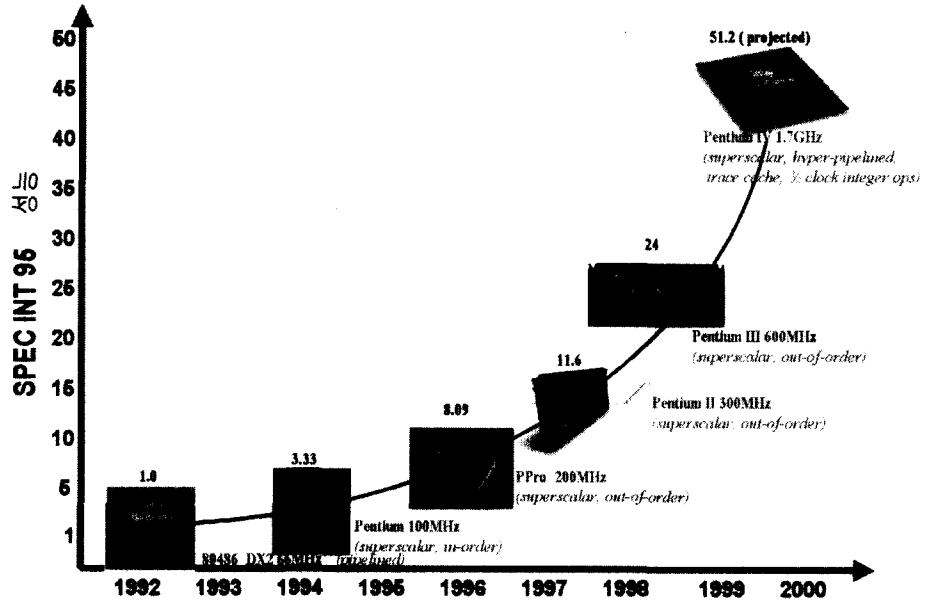
특 집

차세대 마이크로프로세서 기술 동향

최 린
고려대학교 전기전자전파공학부

I. 소 개

1971년에 발표된 세계 최초의 마이크로프로세서인 인텔사의 i4004이후 인텔사 창업주인 고든 무어의 예측대로 마이크로프로세서의 성능은 1년 반에 두 배씩 초고속 성장을 계속하여 왔다. 〈그림 1〉에서 보듯이 현재의 Pentium 4 프로세서는 1993년에 소개된 i486DX 66MHz 프로세서의 50배 이상의 성능을 제공하며 이는 1982년에 소개된 CRAY X-MP 슈퍼컴퓨터 성능을 3배 이상 능가하는 것으로 추정된다. 이러한 마이크로프로세서의 성능 향상은 두 가지 요소, 즉 (1) 클록 속도의 증가와 (2) 사이클 당 실행되는 명령어 수(IPC, Instructions Per Cycle)의 증가에 의해 주도되어 왔다. 전자가 반도체 제조 공정 기술에 의하여 주도되는 반면, 후자는 명령어 수준의 병렬 처리(ILP, Instruction-Level Parallelism)를 기반으로 하는 컴퓨터 구조 연구에 의해 주도되어 왔다. 〈그림 2〉는 최근 발표된 마이크로프로세서들의 구조적 특징과 성능을 보여준다. 시스템 설계 시, 클록 속도와 ILP의 두 목표는 종종 상충되는 결과를 초래하기 때문에 계속되는 도전이 하드웨어의 복잡성과 클록 속도의 균형을 이루는 것이다. 하지만, 슈퍼스칼라(Superscalar) 또는 VLIW(Very Long In-



〈그림 1〉 인텔 마이크로프로세서의 발전

struction Word)로 대표되는 현재의 와이드 이슈 프로세서 구조는 다음과 같은 이유로 계속적인 성능 증가를 방해하는 전례없는 도전에 직면하고 있다.

