Journal of The Korea Society of Computer and Information
Vol. 20 No. 9, pp. 1-10, September 2015

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.9.001

Variable latency L1 data cache architecture design in multi-core processor under process variation

Joonho Kong *

Abstract

In this paper, we propose a new variable latency L1 data cache architecture for multi-core processors. Our proposed architecture extends the traditional variable latency cache to be geared toward the multi-core processors. We added a specialized data structure for recording the latency of the L1 data cache. Depending on the added latency to the L1 data cache, the value stored to the data structure is determined. It also tracks the remaining cycles of the L1 data cache which notifies data arrival to the reservation station in the core. As in the variable latency cache of the single-core architecture, our proposed architecture flexibly extends the cache access cycles considering process variation. The proposed cache architecture can reduce yield losses incurred by L1 cache access time failures to nearly 0%. Moreover, we quantitatively evaluate performance, power, energy consumption, power-delay product, and energy-delay product when increasing the number of cache access cycles.

▸ Keyword : Process variation, Variable latency cache architecture, Cache access time failure, Processor yield loss, Multi-core architecture

I. Introduction

공정 기술이 발전함에 따라서, 소자의 크기가 작아지고 이에 따라 반도체의 성능 및 에너지 효율성도 향상되고 있다. 공정 스케일링 (process scaling) 은 무어의 법칙 (Moore's law)을 유지시켜온 요소들 중 가장 중요한 한 가지이다. 작아진 소자의 크기는 정해진 공간 안에 더 많은 소자들을 집적할 수 있게 하였고 전력 소모량도 줄였다. 결과적으로, 이는 반도체의 성능과 에너지 효율성을 향상시켰다.

그러나, 공정 기술 축소 (process technology shrink)는 공정 변이 (process variation)의 심화라는 역효과를 가져왔다. 공정 변이는 반도체 제조 시 의도치 않은 작은 입자들이 소자에 영향을 주거나, 혹은 소자들의 파라미터에 영향을 주어 각 소자들의 파라미터가 불균등하게 분포되도록 한다. 이러한 불균등 분포는 칩의 로직 혹은 메모리 소자에 영향을 주어 스위칭 속도가 느려지거나, 의도치 않게 많은 누수 전력 소모를 일으킬 수 있다.

공정 변이는 마이크로프로세서 혹은 CPU의 설계에도 지대한 영향을 주고 있다. 파이프라인 구조로 설계된 프로세서의 경우, 파이프라인 임계 경로 (critical path) 지연 시간이 특정 클럭 사이클 시간 (clock cycle time)을 만족시키지 못할 경우, 프로세서가 정해진 사양 (specification)을 만족시키지 못하여 제조된 프로세서 칩을 버리게 되는 상황이 발생한다. 이를 수율 손실 (yield loss) 이라고 하고, 수율 손실이 많아질수록 프로세서 설계/제조 회사의 이익 감소하거나 손실이 커지게 된다.

프로세서 안에서는 L1 캐쉬 메모리 (cache memory)가 특히 공정 변이에 취약한 것으로 알려져 있는데, 그 이유는 캐쉬 메모리를 구성하는 메모리 소자가 6T 정적 임의 접근 메모리 (Static Random Access Memory: SRAM)로 이루어져 있기 때문이다. 6T SRAM 셀들은 공정 변이로 인해 여러 가지 결함

• First Author: Joonho Kong, • Corresponding Author: Joonho Kong
*Joonho Kong (joonho.kong@knu.ac.kr), School of Electronics Engineering, Kyungpook National University
• Received: 2015. 06. 15, Revised: 2015. 07. 05, Accepted: 2015. 08. 24.

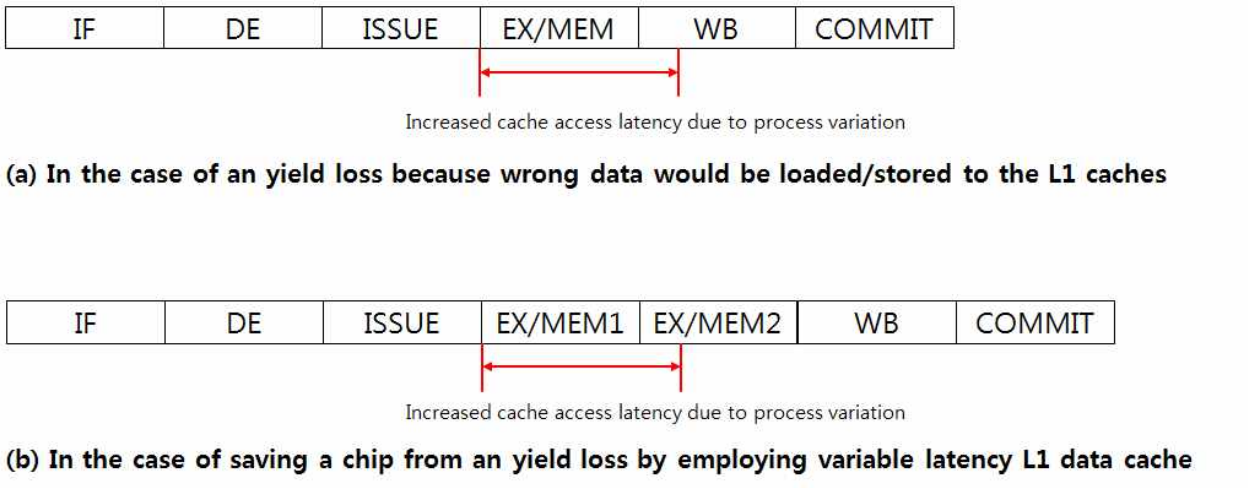

Fig. 1. Comparison of processor pipeline stages between the cases with and without variable latency cache

이 야기될 수 있는데, 가장 대표적으로 접근 시간 실패 (access time failure)가 있다. 접근 시간 실패는 캐쉬 메모리 접근 시 캐쉬 접근 시간이 설계 시 정해진 시간 이상으로 소모되는 것을 의미한다. 이는 파이프라인 구조로 설계된 프로세서에서 셋업 및 홀드 시간 위반 (setup and hold time violation) 으로 인해 잘못된 데이터를 전달하거나, 이를 피하기 위해서 어쩔 수 없이 클럭 주파수를 낮춰서 동작해야하는 문제점을 야기한다. 위에서 설명한 것처럼, 클럭 주파수를 낮췄을 때 정해진 사양을 만족시키지 못할 경우 프로세서의 수율 손실이 발생하게 된다.

이를 해결하기 위해서 가변 접근 시간 캐쉬 (variable latency cache) 구조가 소개되었다 [1][2][3]. 가변 접근 시간 캐쉬는 캐쉬 메모리의 접근 클럭 사이클 수를 가변적으로 사용할 수 있는 구조로써, L1 캐쉬 메모리의 6T SRAM 셀들이 공정 변이에 영향을 받아서 접근 시간이 늘어났을 경우, 캐쉬 메모리의 접근 사이클을 가변적으로 사용함으로써 프로세서의 클럭 주파수를 낮추지 않고도 프로세서 사용이 가능하다.

