通信情報保護學會論文誌
第 11 券 第 2 號, 2001. 4

블록 암호화 알고리즘 RC6 및 Rijndael에서의 병렬성 활용

정 용 화*, 정 교 일*, 손 승 원*

Exploiting Parallelism in the Block Encryption Algorithms RC6 and Rijndael

Yong-wha Chung*, Kyo-il Chung*, Sung-won Sohn*

요 약

현재 대부분의 상용 마이크로프로세서는 슈퍼스칼라 구조를 채택하고 있으나, 반도체 집적도가 증가함에 따라 슈퍼스칼라 구조를 대신할 새로운 마이크로프로세서 구조가 제안되고 있다. 본 논문에서는 최근 새로운 마이크로프로세서 구조로 급부상하고 있는 다중처리 마이크로프로세서 구조가 차세대 블록 암호화 알고리즘에 적합한지를 분석한다. 즉, 차세대 블록 암호화 알고리즘인 RC6와 Rijndael에서의 병렬성을 분석하기 위하여 프로그램 구동방식의 시뮬레이션을 수행한 결과, 명령어 수준 병렬성만으로는 성능의 한계를 갖지만 쓰레드 수준 병렬성을 동시에 활용함으로써 추가적인 성능 향상을 얻을 수 있음을 확인하였다.

ABSTRACT

Currently, the superscalar architecture dominates today's microprocessor marketplace. As more transistors are integrated onto larger die, however, an on-chip multiprocessor is regarded as a promising alternative to the superscalar microprocessor. This paper examines the behavior of the next generation block encryption algorithms RC6 and Rijndael on the on-chip multiprocessing microprocessor. Based on the simulation results by using a program-driven simulator, the on-chip multiprocessor can exploit thread level parallelism effectively and overcome the limitation of instruction level parallelism in the next generation block encryption algorithms.

keyword : Block Encryption Algorithm, Parallel Processing, On-Chip Multiprocessor

Ⅰ. 서 론

컴퓨터를 이용한 주요 정보의 저장, 처리, 전송 등이 필요함에 따라 그 정보를 안전하게 보호하기 위한 많은 방법들이 연구되고 있다. 특히 암호화를 이용한 정보 보호 방법이 오랜 연구를 통해 구현되어 왔으며 실용화되고 있다. 암호화 기술을 구현하기 위한 방법으로는 키의 형태에 따라 암복호 키가 같은 대칭형(symmetric) 암호화와 암복호화 키가 서로 다른 비대칭형(asymmetric) 암호화로 크게 구분할 수 있다. 또한 대칭형 암호화 알고리즘은 데이터 처리 형식에 따라 블록(block) 암호화 알고리즘과 스트림(stream) 암호화 알고리즘으로 나누어진다.

블록 암호화 알고리즘에서는 미국의 국가표준국(National Institute of Standards and Tech-

* 한국전자통신연구원 정보보호기술연구본부(ywchung@etri.re.kr, kyoil@etri.re.kr, swsohn@etri.re.kr)

nology, NIST)이 1977년 국가 표준으로 채택하여 전세계적으로 사용되고 있는 Data Encryption Standard(DES)[1]가 대표적이다. 그러나 DES가 발표된지도 30여년 되었고, 키의 길이도 56비트로 현재의 컴퓨터 계산 능력과 암호 해독 기술로는 하루 이내에 DES를 깰 수 있음이 최근 미국에서 열린 DES 암호해독대회에서 드러났다. 이에 따라 NIST에서는 1998년을 기점으로 DES의 표준 기한이 만료되므로, 1998년 이후에는 DES를 더 이상 표준 알고리즘으로 사용하지 않기로 결정하였다. 이러한 배경 하에서 향후 30년간 사용하게 될 새로운 표준 블록 암호화 알고리즘을 선정하기 위한 기반 작업이 NIST 주도로 진행되어, 1997년 초 표준 암호로 선정되기 위한 최소 요건, 평가 기준 등에 대한 발표와 이에 대한 공개적인 검토를 요청하였다. 이를 바탕으로 NIST에서는 1997년에 Advanced Encryption Standard(AES)[2]를 공모하는 방안을 발표하였고, 5개의 후보 알고리즘(Mars, RC6, Rijndael, Serpent, Twofish)이 선정되어 마지막 평가를 받은 결과 Rijndael이 최종안으로 최근 선정되었다.

이러한 AES 평가와 관련된 연구로는, 범용 마이크로프로세서에서의 후보 알고리즘 성능에 대한 연구가 수행되었고[3~11], 신호처리 전용 칩인 Digital Signal Processor(DSP)[12], Field Programmable Gate Array(FPGA)를 이용한 전용 하드웨어에서의 성능[13]이 분석되었다. 또한, 슈퍼스칼라(superscalar) 구조[14,15]의 범용 마이크로프로세서 상에서 각 알고리즘에 내재되어 있는 병렬성에 대한 연구도 발표되었다[16]. 그러나 슈퍼스칼라 구조는 명령어 수준의 병렬성(Instruction-Level Parallelism, ILP)을 활용하여 하나의 싸이클에 여러 개의 명령어를 수행할 수 있으나, 일반적으로 순차 프로그램내의 ILP가 크지 않기 때문에 성능상의 한계를 갖고 있다[17]. 특히, 복잡한 마이크로프로세서를 설계하는데 엄청난 비용이 든다는 사실은 슈퍼스칼라 구조를 대신할 새로운 마이크로프로세서 구조에 대한 연구를 가속화시키고 있다[18~23]. 이중 다중처리 마이크로프로세서(on-chip multiprocessor microprocessor)[20~22]는 ILP의 한계를 극복함과 동시에 구현의 용이성을 제공한다는 점에서 차세대 마이크로프로세서 구조로 급부상하고 있다. 즉, ILP 외에 쓰레드 수준의 병렬성(Thread-Level Parallelism, TLP)[18~23]을 추가로 제공하고, 반도체 집적도가 증가함에 따라 간단한 구조의 프로세서 유니트를 여러 개 장착함으로써 슈퍼스칼라 구조의 단점을 보완할 수 있다.

