논문 2010-47SD-9-2

개인용 정보 단말장치를 위한 내장형 멀티스레딩 프로세서 구조

(Embedded Multithreading Processor Architecture for Personal Information Devices)

정 하 영*, 정 원 영*, 이 용 석**

(Hayoung Jeong, Wonyoung Chung, and Yong-surk Lee)

요 약

본 논문은 스마트폰, 타블렛 PC와 같은 개인용 정보 단말장치 응용에 적합한 프로세서 구조를 제안한다. 고성능 내장형 프로세서 개발은 아키텍처의 변화가 필요하고, 오버헤드가 크기 때문에, 업계에서는 높은 동작 주파수의 고성능 내장형 프로세서의 개발에 전념하고 있다. 고성능 프로세서 구조 중 비순차 슈퍼스칼라(out-of-order superscalar)는 하드웨어 복잡도가 과도하게 증가하며, 그에 비해 성능 향상이 적으므로 내장형 응용에 적합하지 않다. 따라서 하드웨어 복잡도가 낮은 고성능 내장형 프로세서 구조의 개발이 필요하다. 본 논문에서는 스칼라, 슈퍼스칼라, 멀티프로세서 방식에 비하여 복잡도가 낮은 새로운 SMT(Simultaneous Multi-Threading) 구조를 제안한다. 최근의 개인용 정보단말기는 많은 작업을 동시에 수행하기 때문에, SMT나 CMP는 이에 적합한 구조라 할 수 있다. 또한, 시뮬레이션 결과 SMT는 여러 프로세서 구조 중 가장 효율이 높은 프로세서로 보인다.

Abstract

In this paper, we proposed a processor architecture that is suitable for next generation embedded applications, especially for personal information devices such as smart phones, tablet PC. Latest high performance embedded processors are developed to achieve high clock speed. Because increasing performance makes design more difficult and induces large overhead, architectural evolution in embedded processor field is necessary. Among more enhanced processor types, out-of-order superscalar cannot be a candidate for embedded applications due to its excessive complexity and relatively low performance gain compared to its overhead. Therefore, new architecture with moderate complexity must be designed. In this paper, we developed a low-cost SMT architecture model and compared its performance to other architectures including scalar, superscalar and multiprocessor. Because current personal information devices have a tendency to execute multiple tasks simultaneously, SMT or CMP can be a good choice. And our simulation result shows that the efficiency of SMT is the best among the architectures considered.

Keywords : 내장형 프로세서, 멀티스레드 프로세서, 멀티프로세서

I. 서 론

스마트 폰, 타블렛 PC와 같은 개인용 정보 단말기는 고성능 내장형 프로세서의 주요 응용 분야이다. 최근 개인용 단말기의 시장이 급속도로 커지고 있어 시장에서의 고성능 내장형 프로세서의 요구가 커지고 있다. 하지만 내장형 프로세서에서는 칩 면적과 전력 소비 문제로 고성능 프로세서 구조를 채택하기가 쉽지 않다. 최근 상용으로 발표된 ARM Cortex-A9 코어는 3이슈 비순차 슈퍼스칼라 프로세서 구조와 같은 간단한 구조를 채택하고 있으며, 동작 주파수를 높이고 더 많은 파이프라인을 사용하여 성능을 높이는데 집중하고 있다[1].

* 학생회원, ** 정회원, 연세대학교 전기전자공학과
(School of Electrical & Electronic Engineering, Yonsei University)

※ 본 논문은 지식경제부 출연금으로 ETRI 시스템반도체진흥센터에서 수행한 시스템반도체 융복합형 설계인재양성사업의 연구결과입니다.
접수일자: 2010년7월21일, 수정완료일: 2010년9월6일

또한 최근에는 더 높은 성능을 위해 여러 개의 코어를 사용하는 멀티 프로세서 구조로 전환하고 있다. 하지만 메모리와 입출력 장치의 지연으로 인한 유휴 시간(idle time)의 증가로 프로세서 코어의 리소스를 최대로 활용할 수 없기 때문에 칩 면적에 비하여 얻을 수 있는 성능 향상은 제한적이다.

