韓國 컴퓨터情報學會 論文誌
第11卷 第2號, 2006. 5.

2006-11-2-1-5

Parsec 기반 시뮬레이터를 이용한 다중처리시스템의 성능 분석

이 원 주*, 김 선 욱**, 김 형 래*

Performance Analysis of a Multiprocessor System Using Simulator Based on Parsec

Won Joo Lee*, Sun Wook Kim**, Hyeong Rae Kim*

요 약

본 논문에서는 Parsec을 이용하여 병렬디지털신호처리용 분산공유메모리 다중처리시스템의 성능 분석을 위한 시뮬레이터를 구현한다. 이 시뮬레이터의 특징은 TMS320C6701 DSP 칩의 DMA 기능과 빠른 접근시간을 가진 지역메모리를 사용하는 시스템의 시뮬레이션에 적합하다는 것이다. 또한 시스템의 성능 매개변수 수정과 하드웨어 구성요소들에 대한 재구성이 쉽기 때문에 다양한 실행 환경에서 시스템의 성능을 분석할 수 있다. 시뮬레이션에서는 DSP 알고리즘에서 널리 사용하는 FFT, 2D FFT, Matrix Multiplication, Fir Filter를 사용하여 프로세서의 수, 데이터 크기, 하드웨어 요소의 변화에 따른 실행 시간을 측정한다. 그리고 그 결과를 비교함으로써 본 논문에서 구현한 시뮬레이터의 성능을 검증한다.

Abstract

In this paper, we implement a new simulator for performance analysis of a parallel digital signal processing distributed shared memory multiprocessor systems, using Parsec. The key idea of this simulator is suitable in simulation of system that uses DMA function of TMS320C6701 DSP chip and local memory which have fast access time. Also, because correction of performance parameter and reconfiguration for hardware components are easy, we can analyze performance of system in various execution environments. In the simulation, FFT, 2D FFT, Matrix Multiplication, and Fir Filter, which are widely used DSP algorithms, have been employed. Using our simulator, the result has been recorded according to different the number of processor, data sizes, and a change of hardware element. The performance of our simulator has been verified by comparing those recorded results.

▸ Keyword : Parsec, PCI 버스, TMS320C6701 DSP, D FFT(fast fourier Transform), FIR(finite impulse response)

• 제1저자 : 이원주
• 접수일 : 2006.02.16 심사완료일 : 2006.05.18
* 두원공과대학 인터넷프로그래밍과 부교수
** 한국전자통신연구원 서버플랫폼연구팀 연구원

Ⅰ. 서 론

현재 컴퓨터 기술이 빠른 속도로 발전함에 따라 많은 양의 데이터 처리와 빠른 실행시간을 요구하는 응용프로그램이 점차 증가하고 있다. 이러한 응용프로그램들을 효율적으로 처리하기 위한 하나의 방법은 다수의 고성능 프로세서를 장착한 다중처리시스템을 이용하는 것이다[1]. 다중처리시스템의 개발 과정에는 시스템의 특성과 성능을 예측하는 성능분석 과정이 필수적으로 포함된다.

다중처리시스템의 성능을 분석하기 위해서는 시뮬레이션을 많이 사용하고 있다. 시뮬레이션의 장점은 실제 시스템과 유사한 시스템을 모델링하여 성능을 분석할 수 있다는 것이다[2]. 시뮬레이션에는 이산사건(discrete -event) 모델용 시뮬레이터[3]와 Multisim [4], SimOS[5], PARSIM[6] 시뮬레이터 등이 사용된다. 이러한 시뮬레이터는 시스템 구성요소들의 특징을 표현하기에 부족하기 때문에 상용 프로세서와 버스를 사용하는 병렬디지털신호처리용 분산공유메모리 다중처리시스템의 성능 분석에 적합하지 않다. 또한 기존의 시뮬레이터를 구현한 언어를 이해하여 새로운 기능을 추가하는 과정에 많은 시간과 비용이 소요된다.