그러나 다중처리 마이크로프로세서가 과학계산 응용에서는 우수한 성능을 나타내는 것으로 발표되고 있지만[20~22], 아직까지 암호화 응용에 적용시킨 결과는 발표되지 않고 있다. 따라서 본 논문에서는 새로운 마이크로프로세서 구조로 급부상하고 있는 다중처리 마이크로프로세서 구조가 차세대 블록 암호화 알고리즘에 적합한지를 살펴본다. 특히, 암호화 알고리즘을 고속으로 처리하기 위한 관점에서 ILP 및 TLP를 분석한다. 그리고 본 연구에서는 차세대 블록 암호화 알고리즘 후보 중 ILP가 가장 떨어진다고 알려진 RC6[24]와 ILP가 가장 높다고 알려진 Rijndael[25]에 대해서 분석한다.

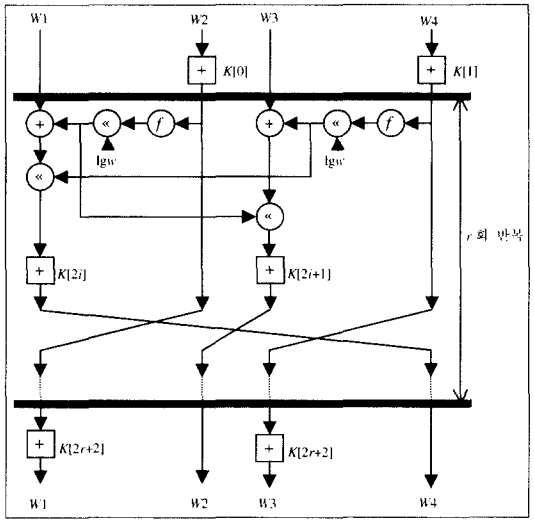
본 논문의 구성은 다음과 같다. 먼저 II장에서 블록 암호화 알고리즘에 대해서 서술하고, III장에서는 다중처리 마이크로프로세서를 설명한다. 그리고 프로그램 구동방식의 마이크로프로세서 시뮬레이터 상에서 수행한 병렬성 분석 결과를 IV장에서 설명하며, 마지막으로 V장에서 결론을 맺는다.

II. 블록 암호화 알고리즘

2.1 RC6

RC6는 RC5[26]를 수정하여 NIST의 AES 후보로 제출된 새로운 128-bit 데이터 블록 암호화 알고리즘이다. 수정된 부분은 주로 AES 요구사항에 따른 수정과 보안성을 증가시키기 위한 수정, 그리고 성능을 향상시키기 위한 수정으로 이루어진다. 일반적인 RC6는 워드 크기 w, 라운드 수 r, 키 크기 k 3개의 변수로 규정되는데, AES에 제출된 RC6는 워드 크기 32-bit, 라운드 수 20, 키 크기 128/196/256비트를 갖는다. 또한 RC6의 기본 동작은 다음의 연산으로 이루어지며, 이러한 기본 연산으로 구성된 암호화 알고리즘의 흐름도를 [그림 1]에 나타낸다.

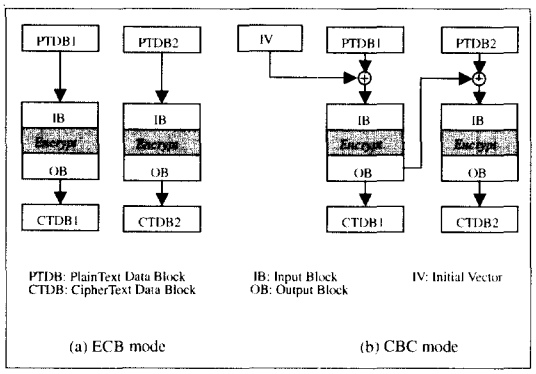
- a+b : 정수 덧셈 모듈러 2^w
- a-b : 정수 뺄셈 모듈러 2^w
- a⊕b : w-bit 워드의 비트레벨 exclusive-or
- a×b : 정수 곱셈 모듈러 2^w
- a≪b : b의 least significant lg w 비트 크기만큼 w-bit 워드 a를 left rotate



〔그림 1〕 RC6 블록 암호화 알고리즘의 흐름도[24]

〔그림 1〕에서 암호화할 입력 블록은 4개의 w비트 레지스터 $W1$, $W2$, $W3$, $W4$에 저장되고, w비트 크기를 갖는 라운드 키는 $K[0,\cdots,2r+3]$에 저장되어 있다고 가정한다. 그리고 $f(x)$는 $x\times(2x+1)$의 함수를 의미하며, r회의 라운드를 거친 후 생성된 암호문(ciphertext) 블록은 다시 $W1$, $W2$, $W3$, $W4$에 저장된다.