이를 해결할 수 있는 연구로 멀티스레딩(multi-threading) 기술을 내장형 프로세서에 적용하려는 연구가 진행되고 있다[2~3]. 이러한 연구는 다수의 스레드의 상태를 저장할 수 있는 레지스터 파일과 매 사이클마다 다수의 스레드에서 명령어를 패치 할 수 있는 구조를 사용한다. 이러한 프로세서 구조를 fine-grain 멀티스레딩이라 한다. Fine-grain 멀티스레딩의 대표적인 예는 SUN의 UltraSPARC T3 프로세서이다[4]. T3 프로세서는 실행 유닛을 최대한 공유하기 위해서 8개의 스레드가 1세트의 정수 유닛과 부동소수점 유닛으로 구성된 실행 유닛을 공유한다. 이러한 구조에서는 스레드 전환이 용이하여 인터럽트 처리나 컨텍스트 전환을 빠르게 수행할 수 있는 장점이 있다. 하지만 멀티스레딩을 지원하기 위해서는 다수의 레지스터 셋과 같은 하드웨어 자원이 필요하다는 단점이 있다.

이러한 단점을 극복하기 위해서 SMT 구조에서는 수퍼스칼라 구조의 리소스를 여러개의 스레드가 공유하여 사용하기 때문에 멀티 프로세서를 구성할 경우 높은 효율을 얻을 수 있다. 하지만 SMT 구조는 이러한 슈퍼스칼라 구조에 더 큰 레지스터 파일과, 다수의 레지스터 매핑 테이블과 리오더 버퍼가 필요하다[5~6]. 따라서 복잡한 구조로 인해서 증가하는 면적 때문에 내장형 프로세서에는 적합하다고 할 수 없었다. 하지만, 최근 SMT의 구조를 개선하여 내장형 프로세서에 적용하려는 연구가 꾸준히 이루어지고 있다[7~8]. 본 논문에서는 내장형 기기에서의 SMT 구조의 효용성을 분석하기 위해서 성능과 칩 면적을 비교하여 이동형 기기에 적합한 고성능 프로세서 구조를 분석하였다.

II. 본 론

1. 기준 프로세서 모델

기준 프로세서는 현재 가장 널리 사용되는 내장형 프로세서와 같이 긴 파이프라인을 가지고, 한 번에 하나의 명령어를 수행할 수 있다. 그림 1은 기준 프로세서 모델의 파이프라인 구조를 보여준다. 기준 모델의 파이프라인은 7단계로 구성되어 높은 클럭 주파수에서 동작할 수 있도록 하였다. 또한 브랜치 타겟 버퍼(Branch target buffer)를 가지고 있으며, 명령어 간의 의존성을 추적하기 위해 스코어보드(scoreboard)를 사용한다. 각 스코어보드 엔트리에는 아키텍쳐 레지스터가 연결되어 있으며, 각 엔트리에는 지연 업데이트 비트(pending update bit)와 소스 오퍼랜드가 사용 가능한 예상 지연 시간을 기록한 지연 사이클 필드(latency cycle field)가 있다. 각 스코어보드를 액세스하거나, 소스 오퍼랜드가 사용 가능한 시간을 계산하기위해서는 더 많은 시간이 필요하므로 별도의 이슈 사이클이 필요하게 된다.

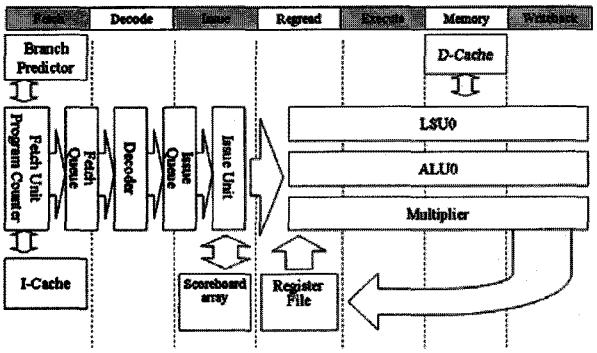


그림 1. 기준 프로세서 파이프라인 구조
Fig. 1. Pipeline structure of baseline processor.

가. 멀티프로세서 구조

여러 개의 스칼라 프로세서나 슈퍼스칼라 프로세서를 사용하여 멀티프로세서 모델을 구성할 수 있다. 본 논문에서는 2개 또는 4개의 스칼라 및 슈퍼스칼라 프로세서를 사용하여 멀티프로세서 모델을 만들었다. 각 프로세서 코어는 L1 캐쉬를 가지고 있으며, 각 코어들은 메모리 버스를 사용하여 통신을 한다. 멀티프로세서 모델에서는 4개의 스칼라 코어 또는 2개의 슈퍼스칼라 코어를 사용하여 매 사이클 마다 4개의 명령어를 수행할 수 있다.