그러나, 이러한 가변 접근 시간 캐쉬 구조는 캐쉬 메모리 접근 시 클럭 사이클 숫자가 늘어남으로써, 성능 저하 및 에너지 소모 증가의 역효과를 가져올 수 있다. 특히, 메모리 접근 명령어 (load/store instructions)가 많이 분포하는 프로그램을 수행하였을 경우, 이러한 역효과가 더 심화되는 현상이 있다.

이전에 발표되었던 논문들은 가변 접근 시간 캐쉬 구조를 수율 [1] 혹은 성능 [2][3] 측면에서만 평가하였다. 특히, [2]와 [3]에서는 프로그램의 특성에 따른 분석보다는 성능의 정량적인 평가에만 초점이 맞추어져있었다. 또한, 다중코어 프로세서 구조가 널리 쓰이는 현 시점에서 다중 코어 프로세서 구조에 맞는 가변 접근 시간 캐쉬 구조가 아직 소개되지 않았다.

본 논문에서는 다중 코어 프로세서 구조에서의 가변 접근 시간 L1 데이터 캐쉬 구조 설계를 제안하고, 이를 성능 및 에너지 측면 평가하고 분석한다. 특히, 다수 쓰레드 프로그램 (multi-threaded program)을 수행하였을 때, 성능 및 에너지를 평가함으로써 최신 프로세서 구조에서의 가변 접근 시간 L1 데이터 캐쉬 구조의 효율성에 대해서 평가한다. 성능 측면 평가에서는 프로그램의 특성을 나타낼 수 있는 단위인 MPKI (Misses Per Kilo Instructions) 및 캐쉬 접근 빈도를 활용하여 프로그램 특성에 따른 성능 평가 및 분석을 수행하였다. 또한, 전력-시간 곱 및 에너지-시간 곱의 측정을 통해 전력-시간 및 에너지-시간 간의 균형 (trade-off)도 살펴본다.

II. Related Work

공정 변이를 고려한 아키텍쳐 수준의 프로세서 설계는 이제까지 많은 연구들이 이루어져왔다. [4]와 [5]에서는 공정 변이 하에서 캐쉬 메모리 셀 (SRAM)에서 발생할 수 있는 결함에 대해서 분석하고 추가적인 셀들을 이용하여 공정 변이에 강한 캐쉬 메모리 구조를 소개하였다. 만약 결함이 있는 캐쉬 메모리 셀들에 접근하는 경우에는 재배치표 (remapping table)를 통해 추가적인 셀들에 접근할 수 있게 하여, 프로세서의 수율을 향상시켰다. [1]에서는 수율을 고려한 캐쉬 구조를 소개하였다. YAPD (Yield-aware power down)의 경우에는 과도한 전력 누수로 인한 수율 손실을 방지하고, VACA (Variable latency cache architecture)는 접근 시간 실패로 인한 수율 손실을 방지하여 캐쉬 메모리와 프로세서의 수율을 높인다. [3]에서도 역시 프로세서의 수율을 높이기 위해서 가변 접근 시간 캐쉬 메모리와 워드라인 전압을 높이는 방법을 사용하였다. [6]에서는 캐쉬 접근 시간 결함이 있는 캐쉬 라인에만 선택적으로 높은 전압을 인가하는 기법이 소개되었다. [7]에서는 [6]의 기법을 확장하여 전력 누수로 인한 수율 손실 또한 줄이기 위해서 SRAM 셀들에 낮은 전압을 인가하는 동시에 워드라인에는 높은 전압을 선택적으로 인가하는 기법이 소개되었다. [8]에서는 비슷한 캐쉬 접근 시간을 가지는 캐쉬 블록끼리 같은 그룹을 구성하여 공정 변이 하에 늘어난 캐쉬 접근 시간으로 인한 성

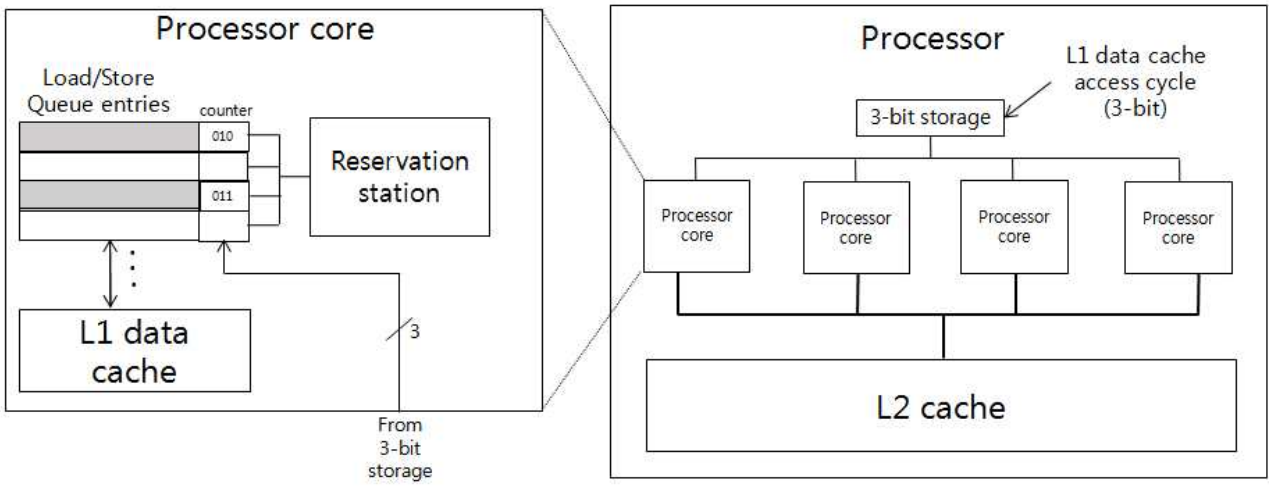

Fig. 2. Proposed architecture for variable latency L1 data cache in multi-core processors

능 저하를 최소화하는 기법이 소개되었다. 최근에는 3D 프로세서에도 공정 변이로 인한 캐쉬 메모리의 결함 문제를 해결하기 위해서 여러 가지 기법이 소개되었다. [9]에서는 캐쉬 안에서의 임계 경로를 줄이기 위한 CLAPS (Cross LAyer Path Splitting) 기법이 소개되었다. 임계 경로의 지연 시간을 줄임으로써 캐쉬 접근 시간 실패를 줄일 수 있다. [10]에서는 내로우 밸류 (narrow value)의 특성을 이용한 3D last-level 캐쉬 구조가 소개되었다. [11]에서는 [10]의 연구를 확장하여 에너지 소모량을 더욱 더 최적화한 3D last-level 캐쉬 구조가 소개되었다.