또한 임의 길이를 갖는 평문(plaintext)을 암호화시키는 방법으로 AES에서는 두가지 모드를 지정하는데, Electronic Codebook(ECB) 모드에서는 하나의 평문 데이터 블록이 입력 블록으로 직접 사용된 후, 〔그림 1〕의 과정을 거쳐서 암호문 블록을 생성해 낸다. 반면, Cipher Block Chaining(CBC) 모드에서는 임의 길이의 평문을 여러 개의 평문 데이터 블록으로 분해한 후, 처음 평문 데이터 블록과 128비트 크기의 초기화 벡터 IV를 exclusive-or하여 처음 입력 블록을 생성해 낸다. 그리고 〔그림 1〕의 암호화 과정을 거쳐서 생성된 출력 블록은 그 블록의 암호문 블록으로 사용됨과 동시에, 두번째 평문 데이터 블록과 exclusive-or되어 두번째 입력 블록을 생성해 낸다. 즉, 평문 블록과 암호문 블록을 "사슬" 방식으로 묶어서 암호화 과정을 수행한다[2]. 두가지 모드를 차이를 요약하면 〔그림 2〕와 같다. 예를 들어, 256비트 크기의 평문이 주어졌을 때, 먼저 두개의 평문 데이터 블록(PTDB1, PTDB2)로 나눈다. ECB 모드에서는 두개의 블록 암호화 과정에 데이터 종속성이 없으므로 병렬로 암호가 가능하나, CBC 모드에서는 처음 평문 데이터 블록(PTDB1)의



〔그림 2〕 ECB 모드와 CBC 모드[2]

암호화 결과를 두번째 평문 데이터 블록(PTDB2) 암호화에 이용하므로 두개의 데이터 블록을 순차적으로 암호화해야 한다.

2.2 Rijndael

Rijndael은 SQUARE[27]를 수정하여 NIST의 AES 후보로 제출된 새로운 블록 암호화 알고리즘으로 다음과 같은 특징을 갖는다.

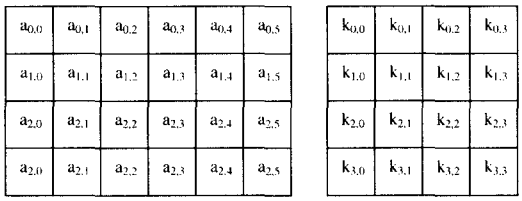
- 블록 크기: 가변 길이
- 키 길이: 가변 길이
- 라운드 수: 가변적

즉, Rijndael은 각각 128/192/256-비트를 갖는 가변 길이 블록과 가변 길이 키를 갖는 블록 암호로, 라운드 수는 블록 크기와 키 길이에 의해 결정된다. 32비트 워드로 표현된 라운드 수 N_r, 블록 크기 N_b, 키 길이 N_k의 관계를 요약하면 〔표 1〕과 같다.

〔그림 3〕에 $4\times N_b$ 배열로 표현되는 중간 암호 결과인 state와 $4\times N_k$ 배열로 표현되는 암호 키를 나타낸다. 또한, Rijndael에 의해 사용되는 입력과 출력은 0부터 $4*N_b-1$까지의 인덱스를 갖는 바이트의 일차원 배열로 표현되며, 암호 키 역시 0부터

〔표 1〕 라운드 수, 블록 크기, 키 길이의 관계

		N_b		
		4	6	8
N_k	4	10	12	14
	6	12	12	14
	8	14	14	14



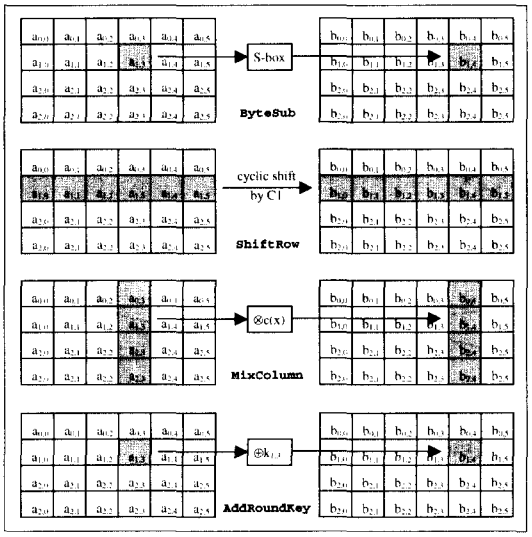
〔그림 3〕 State(N_b=6)와 암호 키(N_k=4) 표시의 예[27]

$4*N_k-1$까지의 인덱스를 갖는 바이트의 일차원 배열로 표현된다. 그리고 입력 바이트는 $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$, $a_{1,1}$, $a_{2,1}$, $a_{3,1}$,… 순서로 state 바이트에 매핑되고, 암호 키 바이트는 $k_{0,0}$, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, $k_{0,1}$, $k_{1,1}$, $k_{2,1}$, $k_{3,1}$,… 순서로 매핑된다.

그리고, 일반적인 라운드는 ByteSub, ShiftRow, MixColumn, AddRoundKey의 4가지 변환으로 구성되지만, 마지막 라운드에서는 ByteSub, ShiftRow, AddRoundKey 3가지 변환만 수행된다.