나. 순차적 SMT 구조

기존의 SMT 구조는 비순차적 슈퍼스칼라 구조에 기반을 두었다. 파이프라인의 전반부는 매 사이클 마다 여러 스레드에서 동시에 명령어를 처리할 수 있다. 패치 된 명령어들은 디코드 되고, 스레드 당 하나씩 있는 레지스터 앨리어스 테이블(register alias table)을 사용하여 리네이밍(renaming)된다. 레지스터 리네이밍 이후에는 각 스레드간의 구분이 무의미 하므로 스레드

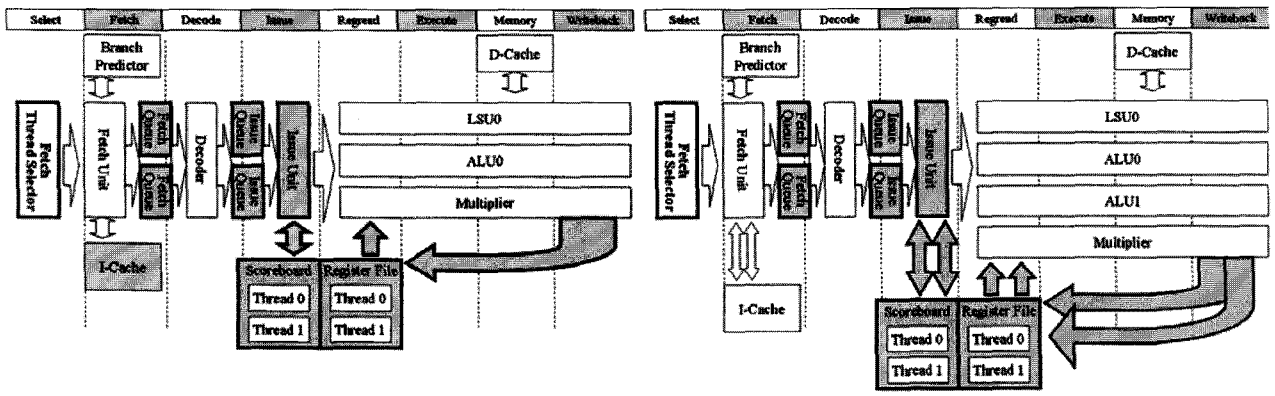


그림 2. 멀티스레드 구조 모델((a) fine-grain 모델 (b) 순차적 SMT 모델)
Fig. 2. 2 models of a multithread architecture ((a) fine-grain model (b) in-order SMT model).

단위의 예외처리(exception handing)와 분기 예측처리(branch prediction handling)를 제외하면 슈퍼스칼라 방식과 동일하게 처리할 수 있다. 이러한 아이디어는 많은 오버헤드를 가지고 있다. 첫째, 다중 스레드의 상태 정보를 모두 저장해야 하므로 물리적 레지스터 파일의 크기가 매우 크다. 둘째, 다중 스레드에서 명령어를 패치 하여야 하므로, 멀티 포트 논-블로킹 캐쉬(multi-port non-blocking cache)가 필요하다. 셋째, 분기 예측기나 캐쉬와 같은 저장 공간, 또는 실행 유닛을 여러 스레드가 공유하므로 더 많은 저장 공간이 필요하게 된다. 그 결과 SMT 구조는 고성능 프로세서 영역에서도 2개의 스레드를 동시에 실행하는 것에 그치고 있다.

본 논문에서는 기존의 SMT의 단점을 개선한 새로운 순차적 SMT 구조를 제안한다. 제안되는 구조의 명령어 패치 단에서는 다중 스레드에서 명령어를 패치 하기 위해서 스레드 선택 단(thread select stage)이 추가 되었다. 스레드 선택 단에서는 각 스레드의 우선순위를 계산하여, 다음 사이클에 어떤 스레드가 패치 될 것인지를 결정한다. 매 사이클 마다 명령어가 패치 되는 스레드의 개수는 명령어 캐쉬의 포트 수와 같다. 이는 명령어 큐(instruction queue)에 각 스레드의 명령어들이 적절하게 섞기기 위해서는 명령어 캐쉬의 포트가 중요하다는 것을 의미한다[9]. 패치 된 명령어는 디코드 단에서 디코드 되며, 디코드 단에는 스레드 당 하나의 명령어 큐를 가지고 있다. 이러한 큐는 스레드 별로 명령어의 의존성을 검사하거나 이슈를 하기 위한 정보를 유지하기 위하여 사용된다. 스칼라 구조에서는, 이슈 유닛은

표 1. 시뮬레이션 모델의 파라미터
Table 1. Parameters of simulation models.