아키텍쳐 수준의 기법들 이외에도 회로 수준 기법들도 소개되었는데 가장 대표적인 것은 ABB (Adaptive Body Biasing) 기법이다 [12][13][14]. ABB는 소자의 임계 전압 (threshold voltage)를 조정할 수 있는 기법으로써, 접근 시간 실패의 경우 접근 시간을 줄이기 위해 임계 전압을 낮출 수도 있고, 전력 누수가 심한 경우 임계 전압을 높여 전력 누수를 높일 수 있는 기법이다. ASV (Adaptive Supply Voltage)기법 [15] 도 소개되었는데, 공급 전압을 높임으로써 접근 시간 실패를 줄일 수도 있고, 공급 전압을 낮춤으로써 전력 누수를 줄일 수도 있다.

본 장에서 소개한 관련 연구들은 마이크로프로세서에서의 공정 변이 문제를 해결하기 위한 연구들이다. 그러나, 단일 코어 (single-core) 구조에서의 가변 접근 시간 캐쉬 구조만 소개되었고, 다중 코어 구조에서의 가변 접근 시간 캐쉬 구조는 소개된 바가 없다. 본 논문에서는 다중 코어 구조에서의 가변 접근 시간 캐쉬 구조를 제안하고 이를 성능, 에너지 및 전력 소모 측면에서 평가한다.

국내 연구 동향으로는 고성능 L1 캐쉬 메모리를 위한 직접 사상 캐쉬 구조, 연관 사상 버퍼 및 선택 테이블로 구성된 캐쉬 메모리 구조가 제안되었다 [16]. 또한, 저전력 명령어 캐쉬 메모리 설계 기법 [17] 및 저전력 필터 캐쉬 설계 기법 [18] 이 제안되었다. 그러나, 위 기법들은 공정 변이를 고려하지 않았다.

III. Variable Latency L1 Data Cache Architecture for Multi-core Processors

가변 접근 시간 캐쉬 구조의 경우, 캐쉬 메모리를 하나의 비동기적 구조로 취급한다. 기존 캐쉬 구조의 경우, 캐쉬 메모리를 접근할 때, 프로세서의 사양에 명시된 클럭 사이클이 소모된다. 그러나, 가변 접근 시간 캐쉬 구조에서는 프로세서 칩마다 다양한 클럭 사이클이 소모될 수 있다. 이를 통해 공정 변이가 프로세서의 클럭 주파수에 주는 악영향을 최소화 함으로써 수율 손실을 막을 수 있다.

그림 1은 가변 접근 시간 캐쉬의 기본적인 파이프라인 구조를 나타낸다. 그림 1 (a) 의 경우, 공정변이로 인해 늘어난 캐쉬 접근 시간이 정해진 클럭 사이클 시간을 만족시키지 못할 경우, 프로세서 사양에 명시된 클럭 주파수를 만족시키지 못하고, 이는 결국 수율 손실로 이어지게 된다. 반면, 그림 1 (b) 의 경우, 캐쉬 메모리에 접근하는 클럭 사이클 숫자를 늘리는 방식을 취한다. 이렇게 함으로써 프로세서의 사양에 명시된 클럭 주파수를 만족시키면서 정상적인 캐쉬 메모리의 동작을 보장할 수 있다. 이는 결국 프로세서의 수율 향상으로 이어진다.

본 논문에서는 다중 코어 구조에서의 가변 접근 시간 L1 데이터 캐쉬 구조를 제안한다. 다중 코어 구조에서의 가변 접근 시간 캐쉬 구조는 단일 코어 구조에서의 가변 접근 시간 캐쉬 구조를 확장한 형태가 된다. 그러나, 다중 코어의 경우 각 코어마다 L1 데이터 캐쉬 메모리가 존재한다. 따라서, 코어 별로 각기 다른 접근 시간을 사용할 수도 있고, 동일한 접근 시간을 사용할 수도 있다. 본 논문에서는 설계의 단순성을 위해 동일한 접근 시간을 사용하는 캐쉬 구조를 제안한다. 코어 별로 다른 접근 시간을 사용하게 되면 성능을 약간 더 높일 수 있지만, 코어 마다 각기 다른 L1 데이터 캐쉬 접근 시간을 지원하기 위해서 하드웨어 자원이 더 필요하다.

그림 2는 다중 코어 구조에서의 가변 접근 시간 L1 데이터

캐쉬 구조를 지원하기 위해 각 코어 내부의 추가적인 로직을 나타낸 것이다. Out-of-order 수행을 지원하기 위해 캐쉬 메모리로부터 읽어온 데이터 (load 명령어를 사용하여)를 바로 reservation station 등에 포워딩하기 위해 load/store queue (LSQ) 엔트리에 계수기(counter)를 추가하였다. 계수기는 LSQ로부터 load 명령어로 인해 L1 데이터 캐쉬를 접근을 시작할 때 해당 프로세서의 L1 데이터 캐쉬 접근 시간 (그림 2에서 3-bit storage로 표기된 부분)으로 초기화되며 클럭의 상승 모서리 (positive edge) 에 1씩 감소한다. LSQ에서는 엔트리의 계수기 값이 0이 되고 L1 데이터 캐쉬에서 캐쉬 hit이 발생하면, L1 데이터 캐쉬로부터 데이터가 도착한 것으로 인식하고 데이터를 해당 reservation station (혹은 물리적 레지스터 파일) 에 전달한다. 만약 L1 데이터 캐쉬 miss가 발생하면 counter 값은 무시된다.

Table 1. Evaluation parameters

Category	Value
CPU architecture	Intel Atom [19]
The number of CPU core	8
Process node	22nm
Clock frequency	2.4GHz
L1 data cache size	24KB
L1 instruction cache size	32KB
L1 data cache access cycle	4 clock cycle
L2 cache size	1MB * 4 = 4MB

예를 들어, L1 데이터 캐쉬 접근 시간이 4 사이클일 경우에, L1 데이터 캐쉬에 접근 시 해당 LSQ 엔트리의 계수기 값은 4로 초기화되며 클럭 사이클이 지나갈 때 마다 1씩 감소하여 캐쉬 hit시에 데이터가 LSQ에 도착하는 사이클에는 LSQ 엔트리의 계수기 값이 0이 된다. Reservation station에서는 해당 LSQ 엔트리를 모니터링하고 있다가 캐쉬 hit이 발생하고 계수기 값이 0이 되면 해당 데이터가 준비 (ready) 상태가 된 것으로 간주하게 되어 LSQ로부터 해당 데이터를 읽어오게 된다.

그림 2에 나오는 L1 데이터 캐쉬 접근 사이클 (3-bit) 은 코어 간 공유되는 저장 공간으로서 프로세서가 제조되고 나서 L1 캐쉬 접근 사이클 테스트를 통해 정해진다. 제안하는 구조는 모든 코어가 같은 캐쉬 접근 사이클을 사용하므로 코어들 중에 가장 높은 L1 데이터 캐쉬 접근 사이클이 모든 코어의 L1 데이터 캐쉬 접근 사이클이 된다. 이 저장 공간에 저장된 L1 데이터 캐쉬 접근 사이클 정보는 LSQ에서 L1 데이터 캐쉬를 접근할 때 해당 엔트리의 계수기 값을 초기화하기 위해 사용된다.

