논문 99-24-12T-9 한국통신학회논문지 '99-12 Vol.24 No.12T

오퍼랜드 참조 요구 대역폭의 개선

정회원 김 홍 준*, 김 창 근*, 김 봉 기*

Bandwidth Improvement of Operand Fetching with The Operand Reference Prediction Cache

Heung-Jun Kim*, Chang-Geun Kim*, Bong-Gi Kim* *Regular Members*

요 약

한 사이클에 요구되는 여러 오퍼랜드 참조의 효율적인 수행은 자료 캐쉬/TLB의 대역폭에 의해 영향을 받는다. 본 논문에서는 자료 캐쉬 및 TLB의 처리 대역폭 증가 대신 오퍼랜드 참조의 예측 특성이 반영된 오퍼랜드 참조 예측 캐쉬를 활용하여 대역폭 요구를 줄이는 방법을 제시하였다. shade를 통해 수집된 5개의 벤치마크 프로그램의 작업부하에 대한 trace-driven 시뮬레이션을 통해 분석되었고 자료 캐쉬 및 TLB에 대한 대역폭이 개선됨을 보였다. 5개의 벤치마크 프로그램에 대해, 512항목의 오퍼랜드 참조 예측 캐쉬는 평균적으로 오퍼랜드 적재 참조의 45%에 대해 정확히 오퍼랜드를 예측하고, 전체 오퍼랜드 참조에 대해 99%의 주소 변환 정보를 제공함으로서 자료 캐쉬와 TLB의 대역폭 요구가 감소함을 보인다.

ABSTRACT

We propose methods to reduce the bandwidth requirement of data cache/TLB in the processor which issues multiple instructions per cycle. To reduce the data cache/TLB bandwidth requirement, we use operand reference prediction cache which predicts operand value and address translation during the instruction fetch stage. Through the trace-driven simulation of five benchmark programs, the performance improvement by this method is analyzed and validated. The operand reference prediction cache with 512 entries shows the prediction accuracy of 90%, the prefetch rate of 43%, and the address translation rate of 99% on average.

I. 서 론

사이클 당 여러 명령어를 수행하는 수퍼스칼라 프로세서 구조에서는 오퍼랜드 참조의 처리 능력을 증가시키기 위하여 TLB 및 캐쉬를 명령어와 자료에 대해 각각 별도로 구성하는 것이 일반적이며, 또한 자료 TLB 및 자료 캐쉬의 구현에서 처리 대역폭을 증가시키는 구조인 멀티 포트 구조와 인터리빙 구조 및 멀티 뱅크 구조가 사용된다[4][13].

이 구조들은 수퍼스칼라 프로세서 구조와 다중 사이클 히트 시간(multi-cycle hit time)을 갖는 캐쉬에서의 처리 대역폭을 개선할 수 있는 효과적인 구조이다. 하지만, 멀티 뱅크 구조는 뱅크 충돌(bank conflict)과 뱅크 수에 따른 연결선에 의한 접근시간(access time)의 증가 문제를 발생한다[5]. 또한, 멀티 포트 구조는 포트의 수에 따라 접근 지연 및 소요 공간이 증가하는 문제가 발생하며[4], 비연속 주소로 참조되는 오퍼랜드의 특성상 인터리빙 구조에 의한 대역폭 개선도 제한적이다. 따라서, 명령어 수준의 병렬성(ILP) 증가에 따른 처리 대역폭을 증가시키기 위해 앞의 구조들을 적용하는 것은

* 진주산업대학교 컴퓨터공학과(thinkthe@cjcc.chinju.ac.kr), (cgkim@cjcc.chinju.ac.kr), (bgkim@cjcc.chinju.ac.kr)
논문번호 : 99025-0910 접수일자 : 1999년 9월 10일

제한이 있으며, 이들 구조를 보다 효율적으로 관리하기 위한 별도의 운영과 구성이 필요하다. 예를 들어 [5]에서는 캐쉬 포트의 효과적 운영 방법을 제안하였고, [4]에서는 뱅크 충돌에 의한 영향을 감소시키는 운영 방법을 제시하였다. 이들은 대역폭을 증가시키는 구조의 효율성을 높이기 위한 보완적 운영 구조이며, 이와 같이 오퍼랜드 참조의 처리 대역폭을 증가시키는 것은 제약성이 뒤따른다.