	Scalar	Superscalar	FGMT	SMT_sm	SMT_lg	CMP_isN_prM
스레드	1	1	4	4	4	M
파이프라인	7-stage	7-stage	8-stage	8-stage	8-stage	7-stage
이슈 폭	1	2	1	2	4	N
실행 유닛	1 ALU 1 LSU 1 Multiplier	2 ALU 1 LSU 1 Multiplier	1 ALU 1 LSU 1 Multiplier	2 ALU 1 LSU 1 Multiplier	4 ALU 2 LSU 1 Multiplier	N*M ALU N*M/2 LSU M Multiplier
명령어 캐시	8KB/8-way 1 port	8KB/8-way 1 port	8KB/8-way 2 port	8KB/8-way 2 port	8KB/8-way 2 port	(8KB/8-way 1 port) M개
데이터 캐시	8KB/8-way 1 port	8KB/8-way 1 port	8KB/8-way 1 port	8KB/8-way 1 port	8KB/8-way 2 port	(8KB/8-way 1 port) M개
BTB	128	128	128	128	128	128*M

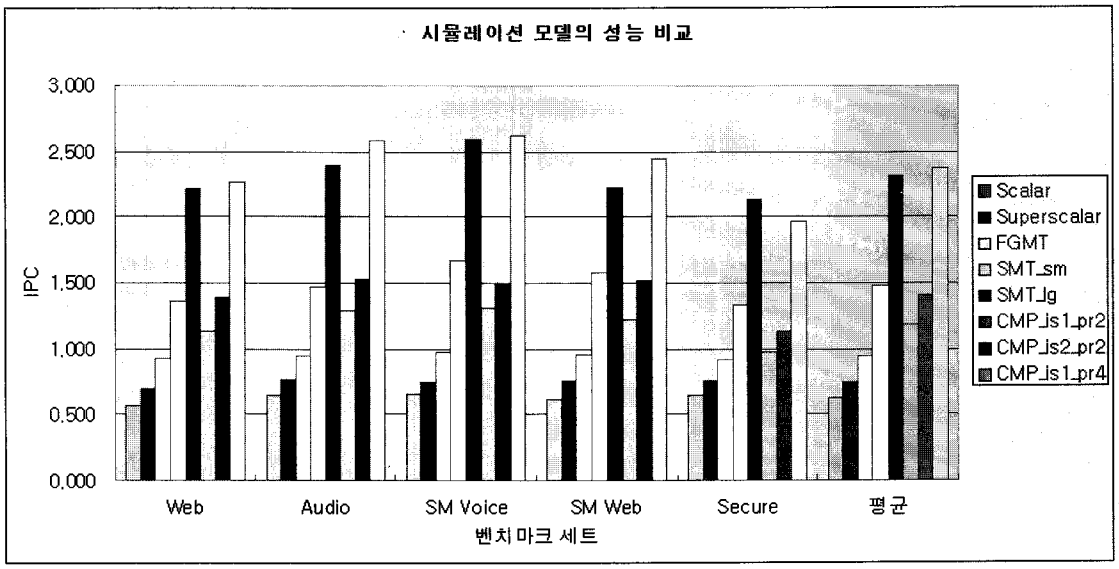


그림 3. 시뮬레이션 결과 IPC 비교
Fig. 3. Simulation result and IPC comparison.

스코어보드를 사용하여 오래된 명령어의 이슈 여부를 검사했지만, 제안된 순차적 SMT 구조에서는 이러한 이슈 결정 방식을 확장하였다. 만약 매 사이클 마다 n개의 명령어가 이슈 될 수 있다면, 각 스레드별 의존성 검사기는 선행하는 n개의 명령어의 이슈가 가능한지 검사한다. 이러한 의존성 검사가 끝나면, 선택 로직은 이슈 가능한 모든 명령어 중에서 n개의 명령어만을 선택하게 된다.