본 논문에서는 병렬디지털신호처리용 분산공유메모리구조 다중처리시스템의 성능 분석을 위한 시뮬레이터를 설계한다. 그리고 이산사건 시뮬레이션 언어인 Parsec[7][8]을 이용하여 시뮬레이터를 구현한다. 또한 이 시뮬레이터를 사용하여 다양한 환경에서 다중처리시스템의 성능을 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 성능분석의 대상인 분산공유메모리 다중처리시스템에 대해 설명하고 3장에서는 Parsec 기반 시뮬레이터에 대해 설명한다. 그리고 4장에서는 시뮬레이터를 사용하여 2D FFT (fast fourier transform) 및 2차원 행렬 곱셈, FIR (finite impulse response) 필터 응용프로그램을 다양한 환경에 적용시켜 얻은 결과를 통해 전체적인 시스템의 성능을 살펴보고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1 분산 공유메모리 다중처리시스템

본 논문에서는 고속신호처리용 분산공유메모리구조 다중처리시스템의 성능을 분석한다. 이 시스템은 여러 개의 프로세서가 버스를 통해 연결된 SIMD(Simple Data Multiple Data)형 시스템으로 그림 1과 같은 구조를 가진다.

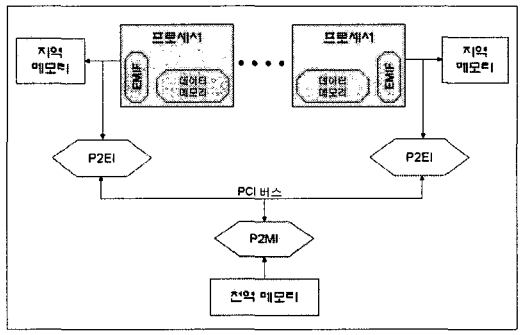

그림 1. 분산공유메모리 다중처리시스템 구조
Fig.1 Distributed shared memory multiprocessor system architecture

그림 1에서 지역 메모리는 각각의 프로세서에 분산되어 있지만 단일 주소공간을 사용하기 때문에 모든 프로세서가 접근할 수 있다. 전역 메모리는 지역 메모리와 시스템에 사용되는 대량의 데이터를 저장하는 기능을 한다[9]. 프로세서는 TI(texas instruments) 사의 TMS320C6701 DSP(digital signal processor) 칩으로 산술연산 기능이 우수하며, 내부에 데이터 메모리를 가지고 있어 프로그램 수행 속도가 빠르다. 또한 프로세서는 32bit EMIF(external memory interface)를 통하여 지역메모리에 접근 할 수 있다. 지역 메모리는 EMIF에서 지원하는 SDRAM 및 SBSRAM을 사용하며 전역 메모리는 대량의 데이터 전송을 위해 SDRAM을 사용한다. 전역메모리와 프로세서 또는 지역메모리와의 데이터 전송은 PCI 버스를 통하여 이루진다. PCI 버스와 EMIF간의 연결은 PCI-to-EMIF (P2EI) 인터페이스를 사용하고, PCI 버스와 전역메모리의 연결은 PCI-to-Memory (P2MI) 인터페이스를 사용한다.

2.2 Parsec

Parsec(PARallel Simulation Environ- ment for Complex Systems)은 이산사건 기반의 시뮬레이션 환경이다. Parsec의 장점은 시뮬레이터 구현 과정에서 복잡한 자료구조 선언 및 메모리 관리가 용이하며 다양한 기능의 C 언어 함수들을 사용할 수 있다는 것이다. 또한 Parsec은 워크스테이션이나 PC, MPP (massively parallel processor), SMP (symmetric multiprocessor) 등의 다양한 시스템에 적용할 수 있다. 그리고 병렬 프로그램이나 이동무선멀티미디어 네트워크, ATM 네트워크, VLSI 설계 등의 다양한 응용 프로그램 시뮬레이션도 가능하다.

Parsec에서 실제 시스템의 하드웨어 구성요소는 엔티티(entity)로 표현한다. 엔티티는 엔티티명과 파라미터를 포함하는 헤드 영역과 지역변수, 실행코드로 구성되며 C언어의 함수 선언과 유사한 형태로 표현한다. 시뮬레이션 프로그램에는 C언어의 메인(main) 함수와 같은 기능을 수행하는 드라이버(driver) 엔티티가 존재한다. 이 엔티티는 프로그램 실행의 시작점으로 다른 엔티티를 생성하고 엔티티간의 메시지 전송 경로를 설정한다.