만약 오퍼랜드 참조의 특성을 활용하여 TLB나 캐쉬에 대한 요청 비율을 감소시킬 수 있다면, 처리 대역폭을 증가시키는 구조와 동일한 효과를 얻을 수 있다.

본 논문에서는 한 사이클 내에 다중 명령어를 발생하는 프로세서에서, 자료 캐쉬 및 자료 TLB의 처리 대역폭을 증가시키는 방법이 아니라 오퍼랜드 참조 요구 대역폭을 감소시킴으로서 자료 캐쉬 및 TLB의 대역폭을 개선하기 위해 오퍼랜드 예측 방법이 적용된 오퍼랜드 참조 캐쉬[1][3]를 활용하는 구성을 제시하고, 시뮬레이션을 통해 자료 캐쉬 및 TLB의 대역폭 개선에 의한 성능 기여도를 분석한다.

Ⅱ. 오퍼랜드 참조 예측 캐쉬

예측 기법은 다음 순서에 참조되리라 예상되는 명령어 또는 자료 및 그 정보가 저장된 주소를 미리 예측하여 메모리 참조를 보다 신속히 처리할 수 있는 방법으로, 최근에 오퍼랜드의 값 또는 주소를 예측하여 오퍼랜드 참조의 시간을 감소하려는 LTB (load target buffer)[6], 적재 값 예측(load value prediction) 기법[8], 오퍼랜드 선취 캐쉬(OPC) 기법[7]들이 제안되었다. 하지만, 이들은 오퍼랜드 참조 시간의 감소에 주력하였고, 예측의 검증을 위해 실제 오퍼랜드를 참조함으로서 자료 TLB/캐쉬에 대한 대역폭 감소에는 크게 영향을 주지 못하였다.

오퍼랜드 참조 예측 캐쉬[1]는 예측 기법을 사용하여 오퍼랜드 참조 처리과정의 일부분을 하이딩(hiding)하고 오퍼랜드의 선취 시점을 앞당김으로서 자료 캐쉬 및 TLB에 대한 요구 대역폭을 감소시킴으로서 사이클당 오퍼랜드 처리 수를 증가시킴으로서 대역폭 개선의 효과를 얻을 수 있다.

오퍼랜드 참조 예측 캐쉬는 그림 1과 같은 항목 구성을 가지며, 각 오퍼랜드에 대해 가상 주소와 함께 변환된 물리 주소가 추가되어, TLB 대신에 주소 변환 정보의 제공이 가능하도록 구성하고 있다. 따라서, 오퍼랜드 참조 예측 캐쉬는 명령어 주소에 의한 오퍼랜드의 예측이 실패한 경우에는 그 명령어 주소에 의한 신속한 주소 변환이 가능하다. 만약 명령어 주소에 의한 주소 변환도 실패한 경우 계산된 유효주소로 오퍼랜드 참조 예측 캐쉬를 조사하여 주소 변환이 가능하도록 운영함으로서, 오퍼랜드 참조 예측 캐쉬의 기능은 자료 TLB의 대체 기능을 할 수 있다.

