논문번호 04-02-05

계층버스 다중처리기에서 캐시 일관성 프로토콜의 민감도 분석

Sensitivity Analysis of Cache Coherence Protocol for Hierarchical-Bus Multiprocessor

李 興 宰*, 崔 眞 圭*, 奇 長 根**, 李 圭 晧***
Heung-Jae Lee*, Jin-Kyu Choe*, Jang-Geun Ki***, Kyou-Ho Lee***

요 약

계층버스 다중처리기 시스템에서 캐시 일관성 프로토콜은 시스템 성능에 영향을 준다. 특정 캐시 일관성 프로토콜 하에서 시스템의 성능은 버스의 대역폭 및 메모리크기, 메모리 블록의 크기에 따라 영향을 받는다. 따라서 시스템 성능에 영향을 미치는 요소들에 대한 민감도 분석이 필요하다.

본 논문에서는 계층버스 다중처리기에 캐시 일관성 프로토콜을 적용하고, 프로토콜에서 정의된 상태가 나타날 확률을 구하였다. 구해진 확률값을 분석적 모델에 적용하여 시뮬레이션을 하였다. 그리고 시뮬레이션 결과를 기반으로 시스템의 성능에 영향을 미치는 요소에 대한 민감도 분석을 하였다.

Abstract

In a hierarchical-bus multiprocessor system, cache coherence protocol has effect on system performance. Under a particular cache coherence protocol, system performance can be affected by bus bandwidth, memory size, and memory block size. Therefore sensitivity analysis is necessary for the part of multiprocessor system.

In this paper, we set up cache coherence protocol for hierarchical-bus multiprocessor system, and compute probability of state of protocol, and analyze sensitivity for part of system by simulation.

Keyword: bus, multiprocessor, cache, coherence protocol, sensitivity

Ⅰ. 서론

다중처리기의 구조로 공유버스(Common Bus)가 구성비용이 저렴하고 구성의 용이성으로 선호되고 있다. 그러나 공유버스 구조의 다중처리기는 버스를 사용하고자 하는 다수의 요구로 인하여 버스에서 병목현상을

본 연구는 2004년도 한국전자통신연구원의 연구비 지원에 의해 수행된 연구 결과임.
接受日:2004年 8月 23日, 修正完了日:2004年 12月 15日

* 韓南大學校 韓南大學校 情報通信멀티미디어工學部
(School of Info. Tech. & Multimedia Eng., Hannam Univ.)
** 公州大學校 情報通信工學部
(Division of Info. & Communication Eng., Kongju Nat. Univ.)
*** 韓國電子通信硏究院
(Electronics and Telecommunications Research Institute)

초래한다. 또한 메모리는 프로세서의 처리속도에 비해 액세스 속도가 느리므로 많은 지연이 발생한다. 버스의 병목현상과 메모리의 지연은 시스템의 성능을 저해하는 가장 큰 문제점이다. 따라서 메모리에서의 지연과 버스의 병목현상을 최소화함으로써 시스템의 성능을 향상시키기 위한 연구가 이루어지고 있다.

메모리 지연과 버스의 병목현상을 최소화하는 공통적인 방법으로 캐시 메모리를 사용한다. 캐시 메모리의 사용은 시스템 성능을 크게 향상시키는 반면에 복사된 데이터간의 불일치 문제(cache coherence)가 발생한다. 이러한 캐시 일관성 문제를 해결하기 위하여 여러 가지 프로토콜들이 제안되어 왔다. 그러나 공유버스 다중처리기에서는 아무리 성능이 우수한 프로토콜을 사용한다고 할지라도 프로세서의 수를 어느 한계 이상 증가시키면 버스의 병목 현상이 발생하여 확장성을 제한한다.

이러한 확장성의 문제를 해결하기 위한 방법으로 다중 버스와 계층버스가 사용된다. 다중버스 시스템은 연결과 인터페이스를 위한 비용이 크며 동기화 문제를 해결하는데 어려움이 있다. 계층적 구조의 다중프로세서 시스템에서는 각 버스상의 전송부하를 감소시키기 위하여 버스와 버스를 연결하는 인터페이스 부분에 별도의 캐시를 장착하는 것이 일반적이며, 이러한 공유캐시는 관련된 버스들 각각에 대하여 스누피(Snoopy) 프로토콜을 주로 사용하여 단일 버스 구조에서의 캐시 일관성 방법을 확장하여 구현할 수 있다[1][2].