- ByteSub(*state*) 변환은 *state* 바이트 각각에 독립적으로 적용되는 non-linear byte substitution이며, ByteSub의 inverse는 inverse S-box가 적용되는 byte substitution이다.
- ShiftRow(*state*) 변환에서는 *state* 행이 다른 값으로 순환 쉬프트되는데, 이때 쉬프트 값 C1, C2, C3는 블록 크기 N_b에 의해 결정된다. ShiftRow의 inverse는 하위 3행을 각각 N_b-C1, N_b-C2, N_b-C3의 순환 이동한다.
- MixColumn(*state*) 변환에서 *state* 열은 $GF(2^8)$ 상의 polynomial로 간주하고, 고정된 polynomial c(x)와 modulo x^4+1로 곱해진다. 즉, c(x)='03'x^3+'01'x^2+'01'x+'02'이며, 이는 행렬곱셈으로 표시될 수 있다. 여기서 '03', '01', '02' 등은 hexa decimal 값을 의미한다. 또한 MixColumn의 inverse는 MixColumn과 유사하며, 각 열이 d(x)와 곱해진다. 즉, d(x)='0B'x^3+'0D'x^2+'09'x+'0E'로 표현된다.
- AddRoundKey(*state, roundkey*) 변환은 *roundkey*를 *state*와 비트별로 XOR하며, *roundkey*는 키 스케줄을 이용하여 암호 키로부터 유도된다. *roundkey* 길이는 블록 크기 N_b와 동일하다.

이러한 4가지 변환 동작을 요약하면 〔그림 4〕와 같고, 키 스케줄 등 Rijndael 알고리즘에 대한 보다 자세한 내용은 〔27〕에 설명되어 있다.



〔그림 4〕 ByteSub, ShiftRow, MixColumn, AddRoundKey 변환[27]

Ⅲ. 다중처리 마이크로프로세서

3.1 마이크로프로세서 구조

명령어 수준의 병렬성(Instruction-Level Parallelism, ILP) 지원을 위한 마이크로프로세서 구조로 슈퍼스칼라(superscalar)와 Very Long Instruction Word(VLIW)가 활발히 연구되고 있으며, 현재 발표되고 있는 상용 마이크로프로세서들은 대부분 슈퍼스칼라 방식을 채택하고 있다. 슈퍼스칼라[14,15] 방식의 가장 큰 특징은 동일한 파이프라인 구조를 갖는 연산장치를 여러 개 병렬로 사용하여 여러 개의 명령어를 동시에 수행시키는 기법이다. 현재 사용되고 있는 상용 마이크로프로세서 중 Pentium, PA7100, PowerPC603, Alpha는 동시에 2개의 명령어를 수행하며(2-issue), 4개 이상의 명령어를 동시에 수행시키는 Power2, UltraSparc, PowerPC620, PA8000 등도 최근에 개발되었다. 특히, 슈퍼스칼라 방식은 선형으로 구성된 명령어 흐름에서 하드웨어가 상호의존성이 없는 명령어를 골라내어 동시에 수행시키는 방식이기 때문에, 명령어간 상호의존성 조사를 위한 하드웨어가 필요하다. 또한, 슈퍼스칼라 방식의 성능을 극대화하기 위하여 분기 예측, 레지스터 재명명, 비순서 실행 등의 기술들이 요구된다. 따라서 슈퍼스칼라 방식은 동시에 처리하는 명령어의 수가 증가할수록 하드웨어 복잡도가 기하급수적으로 증가하여 실행 싸이클 시간을 증가시킨다. 또

한, 명령어 구성 비율에 따라 자원활용도의 편중이 발생하여, 일반적으로 8-issue 이상에 대해서는 회의적인 전망이 지배적이다.

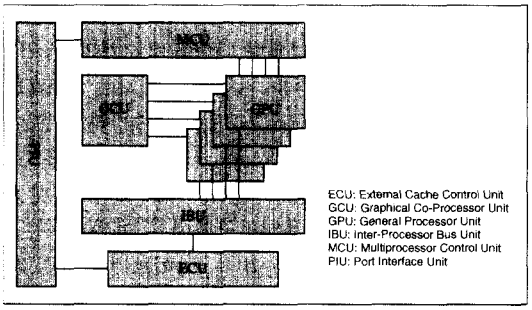
슈퍼스칼라 방식이 ILP를 위하여 하드웨어에 의한 동적 스케줄링 방식을 사용하는 반면, VLIW[28] 방식은 컴파일러에 의한 정적 스케줄링 방식을 사용한다. 즉, VLIW 방식에서는 컴파일러가 선형 명령어 흐름에서 병렬 수행 가능한 명령어들을 찾아내고, 이 명령어들을 하나의 긴 명령어로 만든다. 이때 하드웨어는 복수 개의 단위 명령어로 구성된 하나의 긴 명령어를 받아 단위 명령어들을 동시에 수행시킨다. VLIW 방식은 슈퍼스칼라 방식에 비하여 하드웨어 복잡도가 낮은 장점이 있지만, 컴파일러 개발에 어려움이 많고 마이크로프로세서 상호간 호환성을 보장할 수 없는 단점이 있다.

따라서 ILP 기반의 마이크로프로세서 구조를 대신할 새로운 구조에 대한 연구가 활발히 진행 중에 있으며[18~23], 이중 다중처리 마이크로프로세서(on-chip multiprocessor microprocessor)[20~22]는 ILP의 한계를 극복함과 동시에 구현의 용이성을 제공한다는 점에서 차세대 마이크로프로세서 구조로 급부상하고 있다. 즉, 간단한 구조의 프로세서 유니트를 이용하여 한 쓰레드 내의 크지 않은 ILP를 지원하면서, 여러 개의 프로세서 유니트를 이용하여 여러 개의 쓰레드를 동시에 실행시킴으로써 쓰레드 수준의 병렬성(Thread-Level Parallelism, TLP)[18~23]을 추가로 제공한다. 예를 들어, IBM에서는 두개의 프로세서 유니트를 한 칩에 내장한 Power4를 개발하여, 2001년에 시장에 출하할 예정이다[29].