시뮬레이션 프로그램은 순차적으로 실행되는 이벤트로 구성된다. 이벤트가 발생하면 엔티티 간의 메시지 교환이 이루어진다. 실제 시스템에서는 이벤트 발생시간을 통하여 실행시간을 측정할 수 있다. 이벤트 발생시간은 각 메시지의 타임스탬프 (time stamp)로 알 수 있다. 하지만 시뮬레이션에서는 Parsec의 hold()함수를 이용하여 이벤트 처리 과정에서 요구되는 지연시간을 측정함으로써 알 수 있다.

III. Parsec 기반 시뮬레이터 구현

본 논문에서는 Parsec을 이용하여 고속신호처리용 분산 공유메모리 다중처리시스템의 성능을 분석할 수 있는 시뮬레이터를 구현한다. 이 시뮬레이터는 고속신호처리용 분산 공유메모리 다중처리시스템의 구성요소를 프로세서 모듈, 메모리 모듈, 버스 모듈, 인터페이스 모듈로 모델링 한다. 각 모듈의 하드웨어 구성요소들은 Parsec의 엔티티로 구현한다. 각 모듈의 엔티티는 표 1과 같다.

표 4. 하드웨어 구성요소 및 엔티티
Table 1. H/W component and entity

모듈	엔티티	기능
프로세서	CPU	TMS320C6701의 명령어 수행
	EMIF	프로세서와 지역메모리, PCI 버스와의 연결 기능
	DMA	지역메모리의 데이터 전송을 위한 DMA 기능
	DATA_ MEMORY	각 프로세서 내부의 데이터 메모리 기능
메모리	LOCAL_ MEMORY	각 프로세서의 지역 메모리 기능
	GLOBAL_ MEMORY	대량의 데이터 저장을 위한 전역 메모리 기능
버스	PCI	시스템의 구성요소 연결을 위한 PCI 버스 기능
인터페이스	P2EI	PCI 버스와 EMIF를 연결하는 인터페이스 기능
	P2MI	PCI 버스와 전역 메모리를 연결하는 인터페이스 기능

각 모듈은 엔티티이외에 드라이버 엔티티로 구현된다. 드라이버 엔티티에 의해 설정된 엔티티간의 메시지 전송 경로는 그림 2와 같다.

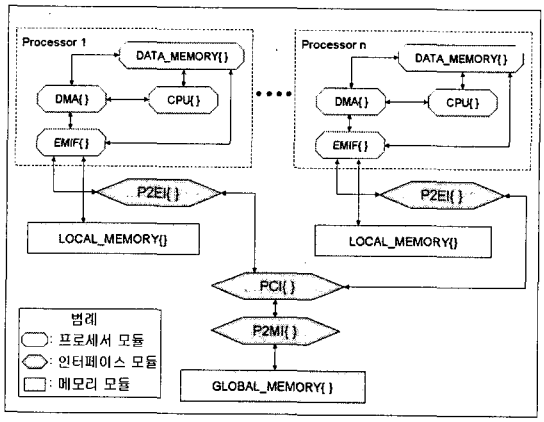

그림 2. 메시지 전송 경로
Fig. 2. Message transmission path

그림 2에서 프로세서 모듈과 메모리 모듈의 지역 메모리 간의 메시지 전송은 EMIF 엔티티를 통하여 이루어진다. 그리고 버스 모듈의 PCI 엔티티는 인터페이스 모듈의 P2EI와 P2MI 엔티티에 연결된다.

3.1 프로세서 모듈

TMS320C6701 DSP[10]는 CPU와 데이터 메모리, 프로그램 메모리, 호스트 포트 인터페이스, EMIF, DMA 등과 각 구성요소의 제어기로 구성되어 있다. 따라서 본 논문에서는 프로세서 모듈을 CPU와 데이터메모리, EMIF, DMA로 구성한다. 이외의 구성요소들은 사용되지 않거나 사용빈도가 적어 시스템의 성능에 영향을 주지 않기 때문에 프로세서 모듈에 포함하지 않는다.

CPU 엔티티는 TMS320C6701에서 제공하는 고정 소수점 연산과 부동 소수점 연산, 메모리 읽기/쓰기, DMA 수행을 위한 인터럽트 처리 부분으로 구현한다.

DATA_MEMORY 엔티티는 프로세서 내부의 데이터 메모리를 모델링한 것이다. 메모리는 2차원 배열 형태로 표현하고, 크기는 TMS320C6701에서 지정한 64KB 이다. 이 메모리는 부동 소수점 데이터의 경우 최대 16,384개, 고정 소수점 데이터의 경우 최대 32,768개까지 저장이 가능하다[11].