명령어 주소	오퍼랜드 주소(가상)	오퍼랜드 주소(물리)	오퍼랜드 값	유효비트	카운터 비트	LRU 비트

그림 1. 오퍼랜드 참조 예측 캐쉬의 항목 구성

동작 과정은 그림 2와 같고, 오퍼랜드 참조 예측

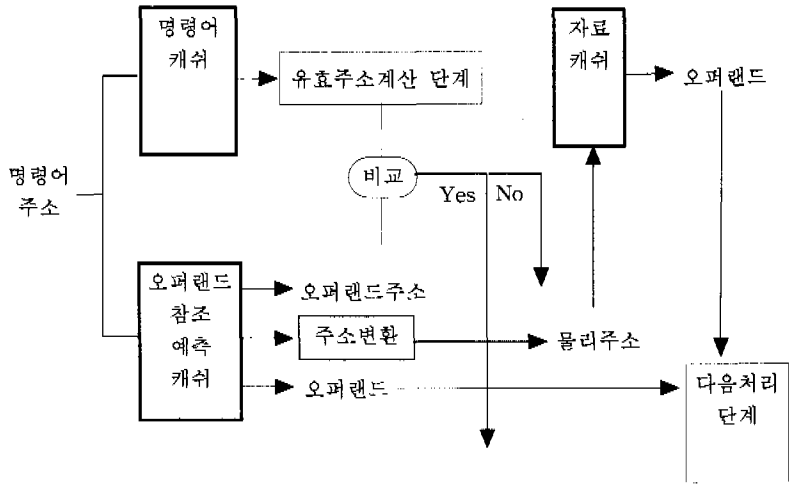

그림 2. 오퍼랜드 참조 예측 캐쉬의 동작

캐쉬에서 적재 명령어의 오퍼랜드 참조 동작은 다음과 같이 요약된다.

단계 1) 명령어 페치 단계와 동시에 오퍼랜드 참조 예측 캐쉬를 참조하여, 명령어 주소에 해당하는 항목으로부터 오퍼랜드가 유효하면 오퍼랜드와 오퍼랜드 가상주소 및 물리주소를 페치한다.

단계 2) 페치 된 명령어로부터 계산된 오퍼랜드 주소를 오퍼랜드 참조 예측 캐쉬의 오퍼랜드 주소(가상)와 비교하여 일치하면 예측된 오퍼랜드를 이용하여 다음 단계의 작업을 수행한다. 만약 일치하지 않으면, 계산된 오퍼랜드 주소와 오퍼랜드 참조 예측 캐쉬로부터의 가상 및 물리주소를 이용하여 주소를 변환한다.

단계 3) 변환된 주소로 자료 캐쉬로부터 오퍼랜드를 페치 한다.

앞의 방법에 의한 주소 변환이 불가능한 경우에는 계산된 유효 주소로 오퍼랜드 참조 예측 캐쉬(가상주소)를 참조하여 주소 변환을 실시한다.

Ⅲ. 작업 부하 및 결과 분석

오퍼랜드 참조 예측 캐쉬에 의한 자료 캐쉬 및 TLB로의 대역폭 영향을 분석하기 위해, C++로 작성한 객체 지향 시뮬레이터를 이용하여 trace-driven 시뮬레이션을 수행하였다. 시뮬레이션에 사용된 메모리 참조 주소는 SUN사에서 제공하는 shade를 사용하여 trace 하였다[12][11][10].

작업부하의 생성을 위해 사용된 벤치마크 프로그램은 동적 메모리 할당을 중심으로 c 언어로 작성한 benchmark 프로그램[9] 중 espresso, p2c, make 및 gs와 유닉스 운영체제에서 사용 빈도가 높은 응용 프로그램 ls의 실제 수행을 통해 얻었으며, 각각의 프로그램 실행 시에 수행되는 명령어 중에서 천만개씩을 수집하였다[2]. 각 벤치마크의 프로그램 특성[9]과 수행 특성은 표 1 같다.

시뮬레이션 결과에서 TLB의 경우 주소 변환 성공률을 중심으로, 자료캐쉬의 경우 IPC(Instruction Per Cycles)를 중심으로 분석한다. 자료 캐쉬의 영향 분석을 위한 시뮬레이션은 오퍼랜드 참조 예측 캐쉬의 항목이 512인 경우로 고정하여 수행하였다.