본 논문에서는 L1 데이터 캐쉬 접근 사이클을 저장하는 공간을 3-bit으로 소개하였지만, 설계에 따라 저장 공간의 크기를 늘릴 수도 있고, 줄일 수도 있다. 늘리게 되면 약간의 하드웨어 자원이 더 필요하지만, 공정 변이로 인해 캐쉬 접근 시간이 많이 늘어나는 프로세서 칩이 있을 경우 이를 수율 손실로부터 보호할 수 있는 장점이 있다. 본 논문에서 제안하는 다중 코어 프로세서에서의 가변 접근 시간 L1 데이터 캐쉬 메모리 구조는 L1 데이터 캐쉬 접근 사이클을 저장하는 저장 공간만 충분하다면 L1 캐쉬 메모리에서 발생하는 캐쉬 접근 시간 실패로 인한 수율 손실을 거의 0%로 낮추는 것이 가능하다.

IV. Evaluation Framework

본 논문에서 제안하는 다중 코어 구조에서의 가변 접근 시간 L1 데이터 캐쉬 구조는 프로세서의 수율 손실을 최소화할 수 있는 장점이 있지만, 늘어난 캐쉬 메모리 접근 사이클로 인해 성능 및 에너지 측면에서의 부작용이 있을 수 있다. 본 장은 이를 평가하기 위한 체계에 대해서 설명한다. 참고로 수율은 앞 장에서 설명한대로 캐쉬 접근 사이클 저장 공간만 충분하다면 접근 시간 실패로 인한 수율 손실은 0%이므로 평가 대상에서 제외하였다.

1. Reference Processor Architecture

본 장에서는 본 논문에서 사용된 참고 (reference) 프로세서 구조에 대해서 기술한다.

표 1에서는 본 논문에서 사용된 참고 프로세서 구조의 개략적인 사양을 나타낸다. 참고 프로세서의 내부 마이크로아키텍쳐 (microarchitecture) 사양은 Intel사의 Atom 구조의 사양을 따른다. CPU core의 개수는 8개이고 공정 노드는 22nm 공정을 사용하였다. 기본 클럭 주파수는 2.4GHz이다. L1 캐쉬의 경우 각 코어당 L1 데이터 캐쉬와 명령어 캐쉬가 분리되어있다. 데이터 캐쉬의 경우 24KB 크기이고 기본 (i.e., 공정 변이에 영향을 받지 않았다고 가정할 때) 접근 시간은 4 클럭 사이클이다. 공정 변이에 영향을 많이 받으면 L1 데이터 캐쉬 접근 시간은 4 사이클보다 더 늘어날 수 있다. L1 명령어 캐쉬는 32KB로 데이터 캐쉬 크기에 비해 8KB가 더 많다. L2 캐쉬의 경우는 2개의 코어가 1MB의 L2 캐쉬 메모리를 공유하는 구조로써, 본 프로세서 구조에서는 총 8개의 코어를 사용하므로 프로세서에는 4MB의 L2 캐쉬 메모리가 있다.

2. Simulation Framework

가변 접근 시간 캐쉬 구조를 가진 프로세서의 성능을 평가하기 위해서 sniper-sim [20] 컴퓨터 구조 시뮬레이터를 사용하였다. sniper-sim은 컴퓨터 구조 및 마이크로 구조 (microarchitecture)의 파라미터를 바꾸어 가면서 프로세서의 성능을 평가할 수 있는 시뮬레이터로써, 대표적으로 분기 예측기 (branch predictor) 나 캐쉬 메모리 구조 등의 사양을 바꾸어가면서 프로세서의 성능을 측정할 수 있다. 본 논문에서는 가변 길이 캐쉬 메모리 구조를 평가하기 위해서 L1 데이터 캐쉬의 접근 클럭 사이클을 달리하면서 평가한다. 표 1에서 나온 것처럼, 기본 L1 데이터 캐쉬 접근 클럭 사이클은 4 사이클이나,

본 논문에서는 가변 접근 시간을 적용하여 5 사이클, 6 사이클, 7 사이클 일 경우에 대해서 각각 평가하도록 한다.

에너지 소모량 측정을 위해서 본 논문에서는 McPAT [21]을 사용하였다. McPAT은 컴퓨터 구조 및 마이크로구조의 사양에 따라서 해당 프로세서의 에너지 및 전력 소모량을 측정해 주는 시뮬레이터이다. 본 논문에서는 다수쓰레드 프로그램의 성능 및 에너지를 측정하기 위하여 Splash2와 PARSEC 벤치마크를 사용하였다. 표 2는 본 논문에서 사용된 Splash2와 PARSEC 프로그램들을 나타낸 표이다. Splash2의 경우에는 모든 프로그램을 전부 사용하였고, PARSEC의 경우에는 선택된 6개의 프로그램들을 사용하였다.

Table 2. Splash2 and PARSEC benchmark programs used for our evaluation

Benchmark programs	
Splash2	barnes
	cholesky
	fft
	fmm
	lu
	ocean
	radiosity
	radix
	raytrace
	water
PARSEC	blackscholes
	canneal
	fluidanimate
	streamcluster
	swaptions
	vips

V. Performance Evaluation Results

본 논문에서는 성능 평가를 위해 IPC (Instruction Per Clock cycle)을 사용한다. 클럭 주파수가 전부 동일하다고 가정하였기 때문에, IPC를 성능 평가 척도로 사용 가능하다 [22].

그림 3은 각 L1 데이터 캐쉬 접근 시간을 달리 하였을 때, Splash2와 PARSEC 벤치마크 프로그램 별 IPC 비교와 MPKI (Misses Per Kilo Instructions)를 나타낸 그림이다. IPC는 각각의 프로그램 별로 L1 데이터 캐쉬 접근 사이클이 4 사이클일 때를 기준으로 정규화된 결과이다. MPKI는 절대 값을 보여준다.

그림 3에서 보이듯이, L1 데이터 캐쉬 접근 사이클 수가 늘어남에 따라 IPC는 감소하는 것을 확인할 수 있다. 프로그램별로 차이가 있지만 L1 데이터 캐쉬 접근 시간이 4 사이클일 경우와 비교하여, 5사이클일 경우에는 평균 2.8%, 6사이클일 경우에는 평균 4.8%, 7사이클일 경우에는 평균 7.2%의 IPC 감소를 보여주었다. 공정 변이가 심화될수록 더 높은 L1 데이터 캐쉬 접근 사이클을 사용해야할 가능성이 높은데, 공정 변이가 상당히 심한 경우 (즉, L1 데이터 캐쉬 접근 사이클 = 7사이클)에도 7% 정도의 성능 저하로 수율 손실을 막을 수 있다.