3.2 Raptor

본 논문에서는 다중처리 마이크로프로세서로 *Raptor*[30]를 사용한다. Raptor는 한국전자통신연구원에서 차세대 마이크로프로세서로 개발한 단일 칩 다중처리 마이크로프로세서로, General Processor Unit(GPU)라는 4개의 독립적인 프로세서 유니트와 Graphic Co-processor Unit(GCU)라는 하나의 그래픽 연산 유니트로 구성된다. Raptor의 주요 특징은 다음과 같다:

- 각각의 프로세서 유니트는 SPARC V9 명령어 세트 구조를 가지며, 64비트 정수 및 부동 소수점 연산을 한다.
- 4개의 독립된 프로세서 유니트가 그래픽 연산 유니트를 공유하여 그래픽 연산을 처리한다.
- 크기가 비교적 작은 내장 1차 캐쉬와 대용량의 외부 2차 캐쉬로 구성된 다중 캐쉬 구조를 갖는다.
- 2차 캐쉬 제어기를 내장하며 여러 개의 Raptor 프로세서를 이용한 시스템 구현을 지원한다.



[그림 5] Raptor 마이크로프로세서의 블록 구성도

Raptor의 블록 구성도를 나타내면 [그림 5]와 같다. 여기서 Inter-processor Bus Unit(IBU)은 GPU와 External cache Control Unit(ECU)을 연결하는 공유 버스이다. 그리고 Multiprocessor Control Unit(MCU)는 인터럽트를 GPU로 분배하고 GPU간의 동기를 지원한다. 반면, Port Interface Unit(PIU)는 칩 내부에서 발생하는 메모리 접근 요구, 입출력 장치 접근 요구, 인터럽트 및 기타 유틸리티 신호선 사용 요구를 처리한다. 특히, 부가적인 외부 로직없이 다중처리 시스템을 구성할 수 있는 multiprocessor-ready 형태의 인터페이스를 제공한다. 그리고 4개의 GPU는 각각의 레지스터 파일과 프로그램 카운터로 그래픽 명령어를 제외한 모든 명령어를 실행하지만, IBU를 통하여 ECU를 공유한다. 마지막으로 GCP는 4개의 GPU에 의해 Single Instruction stream Multiple Data stream(SIMD) 방식으로 그래픽 명령어를 처리한다.

3.3 RapSim

본 연구에서는 다중처리 마이크로프로세서에서 차세대 블록 암호화 알고리즘 RC6의 병렬성을 정성적으로 분석하기 위하여 한국전자통신연구원에서 개발한 *RapSim*[31]을 이용하였다. RapSim은 4개의 GPU와 이들에 의해 공유되는 메모리 계층을 모델링하는 프로그램 구동 방식의 마이크로아키텍쳐 시뮬레이터이다. 여기서 각각의 GPU 모델은 명령어세트 시뮬레이터인 전처리기(Pre-Processing

Unit)와 성능 시뮬레이터인 후처리기(Post-Processing Unit)로 구성된다. 또한 복수개의 GPU 상에서 멀티쓰레드 환경을 지원하기 위하여 *MMOS* (Multi- threaded Mini-OS)[31]라고 하는 프로그래밍 환경을 이용하였다. 즉, MMOS는 응용 프로그램에게 사용자 수준의 멀티쓰레드 환경을 제공하기 위하여 한국전자통신연구원에서 개발한 프로그래밍 환경으로, Pthread[32] 라이브러리, C 라이브러리, RapSim 인터페이스를 제공한다. RapSim의 주요 특징은 다음과 같다:

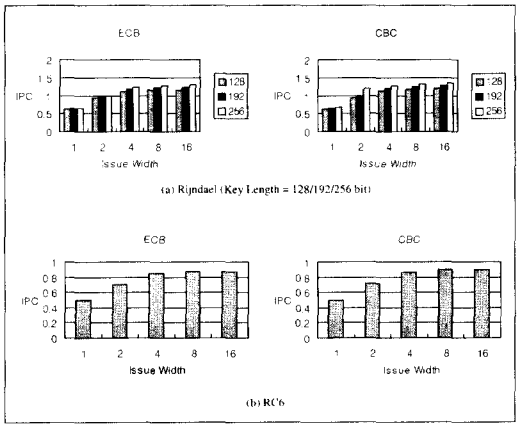
- SPARC V9 명령어와 그래픽 연산 유니트용 명령어의 수행
- 시간 정보를 갖는 프로그램 구동 방식의 시뮬레이터
- 4개의 프로세서 유니트 및 1개의 그래픽 연산 유니트로 구성된 다중처리기 모델
- 프로세서 유니트의 비순서(out-of-order) 실행 지원
- 멀티쓰레드 프로그래밍 모델 지원
- 성능 측정을 위한 정보 수집

Ⅳ. 병렬성 분석

본 논문에서는 1개의 GPU상에서 수행한 순차 실행(Sequential), 2개의 GPU상에서 2개의 쓰레드를 이용한 실행(2-Threads), 4개의 GPU상에서 4개의 쓰레드를 이용한 실행(4-Threads)에 대한 시뮬레이션을 수행하였다. 그리고 RC6 및 Rijndael 알고리즘의 구현은 AES에 제출된 C 프로그램[2]을 이용하였고, 다중처리 마이크로프로세서의 성능 지수로는 Instruction Per Cycle(IPC)를 설정하였다. [표 2]에 본 논문에서 사용한 시뮬레이션 변수의 기본값을 나타내었다.