공유버스 기반 계층버스 구조에 사용되는 캐시 일관성 프로토콜은 일반적으로 하위 레벨에는 Write-through 프로토콜을 사용하고, 상위 레벨에는 Write-back 프로토콜을 사용하고 있다. HiFi 버스는 Write-through 프로토콜과 Write-back 프로토콜을 지원하도록 설계되었다[3][4]. 따라서 HiFi 버스를 계층버스 구조로 확장할 경우 기존의 단일 레벨에서 적용되었던 캐시 일관성 프로토콜을 두 레벨에 그대로 적용하고 레벨간의 영향을 고려하면 확장성 문제를 해결하면서 캐시 일관성 문제를 해결할 수 있다.

특정 캐시 일관성 프로토콜을 선택할 경우 초기 시스템 설계단계에서 최적의 성능을 발휘할 수 있는 구조를 생각해 보지 않을 수 없다. 시스템 성능에 영향을 미칠 수 있는 시스템 구성요소들의 민감도를 구하는 문제는 매우 중요하다. 그러나 변수들의 민감도를 구하는 것은 매우 어렵고 시간이 많이 걸리는 문제로, 분석적 모델의 사용은 변수들에 대한 민감도 적용을 용이하게 할 뿐만 아니라 성능분석 기간을 단축시킬 수 있다는 장점이 있다[5][6].

따라서 본 논문에서는 계층버스 다중처리 마이크로프로세서에서 캐시 일관성 프로토콜의 성능평가 및 민감도 분석에 관한 연구를 하였다. 계층버스구조의 시스템을 모델링하고 캐시 일관성 프로토콜을 적용하여 성능을 평가하였다. Markov 정적 상태도[7]를 이용하여 각 상태에 존재할 확률을 구하고, 이 확률 값을 시뮬레이션의 입력으로 사용하였다. 모델링 및 시뮬레이션은 Awesim[8]을 사용하였다. 민감도 분석을 위하여 메모리 크기와 버스의 대역폭을 변화시키면서 프로세서의 효율, 버스의 이용률을 구하였다.

본 논문의 구성은 다음과 같다. 2장에서는 계층버스 시스템의 구성과 캐시 일관성 프로토콜에 관해 설명하고, 3장에서는 시뮬레이션을 위한 각 상태의 확률을 구하고, 시뮬레이션 모델링을 하였다. 4장에서는 시뮬레이션 결과에 대한 분석을 하였으며, 마지막으로 5장에서는 결론을 기술하였다.

II. 계층버스 구조와 프로토콜

본 논문에서는 한국전자통신연구원에서 제안한 계층버스 다중처리 마이크로프로세서인 Raptor[9]의 구조를 모델로 Raptor 프로토콜을 적용하여 성능 평가 하였다. Raptor는 하나의 칩에 4개의 64비트 범용 처리장치(General Processor Unit)로 구성되며, 각 범용 처리 장치는 on-chip 캐시를 1차 캐시로 사용하며, 외부의 2차 캐시와 연결된다.

그림 1은 시뮬레이션 모델이다. 1차 캐시(L1 cache) 미스에 의해 발생하는 메모리 요구에 대한 지연을 최소화하고 효과적인 대역폭을 제공하기 위하여 4개의 L1 캐시에 2차 캐시(L2 cache)를 연결하고, 이 L2 캐시들을 넓은 대역폭을 갖는 2차 캐시 버스로 연결한다. 주기억장치는 병렬 메모리를 사용함으로서 프로세서의 캐시 미스 시에 발생하는 메모리 접근에 대하여

지연을 최소화하도록 하였다.