각 프로그램별로 성능 저하 정도의 차이가 있다. 메모리 접근 명령어의 비율이 많고, 캐쉬 접근 실패율이 높은 (memory-bound) 프로그램의 경우, L1 데이터 캐쉬 접근 사이클이 높다고 하더라도, 늘어난 L1 데이터 캐쉬 접근 사이클로 인해 일어나는 성능 저하가 크게 감춰지는 효과가 있다. 이는 캐쉬 접근 실패로 인한 L2 캐쉬나 메인 메모리에 접근하는 시간 때문에 늘어난 L1 데이터 캐쉬 접근 시간이 감춰지는 효과가 있기 때문이다. 반면, 캐쉬 접근 실패율이 낮고 메모리 접근 명령어의 비율이 높은 프로그램들의 경우, 늘어난 캐쉬 접근

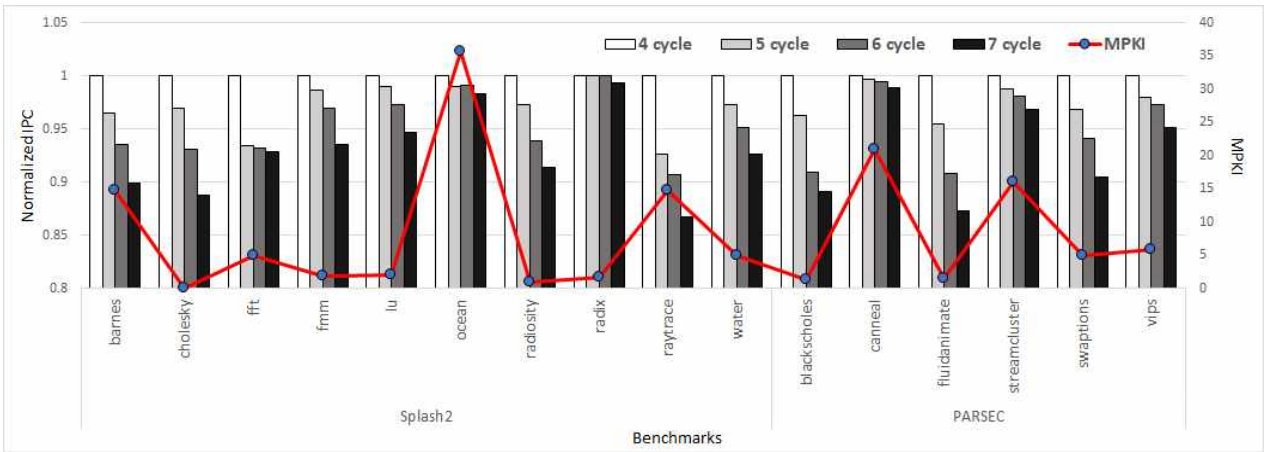

Fig. 3. Misses per kilo instructions (MPKI) and normalized IPC results across different cache access cycles

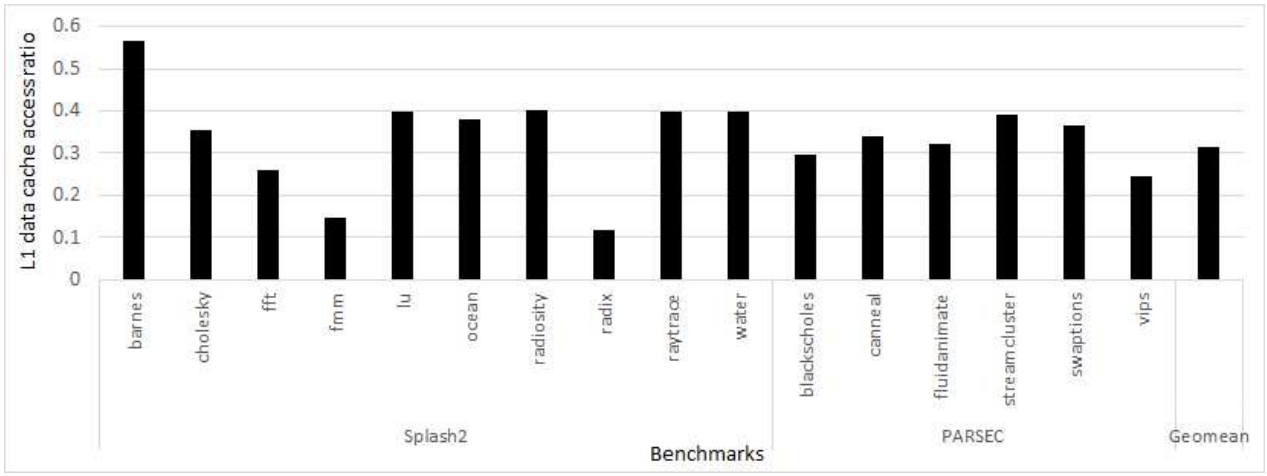

Fig. 4. Results of L1 data cache access frequencies for each benchmark program

시간이 성능저하에 기여하는 정도가 크기 때문에 L1 데이터 접근 사이클 수가 클수록 성능 저하가 두드러지는 경향이 있다. 그림 3에서 보면 전체적인 경향이 MPKI가 높을수록 늘어난 L1 캐쉬 접근 사이클로 인한 성능 저하가 적은 것을 확인할 수 있다.

위에서 언급한 경향과 배치되는 결과를 보여주는 프로그램들도 있는데, Splash2 Radix의 경우, MPKI가 낮음에도 불구하고 L1 캐쉬 접근 사이클이 늘어났을 때 성능 저하가 다른 프로그램에 비해 적은 것을 확인할 수 있다. 이 이유로는 Radix 프로그램의 경우, L1 데이터 캐쉬 접근 횟수 자체가 다른 프로그램들에 비해 상대적으로 적다. 그림 4에서 보이듯이, 위의 결과에서 나온 Splash2와 PARSEC 벤치마크 프로그램들의 평균 L1 데이터 캐쉬 접근 비율 (=L1 데이터 캐쉬 접근 횟수 / 전체 명령어 개수) 이 31.6%인데 반해 radix의 경우 11.6%에 불과하다. 이 때문에 L1 데이터 캐쉬 접근 사이클이 늘어나도 이에 따른 성능 저하는 상대적으로 낮게 나타난다.

VI. Power and Energy Evaluation Results

그림 5는 L1 데이터 캐쉬 접근 사이클 별 프로세서의 전력 소모 결과를 보여준다. L1 데이터 캐쉬 접근 사이클이 4 사이클일 경우에 정규화된 결과이다. 그림에서 보이듯이, 접근 사이클이 늘어날수록 전력 소모량은 감소하는 추세를 보인다. 평균적으로, L1 데이터 캐쉬 접근 사이클이 5 사이클일 경우 1.1%, 6일 경우 2.2%, 7일 경우 3.4%의 전력 소모 감소를 보여준다. 이 이유로는 L1 데이터 캐쉬 접근에는 동일한 에너지가 소모되지만, 접근 시간이 늘어나는 경우는 더 긴 시간동안 동일한 에너지가 소모됨으로써 상대적으로 전력 소모는 낮추는 효과가 있다. 이는, 가변 접근 시간 캐쉬 구조가 프로세서 코어의 발열을 낮추는 부수적인 효과도 있을 것으로 예상된다. 프로그램 별로 살펴보면 그림 3에서 보여주었던, IPC결과와 거의 동일한 경향을 보여준다.