오퍼랜드 참조 예측 캐쉬는 자료 캐쉬의 요구를 감소시켜 오퍼랜드 참조 지연을 줄이고, TLB의 요청 대역폭을 현저하게 감소시키는 구조이다. 예로서 espresso 응용 프로그램의 시뮬레이션 결과인 그림 3에서 오퍼랜드 참조 예측 캐쉬의 항목 수에 따른 오퍼랜드 값 및 주소 변환(명령어 주소 및 오퍼랜드 주소에 의한)에 대한 예측의 정확도를 보여준다. 512 항목의 경우에는 예측에 의해 55.9%의 오퍼랜드 참조에 대해 참조 지연과 자료 캐쉬의 참조를 줄일 수 있고, 99.7%까지 주소 변환을 수행하는 것이 가능하다.

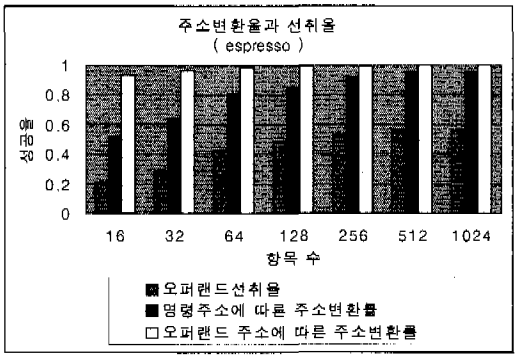

그림 3. erspresso에 대한 예측 정확도

표 1. 벤치마크 프로그램과 수행 특성

벤치마크 프로그램	작업 정도	자료 페이지 수	적재 명령	저장 명령	프로그램 특성
espresso	z5xp1	139	0.218	0.037	PLA 논리 최적화 프로그램
gs	small	241	0.190	0.060	포스트 스크립트 페이지-묘사 언어에 대한 해석 수행 프로그램
make	perl	212	0.205	0.030	GNU make 프로그램
p2c	grade	137	0.212	0.039	Pascal 원시 프로그램을 C 원시 프로그램으로 변경
compress	compress	207	0.217	0.098	자료 압축 프로그램
ls	ls -lR	154	0.135	0.045	디렉토리 나열 프로그램
평 균		181.667	0.196	0.052	

3.1 TLB 대역폭에 대한 영향

표 2는 5개의 벤치마크 프로그램에 대해 시뮬레이션을 통해 얻어진 오퍼랜드 참조 예측 캐쉬에서 응용 프로그램 별 항목 수 변화에 의한 주소변환 성공률을 보인다. 동일한 적재 명령어가 동일한 페이지내의 오퍼랜드 주소를 참조하는 경우에는 주소 변환 정보를 제공하여 신속한 주소 변환을 수행할 수 있다. 적재 명령어의 주소를 인덱스로 하여 오퍼랜드 참조 예측 캐쉬로부터 오퍼랜드 주소 변환이 가능한 경우는 512항목에서 평균 90%이다. 이 중에서 오퍼랜드 값 예측이 성공한 경우에는 주소 변환이 필요 없으므로 이 부분을 뺀 나머지 부분만큼 40%(90%-50%)의 오퍼랜드 참조에 대해 직접 주소 변환이 가능하다. 오퍼랜드 참조 예측 캐쉬 구조에서 오퍼랜드의 계산된 유효 주소를 인덱스로 하여 오퍼랜드 참조 예측 캐쉬로부터 오퍼랜드 주소 변환 정보를 얻을 수 있는 비율은 512항목인 경우에는 주소 변환 비율이 99%이므로, 오퍼랜드의 예측이 실패하더라도 오퍼랜드 참조 예측 캐쉬는 TLB의 기능을 대신할 수 있다. 또한 총 주소 변환 율에 있어서 512 항목 이상에서 99%를 보이며, 따라서 TLB의 대역폭 요청을 최대 99%까지 감소시킴을 의미한다.