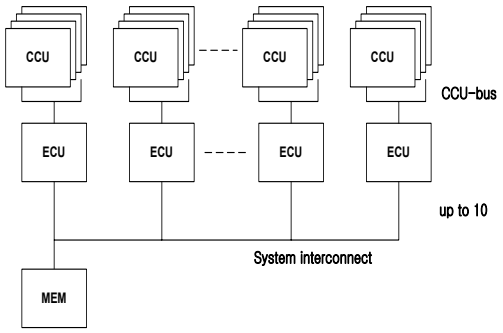

그림 1. 다중처리시스템 모델
Fig. 1. Multiprocessor model

그림 2는 CCU(Cache Control Unit)의 캐시 일관성 프로토콜을 위한 상태 천이도이다. 여기에는 ECU로부터의 영향을 고려하였다. CCU의 상태는 무효상태(invalid - I), 수정되지 않고 공유할 수 있는 상태(Exclusive - E), 수정되고 공유할 수 있는 상태(Owned - O), 수정되고 공유된 상태(Owned/Shared - OS), 수정된 상태(Modified - M), 수정되지 않고 공유할 수 있는 상태(Shared - S)의 여섯 가지 상태로 정의된다. CCU에서 O, OS, M의 세 가지 상태를 정의함으로써 프로세서의 쓰기 요구에 대한 버스의 사용을 최소화할 수 있다.

그림 3은 ECU(External Cache control Unit)의 캐시 일관성 프로토콜을 위한 상태 천이도이다. 여기에는 CCU로부터의 영향을 고려하였다. ECU의 상태는 무효상태(Invalid - I), 수정되지 않고 공유할 수 있는 상태(Exclusive/Shared - ES), 수정되지 않고 공유할 수 있는 상태(Shared/Shared - SS), 수정되고 공유할 수 있는 상태(Owned/Shared - OS), 쓸모없는 상태(Stale/Modified - SM)의 다섯 가지 상태로 정의된다.

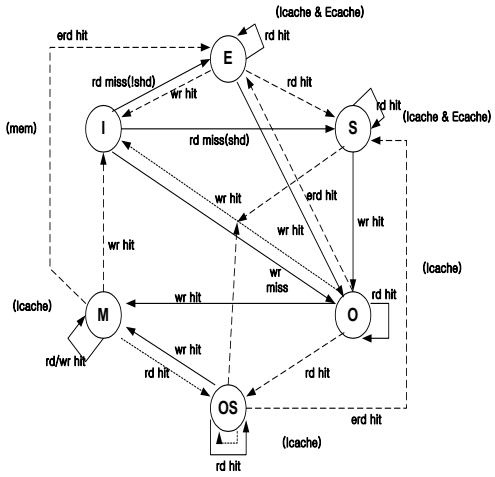

그림 2. CCU(L1 cache)의 상태천이도
Fig. 2. State transition diagram of CCU

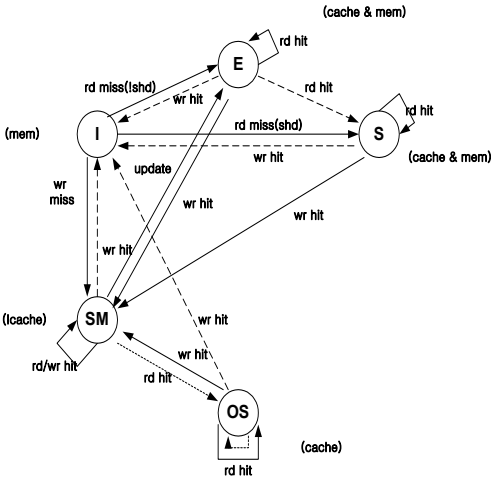

그림 3. ECU(L2 cache)의 상태천이도
Fig. 3. State transition diagram of ECU

III. 확률계산 및 시뮬레이션

그림 1의 시스템 모델을 기반으로 Markov 정적 상태도를 작성하고, 확률을 구한다. 구한 확률 값을 입력으로 시뮬레이션을 하였다.

3.1 확률 계산

시뮬레이션에 사용할 확률을 구하기 위하여 다음과 같이 정의한다.

a : 1차 캐시의 블록 하나를 택하여 읽기 동작을 수행할 확률

b : 1차 캐시의 블록 하나를 택하여 쓰기 동작을 수행할 확률

c : 1차 캐시의 블록 하나를 택하였을 때 미스될 확률

d : 두 개 이상의 1차 캐시가 같은 블록을 가질 확률

e : 요구한 블록이 다른 1차 캐시에 존재할 확률

a′ : 2차 캐시의 블록 하나를 택하여 읽기 동작을 수행할 확률

c' : 2차 캐시의 블록 하나를 택하였을 때 미스될 확률

d′ : 전체 캐시에서 두 개 이상의 캐시가 같은 블록을 가질 확률

e′ : 전체 캐시에서 요구한 블록이 다른 캐시에 존재할 확률

f : 2차 캐시에 대한 1차 캐시의 크기의 비

f' : 메모리에 대한 2차 캐시의 크기의 비

1차 캐시의 확률은 다음 식으로 구해진다.