(1) 명령어 인출 및 이슈 대역폭 : 현재 고성능 프로세서에서 널리 쓰이고 있는 슈퍼스칼라 구조는 현재의 3-way 또는 4-way에서 8-way 또는 16-way 슈퍼스칼라로 대역폭을 확장할 때, 그 하드웨어의 복잡성으로 인해 파이프라인의 여러 단계에서 문제가 발생한다[9]. 예를 들어, 이슈 비율이 증가되면 사이클 당 더 많은 독립적인 명령어를 찾기 위해 명령어 윈도우 크기는 증가되어져야 한다. 하지만, 명령어 wakeup과 이슈 하드웨어의 복잡성은 윈도우 크기가 증가함에 따라 비선형적으로 증가한다. 또한, 이슈 비율의 증가와 더불어 레지스터 renaming 대역폭과 레지스터 파일 대역폭이 증가되어져야 하며, 이 두 가지 모두가 프로세서의 클록 사이클 시간을 증가시키게 된다. 한편, 이슈 대역폭의 증가는 명령어 인출(fetch) 대역폭의 비례적인 증가없이는 지속될 수 없

〈표 1〉 최신 마이크로프로세서의 파이프라인 특징 및 속도

구 분	클록 속도 MHz	반도체 공정	파이프라인 특징	분기 예측 기법	캐쉬 ▸L1(I+D) ▸L2	시스템 버스 대역폭	트랜지스터 카운트 million	패키지	기타 마이크로 구조 특징	성능 측정 ▸SpecINT2k ▸SpecFP2k
Intel Pentium4	1300 1400 1500 1700	0.18μm	▸3-6way OOO (out-of-order) 20 stages ▸4int/2float units	▸static and 2 level adaptive dynamic branch prediction, 4K entry BTB	▸12KB(μop) +8KB(4 way) ▸256KB (8 way)	400MHz 3.2GB/s 64bit	42	423 pin OLGA	▸basic integer operations in 1/2 processor clock tick	591@1700MHz 609@1700MHz
AMD Athlon MP Model 6	1400	0.18μm	▸3-9way OOO 10 stages ▸3int/3float/ 3mem units	▸2K entry dynamic branch prediction	▸64KB+64KB (2 way) ▸256KB (16 way)	266MHz 2.1GB/s 64 bit	37	452 pin PGA	▸512-entry TLB	554@1400MHz 458@1400MHz
SUN Ultra SPARC III	900	0.18μm	▸3-way in-order 14 stages ▸2int/2float/ 2mem units	▸16K entry dynamic branch prediction	▸32KB+64KB (4 way) ▸1, 2, 8 MB	300MHz 2.4GB/s 64 bit	29	1368 pin BGA	▸L1 cache: 2KB prefetch, 2KB write-buffers	467@900MHz 482@900MHz
Motorola PowerPC MPC 7450	533 667 733	0.18μm	▸4-way OOO, 7 stages ▸4int/1float units	▸2K entry 2 level adaptive dynamic branch prediction, 128-entry BTB	▸32KB+32KB (8 way) ▸256KB (8 way)	133MHz 1.0GB/s 64 bit	33	483 pin PGA	▸1 or 2MB L3 cache (8-way)	*346@733MHz **220@733MHz
Compaq Alpha 21264A	833 1000	0.18μm	▸4-way OOO 7 stages ▸4int/2float units	▸1K entry local+ 4K entry global hybrid branch prediction	▸64KB+64KB ▸2, 4, 8 MB	200MHz 1.6GB/s 64 bit	15.2	587 pin CPGA	▸integer register files are decentralized into 2 clusters	552@833MHz 705@833MHz
HP PA-8700	1000	0.18μm	▸4 way OOO 7 stages ▸4int/ 4float/ 2mem units	▸static and 2K entry 2-level adaptive dynamic branch prediction, 32 entry BTB	▸0.75MB+ 1.5MB ▸No L2 cache	120MHz 0.96GB/s	186	1.7V 304mm²	▸Large 1.5MB on-chip cache ▸240entry TLB	603@1000MHz 581@1000MHz

* SpecINT2000으로 환산한 수치(SpecINT95 32.1)

** SpecFP2000으로 환산한 수치(SpecFP95 23.9).

다. 하지만, 빈번한 분기(branch) 명령으로 인해 메모리 시스템으로부터 한 클록에 인출 가능한 명령어의 수는 4-6으로 제한된다. 즉, 제한된 명령어 인출이 와이드 이슈 구조의 병목 현상을 일으키는 것이다.