EMIF 엔티티는 프로세서와 지역 메모리간의 데이터 전송 메시지를 처리한다. 또한 다른 프로세서의 지역 메모리와 전역 메모리의 데이터 전송 메시지를 인터페이스 모듈에 전송한다.

DMA 엔티티는 데이터 메모리와 지역 메모리간 다량의 데이터를 전송하는 기능을 수행한다. 또한 다른 프로세서의 지역 메모리 또는 전역 메모리에 다량의 데이터를 전송할 때 전송 유형과 범위를 결정하여 DATA_ MEMORY 엔티티에 전송 요청 메시지를 전달한다.

프로세서 모듈의 각 엔티티들이 요청된 명령들을 실행하는데 소요되는 시간은 TMS 320C6701의 CPU 사이클 시간을 기준으로 한다. 일부 명령들에 대한 CPU 사이클 시간은 표 2와 같다[11].

표 5. TMS320C6701의 CPU 사이클 시간
Table 2. CPU cycle time of TMS320C6701

명령	CPU 사이클 타임	
	고정 소수점	부동 소수점
abs	1 CPU 사이클	1 CPU 사이클
+	1 CPU 사이클	4 CPU 사이클
==	1 CPU 사이클	1 CPU 사이클
〉	1 CPU 사이클	1 CPU 사이클
〈	1 CPU 사이클	1 CPU 사이클
*	2 CPU 사이클	4 CPU 사이클
-	1 CPU 사이클	4 CPU 사이클
/	1 CPU 사이클	30 CPU 사이클

3.2 메모리 모듈

LOCAL_MEMORY 엔티티는 각 프로세서와 연결된 지역 메모리로 8MB SDRAM을 대상으로 모델링하고 데이터 메모리와 동일하게 2차원 배열로 구현한다. 본 논문에서는 시뮬레이션을 위한 입력 데이터의 크기를 최대 1024로 제한했기 때문에 배열의 크기는 8MB 메모리를 2048*1024로 나타낸다.

GLOBAL_MEMORY 엔티티는 16MB SDRAM을 대상으로 하였으며 지역 메모리와 동일하게 2차원 배열로 구현한다. 메모리 모듈 시뮬레이션에서는 읽기, 쓰기 메시지를 받아서 지정 위치의 데이터를 처리하고 결과 메시지를 CPU 엔티티에 전송하는데 소요되는 지연시간을 측정한다. 각 메모리의 데이터 접근 시간은 메모리의 종류에 따라 조금씩 차이가 있다. 메모리 종류에 따른 지역 메모리의 접근시간을 CPU 사이클 시간으로 표현하면 표3과 같다.

표 3 지역 메모리의 접근시간
Table 3. Local memory access time

메모리 종류	데이터		비고
	읽기	쓰기	
SDRAM	17 / 18	7 / 8	메모리 읽기를 위한 정보 및 결과 전송시 EMIF 에서 소요되는 시간 포함
SBSRAM	15 / 16	7 / 8	메모리 쓰기를 위한 정보 전송시 EMIF에서 소요되는 시간 포함

전역 메모리에 대한 접근시간은 지역 메모리의 접근시간과 인터페이스 모듈에서 요구되는 소요시간을 합산하여 구한다.

3.3 버스 및 인터페이스 모듈

PCI 엔티티는 각 모듈을 연결해주는 PCI 버스를 모델링한 것으로 프로세서들에 의한 PCI 사용 요청을 FCFS (first come first service) 버스 중재방식으로 처리한다.

인터페이스 모듈에는 P2EI와 P2MI 엔티티가 존재한다. P2EI 엔티티는 PCI와 EMIF 엔티티를 연결하는 기능을 한다. P2MI 엔티티는 전역메모리와 PCI 버스를 연결하는 기능을 한다. 인터페이스 모듈의 각 엔티티들은 단지 하드웨어 구성요소들을 연결해주는 기능을 수행하기 때문에 시뮬레이션동안 데이터 전송에 소요되는 지연시간을 측정하여 전체 시뮬레이션 시간에 추가하면 된다. 즉, PCI 버스의 경우 프로세서에 의한 PCI 사용요청에 대한 처리지연시간과

데이터 전송에 소요되는 지연시간을 전체 시뮬레이션 시간에 추가한다. 각 인터페이스 및 PCI 버스의 소요시간은 PCI 엔티티 내부에서 변경할 수 있다.