$$P_I = \frac{bd + c}{a + b + c + bd}$$

$$P_M + P_O = \frac{b}{a'(1-e') + bd + c + ad + b}$$

$$P_{OS} = \frac{ad(P_M + P_O)}{bd + c + a'd' + b}$$

$$P_E = \frac{a(1-e)P_I + a'(1-e')(P_M + P_O)}{bd + c + ad + b}$$

$$P_S = \frac{adP_E + aeP_I + a'd'P_{OS}}{bd + c + b}$$

$$P_O = \frac{b(P_E + P_I + P_S)}{a'(1-e') + bd + c + b + ad}$$

$$P_M = 1 - P_I - P_E - P_O - P_{OS} - P_S$$

2차 캐시의 확률은 다음 식으로 구해진다.

$$P_I = \frac{bd' + c'}{a' + b + c' + bd'}$$

$$P_{SM} = \frac{b}{a'(1-e') + a'd' + bd' + c' + b}$$

$$P_{ES} = \frac{a'(1-e')(P_{II} + P_{SM})}{a'd' + b + c' + bd'}$$

$$P_{SS} = \frac{a'd'P_{ES} + a'e'P_I}{bd' + c' + b}$$

$$P_{OS} = \frac{a'd'P_{SM}}{bd' + c' + b}$$

설정된 성능 분석 요소에 대하여 기준 값 적용시의 이용률과 변이 값 적용시의 이용률을 계산식에 적용하여 민감도(SR)를 구할 수 있다. 민감도는 다음 식으로 구할 수 있다.

$$SR(VAR) = \frac{U(VAR) - U(\triangle VAR)}{U(VAR)} \times 100$$

여기에서 VAR은 메모리 크기, 버스 대역폭이다. $SR(VAR)$은 각 변수(VAR)에 대해서 프로세서 효율, CCU의 어드레스 버스 이용률, CCU 버스의 데이터 버스 이용률, ECU 버스의 어드레스 버스 이용률, ECU 버스의 데이터 버스 이용률에 대한 민감도를 구할 수 있다.

3.2 시뮬레이션 모델링

시뮬레이션은 캐시간의 전송(cache to cache)이 가능한 수정된 HiPi 버스를 사용하였으며, 시뮬레이션의 변수로 사용된 값은 프로세서의 수, 읽기를 수행할 확률, 히트율이다. 이러한 변수에 따라 각 상태의 확률을 구하여 시뮬레이션의 입력 값으로 사용하였다. 버스의 사이클 타임은 10ns부터 80ns까지 변화시키고, 메모리 비율은 기준(0.000125)의 1/2부터 ×2까지 변화시켰다. 주기억장치의 크기는 메모리는 16개의 블록으로 세분화하여 프로세서의 entity 발생 시에 랜덤하게 메모리를 선택하도록 하였다. 프로세서의 수는 8개에서 48개까지, 히트율은 0.90에서 0.98까지 변화시키며 시뮬레이션을 수행하였다.

Ⅳ. 시뮬레이션의 결과 분석

4.1 시뮬레이션 결과

공유 버스 구조의 캐시 일관성 방법은 스누핑을 사용한다. 스누핑에서는 버스에 연결된 모든 캐시가 버스상의 요청을 항상 감시함으로써 버스의 요청에 대해 각 캐시가 일관성 동작을 수행한다. 따라서 계층 버스 구조에서 캐시 일관성을 통합망하기 위해서는 온 칩 캐시에서 발생한 버스 요청이 다른 온 칩 캐시에서 감시될 수 있도록 전달되어야 한다. 이러한 일관성 통합망을 위한 버스 요청을 상위 단계의 버스를 통하여 다른 온 칩 캐시로 전달되어야하기 때문에 상위 단계의 버스 통행량이 증가하게 되어 전체 시스템의 병목으로 작용하여 시스템의 확장성을 제한하는 요소가 된다. 이러한 버스의 통행량을 줄이기 위한 방법으로 일관성 프로토콜에서는 1차 캐시에 O, OS, M상태를 정의하였으며, 2차 캐시에 SM 상태를 정의하였다.