그림 6은 L1 데이터 캐쉬 접근 사이클 별 프로세서의 에너

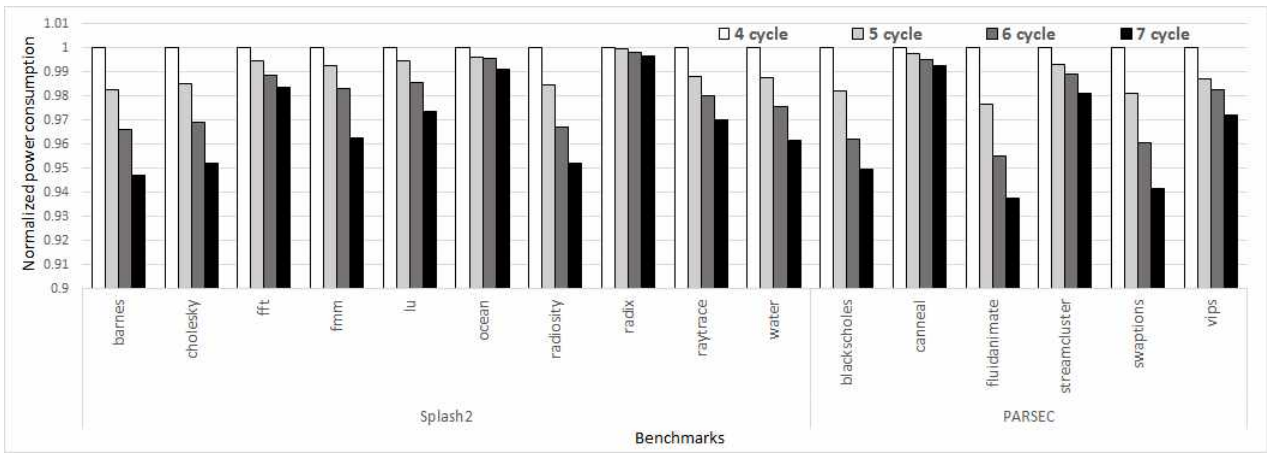

Fig. 5. Normalized power consumption results for each benchmark program across different cache access cycles

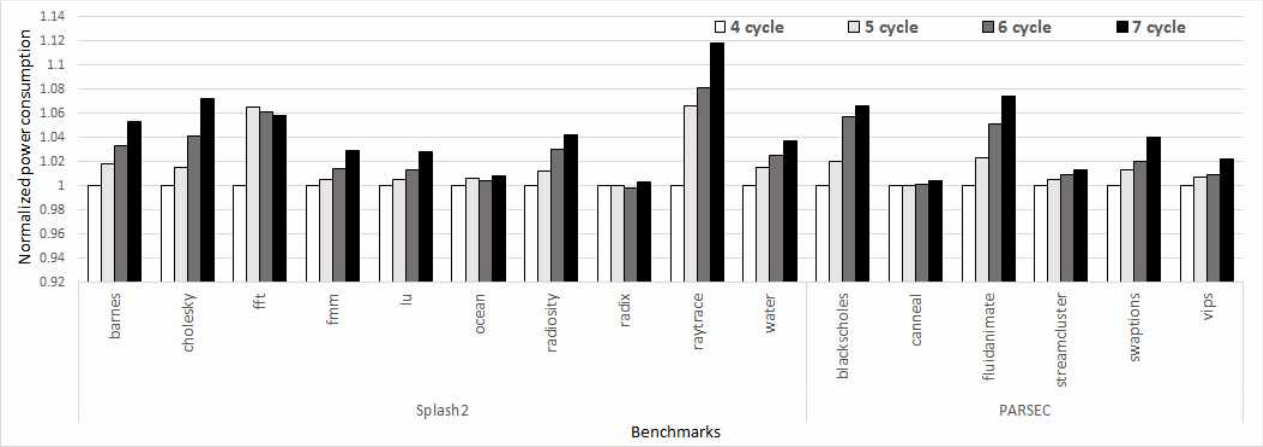

Fig. 6. Normalized energy consumption results for each benchmark program across different cache access cycles

지 소모 결과를 보여준다. 에너지 소모의 경우에는 전력 소모와 다른 결과를 보여주는데 이유는 에너지 소모량은 성능에도 영향을 받기 때문이다. 평균적으로, L1 데이터 캐쉬 접근 사이클이 4사이클일 경우와 비교하여, 5사이클일 경우에는 1.7%, 6사이클일 경우에는 2.8%, 7사이클일 경우에는 4.1%의 에너지 소모량 증가를 보여주었다. 위에서 보여주었듯이, L1 데이터 캐쉬 접근 사이클이 증가할수록 전력 소모는 감소하였지만, 성능 저하로 인해 실행시간이 늘어나면서 전체적인 에너지소모는 오히려 증가하는 양상을 보여준다. 각 프로그램 별로 경향의 차이가 있는데, Splash2 Radix의 경우 L1 데이터 캐쉬 접근 사이클이 6일 때 가장 낮은 에너지 소모량을 보여준다. 그림 3에서 보이듯이, Radix의 경우, 접근 사이클이 6일 때 접근 사이클이 4인 경우와 성능 차이가 거의 없다. 이 경우, 줄어든 전력 소모량이 거의 동일하게 에너지 소모량 감소로 반영되어 접근 사이클이 6일 때 가장 에너지 소모량이 적은 결과를 보여준다. Splash2 fft의 경우, L1 데이터 캐쉬 접근 사이클이 5일 때가 에너지 소모량이 가장 많은 것으로 나오는데, 성능 결과 (그림 3)에서 보면 접근 사이클이 4일 경우와 5일 경우는 성능 차이가 크게 나지만, 5, 6, 7사이에는 성능 차이가 크지 않다. 반면, 전력 소모량의 경우는 접근 사이클이 4에서 7로 증가하면서 거의 일정한 양으로 감소하는 결과를 보여준다. 따라서, 에너지 소모량의 경우, 접근 사이클이 5인 경우가 가장 높게 나온다.

VII. Power-Delay Product and Energy-Delay Product Evaluation Results

성능 (실행시간)과 전력/에너지 소모는 프로세서 및 컴퓨터 시스템 설계에 있어서 매우 중요한 두 요소지만 이들 단위들은 서로 상충되는 측면이 있다. 따라서, 이들 간의 균형 (trade-off)이 상당히 중요하다. 본 논문에서는 성능과 전력/에너지 요소들을 복합적으로 고려하였을 때, 캐쉬 접근 사이클을 변경 시 정량적으로 어떤 영향이 있는지 확인하기 위해 전력-시간 곱 (power-delay product: PDP)와 에너지-시간 곱 (energy-delay product: EDP) 결과를 보여준다.