IV. 다중처리시스템의 성능 분석

먼저 본 논문에서 구현한 시뮬레이터의 신뢰도를 검증한다. 그리고 다양한 응용프로그램을 실행시켜 고속신호처리용 분산공유메모리구조 다중처리시스템의 성능을 분석한다.

4.1 시뮬레이터 신뢰도

본 논문에서 구현한 시뮬레이터에서 1D FFT 응용프로그램을 실행한 결과와 TI 사에서 제공하는 TMS320C6701의 벤치마크 자료[12]를 비교함으로써 본 논문에서 구현한 시뮬레이터에 대한 신뢰도를 검증한다. 시뮬레이터 검증에 대한 신뢰도를 높이기 위해 TI 사에서 가정한 실행 환경과 동일한 환경에서 시뮬레이션을 진행한다. TI 사에서 가정한 실행 환경은 다음과 같다. 첫째 프로그램 실행에 필요한 모든 데이터는 데이터 메모리에 존재한다. 둘째 프로그램 수행 중에 CPU의 메모리 접근 충돌은 발생하지 않는다.

그림 3은 본 논문의 시뮬레이터에서 1D FFT[13] 응용프로그램을 실행한 결과와 TI사의 벤치마크 데이터를 비교한 것이다.

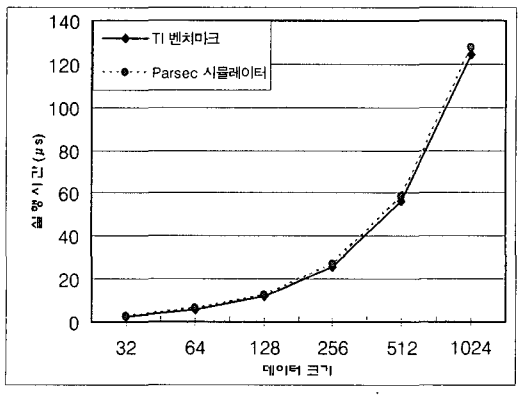

그림 3. 1D FFT 실행시간
Fig. 3. Execution time with 1D FFT

그림 3을 살펴보면 본 논문의 시뮬레이션 결과와 TI사의 벤치마크 결과가 유사함을 알 수 있다. 따라서 본 논문에서 구현한 시뮬레이터로 얻은 결과에 대하여 신뢰할 수 있다.

4.2 2D FFT

본 논문에서는 2D FFT를 사용하여 4가지의 시뮬레이션을 실시한다. 2D FFT는 2차원 데이터에 대해서 FFT를 수행하는 알고리즘으로 1D FFT를 2회 실행한다.

첫 번째 시뮬레이션에서는 프로세서의 수를 2~64, 데이터 크기를 64*64~1024*1024로 증가시키면서 실행시간을 측정한다. 그 결과는 그림 4와 같다.

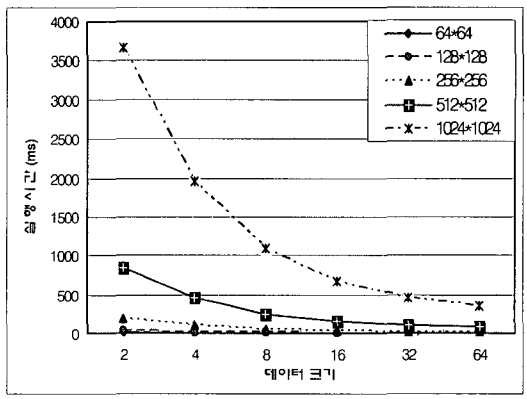

그림 4. 프로세서 수에 따른 실행시간
Fig. 4. Execution time vs. the number of Processor

그림 4에서 프로세서의 수가 4일 경우 데이터 크기 512*512와 1024*1024에서 실행시간이 각각 450ms와 1956ms이다. 즉, 프로세서의 수가 동일할 경우 데이터 크기가 2배 증가하면 실행시간이 4배 이상 증가함을 알 수 있다. 이것은 2D FFT에서 1D FFT을 2회 실행함에 따라 실행시간이 2배 증가하고, 데이터 전송에 따른 통신시간이 증가하기 때문이다.

두 번째 시뮬레이션에서는 전역메모리가 없는 시스템의 성능분석도 가능함을 보인다. 전역메모리의 유무에 따른 시뮬레이션 결과는 그림 5와 같다.