그림 4와 그림 5는 히트율 변화에 따른 이용률이다. 프로세서의 수를 48개로 고정시킨 상태에서 이용률을 보면 히트율 변화에 따라 프로세서의 효율이 증가를 하고, 어드레스 버스, 데이터 버스의 이용률은 감소하는 것을 볼 수 있다. 이것은 캐시간의 전송이 가능하고, 히트율이 증가할수록 메모리까지 프로세서의 요구가 내려갈 확률이 적기 때문이다.

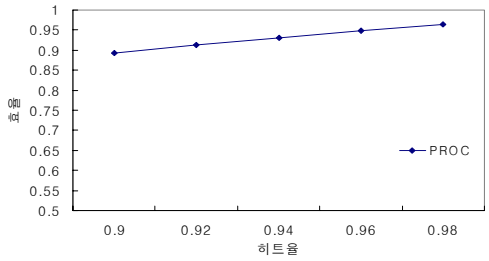

그림 4. 히트율 변화시 프로세서 효율
Fig. 4. Processor efficiency vs. Hit ratio

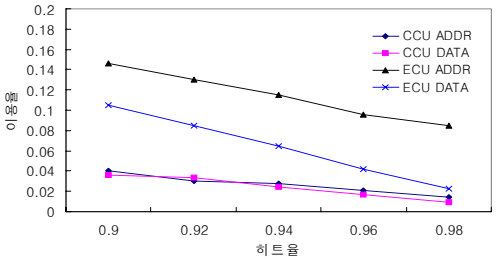

그림 5. 히트율 변화시 버스의 이용율
Fig. 5. Utility vs. Hit ratio

그림 6과 그림 7은 프로세서 수 변화에 따른 이용률이다. 히트율을 0.98로 고정시킨 상태에서 프로세서의 수를 증가시킬 때, ECU 버스의 어드레스 버스와 데이터 버스는 병목에 의하여 약간의 증가를 보이지만, 성능에 큰 영향을 주지 않는다. CCU버스의 어드레스 버스와 데이터 버스는 ECU의 병목이 작기 때문에 프로세서 수에 영향을 받지 않는다.

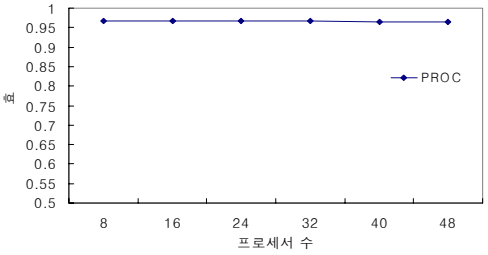

그림 6. 프로세서 수 변화에 따른 프로세서 효율
Fig. 6. Efficiency vs. No. of processor

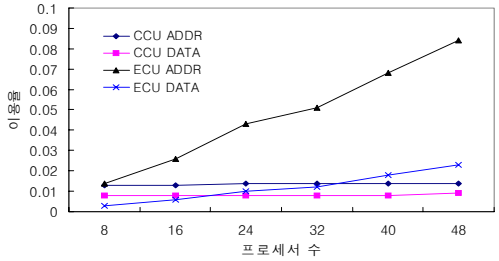

그림 7. 프로세서 수 변화에 따른 버스 이용률

Fig. 7. Utility vs. No. of processor

그림 8과 그림 9는 버스의 주기를 변화시켰을 경우이다. 버스의 주기를 증가시킬수록 프로세서의 효율은 떨어지고, CCU 버스와 ECU 버스의 이용률은 증가한다. 버스의 주기가 커질수록 버스에서의 병목 현상과 메모리의 지연에 의해서 성능이 떨어진다.