(2) 온-칩 지연시간 : 반도체 기술은 게이트와 배선 지연 모두에 많은 영향을 미친다. 게이트 지연이 반도체 세대마다 25%씩 감소하는 동안, 배선 지연은 매 세대마다 두 배로 증가한다. 온-칩 배선지연은 현재 0.25μ 기술에서의 2~4 클록 사이클로부터 0.1μ 이하의 선 폭 기술에서는 20~30 클록 사이클에 달하는 것으로 예측된다[3]. 이것은 새로운 반도체 기술이 소개됨에 따라 프로세서가 공유하는 자원들의 물리적 위치를 더욱 떨어지게 함으로써 현재의 슈퍼스칼라 프로세서 구조의 확장을 어렵게 하는 이유가 된다. 예를 들어, 데이터 bypass와 레지스터 파일 액세스가 한 사이클이 아닌 여러 개의 클록 사이클이 걸리게 하는 것이다.

(3) 단일 명령어 스트림의 한계 : 슈퍼스칼라 프로세서가 이러한 한계를 극복하더라도 명령어의 병렬성을 발견하는 데에는 한계가 있다. 비록 쓰이지 않는 상당량의 병렬성이 오늘날 프로그램에 존재하기는 하지만[4] 단일 명령어 스트림 모델은 독립적인 명령어를 찾는 범위를 명령어 윈도우의 최근 명령들로 국한시킴으로써 프로그램에 널리 퍼져있는 고유한 병렬성을 노출시키기에는 충분하지 않다.

이슈대역폭과 온-칩 지연 시간 문제 모두 클록 사이클 시간을 증가시키거나 파이프라인 단계의 수를 증가시키는데 이것은 슈퍼스칼라 마이크로 구조의 확장성 이슈로 요약되어질 수 있다. 이것은 분산된 파이프라인 구조에 의해 해결될 수 있다. 이러한 구조는 자원을 분할하고 분배함으로써, 비록 적은 공유로 인해 낮은 자원 이용률을 초래하기도 하지만, 사이클 시간을 증가시키지 않고 확장성 이슈를 해결할 수 있다. 여러 형태의 분산 구조가 제안되었으며 이는 분할된 레지스터 파일 구조와 같이 오직 특정한 자원을 분산화 하는 방법부터 단일 칩 상에서의 멀티프로세서처럼 공유된 자원을 모두 분산시키는 방법까지 다양하다.

세 번째 이슈를 해결하기 위해, 쓰레드 수준의 병렬성(thread-level parallelism : TLP)이 새로운 대안으로 여러 대학에서 연구되어 왔다[1][2][5][6][7][8][10][11]. 단일 프로그램을 병렬 작업으로 분해하고 그들을 여러 개의 처리장치에 분배함으로써 멀티쓰레드 프로세서는 프로그램 순서보다 앞서서 명령어들을 찾음으로써 더 많은 병렬성을 노출시킬 수 있다. 게다가 멀티 프로그래밍 환경에서는 많은 독립적인 쓰레드들이 하나의 파이프 라인을 공유함으로써 처리량(throughput)과 자원 이용도(utilization)를 향상시킬 수 있다. 또한, 이러한 멀티쓰레딩(Multithreading) 기술은 종종 쓰레드마다 독립적인 파이프라인과 레지스터 파일을 사용함으로써 멀티프로세서와 함께 분산 구조의 마이크로프로세서를 추구한다.

본 논문에서는 차세대 마이크로프로세서의 대안으로 제시되고 있는 멀티쓰레딩과 단일 칩 멀티프로세싱 기술을 소개하고 이러한 구조들이 앞으로의 차세대 마이크로프로세서 개발에 어떻게 반영되고 있는지 살펴보기로 한다.

II. 마이크로프로세서에서의 다중 명령어 스트림(Multiple Instruction Stream) 모델

단일 명령어 스트림 모델이 하드웨어에 의해 이용될 수 있는 ILP를 제한함에 따라, 좀 더 많은 ILP를 이용하기 위해 다중 명령어 스트림 모델에 대한 연구가 진행되어 왔다. 다중 명령어 스트림 모델은 여러 개의 명령어 스트림을 독립적인 파이프라인에 실행함으로써 파이프라인 해저드 발생 시 모든 명령어 스트림을 지연(stall)시

킬 필요가 없기 때문에 파이프라인 해저드(hazard)를 다루는데 더 효과적이다. 예를 들어, 각각의 개별적인 명령어 스트림에서 야기된 분기 예측 실패로 인한 파이프라인 플러쉬(flush)나 지연(stall)은 다른 명령어 스트림에서의 명령어 실행에 영향을 미치지 않을 수 있다. 또한, 다중 명령어 스트림 모델에서는 하나의 커다란 논리적인 명령어 윈도우가 여러 개의 파이프라인의 작은 윈도우들로부터 구성되기 때문에 하드웨어의 복잡성을 증가시키지 않고 명령어 수준 이상의 병렬성, 즉 루프 수준 또는 프로시저 수준의 병렬성을 이용하는 데 효과적이다. 이 다중 명령어 스트림 모델은 여러 개의 명령어 스트림으로부터 명령어를 검사할 수 있도록 여러 개의 프로그램 카운터를 필요로 하며 다양한 형태로 구현될 수 있다.