그림 7은 캐쉬 접근 시간에 따른 각 벤치마크 프로그램 별 정규화된 PDP를 보여준다. PDP값은 캐쉬 접근 사이클이 4 사이클일 때를 기준으로 정규화되었다. 그림에서 보이듯이, 일반적으로 캐쉬 접근 사이클이 낮을수록 더 낮은 (i.e., 더 좋은) PDP값을 보여준다. 그럼 5에서 확인할 수 있듯이 평균적으로는 캐쉬 접근 사이클이 늘어날수록 더 낮은 전력 소모를 보인다. 그러나, 캐쉬 접근 사이클이 늘어남에 따른 성능 오버헤드가 전력 감소량에 비해 많기 때문에 PDP는 더 늘어나는 경향을 보인다. 평균적으로 접근 사이클이 5 사이클일 때는 1.7%정도 PDP가 증가하는 모습을 보였다. 6 사이클일 경우와 7 사이클일 경우는 각각 2.8%와 4.1%의 PDP 증가를 보였다.

그림 8은 캐쉬 접근 시간에 따른 정규화된 EDP결과를 보여준다. EDP값도 PDP의 경우와 마찬가지로 캐쉬 접근 사이클이 4 사이클일 경우를 기준으로 정규화되었다. 에너지는 전력과 수행시간의 곱이므로 PD^2P와 동일하다. 따라서, PDP와 비교하여 수행 시간에 더 높은 가중치가 가해진다. 그림 7에서의 PDP 결과에서 보이듯이, 수행시간이 PDP에 미치는 영향이 전력 소모가 PDP에 미치는 영향보다 크다. 따라서, EDP의 경우에는 PDP와 경향은 비슷하되 정도는 더 심화되는 모습을 보인다. 결과적으로 캐쉬 접근 사이클이 5 사이클일 경우는 4 사이클일 경우 대비 4.6%의 EDP 증가를 보여주었다. 6 사이클일 경우와 7 사이클일 경우는 각각 8.0%와 12.2%의 EDP 증가를 보여주었다.

위 결과에서 보이듯이 캐쉬 접근 사이클을 늘리면 PDP와 EDP에 악영향을 준다. 따라서, 프로세서 설계자들은 가변 접근 시간 캐쉬 구조 설계 시 너무 과도하게 캐쉬 접근 시간을 늘리기 보다는 프로세서 수율과 PDP 혹은 EDP와의 적절한 균형 (trade-off)을 통해 L1 데이터 캐쉬의 접근 시간을 정할 수 있다.

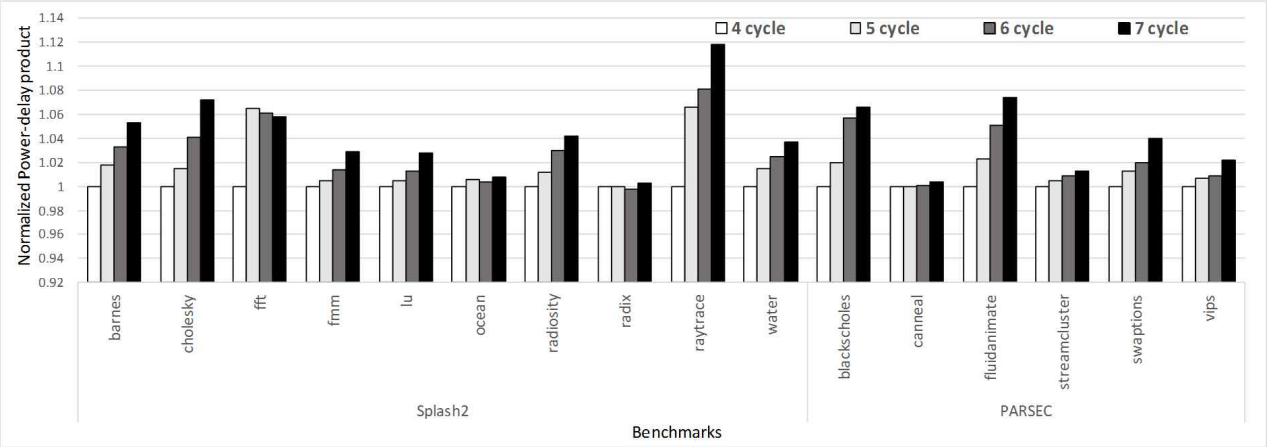

Fig. 7. Normalized power-delay product results for each benchmark program across different cache access cycles

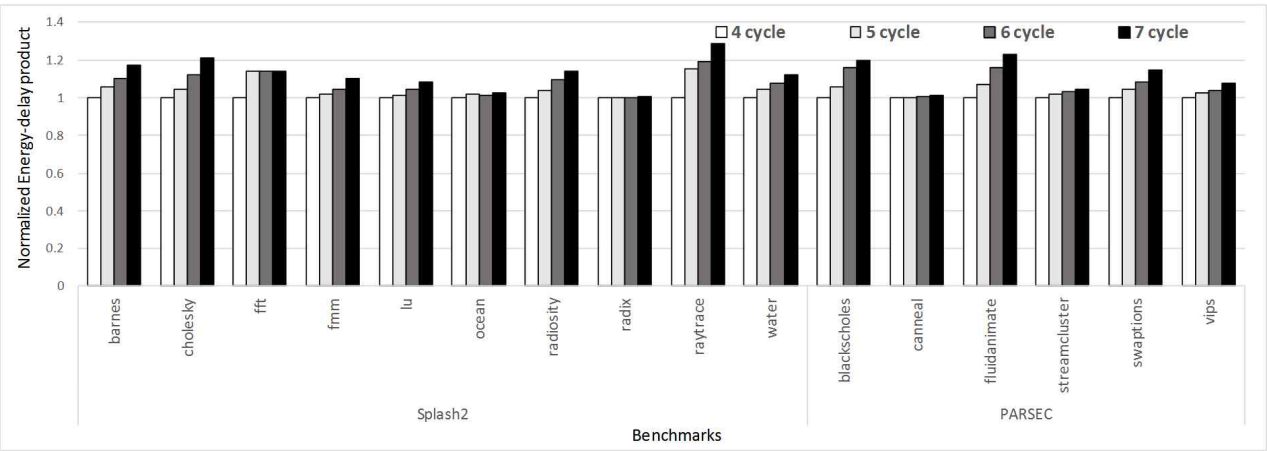

Fig. 8. Normalized energy-delay product results for each benchmark program across different cache access cycles