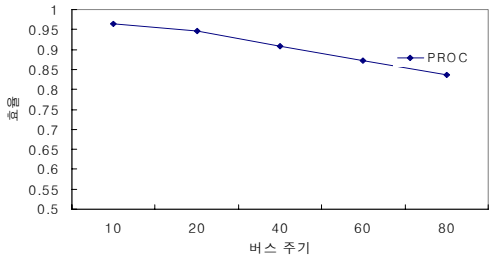

그림 8. 버스 주기 변화에 따른 프로세서 효율

Fig. 8. Processor efficiency vs. Bus cycle

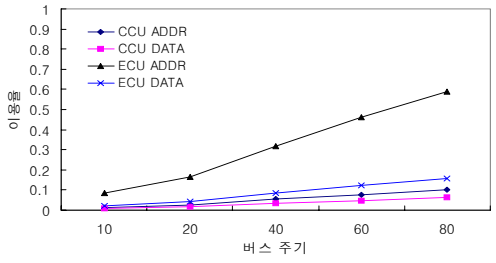

그림 9. 버스 주기 변화에 따른 버스 이용률

Fig. 9. Bus utility vs. Bus cycle

그림 10, 11, 12는 프로세서의 수를 48개, 히트율을 0.98, 버스의 주기를 10ns로 놓고 메모리 비율을 변화시킬 경우의 프로세서 효율과 버스의 이용률이다. 프로세서의 효율은 f에 따라 크게 변하고, 버스 이용률은 f'에 따라 큰 차이를 보인다. 메모리 사이즈를 크게 할수록 성능이 좋아지지는 않는다. 그 이유는 캐시의 히트율이 캐시의 크기와 블록 사이즈에 따라 큰 영향을 받기 때문이다.

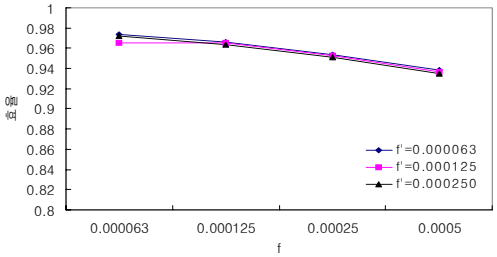

그림 10. 메모리 비율 변화에 따른 프로세서 효율

Fig. 10. Processor efficiency vs. Memory ratio

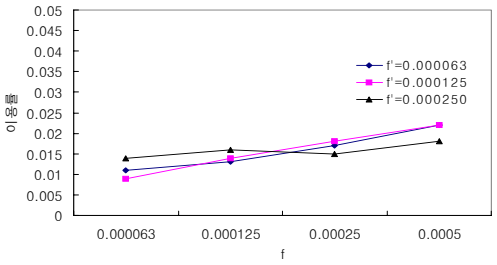

그림 11. 메모리 비율 변화에 따른 CCU 어드레스 버스 이용률

Fig. 11. CCU address bus utility vs. Memory ratio

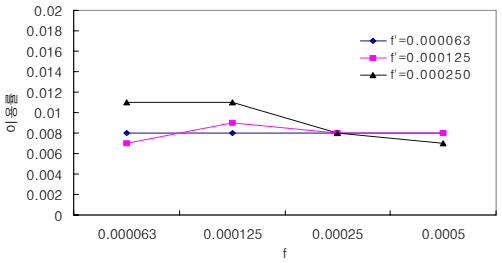

그림 12. 메모리 비율 변화에 따른 CCU 데이터 버스 이용률

Fig. 12. CCU data bus utility vs. Memory ratio

4.2 민감도 분석

민감도의 기준 값은 버스의 주기는 20ns, 메모리 비율 0.000125로 해서 변수에 따른 민감도를 구하였다{9}. 기준 값에 대한 프로세서 효율 및 버스 이용률에 대한 민감도는 표 1과 표 2에 정리하였다.

표 1과 표 2에서, 기준 값의 2배에서의 각 변수에 대한 민감도를 비교해 보면, 프로토콜의 성능에 가장 민감하게 영향을 미치는 변수를 찾아낼 수 있다. 프로세서 효율에 대한 민감도를 비교해보면, 버스 주기 변화에 대해서 SR(버스 주기)은 4.01% 증가, 메모리 비율 변화에 대해서 SR(메모리 비율)은 1.35% 감소를 보인다. CCU 버스의 어드레스 버스 이용률에 대한 민감도를 비교해보면, 버스 주기 변화에 대해서 SR(버스 주기)은 96.30% 감소, 메모리 비율 변화에 대해서 SR(메모리 비율)은 6.25% 증가를 보인다. CCU 버스의 데이터 버스 이용률에 대한 민감도를 비교해보면, 버스 주기 변화에 대해서 SR(버스 주기)은 88.24% 감소, 메모리 비율 변화에 대해서 SR(메모리 비율)은 27.27% 증가를 보인다. ECU 버스의 어드레스 버스 이용률에 대한 민감도를 비교해보면, 버스 주기 변화에 대해서 SR(버스 주기)은 90.96% 감소, 메모리 비율 변화에 대해서 SR(메모리 비율)은 36.17% 감소를 보인다. ECU 버스의 데이터 버스 이용률에 대한 민감도를 비교해보면, 버스 주기 변화에 대해서 SR(버스 주기)은 95.45% 감소, 메모리 비율 변화에 대해서 SR(메모리 비율)은 11.76% 증가를 보인다.