[표 2] 시뮬레이션 변수

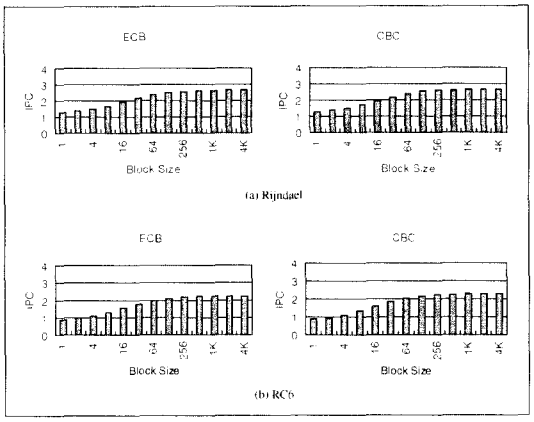
항 목	기 본 값
GPU 수	1,2,4
GPU 이슈 크기	1,2,4,8,16
1차 캐쉬 크기	16Kbyte I-cache, 16Kbyte D-cache/ 32 bytes line
2차 캐쉬 크기	4Mbyte/32 bytes line
쓰기 갱신	1차 캐쉬에서 2차 캐쉬 : write through 2차 캐쉬에서 메인메모리 : write back
1차 캐쉬 접근 시간	1싸이클
2차 캐쉬 접근 시간	4싸이클
메인메모리 접근 시간	10싸이클
명령어 수행	integer ALU=1 싸이클 Integer Multiply=4~34 싸이클 Integer Division=36(single), 68(double) 싸이클 Load/Store=1 싸이클 Control Transfer=1 싸이클 FP Addition/Subtraction=1 싸이클 FP Multiply=4 싸이클 FP Division=12(single), 22(double) 싸이클



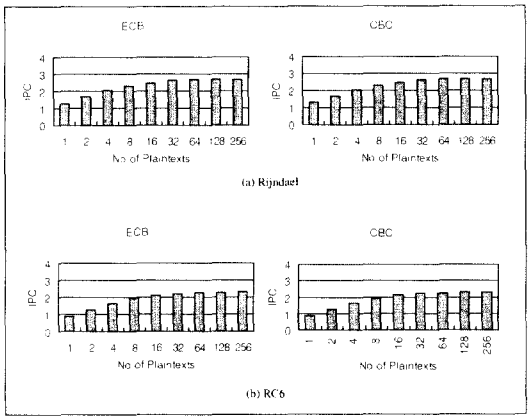
[그림 6] ECB 및 CBC 모드에서 이슈 크기의 효과

먼저 슈퍼스칼라 구조가 제공하는 명령어 수준 병렬성의 성능을 확인하기 위하여, GPU 이슈 크기를 변화시키면서 1개의 GPU 상에서 순차 실행시킨 결과를 [그림 6]에 나타내었다. 먼저 한 블록 크기를 갖는 하나의 평문을 암호화하는 경우, ECB 및 CBC 모드에서 모두 이슈 크기가 8이 될 때까지는 성능이 개선되지만, Rijndael 및 RC6 각각에서 이슈 크기를 16까지 증가시키더라도 명령어 수준의 병렬성만으로는 IPC가 1.4와 0.9를 넘지 못함을 알 수 있다. 또한 [14]에서 언급한바와 같이, Rijndael이 RC6에 비하여 높은 IPC를 갖는데, 이는 Rijndael의 각 round에 보다 많은 병렬성이 내재되어 있기 때문이다.

다음에 이슈 크기를 8로 고정시킨 후, 여러 블록 크기를 갖는 한 개의 평문과 한 개의 블록 크기를 갖는 여러 개의 평문을 암호화하는 경우 IPC의 변화를 살펴보자. [그림 7]에서 알 수 있듯이, 평문당 1024 블록 크기를 가질 때까지 IPC는 계속 증가하여



〔그림 7〕 ECB 및 CBC 모드에서 평문 크기의 효과



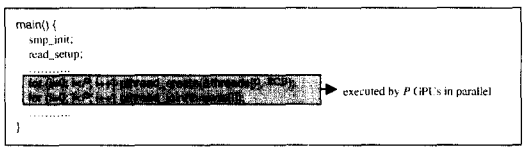
〔그림 8〕 ECB 및 CBC 모드에서 평문 개수의 효과

2.8(Rijndael) 및 2.3(RC6)에서 포화됨을 알 수 있다. 또한 〔그림 8〕에서 알 수 있듯이 평문당 블록 크기를 고정시킨 후 평문의 수를 증가시켜도 유사한 패턴의 그래프를 얻을 수 있다. 즉, 128개의 평문을 암호화하는 경우 2.7(Rijndael) 및 2.2(RC6)까지 IPC가 증가함을 알 수 있다. 그리고 평문당 블록 크기를 증가시키는 경우보다 평문의 수를 증가시키는 경우에 더욱 빨리 포화점에 접근하는데, 이는 평문당 수행하는 키 생성 루틴(makeKey)이 블록당 수행하는 암호처리 루틴(blockEncrypt)보다 더욱 쉽게 명령어 수준 병렬성을 활용하기 때문이다.