멀티스레딩을 지원하기위해서 가장 중요한 부분 중에 하나는 레지스터 파일을 수정하는 것이다. SMT는 다수의 스레드의 프로세서 상태를 저장하여야 하므로, 레지스터 파일, 프로그램 카운터(program counter), 상태 레지스터(status register)와 같은 모든 프로세서 상태를 저장하는 하드웨어는 스레드 당 하나씩 존재하여야 한다. 이것이 멀티스레딩 구조의 가장 큰 단점이라고 할 수 있다. 하지만 본 논문에서 제안한 순차적 SMT 구조는 기존의 SMT 구조에 비하여 더 적은 레지스터 파일을 필요로 한다. 또한 의존성 검사방법을 개선하여, 스레드 당 하나의 스코어보드를 사용하여 매 사이클 마다 각 스코어보드의 엔트리만을 액세스 하게 된다.

본 논문에서는 비교를 위하여 2개의 멀티스레드 구조를 제안한다. 첫 번째 구조는 매 사이클 마다 1개 스레드에서 명령어를 패치 하는 fine-grain 멀티스레딩 구조이다. 두 번째 구조는 동시에 여러 스레드에서 명령어를 패치 하는 SMT 구조이다. 그림 2에서 보는 것과 같이, 이 두 구조의 차이점은 명령어 캐쉬의 포트 개수의 차이와 같다. 두 구조에서는 최대 2개의 스레드를 지원하며, 최대 2개의 명령어를 동시에 이슈 할 수 있다.

III. 실 험

본 연구를 위하여 사이클 기반의 시뮬레이터를 사용하였다. 이 시뮬레이터의 핵심 부분은 이전의 연구 결과를 이용하였다[10]. ARM ELF 바이너리 로더와 시스템 콜(system call)은 ARM Linux와 호환되도록 SimpleScalar ARM 버전을 포팅 하여 사용하였다[11]. 시뮬레이터는 다수의 ARM 바이너리를 동시에 읽어서 수행하며, 수행하는 바이너리의 개수를 1로 설정함으로써 스칼라 모델과 슈퍼스칼라모델을 시뮬레이션 할 수 있다. 표 1은 시뮬레이션을 위한 초기 설정 값을 보여주고 있다.

1. 워크로드

개인용 정보단말 기기에서는 인터넷과 스트리밍 멀티미디어 응용프로그램의 중요성이 매우 커지고 있으므로, 이러한 워크로드들이 멀티태스킹 환경에서 동작한다고 볼 수 있다. 본 연구에서 사용한 워크로드는 MiBench embedded benchmark suite와 IEEE 802.11 WLAN WEP (Wired Equivalent Privacy) 알고리즘를 사용하였다[12]. 위 두 알고리즘은 다양한 상황에 맞추어

표 2. 시뮬레이션에 사용한 스레드 조합
Table 2. Thread combination for simulation.

벤치마크	스레드 조합
웹 브라우징 (Web)	djpeg, toast, untoast, search_large, lout search_small, crc, rawdaudio, rijndael, rawcaudio, madplay, wep
스트리밍 오디오와 E-book (Audio)	madplay, sbcenc, crc. wep
스마트 폰 (SM Voice)	rawaudio, toast, untoast, rawcaudio
스마트 폰 (SM Web)	untoast, toast, lout
암호화된 파일의 전송 (Secure)	rijndael, rx, crc, wep

멀티태스킹 환경을 만들었고 표 2는 각 멀티태스킹 환경에 사용된 MiBench 스레드의 조합을 보여준다.