표의 결과를 그래프로 나타내면 그림 13과 같다. 버스 주기에 대한 민감도가 메모리 크기에 대한 민감도보다 크다.

표 1. 버스 주기 변화에 대한 민감도

Table 1. Sensitivity by Bus cycle

구 분		기준	기준/2	기준*2	기준*3	기준*4
프로세서	이용률	0.947	0.965	0.909	0.873	0.836
	민감도(%)	0.00	-1.9	4.01	7.81	11.72
CCU address	이용률	0.027	0.014	0.053	0.076	0.100
	민감도(%)	0.00	48.15	-96.30	-181.48	-270.37
CCU data	이용률	0.017	0.009	0.032	0.047	0.062
	민감도(%)	0.00	47.06	-88.24	-176.47	-264.71
ECU address	이용률	0.166	0.084	0.317	0.460	0.589
	민감도(%)	0.00	49.40	-90.96	-177.11	-254.82
ECU data	이용률	0.044	0.023	0.086	0.124	0.158
	민감도(%)	0.00	47.73	-95.45	-181.82	-259.09

표 2. 메모리 비율 변화에 대한 민감도

Table 2. Sensitivity by Memory ratio

구분	f' / f	기준/2		기준값		기준*2	
		이용률	민감도(%)	이용률	민감도(%)	이용률	민감도(%)
프로세서	기준값	0.966	0.00	0.965	0.00	0.964	0.00
	기준/2	0.974	-0.83	0.965	0.00	0.972	-0.83
	기준*2	0.954	1.24	0.953	1.24	0.951	1.35
CCU address	기준값	0.013	0.00	0.014	0.00	0.016	0.00
	기준/2	0.011	15.38	0.009	35.71	0.014	12.5
	기준*2	0.017	-30.77	0.018	-28.57	0.015	6.25
CCU data	기준값	0.008	0.00	0.009	0.00	0.011	0.00
	기준/2	0.008	0.00	0.007	22.2	0.011	0.00
	기준*2	0.008	0.00	0.008	11.1	0.008	27.27
ECU address	기준값	0.079	0.00	0.084	0.00	0.094	0.00
	기준/2	0.053	32.9	0.058	30.95	0.071	24.47
	기준*2	0.113	-43.04	0.124	-47.62	0.128	-36.17
ECU data	기준값	0.019	0.00	0.023	0.00	0.034	0.00
	기준/2	0.019	0.00	0.024	-4.35	0.038	-11.76
	기준*2	0.017	10.53	0.021	8.70	0.030	11.76

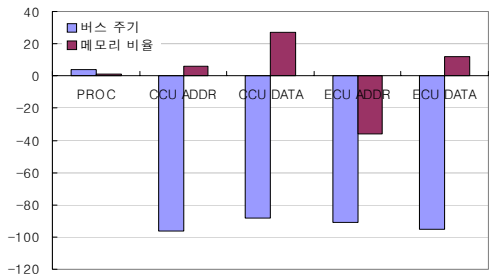

그림 13. 각 변수에 대한 민감도
Fig. 13. Sensitivity of components

V. 결론

본 논문에서는 계층버스 다중처리기에 캐시 일관성 프로토콜을 적용하여 성능평가를 하였다. 성능평가 방법으로는 분석적 모델링에 의한 시뮬레이션 방법을 택하였다. 올바른 가정과 인자의 설정을 위하여 각 프로토콜의 블록의 상태를 Markov 정적 상태도로 나타내고, 각 상태에 존재할 확률을 계산하였다. 계산된 확률 값을 시뮬레이션의 인자로 사용하였다.