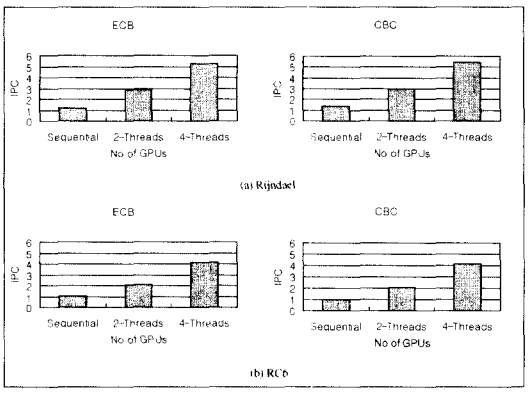
또한 암호화 모드간의 성능을 비교하면, ECB 모드에서 암호화할 블록간 데이터 종속성이 없어서 블록간 데이터 종속성이 있는 CBC 모드에 비하여 더 높은 IPC를 제공할 수 있음에도 실제로는 그렇지 못하다. 이는 슈퍼스칼라 프로세서에서 활용하는 명령어 수준 병렬성의 크기(granularity)가 블록 레벨 병렬성에 비하여 월등히 작기 때문이다. 즉, fine-grain 병렬성에 적합한 슈퍼스칼라 구조로는 coarse-grain 병렬성을 충분히 활용하지 못함을 알 수 있다. 따라서 더 높은 성능을 제공하기 위해서는 쓰레드 수준의 병렬성과 같은 coarse-grain 병렬성의 활용이 추가로 필요함을 알 수 있다.

다음은 이슈 크기를 2로 고정시킨 후, GPU를 여러 개 활용하여 쓰레드 수준의 병렬성을 제공하는 경우에 대해 살펴보자. 쓰레드 수준 병렬성의 효과를 확인하기 위해, Pthread[32]를 이용한 병렬 RC6 프로그램의 예를 〔그림 9〕에 나타내었다. II장에서 언급한대로 ECB 모드에서는 각 블록별 동시수행이 가능하므로, 각각의 평문 또는 각각의 평문 블록을 여러 GPU에 할당하여 쓰레드 수준 병렬성을 활용할 수 있다. 그러나 CBC 모드에서는 하나의 평문을 구성하는 각각의 블록 처리에 데이터 종속성이 존재하기 때문에, 여러 개의 평문을 동시에 처리하는 병렬 수행만이 가능하다. 하지만 다중처리 마이크로프로세서와 같은 고성능 프로세서가 서버에 탑재되고 이러한 고성능 서버에서 여러 개의 평문 데이터 스트림이 암호화 또는 복호화된다고 가정한다면[10], CBC 모드에서의 병렬 수행도 적극 활용 가능하다.

〔그림 10〕에 Pthread로 병렬화된 프로그램을 이용하여 쓰레드 수를 증가시킬 때 수행 싸이클 수와



〔그림 9〕 Pthread를 이용한 ECB 모드 병렬화



〔그림 10〕 순차 및 쓰레드를 이용한 실행시 IPC 비교

IPC의 변화를 나타내었다. 각각의 GPU에서 최소 1개 이상의 평문을 처리하기 위하여 전체적으로는 4개의 평문을 처리해야 한다고 가정한다. [그림 10]에서 알 수 있듯이, 쓰레드 수를 증가시키면 IPC는 증가한다. 즉, Rijndael과 RC6는 작은 오버헤드로 쓰레드 수준의 병렬성을 적절히 활용하여 선형적인 성능 향상을 기대할 수 있다. 여기서 병렬 쓰레드 이용시 싸이클 수는 parent process를 수행하는 GPU0의 싸이클 수를 의미한다.

V. 결 론

본 논문에서는 차세대 마이크로프로세서 구조로 급부상하고 있는 다중처리 마이크로프로세서를 이용하여 RC6 및 Rijndael 블록 암호화 알고리즘을 수행할 때 어떠한 성능 특성을 나타내는지 분석하였다. 성능 분석을 위하여 마이크로프로세서 시뮬레이터를 이용한 프로그램 구동 시뮬레이션을 수행하였고, IPC를 분석하였다.

시뮬레이션 수행 결과, GPU 수가 증가함에 따라 IPC도 거의 선형적으로 증가함을 확인하였다. 즉, 슈퍼스칼라 구조에서 명령어 수준의 병렬성만을 활용할 경우 2.3(RC6) 또는 2.8(Rijndael) 정도의 IPC를 기대할 수 있으나, 4개의 GPU로 구성된 다중처리 마이크로프로세서 구조에서 쓰레드 수준의 병렬성을 추가로 활용할 경우 4.1(RC6) 또는 5.5(Rijndael) 이상의 IPC를 얻을 수 있었다. 따라서 다중처리 마이크로프로세서는 명령어 수준 병렬성이 큰 Rijndael과 적은 RC6 각각에서 매우 효과적인 차세대 마이크로프로세서 구조임을 확인하였다.

참 고 문 헌

[1] National Bureau of Standards, "Data Encryption Standard," *FIPS Pub.*, 46, 1977.

[2] NIST, "Advanced Encryption Standard Development Effort," *http://csrc.nist.gov/encryption/aes.*

[3] H. Lipmaa, "AES Candidates: A Survey of Implementations," *Proc. of the Second AES Conf.*, 1999.

[4] B. Gladman, "Implementation Experience with AES Candidate Algorithms," *Proc. of the Second AES Conf.*, 1999.

