

## 병렬신호처리시스템을 위한 성능 모니터의 구현 및 검증

이 원 주\*, 김 효 남\*\*

### An Implementation and Verification of Performance Monitor for Parallel Signal Processing System

Won-Joo Lee \*, Hyo-Nam Kim\*\*

#### 요 약

본 논문에서는 TMS302C6711을 기본 프로세서로 사용하는 DSP Starter Kit(DSK)를 이용하여 병렬신호처리시스템의 성능을 측정하는 성능 모니터를 구현하고 검증한다. 이 성능 모니터의 특징은 DSP/BIOS의 기능 및 실시간 데이터 전송을 위한 RTDX(Real Time Data Exchange)를 사용하여 DSP 작업부하, 메모리 이용률, 그리고 브릿지 트래픽 등과 같은 병렬신호처리시스템의 성능 평가 척도를 측정할 수 있다는 것이다. 시뮬레이션에서는 DSP 알고리즘에서 널리 사용하는 FFT, 2D FFT, Matrix Multiplication, Fir Filter를 사용한다. 하나의 웨이브 파일에서 각각 다른 주기와 데이터 크기, 버퍼 크기에 따른 결과를 성능 모니터와 TI(Texas Instrument)사의 코드 컴포저 스튜디오로 측정한다. 그리고 그 결과를 비교함으로써 본 논문에서 구현한 성능 모니터의 정확성을 검증한다.

#### Abstract

In this paper, we implement and verify performance monitor for parallel signal processing system, using DSP Starter Kit(DSK) of which the basic processor is TMS302C6711 chip. The key ideas of this performance monitor is, using Real Time Data Exchange(RTDX) for the purpose of real-time data transfer and function of DSP/BIOS, the ability to measure the performance measure like DSP workload, memory usage, and bridge traffic. In the simulation, FFT, 2D FFT, Matrix Multiplication, and Fir Filter, which are widely used DSP algorithms, have been employed. Using performance monitor and Code Composer Studio from Texas Instrument(TI), the result has been recorded according to different frequencies, data sizes, and buffer sizes for a single wave file. The accuracy of our performance monitor has been verified by comparing those recorded results.

▶ Keyword : DSP(Digital Signal Processing), DSK(DSP Starter Kit), TMS302C6711, RTDX(Real Time Data Exchange), 코드 컴포저 스튜디오(Code Composer Studio)

• 제1저자 : 이원주

• 접수일 : 2005.08.03, 심사완료일 : 2005.09.05

\* 두원공과대학 인터넷프로그래밍과 부교수, \*\* 청강문화산업대학 컴퓨터소프트웨어과 부교수

## I. 서론

디지털 신호처리 분야에서는 다양한 데이터를 실시간으로 처리해야 하기 때문에 상용 마이크로프로세서를 대신한 DSP(Digital Signal Processing) 칩을 이용한 고성능 컴퓨터 시스템을 개발하여 사용하고 있다[1].

TI(Texas Instrument)의 TMS320C6711 DSP 칩을 사용하여 신호처리시스템을 개발하면 코드 컴포저 스튜디오라는 시스템 성능 분석 툴을 제공받아 사용할 수 있다. 하지만 이 툴은 신호처리시스템의 작업부하만을 측정할 수 있는 제한적인 기능만을 제공한다[2]. 따라서 다양한 성능 평가를 위해서는 별도의 성능 모니터 개발이 필요하다.

본 논문에서는 TI사의 TMS320C6711을 기본 프로세서로 사용한 신호처리시스템의 성능을 분석하기 위해 새로운 성능 모니터를 구현한다. 이 성능 모니터는 작업부하 뿐만 아니라 메모리 이용률과 브릿지 트래픽 등을 모니터링 할 수 있다. 다양한 DSP 응용 알고리즘을 실행하면서 코드 컴포저 스튜디오와 본 논문에서 구현한 성능 모니터로 측정된 작업부하를 비교 분석함으로써 성능 모니터의 정확성을 검증한다.