그림 5를 살펴보면 전역메모리를 사용하는 경우 실행시간이 빠르다는 것을 알 수 있다. 이는 각각의 프로세서가 첫 번째 1D FFT 실행 결과를 전역메모리로 전송하는 경우와 지역메모리에 전송하는 경우에 나타나는 통신 오버헤드의 차이 때문이다.

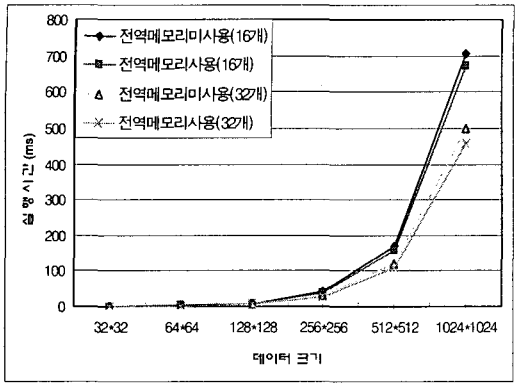

그림 5. 전역메모리 사용에 따른 실행시간
Fig. 5. Execution time using global memory

세 번째 시뮬레이션에서는 다중처리시스템의 하드웨어 구성요소 변화에 따른 시스템의 성능을 분석할 수 있음을 보인다. 동작 특성이 다른 SDRAM 과 SBSRAM을 지역메모리로 사용할 경우 시스템의 성능에 미치는 영향을 알아본다. 시뮬레이션 결과는 그림 6과 같다.

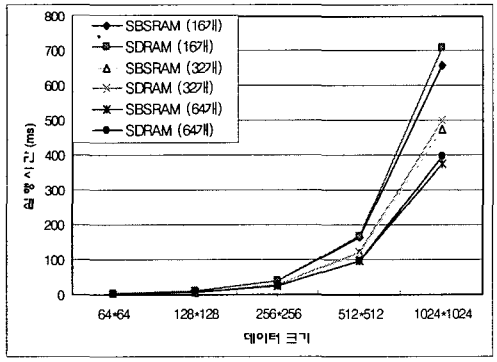

그림 6. 지역메모리 소자에 따른 실행시간
Fig. 6. Execution time with local memory element

그림 6을 살펴보면 SBSRAM을 사용하면 SDRAM에 비해 성능이 향상됨을 알 수 있다. 이것은 SBSRAM이 SDRAM보다 빠른 메모리 접근시간을 제공하기 때문이다. 하지만 DMA 전송시 SBSRAM은 다른 구성요소들과의 동기화를 위해 별도의 오버헤드 시간이 요구되기 때문에 전체적인 실행시간에서는 큰 차이가 없다.

네 번째 시뮬레이션에서는 DMA 전송시 프레임 당 데이터양의 변화에 따른 시스템의 성능 변화를 알아본다. 시뮬레이션 결과는 그림 7과 같다.

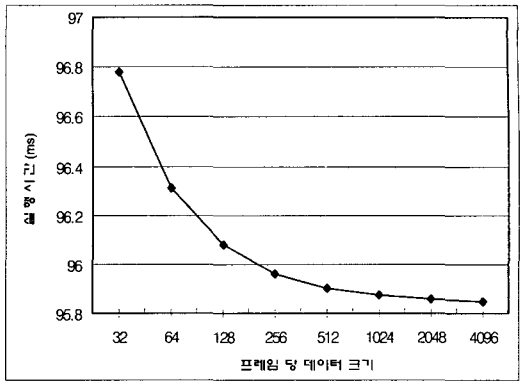

그림 7. 프레임 당 데이터양에 따른 실행시간
Fig. 7. Execution time vs. data size per frame

그림 7을 살펴보면 프레임 당 데이터 크기가 512*512 이상에서는 전체 실행시간에 크게 영향을 미치지 않는다는 것을 알 수 있다.

4.3 행렬 곱셈

행렬곱셈은 그래프이론 및 수치 알고리즘, 신호처리 등에 많이 사용되는 행렬연산이다. 본 시뮬레이터에 적용한 행렬곱셈 알고리즘은 TI사에서 제공한 알고리즘으로 곱셈에 사용될 두 개의 행렬은 정방행렬로 제한다.