1. 단일 칩 멀티프로세싱

다중 명령어 스트림 모델을 구현하는 자연스러운 방법은 스탠퍼드 대학에서 개발되고 있는 Hydra 프로세서[8]나 MIT 대학의 RAW 프로세서[6]와 같이 단일 칩 상에 멀티프로세서를 구성하는 것이다. 멀티프로세서는 각 프로세서가 자신의 레지스터 파일과 파이프라인 그리고 명령어 및 데이터캐쉬를 가지고 있다. 각 프로세서의 클록 사이클 시간에 영향을 줄 수 있는 중앙 집중의 공유된 자원이 없기 때문에 이는 가장 분산된 구조이고 또한 가장 확장이 용이한 구조이다. 멀티프로세서는 두 가지 방법으로 병렬 처리에 이용될 수 있다. 첫 번째로, 멀티프로그래밍 환경에서 여러 독립적인 프로세스들을 병렬로 실행함으로써 처리량을 증가시킬 수 있다. 두 번째로, 병렬 컴파일러에 의해 단일 어플리케이션으로부터 여러 개의 쓰레드를 생성하여 여러 개의 쓰레드를 병렬로 실행할 수 있다. 이러한 경우, 멀티프로세싱이 단일 프로그램 성능을 향상시키는 데 사용될 수 있다.

Hydra에서 온-칩 L2 캐쉬를 제외하고 모든 자원들은 분산된다. 그러므로 프로세서간의 통신은 공유된 L2 캐쉬를 통하여만 이루어진다. RAW에서는 캐쉬나 메모리를 포함한 모든 자원들이 분산되어진다. 따라서 프로세서간의 통신은 오직 사용자 수준의 통신 명령에 의해서만 이루어진다. 즉, Hydra가 공유 메모리(shared memory) 멀티프로세싱 모델을 제공하는 반면 RAW는 분산 메모리(distributed memory) 멀티프로세싱 모델을 지원하는 것이다. 하지만, 이러한 멀티프로세싱 방법은 다른 명령어 스트림 사이의 동기와 통신이 단지 공유된 메모리나 프로세서 상호간의 통신을 통해서만 이루어지기 때문에 다른 다중 명령어 스트림 모델에 비해서 더 coarse-grain 병렬성만 이용할 수 있는 제한이 있다. 이러한 단일 칩 멀티프로세싱 모델은 IBM 및 Motorola가 개발 중에 있는 차세대 마이크로프로세서인 Power 4 프로세서에 구현될 예정이다.

2. 멀티쓰레딩(Multithreading)

단일 칩 상에서의 물리적 근접의 이점을 활용하기 위해 몇 가지 더 단단히 결합된 다른 다중 명령어 스트림 모델들이 제안되었다. 이들은 일반적으로 멀티쓰레딩 구조라 불리며, 자원공유의 정도 또 쓰레드간의 동기화(synchronization) 및 통신을 위해 제공되는 하드웨어와 컴파일러의 지원 정도에 따라 분류된다. 빠른 문맥 전환과 fine-grain 쓰레드를 지원하기 위해 대부분의 제안된 구조들은 각 쓰레드마다 개별적인 레지스터 파일을 사용한다.