2. 시뮬레이션 결과

비교를 위하여 8가지 아키텍처를 시뮬레이션 하였다. 그림 3에서는 성능 평가 결과를 보여주고 있다. 순차적 방식을 사용한 슈퍼스칼라 구조는 성능이 크게 증가하지 못하였다. 평균 19%의 성능 향상을 보이지만 이는 차세대 정보단말기에는 적합하지 않다. 반면에 스레드 수준의 병렬성을 이용한 프로세서 구조는 높은 성능 향상을 보였다. 가장 간단한 싱글 이슈 FGMT 구조는 51% 성능향상을 보였고, 2-issue SMT_sm, CMP_is1_pr2와 같은 모델은 2배에 가까운 성능 향상을 보였다. 4 이슈 구조인 SMT_lg, CMP_is2_pr2, CMP_is1_pr4 구조는 각각 120%, 145%, 280%의 성능이 향상되었다. 성능만을 고려하였을 경우 4개의 스칼라 프로세서로 구성된 CMP가 가장 좋은 성능을 보였다. 하지만 이것은 4개의 프로세서 코어 뿐 아니라 4개의 개별적인 L1 캐쉬가 하나의 칩 안에 집적되어야 한다. 만약 그보다 적은 수의 캐쉬를 사용할 경우 메모리 대역폭의 저하로 인하여 성능이 감소 할 수 있다. 반면에 SMT 구조는 캐쉬와 같은 하드웨어 자원을 스레드 간에 공유할 수 있으므로 CMP 방식 보다 하드웨어가 절약된다. 일반적으로 캐쉬의 크기는 칩 면적에 30~50% 정도를 차지하고 있으므로 SMT 구조는 칩 면적에 많은 영향을 줄 수 있다. 따라서 가격과 성능의 효율성을 비교하였을 때는 SMT 구조가 CMP 보다 우수하다고 할 수 있다. 게다가 CMP_is1_pr4 모델은 멀티태스킹 환경이 아닌 싱글 스레드환경에서는 스칼라 모델과 같은 성능을 보이지만, SMT와 슈퍼스칼라 구조는 19% 성능 향상을 보인다.

3. 하드웨어 복잡도 분석

가격대 성능의 효율이 높은 프로세서 구조를 선택하기 위해서 시뮬레이션한 모든 프로세서 구조의 하드웨어 복잡도를 추정 비교하였다. 본 논문에서는 프로세서 구조를 모두 모델링하지 않고, ARM 프로세서 하드웨어의 실재 게이트 카운트에 근거하여 추산하였다. 제안된 구조의 총 게이트 카운트는 각 하위 블록의 게이트 카운트를 합산하여 구할 수 있다. 이러한 접근은 정확도 면에서는 많이 떨어질 수 있지만, 제안된 프로세서 구조들에 대한 비교의 자료로서는 가치가 있다. 비교 결과는 표 3에 요약되어 있다. SMT_lg와 CMP_is1_pr4 모델이 IPC (Instruction Per Clock)가 높게 나타나지만 성능/면적(IPC/area)은 SMT_lg가 가장 높다. 따라서 순차적 SMT 구조인 SMT_lg가 내장형 프로세서로 가장 적합한 구조라 할 수 있다.

표 3. 프로세서 구조의 면적 예상치
Table 3. Area estimations of processor models.

	Scalar (ARM9)	Superscalar	FGMT	SMT_sm	SMT_lg	CMPis1_pr4
ALU	8.1	16.2	8.1	16.2	32.4	8.1*4
곱셈기	9.8	9.8	9.8	9.8	9.8	9.8*4
레지스터 파일	22	44	88	176	176	22*4
컨트롤 유닛	4.2	6.3	6.3	6.3	8.4	4.2*4
디코드 유닛	2	4	2	4	8	2*4
포워딩 유닛	2	4	2	4	8	2*4
MMU	87	87	87	87	87	87*4
BUS 인터페이스	12	12	12	12	12	12*4
캐시	278	278	278	278	278	278*4
전체 면적	425.1	461.3	493.2	593.3	619.6	1748.4
면적 증가	0	8.52	16.02	39.57	45.75	311.29
성능/면적 (IPC/Area)	1.49	1.62	1.92	2.50	3.73	1.36

Ⅳ. 결 론

본 논문은 개인용 정보 단말기에 적합한 새로운 프로세서 구조를 제안 하였다. 최근의 개인용 정보 단말기의 응용프로그램 특성은 네트워크와 멀티미디어프로그

램의 멀티태스킹이라고 가정하였을 때, 제안된 구조는 복잡한 하드웨어가 필요한 비순차적 실행과 같은 ILP를 높이는 방법이 아닌 TLP를 높이는 것에 주력했다.