시뮬레이션에서는 프로세서의 수를 2~64, 데이터 크기를 64*64~1024*1024로 증가시키면서 실행시간을 측정하였으며 그 결과는 그림 8과 같다.

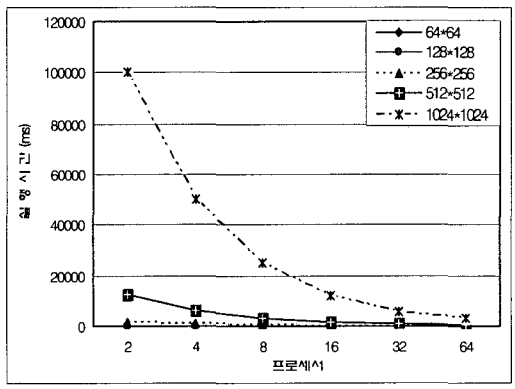

그림 8. 행렬곱셈에서 프로세서 수에 따른 실행시간
Fig. 8. Execution time vs. the number of Processor with matrix multiplication

그림 8을 살펴보면 그림 4와 유사한 형태를 보인다. 즉, 프로세서의 수가 증가할수록 실행시간의 감소율이 점차 작

아짐을 알 수 있다. 이것은 프로세서 수의 증가에 따른 프로세서간의 통신비용이 증가하지만 다수의 프로세서로 병렬처리를 함으로써 실행시간을 감소시키기 때문이다. 행렬 곱셈의 경우 다른 시뮬레이션 결과들도 2D FFT의 시뮬레이션 결과와 유사하지만 실행시간은 커졌다.

4.4 FIR 필터

FIR 필터는 입력 신호의 상태를 증가시키거나, 신호로부터 정보를 알아낼 때, 합쳐진 신호를 분리할 때 사용하는 응용프로그램이다. 본 시뮬레이터에 적용한 FIR 필터 알고리즘은 TI사에서 제공하는 알고리즘이다.[14] 시뮬레이터 적용시 입력신호는 행렬 형태로 입력되며 이 행렬은 각 프로세서의 지역메모리에 저장된다. 변형을 위한 필터도 행렬 형태로 입력되며 데이터 메모리에 저장하여 프로세서간의 통신비용을 감소시킨다.

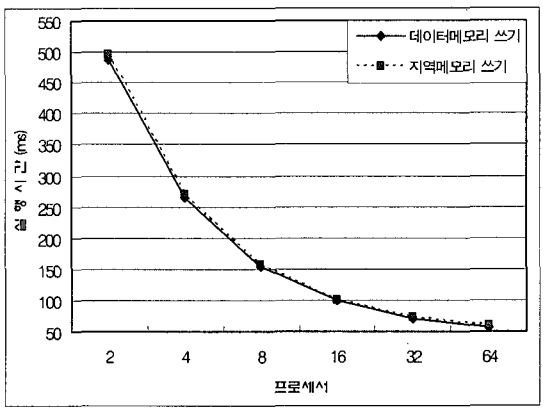

그림 9. 메모리 종류에 따른 실행시간
Fig. 9. Execution time vs. memory type

FIR 필터의 시뮬레이션 결과도 2D FFT의 결과와 유사하기 때문에 본 논문에서 생략한다. 하지만 실행 결과의 저장 위치에 따른 실행시간의 변화를 측정한 결과는 그림 9와 같다. 그림 9를 살펴보면 결과를 데이터 메모리에 저장할 때 실행시간이 빠르다는 것을 알 수 있다. 이것은 행렬의 원소에 대한 계산이 종료될 때마다 결과를 지역메모리에 저장한 후에 행렬에 대한 계산이 종료되면 DMA 전송을 이용해서 지역메모리로 결과를 전송하는 것이 더 효율적이기 때문이다.

V. 결론

본 논문에서는 Parsec을 이용하여 병렬디지털신호처리용 분산공유메모리 다중처리시스템의 성능 분석을 위한 시뮬레이터를 구현하였다. 이 시뮬레이터는 TMS320C6701 DSP 칩의 DMA 기능 및 EMIF에 의한 지역메모리의 빠른 접근과 같은 시스템 하드웨어 요소들의 특징을 잘 나타내도록 모델링한 시스템의 시뮬레이션에 적합하다. 또한 시스템 성능 매개변수의 수정과 하드웨어 구성요소들에 대한 재구성이 쉽기 때문에 다양한 실행 환경에서 시스템의 성능을 분석할 수 있다.

