CPU에 의한 DMA 전송 지연시간의 상한으로 구한 후, 이로부터 DMA 컨트롤러의 최소 서비스 함수 $g_{dma}(t)$를 구해 DMA 태스크의 최악 응답시간을 산출한다. 모의 실험의 결과, 통상적인 DMA I/O 전송에 있어서 20% 오차 범위 이내의 안전한 응답시간이 산출됨을 보였다. 향후과제로는 데이타 참조의 캐쉬 접근 실패로 인한 영향을 분석에 포함시키는 것과 라운드 로빈 버스 프로토콜 환경에서 DMA I/O 요구의 최악 응답시간을 분석하는 것을 고려하고 있다.

참고 문헌

[1] J. Hahn, R. Ha, S.-L. Min, and J. W.-S. Liu. Analysis of Guaranteed DMA Response Time in Fixed-Priority Bus Arbitration Protocol. Technical Report SNU-CE-TR-1999-1, Dept. of Computer Engineering, Seoul National University, Aug. 1999.

[2] T.-Y. Huang. *Worst-Case Timing Analysis of Concurrently Executing DMA I/O and Programs.* PhD thesis, University of Illinois, April 1997.

[3] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The BCS Computer Journal*, 29(5):390–395, Oct. 1986.

[4] J. P. Lehoczky, L. Sha, and Y. Ding. The Rate Monotonic Scheduling Algorithm – Exact Chracterization and Average Case Behavior. In *Proceedings of the 10th Real-Time Systems Symposium*, pages 166–171, 1989.

[5] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[6] K. W. Tindell, A. Burns, and A. J. Wellings. An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. *Journal of Real-Time Systems*, 6(2):133–151, March 1994.