표 2. 오퍼랜드 참조 예측 캐쉬의 주소 변환 성공률

항목수 \ 이름		espresso	gs	ls	make	p2c	평균
1024	a	0.930	0.895	0.888	0.940	0.938	0.91
	b	0.998	0.992	0.990	0.998	0.995	0.99
512	a	0.930	0.887	0.832	0.933	0.930	0.90
	b	0.997	0.989	0.985	0.996	0.994	0.99
256	a	0.901	0.857	0.781	0.902	0.897	0.86
	b	0.995	0.985	0.974	0.994	0.992	0.98
128	a	0.840	0.798	0.735	0.839	0.869	0.81
	b	0.991	0.977	0.963	0.991	0.988	0.98
64	a	0.795	0.654	0.617	0.803	0.869	0.74
	b	0.985	0.960	0.941	0.987	0.988	0.97
32	a	0.623	0.440	0.532	0.724	0.775	0.61
	b	0.966	0.923	0.915	0.975	0.960	0.94

a : 명령 주소에 의한 주소 변환률

b : 오퍼랜드 주소에 의한 주소 변환률

3.2 자료 캐쉬 대역폭 영향 분석

오퍼랜드 참조 예측 캐쉬에 의해 자료 캐쉬의 대역폭 영향을 분석하는 시뮬레이션에서, 자료 캐쉬로의 요청은 모두 히트하는 경우로 가정하며. 오퍼랜드 참조 예측 캐쉬에는 대기 큐(waiting queue)는 없는 것으로 가정한다. 시뮬레이션의 운영 환경 변화는 표 3과 같고, 오퍼랜드 처리를 위한 구성에서 자료캐쉬만으로 구성되는 경우를 단일 구성으로, 오퍼랜드 참조 예측 캐쉬와 함께 구성된 경우를 혼합 구성으로 구분한다.

표 3. 시뮬레이션 운영 환경

명령어 발생 수	오퍼랜드 처리 구성	처리 사이클 수
1개	단일구성	자료 캐쉬 : 2 cycle
	혼합구성	오퍼랜드 캐쉬 : 1 cycle 자료캐쉬 : 2 cycle
2, 3개	단일구성	자료캐쉬 : 1 cycle
	혼합구성	오퍼랜드 캐쉬 : 1 cycle 자료캐쉬 : 1 cycle

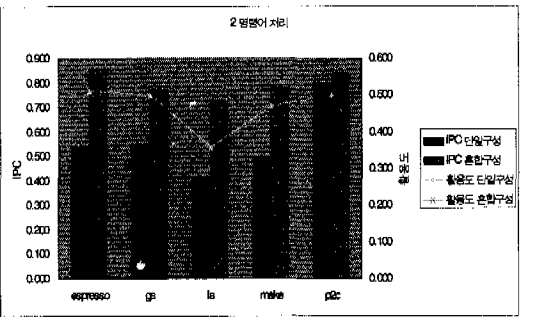

그림 4. 2 명령어 동시 발생/처리시 성능 효과

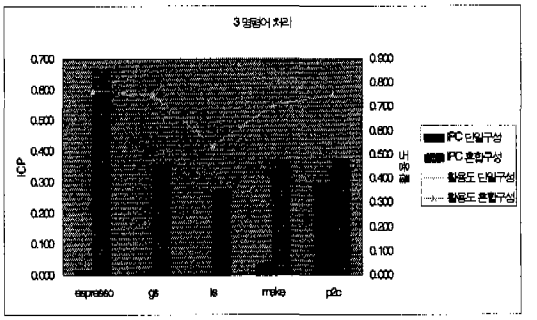

그림 5. 3 명령어 동시 발생/처리시 성능 효과

그림 4과 그림 5는 매 사이클당 명령어를 2개 및 3개씩 발생(issue)하는 경우에 자료 캐쉬만을 갖는 단일구성과 오퍼랜드 참조 예측 캐쉬 및 자료 캐쉬로 구성되는 혼합구성에서 자료캐쉬의 활용도와 사이클당 처리 명령어수인 IPC(Instructions Per Cycle)를 보인다. 각 사이클에 2개의 명령을 동시에 발생하는 경우 혼합구성은 오퍼랜드 참조 캐쉬가