1) 독립적인 쓰레드들의 멀티쓰레딩

멀티쓰레딩 분야에서의 많은 연구는 독립적인 명령어 스트림, 즉 독립적인 응용 프로그램들의 동시 실행을 가정함으로써 프로세서의 처리량 증대를 목표로 하였다. 이러한 멀티쓰레딩 기법의 대표적인 예로는 워싱턴 주립대에서 제안된 simultaneous 멀티쓰레딩(SMT)[11]과 Tera 컴퓨터사에서 개발된 Tera 슈퍼컴퓨터[2]를 들 수 있다. 이러한 멀티쓰레딩 모델에서는 쓰레드 사이의 의존성(dependency)이 존재하지 않으므로 공유된 메모리를 제외한 쓰레드간의 어떠한

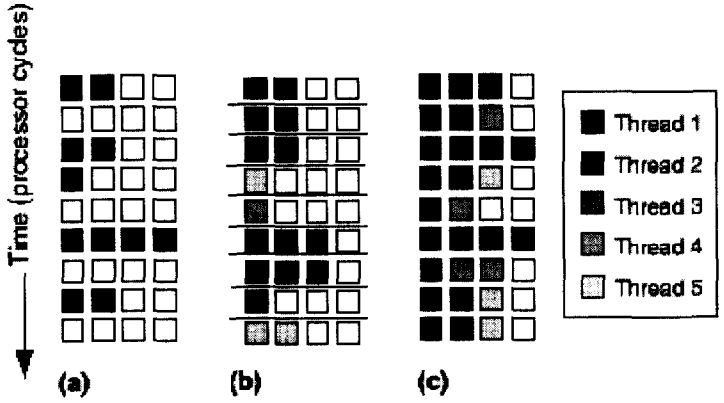
통신이나 동기화에 대한 하드웨어 지원이 없다. 이러한 제한은 시스템 설계를 매우 간단하게 하며 주로 멀티프로그래밍 환경에서 독립적인 프로그램들을 동시에 실행함으로써 처리량을 증가시키는 것을 목표로 한다. 하지만 이 경우 캐쉬 및 TLB 공유에서 발생되는 프로그램 상호간의 충돌과 간섭으로 인해 단일 프로그램의 성능은 약해지기도 한다.

처리량을 기반으로 하는 멀티쓰레딩 기법은 다음 두 가지 방법으로 나뉘어 질 수 있다:

(a) fine-grain 멀티쓰레딩 : 각 사이클마다 단지 하나의 활성화된 쓰레드를 가지는 멀티쓰레딩-현재 쓰레드가 캐쉬 미스나 파이프라인 지연(stall)과 같은 긴 지연에 의해 블록되었을 때 다른 쓰레드로 전환한다.

(b) SMT : 각 사이클마다 여러 개의 활성화 쓰레드를 가지는 멀티쓰레딩-매 사이클마다 여러 쓰레드로부터 명령어를 이슈한다. SMT가 두 번째 방법의 예인 반면, Tera, MIT 대학의 M-Machine[5] 등은 첫 번째 카테고리에 속한다.

〈그림 2〉는 슈퍼스칼라, fine-grain 멀티쓰레딩, 그리고 SMT 프로세서 파이프라인의 차이를 보여준다. 〈그림 2(a)〉는 일반적인 슈퍼스칼라 파이프라인에서의 명령어 이슈 과정을 보여 준다. 프로세서는 매 사이클마다 하나의 쓰레드를 실행하며 이로부터 이슈할 명령어들을 찾는다. 만약에 이슈할 명령이 없는 경우 이슈 슬롯은 사용되지 않으며, 수직적인(vertical waste : 한 사이클의 모든 이슈 슬롯이 낭비) 또는 수평적인 이슈 슬롯의 낭비(horizontal waste : 한 사이클의 일부 이슈 슬롯이 낭비)를 초래한다. 〈그림 2(b)〉는 Tera와 같이 fine-grain 멀티쓰레딩 모델에서의 명령어 이슈 과정을 보여 준다. 매 사이클마다 프로세서는 하나의 쓰레드로부터 명령어를 실행하며 파이프라인 지연이 발생하면 다른 쓰레드 문맥으로 전환하여 새로운 쓰레드로부터 명령어를 실행한다. 그림에서 볼 수 있듯이 fine-grain 멀티쓰레딩 모델의 장점은 긴 지연시간을 갖는 연산을 잘 처리함으로써 수직적인 낭비를 제거하는데 효과적이다. 그럼에도 불구하고 수평적인 낭비를 제거하지는 못한다. 결과적으로, 명령어 이슈 대역폭이 증가함에 따라 fine-grain 멀티쓰레딩 구조는 결국 슈퍼스칼라 프로세서와 같은 운명을 겪을 것이다. 〈그림 2(c)〉는 SMT 프로세서의 명령어 이슈 과정을 보여준다. SMT는 fine-grain 멀티쓰레딩 모델과 마찬가지로



〈그림 2〉 파이프라인 이슈 슬롯의 비교
(a) 슈퍼스칼라 프로세서 (b) fine-grain 멀티쓰레딩 (c) SMT 프로세서 파이프라인.
채워진 박스는 이슈 할 명령이 있는 슬롯을, 빈 박스는 사용되지 않은 이슈 슬롯을 나타낸다.