[5] B. Schneier, et al., "Performance Comparison of the AES Submissions," *Proc. of the Second AES Conf.*, 1999.

[6] E. Biham, "A Note on Comparing the AES Candidates," *Proc. of the Second AES Conf.*, 1999.

[7] B. Schneier and D. Whiting, "A Performance Comparison of the Five AES Finalists," *Proc. of the Third AES Conf.*, 2000.

[8] M. Takenaka, et al., "Performance Comparison of 5 AES Candidates with New Performance Evaluation Tool," *Proc. of the Third AES Conf.*, 2000.

[9] L. Bassham, "Efficiency Testing of ANSI C Implementations of Round 2 Candidate Algorithms for the Advanced Encryption Standard," *Proc. of the Third AES Conf.*, 2000.

[10] R. Weiss and N. Binkert, "A Comparison of AES Candidates on the Alpha 21264," *Proc. of the Third AES Conf.*, 2000.

[11] J. Worley, et al., "AES Finalists on PA-RISC and IA-64: Implementations & Performance," *Proc. of the Third AES Conf.*, 2000.

[12] T. Wollinger, et al., "How Well Are High-End DSPs Suited for the AES Algorithms?," *Proc. of the Third AES Conf.*, 2000.

[13] A. Elbirt, et al., "An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists," *Proc. of the Third AES Conf.*, 2000.

[14] C. Clapp, "Instruction-level Parallelism in AES Candidates," *Proc. of the Second AES Conf.*, 1999.

[15] D. Culler, J. Singh, and A. Gupta, *Parallel Computer Architecture? A Hardware/Software Approach*, Morgan Kaufmann, Pub., 1998.

[16] M. Slater, "The Microprocessor Today,"

IEEE Micro, Vol. 16, No. 6, pp. 32~45, 1996.

[17] D.Wall, "Limits of Instruction Level Parallelism," *WRL Research Report*, Digital Western Research Laboratory, 1993.

[18] J. Wilson, "Challenges and Trends in Processor Design," *IEEE Computer*, Vol. 31, No. 1, pp. 39~50, 1998.

[19] S. Egger, et al., "Simultaneous Mutithreading: A Platform for Next-Generation Processor," *IEEE Micro*, Vol. 17, No. 5, pp. 12~19, 1997.

[20] B. Nayfe, L. Hammond, and K. Olukotun, "Evaluation of Design Alternatives for Multiprocessor Microprocessor," *Proc. of Int' l Symp. on Computer Architecture*, pp. 66~77, 1996.

[21] L. Hammond, et al., "A Single-Chip Multiprocessor," *IEEE Computer*, Vol. 30, No. 9, pp. 79~85, 1997.

[22] S. Amarashinhe, et al., "Multiprocessors From a Software Perspective," *IEEE Micro*, Vol. 16, No. 3, pp. 52~61, 1996.

[23] A. Agarwal, et al., "Performance Tradeoff in Multithreading Processors," *IEEE Tr. on Parallel and Distributed Systems*, Vol. 3, No. 5, pp. 525~539, 1992.

[24] R. Rivest, et al., "The RC6 Block Cipher," *Proc. of the First AES Conf.*, 1998.

[25] R. Rivest, "The RC5 Encryption Algorithm," *LNCS*, 1008, pp. 86~96, 1995.

[26] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," *Proc. of the First AES Conf.*, 1998.

[27] J. Daemen, L. Knudsen, and V. Rijmen, "The Block Cipher SQUARE," *LNCS*, 1267, pp. 149~165, 1998.

[28] J. Fisher, "Very Long Instruction Word Architecture and the ELI-512," *Proc. of Int' l Symp. on Computer Architecture*, pp. 140~150, 1983.

[29] K. Diefendorff, "Power4 Focuses on Memory Bandwidth," *Microprocessor Report*, Vol. 12, No. 12, 1999.

[30] 박경 외 3인, "다중처리 마이크로프로세서 설계," *대한전자공학회* 추계학술대회 논문집, pp. 717~720, 1996.

[31] K. Park, et al., "On-Chip Multiprocessor with Simultaneous Multithreading," *Technical Report*, ETRI, 1999.

[32] IEEE, POSIX P1003.4a: *Threads Extension for Portable Operating* Systems, 1994.

〈著者紹介〉

정 용 화 (Yong-wha Chung) 정회원
1984년 : 한양대학교 전자통신공학과 졸업
1986년 : 한양대학교 전자통신공학과 석사
1997년 : 미국 University of Southern California 컴퓨터공학과 박사
1986년~현재 : 한국전자통신연구원 책임연구원(팀장)
〈관심분야〉 암호알고리즘, 생체인식알고리즘, 병렬처리

정 교 일 (Kyo-il Chung) 정회원
1981년 : 한양대학교 전자공학과 졸업
1983년 : 한양대학교 산업대학원 전자계산학과 석사
1997년 : 한양대학교 전자공학과 박사
1981년~현재 : 한국전자통신연구원 책임연구원(부장)
〈관심분야〉 IC카드, 정보보호, 생체인식, 신호처리

손 승 원 (Sung-won Sohn) 정회원
1984년 : 경북대학교 전자공학과 졸업
1994년 : 연세대학교 전자공학 석사
1999년 : 충북대학교 컴퓨터공학과 박사
1991년~현재 : 한국전자통신연구원 책임연구원(부장)
〈관심분야〉 네트워크 보안, 라우팅 알고리즘, 생체인식기술