이러한 가정에 따라 MiBench, WLAN 워크로드를 사용하여 개인용 정보 단말기의 복잡한 멀티태스킹 환경을 5개의 유형으로 나누어 시뮬레이션 하였다. 성능만을 고려하였을 때는 CMP_is1_pr4와 SMT_lg가 가장 높은 성능을 보였다. 하지만, 하드웨어 복잡도와 프로세서의 성능을 비교하여 보았을 때 개인용 정보단말기의 내장형 프로세서로 가장 적합한 것은 순차적 SMT 모델이다. 순차적 SMT 모델은 기존의 스칼라 구조와 비교하여 면적 대비 성능이 2.5배로 가장 좋다고 할 수 있다. 따라서 제안된 순차적 SMT 프로세서 구조는 개인용 정보단말기의 내장형 프로세서로 가장 적합하다고 할 수 있다.

참 고 문 헌

[1] "The ARM Cortex-A9 Processors", ARM Ltd., 2009.

[2] U. Brinkschulte , C. Krakowski , J. Kreuzinger , Th. Ungerer, "A Multithreaded Java Microcontroller for Thread-Oriented Real-Time Event Handling," Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques, p.34, October 12-16, 1999.

[3] J. Kreuzinger, A. Schulz, M. Pfeffer, T. Ungerer, U. Brinkschulte, C. Krakowski. "Real-time scheduling on multithreaded processors," the proceedings of seventh international conference on Real-Time Systems and Applications, pp. 155, 2000.

[4] J. Shin, K. Tam, D. Huang, B. Petrick, H. Pham, C. Hwang, H. Li, A. Smith, T. Johnson, F. Schumacher, D. Greenhill, A. Leon, A. Strong. "A 40nm 16-Core 128-Thread CMT SPARC SoC Processor". ISSCC 2010.

[5] Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy, "Simultaenous Multithreading: Maximizing On-Chip Parallelism," Proceedings of 22nd Annual International Symposium on Computer Architecture, pp. 392-403, Santa Margherita Ligure, Italy, May 1995.

[6] Keith Diefendorff, "Compaq Chooses SMT for Alpha," Microprocessor Report, December 6, 1999.

[7] Patrick Crowley, Marc E. Fiuczynski Jean_Loup Baer, and Brian N. Bershad, "Characterizing Processor Architectures for Programmable Network Interfaces," Proceedings of the 2000 International Conference on Supercomputing, May 2000.

[8] Peter N. Glaskowsky, "Networking Gets XStream," Microprocessor Report, November 13, 2000.

[9] Dean M. Tullsen, Susan J. Eggers, Joel S. Emer, and Henry M. Levy, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," Proceedings of 23rd Annual International Symposium on Computer Architecture,pp. 191-202, Philadelphia, Pennsylvania, May 1996.

[10] Byung In Moon, Moon Gyung Kim, In Pyo Hong, Ki Chang Kim, and Yong Surk Lee, "Study of an In-order SMT Architecture and Grouping Schemes," International Journal of Control, Automation, and Systems, Volume 1, Number 3, pp.339~350, September 2003.

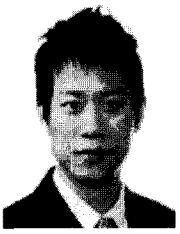
[11] SimpleScalar Version 4.0 Test Releases, SimpleScalarLLC, http://www.simplescalar.com/v4test.html

[12] Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, Richard B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," IEEE 4th Annual Workshop on Workload Characterization, Austin, Texas, December 2001.

저 자 소 개

정 하 영(학생회원)
2003년 중앙대학교 전기전자 공학부 학사 졸업.
2005년 연세대학교 전기전자 공학과 석사 졸업.
2005년~현재 연세대학교 전기 전자공학과 박사과정.
<주관심분야 : 마이크로 프로세서, ASIC, SoC>

이 용 석(정회원)
1973년 연세대학교 전자공학과 학사 졸업.
1977년 University of Michigan Electrical Engineering 석사 졸업.
1981년 University of Michigan Electrical Engineering 박사 졸업.
1993년~현재 연세대학교 전기전자공학과 교수.
<주관심분야 : 마이크로프로세서 설계, VLSI 설계, DSP 프로세서 설계, 고성능 연산기 설계>



정 원 영(학생회원)
2005년 연세대학교 전기전자 공학과 학사 졸업.
2005년~현재 연세대학교 전기 전자공학과 석박사 통합과정.
<주관심분야 : 네트워크 프로세서, 컴퓨터 아키텍처, 메모리구조>