오퍼랜드를 직접 제공하게 되어 자료 캐쉬의 평균 활용도를 50%에서 31%로 감소시키고, IPC는 평균 0.6에서 0.8로 1.33배 증가시킨다. 또한 3개 명령 발생 시에도 혼합구성은 평균 활용도를 74%에서 47%로 감소시키고, IPC는 0.1에서 0.47로 증가된다.

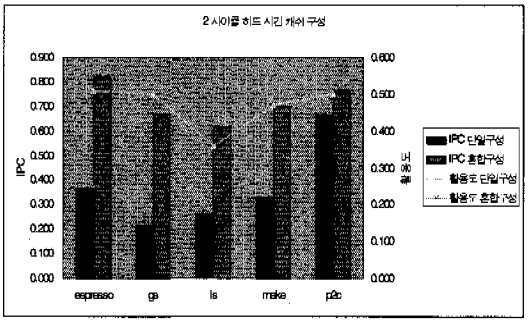

그림 6. 2사이클 히트 시간을 갖는 캐쉬

그림 6은 매 사이클당 1개의 명령어를 발생하는 경우에 자료 캐쉬의 히트 처리시간이 2 사이클 소요되는 단일구성과 혼합구성의 시뮬레이션 결과이다. 오퍼랜드 참조 예측 캐쉬와 함께 동작하는 혼합구성의 경우 단일구성에 비해 자료 캐쉬의 평균 활용도를 47%에서 27%로 감소시키고, IPC는 0.37에서 0.71로 1.92배 증가시킨다.

이 결과는 오퍼랜드 참조 예측 캐쉬 구조가 사이클 당 다중 명령어를 처리해야 하는 프로세서 구조에서 자료 캐쉬의 요청 대역폭을 크게 감소시켜 효율성을 증대시키는 구조임을 제시한다. 또한 오퍼랜드 참조 예측 캐쉬 구조는 2 사이클 히트 타임을 갖는 캐쉬 구성에서도 성능 효율이 증가됨을 알 수 있다.

Ⅳ. 결 론

사이클 당 여러 명령어를 수행하는 프로세서 구조에서 참조 요청의 단순 평균에 의한 처리 대역폭 결정보다는 오퍼랜드 참조 특성에 따라 참조 요청 수를 줄이거나, 참조의 경로를 다양화하여 요청 대역폭을 증가시키는 것이 구현 측면에서 더 효율적이다.

본 논문은 한 사이클에 요구되는 여러 오퍼랜드 참조를 효율적으로 수행하기 위해 멀티 포트 등의 구조를 이용한 자료 캐쉬 및 자료 TLB의 처리 대역폭 증가가 아닌, 오퍼랜드의 예측 특성을 이용하는 오퍼랜드 참조 예측 캐쉬를 활용하여 요구 대역폭을 감소시키는 방법을 제안하였다.

예측 기법을 사용한 오퍼랜드 참조 예측 캐쉬를 활용하는 구성의 trace-driven 시뮬레이션을 수행하기 위해 작업 부하의 추적은 shade의 여러 함수들을 이용해 5개 프로그램에 대해 천만개씩을 수집하였고, 결과로서 TLB의 주소 변환 예측의 정확도와 자료 캐쉬의 IPC 제시하고 TLB 및 캐쉬에 대한 성능 기여도를 분석하였으며, 오퍼랜드 예측 특성을 이용하는 오퍼랜드 참조 예측 캐쉬의 활용을 통해 자료 캐쉬 및 TLB에 대해 대역폭 요구를 감소 시킬 수 있음을 보였다.