본 논문에서는 구현한 시뮬레이터에서 1D FFT 응용프로그램을 실행한 결과와 TI 사에서 제공하는 TMS320C6701의 벤치마크 자료[12]를 비교함으로써 시뮬레이터에 대한 신뢰도를 검증하였다. 그리고 이 시뮬레이터를 사용하여 2D FFT, 행렬곱셈, FIR 필터 응용프로그램으로 다양한 시뮬레이션을 실시하였다.

시뮬레이션 결과 데이터 크기가 동일할 경우 프로세서의 수를 증가시키면 실행시간을 줄일 수 있음을 알 수 있었다. 그리고 프로세서간의 통신비용 증가로 인해 프로세서 수에 비례하여 시스템의 성능 향상이 이루어지지 않는다는 것을 알 수 있었다. 메모리 모듈의 경우 SBSRAM을 지역메모리로 사용하고, 전역 메모리를 사용하면 실행시간이 빨라짐을 알 수 있었다. 또한 2D FFT에서 DMA 전송시 프레임 당 데이터 크기를 256으로 지정하면 가장 우수한 성능을 얻을 수 있음을 알 수 있었다.

참고문헌

[1] 박승권, 유승화, "병렬처리 컴퓨터의 추세," 정보과학회지, 제13권, 제7호, pp. 5-15, Jul. 1995

[2] P.Keltcher, R.M. Owens and M.J. Irwin, "A Simulation Methodology for Evaluating

Parallel Computers, "Proceedings of The 17th IEEE Symposium on Parllel and Distributed Processing, 1995

[3] C.C. Chou, P.Z. Chen and S.N. Chen, "A Program-Driven Parallel Machine Simulation Environment," Proceeding of International Conference on Parallel and Distributed Systems, IEEE CS Press, pp.25-32, 1998

[4] P.K. McKinley and C. Trefftz, "MultiSim: A Simulation Tool for the Study of Large-Scale Multiprocessors," Proceeding of The 1993 International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Networks, pp.57-62, Jan. 1993

[5] 'The SimOS Simulation Environment', Stanford University Computing Systems Lab., Oct. 1997.

[6] http://www.ece.nwu.edu/~sgoh/project.html

[7] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H.Y. Song, "Parsec: A Parallel Simulation Environment for Complex Systems," IEEE Computer, pp. 77-85, Oct. 1998.

[8] PARSEC User Manual, UCLA Parallel Computing Lab., Jan. 1999

[9] 문병표, 박준석, 전창호, 박성주, 이동호, 한기택, "TMS320C67x 기반 병렬신호처리시스템의 설계와 성능분석," 한국정보처리학회논문지, 제 7권 1호, pp. 65-73, Jan. 2000

[10] TMS320C6701 Floating Point Digital Signal Processor, Texas Instruments, May. 2000

[11] TMS320C6000 CPU and Instructions Set Reference Guide, Texas Instruments, Jan. 2000

[12] http://www.ti.com/sc/docs/products/dsp/c6000/benchmarks/67x.htm

[13] TMS320C62x DSP Library Programmer's Reference, Texas Instruments, Mar. 2000

[14] 이원주, 김효남 "병렬신호처리시스템을 위한 성능 모니터의 구현 및 검증," 한국컴퓨터정보학회 논문지, 제10권, 제5호, pp. 313-321, Nov. 2000

저 자 소 개

이 원 주

1989: 한양대학교 전자계산학과 공학사
1991: 한양대학교 컴퓨터공학과 공학석사
2004: 한양대학교 컴퓨터공학과 공학박사
현재: 두원공과대학 인터넷프로그래밍과 부교수
관심분야: 그리드 컴퓨팅, 성능분석, 웹 및 모바일 컴퓨팅

김 선 욱

1996: 충북대학교 전자계산학과 이학사
2001: 한양대학교 컴퓨터공학과 공학석사
현재: 한국전자통신연구원 서버플랫폼연구팀 연구원
관심분야: 성능분석, 클러스터컴퓨팅, 시스템 연결망, 네트워크가속기술

김 형 래

1987: 한양대학교 전자공학과 공학사
1989: 한양대학교 전자공학과 공학석사
1996: 한양대학교 전자공학과 공학박사
현재: 두원공과대학 인터넷프로그래밍과 부교수
관심분야: 웹 및 모바일 컴퓨팅