시뮬레이션 결과, 버스의 주기를 증가시키면 프로세서의 효율과 시스템의 성능이 감소하며, 메모리 비율 f와 f'를 변화시켰을 때, f'=0.000063이고 f는 0.000063에서 프로세서의 효율은 0.974이고, 시스템의 성능은 46.752로 가장 좋은 성능을 보이는 것으로 분석된다.

시뮬레이션에 의한 민감도 분석에서 프로세서 효율, CCU와 ECU의 어드레스버스 및 데이터버스의 이용율 등을 종합 분석하면, 시스템의 성능에 가장 민감하게 작용을 하는 요소는 버스 주기임을 알 수 있었다.

시스템 성능에 영향을 미치는 요소에 대한 민감도 분석은 시스템 설계단계의 초기에 최적의 성능을 발휘할 수 있는 시스템 구성을 제안할 수 있어, 향후 계층버스 다중처리 시스템 개발에 활용할 수 있을 것이다.

참고문헌

[1] Kai Hwang, *Advanced Computer Architecture:Parallelism, Scalability, Programmability*, McGraw-Hill, 1993.

[2] An Do Ki, Byung Kwan Park, Won Sae Sim, Kyeng Yong Kang, Yong Ho Yoon, "Highly Pipelined Bus:HiPi-Bus," The ETRI Journal 전자통신, Vol.13, No.3, pp.33-50, Oct. 1991.

[3] Qing Yang, Laxmi N. Bhuyan, Bao-Chyn Liu, "Analysis and Comparison of Cache Coherence Protocols for a Packet-Switched Multiprocessor," Trans. on computers, Vol. 38, No. 8, Aug., 1989.

[4] Jim Handy, *The Cache Memory Book*, Academic Press, 1993.

[5] A.J. Smith,"Cache Evaluation and the Impact of Workload Choice,"In Proc. Annu. Int. Symp., Computer Architecture, pp. 64-73, 1985.

[6] M. Ajmone Marsan, G. Balbo, G. Conte, *Performance Models of Multiprocessor Systems*, The MIT Press, 1986.

[7] M. Ajmone Marsan, G. Balbo, G. Conte, *Performance Models of Multiprocessor Systems*, The MIT Press, 1986.

[8] A.A.B.Pritskerr, *Simulation with Visual SLAM and AweSim*, Wiley, 1997.

[9] 박경, 한우종, 윤석한,"단일칩 다중처리 마이크로프로세서:Raptor," 한국통신학회학술대회,1997.4.

저 자 소 개

李 興 宰

1968년 2월 27일생.
1998년 한남대학교 전자공학과 졸업(공학사)
2000년 한남대학교 대학원 전자공학과 졸업(공학석사)
2002년 ~ 현재 한남 대학교 대학원 전자공학과 박사과정 재학중
주관심분야 : 네트워크 시뮬레이션, Embedded System 설계

崔 眞 圭

1958년 9월 20일생.
1980년 고려대학교 전자공학과 졸업(공학사)
1982년 고려대학교 대학원 전자공학과 졸업(공학석사)
1987년 고려대학교 대학원 전자공학과 졸업(공학박사)
1987년 9월 - 1990년 8월 대전공업대학 조교수
1999년 - 2000년 미국 Oregon State University 방문교수
1990년 8월 - 현재 한남대학교 정보통신멀티미디어공학부 교수
주관심분야 : 통신망 성능평가, 디지털시스템설계, Embedded System

奇 長 根

1980.3 - 1986.2 고려대학교 공과대학 전자공학과 공학사
1986.3 - 1988.2 고려대학교 대학원 전자공학과 공학석사
1988.3 - 1992.2 고려대학교 대학원 전자공학과 공학박사
2002.6 - 2003.7. University of Arizona, Visiting Professor
1992.3 - 현재 공주대학교 교수
주관심분야 : 컴퓨터 네트워크, 멀티미디어 통신

李 圭 晧

1958년 3월 5일생
1980년: 경북대 전자공학과 공학사
1982년: 경북대 대학원 전자공학과 공학석사
1998년: The University of Gent, Belgium, 정보/컴퓨터공학과 공학박사
1986 - 1988 미국 AIT Inc, 연구원
1983 - 현재 한국전자통신연구원(ETRI) 책임연구원
주관심분야 : 고속인터넷 기술, IP응용기술, 고성능 네트워크프로세서, 고성능 라우터 기술