수직적인 낭비를 제거하는 반면 매 사이클마다 여러 쓰레드들로부터 실행할 명령어들을 선택함으로써 수평적인 낭비 역시 줄일 수 있다. 만약 하나의 쓰레드가 높은 ILP를 가졌다면 병렬성은 만족되어질 수 있으며 또한 여러 개의 쓰레드가 각각 낮은 ILP를 가진 경우 이들을 동시에 수행함으로써 병렬성을 증가시킬 수 있다. 즉, fine-grain 멀티쓰레딩에 비해 SMT 프로세서는 이슈할 수 있는 모든 쓰레드들로부터 명령어를 선택함으로써 ILP를 증가시키고 자원을 실행 시간에 동적으로 스케줄 함으로써 하드웨어를 더욱 효과적으로 이용할 수 있다. 이 SMT 모델은 Compaq에서 개발 예정인 21464 프로세서에서 채택 구현될 예정이다.

2) 종속적인 쓰레드들 사이의 멀티쓰레딩

하나의 프로그램으로부터 생성된 쓰레드들은 서로 의존적일 가능성이 높다. 따라서 단일 프로그램의 성능을 향상시키기 위해서는 의존적인 쓰레드들 사이의 멀티쓰레딩이 필수적이며, 쓰레드간의 의존성 해결(dependency resolution), 통신, 동기화(synchronization)를 지원하기 위해 더 복잡한 하드웨어와 컴파일러가 요구된다. 이러한 단일 프로그램의 성능 향상을 위해 제시된 대표적인 멀티쓰레딩 모델로 위스콘신 대학에서 제안된 멀티스칼라 프로세서가 있다[10].

멀티스칼라 모델은 쓰레드간의 통신 및 동기에 하드웨어와 컴파일러를 모두 사용하는 복합적인 접근 방법을 취한다. 컴파일러는 쓰레드 사이의 레지스터 의존성(register dependency)을 감지하고 쓰레드사이의 레지스터 값의 전달과 동기를 위해 프로그램에 명령들을 삽입하는 반면 하드웨어는 실행 시간에 메모리 의존성을 검증한다. 메모리 의존성 검증(memory disambiguation) 하드웨어는 동시에 실행되고 있는 모든 쓰레드들에 의해 생성된 메모리 주소들을 관리하여 renaming 기법을 통해 불필요한 메모리 의존성(false dependences)들을 제거하는 동시에 실행시간에 발생하는 메모리 데이터의 의존성 위반 사실을 감지한다. 의존성 위반은 만약 현재의 쓰레드가 메모리 위치에 기록하려 할 때 그 뒤의 쓰레드가 이미 같은 위치로부터 읽은 사실이 감지되면 발생한다. 이런 사실이 감지되면, 올바른 메모리 접근 순서를 유지하기 위해 뒤의 쓰레드와 그 후의 모든 뒤따르는 쓰레드들은 취소되어지고 재시작 되어야 한다. 따라서 이러한 멀티스칼라 모델이 추구하는 성능 향상을 위해서는 이러한 재시작의 비용을 최소화하는 것이 중요하다.

컴파일러에 의존하는 멀티스칼라 모델과 달리 최근에 제안된 speculative 멀티쓰레딩(SM) 모델[7]과 DMT 모델[1]에서는 쓰레드사이의 의존성검사, 통신 및 동기화 모두가 하드웨어에 의해 행해진다. 이러한 멀티쓰레딩 모델에서는 컴파일러의 도움 없이 하드웨어가 실행시간에 동적으로 여러 개의 쓰레드를 생성, 쓰레드 수준의 병렬성을 이용할 수 있으며 또한 ISA 변경을 요구하지 않으므로 기존의 어플리케이션 프로그램과 완전한 코드 호환성을 지원하는 장점이 있다.