참 고 문 헌

[1] 김흥준, 조경산 "오퍼랜드 참조 예측 캐쉬(ORPC)를 활용한 오퍼랜드 페치의 성능 개선," 정보처리학회 논문지, 제5권 제6호, pp. 1652-1659, 1998.

[2] 김흥준, 조경산 "오퍼랜드 참조 특성에 관한 연구", 정보처리학회 추계 학술발표논문집, 제5권, 제2호, pp. 1211-1214, 1998.

[3] 김흥준, 조경산 "오퍼랜드 참조 지연 감소와 대역폭 개선을 위한 예측 기반의 오퍼랜드 참조 캐쉬", 단국대학교 박사학위 청구 논문, 1998.

[4] Todd M. Austin, Gurindar S. Sohi, "Hardware High-Bandwidth Address Translation for Multiple-Issue Processors," Proc. of ISCA, pp. 158 - 167, 1996.

[5] Kenneth M. Wilson, Kule Olukotun, "Designing High Bandwidth On-Chip Caches," Proc. of ISCA, pp. 121 - 132, 1997.

[6] M. Golden and Trevor Mudge, Hardware Support for Hiding Cache Latency, CSE-TR-152-93, available at www.cs.umich. edu, U. of Michigan, 1993.

[7] L. Widigen, E. Sowadsky and K. McGrath, "Eliminating Operand Read Latency," Computer Architecture News, Vol. 24, No. 5, pp. 18 - 22, 1996.

[8] M. Lipasti and et al., "Value Locality and Load Value Prediction," Proc. of ASPLOS, pp. 138 - 147, 1996.

[9] Benjamin Zorn and Dirk Grunwald, "Evaluating Models of Memory Allocation," ACM

Transsactions on Modeling and Computer Simulation, Vol. 4, No. 1, pp. 107 - 137, 1994.
[10] David L. Weaver and Tom Germond, The SPARC Architecture Manual, Prentice Hall, 1994.
[11] Sun microsystem, UltraSparc user's manual, Sun microsystem, 1997, avaliable at http://www.sun.com.
[12] Bob Cmelik and David Keppel, Shade: A Fast Instruction-Set Simulator for Execution Profiling, Sigmetrics, ACM, pp. 128 - 137, 1994.
[13] Harvey G. Cragon, Memory Systems and Pipelined Processors, Jones and Bartlett Publishers, 1996.

김 흥 준(Heung-Jun Kim) 정회원

1989년 : 단국대학교 전자계산과 (이학사)
1993년 : 단국대학교 전산통계과 (이학석사)
1999년 : 단국대학교 전산통계과 (이학박사)

1999년~현재 : 진주 산업 대학교 컴퓨터공학과 전임강사
<주관심 분야> 컴퓨터 구성 및 성능 평가, 시뮬레이션

김 창 근(Chang-Geun Kim) 정회원

1985년 : 경상대학교 전산통계과 (이학사)
1991년 : 경남대학교 컴퓨터공학과(공학석사)
1999년 : 경남대학교 컴퓨터공학과(공학박사)

1995년~현재 : 진주 산업 대학교 컴퓨터공학과 조교수
<주관심 분야> 멀티미디어 통신, VOD, 정보통신 시스템

김 봉 기(Bong-Gi Kim) 정회원

1987년 : 숭실대학교 전자계산학과(공학사)
1989년 : 숭실대학교 전자계산학과(공학석사)
1999년 : 숭실대학교 전자계산학과(공학박사)

1994년~1998년 : 한림정보산업대학 컴퓨터응용과 교수
1999년~현재 : 국립진주산업대학교 컴퓨터공학과 교수
<주관심 분야> 멀티미디어 데이터베이스, 웹 데이터베이스, 내용기반검색