VIII. Conclusions

본 논문에서는 최근 많이 사용되고 있는 다중 코어 프로세서 구조에서 가변 접근 시간 L1 데이터 캐쉬 구조를 제안하였다. 제안한 L1 데이터 캐쉬 메모리 구조는 캐쉬 메모리 접근 시간 실패로 인한 프로세서의 수율 손실을 거의 0%로 낮출 수 있다. 또한, 다중 쓰레드 프로그램을 수행할 시 성능, 에너지, 및 전력 소모 결과를 보여줌으로써, 최신 프로세서 구조 및 병렬 처리 프로그램에서 가변 접근 시간 L1 데이터 캐쉬 구조가 성능, 전력, 및 에너지 측면에서 어떠한 영향이 있는지에 대해서 평가 및 분석하였다. 전력-성능, 에너지-성능 간의 균형 관계를 보기 위해서 전력-시간 곱 및 에너지-시간 곱 결과도 평가하였다. 본 논문의 분석 결과에 따르면 L1 데이터 캐쉬의 접근 시간이 늘어날수록 이에 따라서 성능 저하도 비례하는 형태로 나타났다. 기존 L1 데이터 캐쉬의 접근 시간이 4 사이클일 경우와 비교하여 7 사이클일 경우 평균 7.2%의 IPC 감소를 보여주었다. 전력 소모 측면에서는 7사이클의 경우가 4사이클의 경우 대비 3.4% 감소한 결과를 보여준다. 에너지 소모 측면에서는 성능과 전력 두 측면이 모두 반영된다. 7사이클의 경우 4사이클의 경우 대비 평균 4.1% 에너지 소모 증가를 보여주었다. 전력-시간 곱과 에너지-시간 곱의 경우 캐쉬 접근 사이클이 늘어날수록 증가하는 경향을 보였다. 특히, 에너지-시간 곱의 경우가 증가 폭이 더 큰데, 이유는 에너지가 이미 성능의 영향을 받으므로 전력보다는 성능에 더 높은 가중치가 가해졌기 때문이다.

공정 기술이 발전할수록 공정 변이는 더욱 더 심해질 것이고, 이에 따라 캐쉬 메모리에서 발생할 수 있는 결함을 해결하는 것이 상당히 중요한 문제가 될 것이다. 가변 접근 시간 캐쉬 구조는 그러한 문제를 해결할 수 있는 효율적인 기법 중 하나로써 본 논문의 분석 결과가 다중 코어 프로세서 설계자들에게 좋은 가이드가 될 수 있을 것으로 생각된다.

REFERENCE

[1] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, “Yield-Aware Cache Architectures”. In Proc.

of IEEE/ACM Int'l Symposium on Microarchitecture, pp. 15-25, Dec. 2006.

[2] S. Ozdemir, A. Mallik, J. C. Ku, G. Memik, and Y. I. Ismail, "Variable latency caches for nanoscale processor". SC 2007, In Proc. of 2007 ACM/IEEE Conference on Supercomputing, pp. 1-10, Nov. 2007.

[3] M. Mutyam, F. Wang, K. Ramakrishnan, V. Narayanan, M. T. Kandemir, Y. Xie, and M. J. Irwin, "Process Variation-Aware Adaptive Cache Architecture and Management". IEEE Trans. Computers 58(7), pp. 865-877, Jul. 2009.

[4] A. Agarwal, B. C. Paul, H. Mahmoodi. A. Datta, and K. Roy, "A process-tolerant cache architecture for improved yield in nanoscale technologies", IEEE Transaction on VLSI Systems, vol. 13, No. 1, pp. 27-38, Jan. 2005.

[5] A. Agarwal, B. C. Paul, S. Mukhopadhyay, and K. Roy, "Process variation in embedded memories: failure analysis and variation aware architecture", IEEE Jnl. of Solid-State Circuits, vol. 40, Issue 9, Sep. 2005.

[6] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. W. Chung, "Selective wordline voltage boosting for caches to manage yield under process variations". In Proc. of Design Automation Conference, pp. 57-62, Jul. 2009.

[7] J. Kong, Y. Pan, S. Ozdemir, A. Mohan, G. Memik, and S. W. Chung, "Fine-Grain Voltage Tuned Cache Architecture for Yield Management Under Process Variations". IEEE Trans. VLSI Syst. vol. 20, no. 8, pp. 1532-1536, Aug. 2012.

[8] M. Mutyam, and N. Vijaykrishnan, "Working with process variation aware caches". In Proc. of Design, Automation and Test in Europe, pp. 1152-1157, Mar. 2007.

[9] S. Ozdemir, Y. Pan, A. Das, G. Memik, G. H. Loh, and A. N. Choudhary, "Quantifying and coping with parametric variations in 3D-stacked microarchitectures". In Proc. of Design Automation Conference, pp. 144-149, Jun. 2010.

[10] J. Kong, and S. W. Chung, "Exploiting narrow-width values for process variation-tolerant 3-D microprocessors". In Proc. of Design Automation Conference, pp. 1197-1206, Jun. 2012.

[11] J. Kong, F. Koushanfar, and S. W. Chung, "An energy-efficient last-level cache architecture for process variation-tolerant 3D microprocessors". IEEE Trans. Computers, published online

[12] J. Tschanz, K. A. Bowman, and V. De. "Variation-tolerant circuits: circuit solutions and techniques". In Proc. of Design Automation Conference, pp. 762-763, Jun. 2005.

[13] J. Gregg and T. W. Chen. "Post Silicon Power/Performance Optimization in the Presence of Process Variations Using Individual Well Adaptive Body Biasing (IWABB)". In Proc. of IEEE Int'l Symposium on Quality Electronic Design, Mar. 2007.

[14] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Mitigating Parameter Variation with Dynamic Fine-Grain Body Biasing". In Proc. of IEEE/ACM Int'l Symposium on Microarchitecture, pp. 27-42, Dec. 2007.

[15] H. Li, Y. Chen, K. Roy, and C.-K. Koh, "SAVS: a self-adaptive variable supply-voltage technique for process-tolerant and power-efficient multi-issue superscalar processor design". In Proc. of Asia South Pacific Design Automation Conference, Jan. 2006.

[16] B. -S. Jung and J. -H. Lee, "Cache memory system for high performance CPU with 4GHz", Journal of The Korea Society of Computer and Information, Vol. 18, No. 2, pp. 1-8, Feb. 2013.

[17] N. R. Yang, J. M. Kim, and C. H. Kim, "Energy-aware Instruction Cache Design using Backward Branch Information for Embedded Processors", Journal of The Korea Society of Computer and Information, Vol. 13, No. 6, pp. 33-39, Nov. 2008.

[18] Y. -J. Park, J. -M. Kim, and C. H. Kim, "Low-power Filter Cache Design Technique for Multicore Processors", Journal of The Korea Society of Computer and Information, Vol. 14, No. 12, pp. 9-16, Dec. 2009.

[19] Intel® Atom™ Processor C2750 (4M Cache, 2.40 GHz), http://ark.intel.com/ products/77987

[20] T. Carlson, W. Heirman, and L. Eeckhout. "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation". In Proc. of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1-12, Nov. 2011.

[21] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi. “McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures”. In Proc. of 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 469-480, Dec. 2009.

[22] D. A. Patterson and J. L. Hennessy, “Computer Organization and Design The Hardware/Software Interface (ser. Comput. Architecture and Design), 4th edition”. San Mateo, CA, USA: Morgan Kaufmann, 2012.

Authors

Joonho Kong received the B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University, Korea, in 2007, 2009 and 2011, respectively. He is currently an assistant professor in the School of Electronics Engineering, Kyungpook National University.

He is interested in computer architecture design, low-power and energy-efficient computer system design, and hardware security.