〈표 2〉는 자원 공유 및 분산 정도, 쓰레드간의 통신 방법, 처리량 또는 성능 향상의 관점에서 최근의 멀티쓰레딩 모델들의 특징을 요약한 것이다. 공유 자원의 정도와 프로세싱 요소(Processing Elements, PE)들의 연결 방법이 모델의 확장성을 결정한다. RAW모델이 가장 확장 가능한 구조인 반면 SMT는 가장 중앙 집중적인 멀티쓰레드 구조이다. SMT와 Hydra, RAW는 공유 메모리 또는 사용자 수준의 통신 이외에 프로세싱 요소들 사이의 어떠한 데이터 통신이나 동기화 채널도 제공하지 않는다. 이는 고비용의 통신을 야기하여 coarse grain 병렬성의 이용만을 가능하게 함으로써 주로 처리량의 증대를 목표로 한다. 반면에 멀티스칼라 모델은 컴파일러 지원을 통해 단일 프로그램으로부터 쓰레드 수준 병렬성을 추출함으로써 프로그램의 성능을 향상할 수 있다. DMT와 SM 프로세서는 하드웨어에 의해 실행 시간에 쓰레드 수준 병렬성을 끌어내어 컴파일러의 도움 없이 성능을 증대할 수 있는 반면 멀티스칼라, Hydra, RAW 구조에 비교하여 자원의 공유가 많고 따라서 덜 확장 가능한 단점을 가지고 있다.

〈표 2〉 멀티쓰레딩 모델의 분류

구 분	슈퍼스칼라	SMT	DMT	SM	멀티스칼라	Hydra	RAW
공유되는 자원	모두	레지스터 파일을 제외한 나머지	캐쉬, 레지스터, functional units	캐쉬	데이터 캐쉬, L2 캐쉬	L2 캐쉬	핀
PE 사이의 연결	해당 없음	해당 없음	bypass 망	링	링	버스	2차원 mesh
중앙집중 대 분산	중앙 집중 ····· 지방 분산 (확장하기 어렵다) (확장이 용이하다)						
쓰레드 간의 통신	해당 없음	없다	있다	있다	있다	없다	없다
성능 대 처리량	해당 없음	처리량	성능	성능	성능	처리량	처리량

III. 차세대 상용 마이크로프로세서의 동향

〈표 3〉은 인텔, 컴팩 및 IBM에서 개발 중에 있는 차세대 마이크로프로세서의 발표된 마이크로구조의 특징을 보여준다. 인텔 최초의 64비트 마이크로프로세서인 Itanium 프로세서는 인텔의 IA-64 명령어 구조(instruction set architec-

〈표 3〉 차세대 상용 마이크로프로세서의 특징 및 개발 계획

프로세서 (발표예정)	클록속도 MHz	반도체 공정	파이프라인 특징	캐쉬	시스템 버스 대역폭	트랜지스터 카운트 (million)	패키지	기타 마이크로 구조 특징
Intel								
Itanium (2001)	733 800	0.18μm	▸ 6-way in-order 10 stages	▸ 32KB+32KB L1 ▸ 256KB? L2 ▸ 2 or 4MB L3	2.1GB/s		Socket M	▸ register stack ▸ predication support ▸ control and data speculation ▸ software pipelining support ▸ 4 single-precision FMACs
McKinley (2002)	1000+	0.13μm	▸ 6-way in-order	▸ 4MB on-die L3				▸ developed by HP ▸ derivative of Itanium
Madison (2004?)		0.13μm		▸ 8MB on-die L3				▸ 2nd-generation IA-64
Compaq								
21364 (EV7)	1500	0.18μm	▸ 4-fetch, 6-issue OOO	▸ 64KB+64KB L1 ▸ 1.5MB L2	10GB/s	130	FC/SCP 1400	
21464 (EV8)	2000	0.125μm	▸ 8-fetch (16?) 8-issue (10?) OOO		10GB/s	250	FC/SCP 1800	▸ SMT multithreading
IBM								
Power4	1000+	0.18μm	▸ 5-issue OOO	▸ 1.5MB L2 ▸ 32MB L3	10GB/s	680	MCM	▸ 2 processor cores on 1 die ▸ 4 chips on 1 MCM module ▸ 8-way SMP multiprocessing

ture)를 최초로 구현한 프로세서로서 서버 및 고성능 웍스테이션을 겨냥한 고성능 프로세서이다. 그 후속인 McKinley 프로세서는 Itanium의 코어 파이프라인을 기초로 HP 주도하에 개발중인 프로세서이며 Madison 프로세서는 Itanium의 차세대 IA-64 프로세서로서 멀티쓰레딩 또는 멀티프로세싱 등 다중 명령어 스트림 모델이 적용되리라 예측된다. Compaq은 21464 프로세서에서 SMT 방식의 멀티쓰레딩 기법을 사용하여 개발 중에 있으며 IBM은 멀티쓰레딩 기법 대신에 대칭형 멀티프로세싱(SMP, symmetric multiprocessing) 기법을 적용하여 쓰레드 수준의 병렬성을 이용하는 Power4 프로세서를 개발, 곧 발표할 예정이다. Power4는 단일 칩에 2-way 멀티프로세서를 구현하였으며 4개의 칩을 이용 8-way MP 시스템으로 개발되었다.

참 고 문 헌

〔1〕 H. Akkary and M. A. Driscoll, "A Dynamic Multithreading Processor", Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture, pp. 226-236, 1998.

〔2〕 R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, "The Tera Computer System", Proceedings of the ACM International Conference on Supercomputing, pp. 1-6, 1990.

〔3〕 M. T. Bohr, "Interconnect Scaling-The Real Limiter to High Performance ULSI", Solid State Technology, Vol. 39, No. 9, pp. 105-111, September 1996.

〔4〕 M. Butler, T. Yeh, Y. Patt, M. Alsup, H. Scales, and M. Shebanow, "Single Instruction Level Stream Parallelism Is Greater than Two", Proceedings of the 21st Annual ACM/IEEE International Symposium on Computer Architecture, pp. 223-232, 1994.

〔5〕 M. Fillo, S. W. Keckler, W. J. Dally, N. P. Carter, A. Chang, Y. Gurevich, and W. S. Lee, "The M-Machine Multicomputer", Proceedings of the 28 th Annual ACM/IEEE International Symposium on Microarchitecture, pp. 146-156, 1995.

〔6〕 W. Lee, R. Barua, M. Frank, D. Srikrishna, J. Babb, V. Sarkar, and S. Amarasinghe, "Space-Time Scheduling of Instruction-Level Parallelism on a Raw Machine", Proceedings of the 8th International Conference for Architectural Support for Programming Languages and Operating Systems, pp. 46-57, 1998.

〔7〕 P. Marcuello, A. Gonz ez, and J. Tubella, "Speculative Multithreaded Processors", Proceedings of the ACM 1998 International Conference on Supercomputing, pp. 77-84, 1998.

〔8〕 K. Olukotun, K. Chang, L. Hammond, B. Nayfeh, and K. Wilson, "The Case for a Single Chip Multiprocessor", Proceedings of the 7th International Conference for Architectural Support for Programming Languages and Operating Systems, pp. 2-11, 1996.

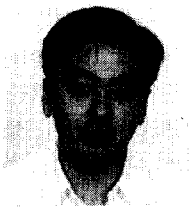
〔9〕 S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-Effective Superscalar Processors", Proceedings of the 24th Annual ACM/IEEE International Symposium on Computer Architecture, pp. 206-218, 1997.

〔10〕 G. S. Sohi, S. E. Breah, and T. N. Vijaykumar, "Multiscalar Processors", Proceedings of the 22nd Annual

ACM/IEEE International Symposium on Computer Architecture, pp. 414-425, 1995.

[11] M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous Multithreading : Maximizing On-chip Parallelism", Proceedings of the 22nd Annual ACM/IEEE International Symposium on Computer Architecture, pp. 392-403, 1995.

저 자 소 개



崔 麟

1964년 2월 10일생, 1986년 2월 서울대학교 공과대학 컴퓨터공학과 졸업 (학사), 1988년 2월 서울대학교 대학원 공과대학 컴퓨터공학과 졸업 (석사), 1996년 3월 미국 University of Illinois at Urbana-Champaign, Dept. of Computer Science (박사), 1988년 3월~1990년 7월 : 한국 통신 연구개발단 전임연구원, 1996년 4월~1998년 8월 : 미국 Intel Corporation, Santa Clara, CA, Senior Design Engineer, 1998년 7월~2000년 8월 : 미국 University of California, Irvine, Dept. of Electrical and Computer Engineering, 조교수, 2000년 9월~현재 : 고려대학교 공과대학 전기전자전파공학부, 조교수 〈주관심 분야 : 마이크로프로세서, 컴퓨터 구조, 컴파일러, Embedded Computing, Mobile Computing〉