본 논문의 2장에서는 성능 모니터링 방법 및 기존의 성능 모니터에 대하여 설명하고, 3장에서는 본 논문에서 개발한 새로운 성능 모니터의 구성 및 구현 방법에 대하여 설명한다. 4장에서는 신호처리시스템의 성능평가를 위해 필요한 성능평가 척도를 정의한다. 그리고 여러 가지 DSP 응용 프로그램 실행 과정의 모니터링 결과를 비교 분석한다. 그리고 5장에서 결론을 맺도록 한다.

## II. 성능 모니터링 방법

### 2.1 성능모니터링 방법

신호처리시스템의 성능 모니터링 방법에는 소프트웨어와 하드웨어를 이용하는 방법과 하이브리드(hybrid) 방법이 있다[3].

- **소프트웨어를 이용한 방법:** 이 방법은 소프트웨어를 이용하여 목적(target) 시스템으로 코드를 삽입하는 방법으로 유연하며 추가적인 하드웨어가 필요 없다는 장점이 있다. 하지만 모니터링 시스템이 목적 시스템의 프로세서나 메모리 공간을 사용하기 때문에 시스템의 성능에 영향을 주는 단점도 있다[3]. 모니터링 과정은 사건 감지와 사건 자료를 수집하는 과정으로 구성된다. 모니터링 과정은 세 가지 방법으로 수행된다. 첫째 목적 프로그램이 사건을 감지하고 사건 자료를 수집한다. 둘째 목적 프로그램이 사건을 감지하고 모니터가 사건 자료를 수집한다. 셋째 커널(kernel)이 사건을 감지하고 모니터가 사건 자료를 수집한다. 세 가지 방법은 동일한 시스템에서 조합하여 사용할 수 있다.

- **하드웨어를 이용한 방법:** 이 방법은 하드웨어 또는 소프트웨어 모듈의 실행시간 속성을 모니터링 하는데 사용한다[4]. 모니터링 시스템은 모니터링 하드웨어와 제어 모듈로 구성된다. 모니터링 하드웨어는 목적 시스템의 버스와 연결되어 있으며, 사건을 감지하고 사건 자료를 수집한다. 제어 모듈은 목적 시스템으로부터 분리되어 있으며 모니터링 하드웨어를 제어한다. 하드웨어 모니터를 구현하는 과정은 첫째, 하드웨어 모니터링 장치를 목적 시스템에 연결한다. 둘째, 모니터링 조절 모듈과 목적 프로그램을 수행한다. 셋째, 하드웨어 모니터링과 관련된 사건을 감지하고 처리한다. 이 방법의 장점은 모니터링 시스템과 목적 시스템은 자원을 서로 공유하지 않으므로 모니터링 시스템이 목적 시스템에 거의 영향을 주지 않는다는 것이다. 하지만 모니터링을 위한 비용이 많이 들고, 특정 프로세서나 시스템에 종속적이라는 단점이 있다.

- **하이브리드(hybrid) 방법 :** 이 방법은 소프트웨어와 하드웨어 방법을 혼합한 방법이다[4]. 모니터링 소프트웨어인 목적 프로그램은 수행되면서 사건을 발생시키고 하드웨어 모니터는 사건을 감지하여 데이터를 수집한다. 하드웨어 모니터는 설계 단계에서 시스템의 한 부분으로서 설계된다. 하지만, 테스트 또는 디버깅 단계에서는 개별적인 장치 또는 목적 시스템으로 통합되는 보조 프로세서로 설계된다. 또한 트리거링(triggering)으로 하드웨어에서 수행되는 것을 기록하기 위해 목적 시스템 내부에 명령어를 삽입한다.

## 2.2 기존의 성능 모니터

기존의 성능 모니터에는 HARTS[3], SHRIMP[4], MSPARC[5], 등이 있다.

- HARTS 성능 모니터: HARTS는 미시간 대학에서 개발된 실험적 분산 시스템으로 6개의 노드로 구성된 메쉬(mesh) 구조 강결합(tightly coupled) 마이크로프로세서 시스템이다[3]. 이 시스템의 성능을 모니터링하기 위한 소프트웨어는 HMON이다. HMON에서는 목적 프로그램과 커널이 이벤트를 감지하고 모니터가 이벤트 데이터를 수집한다. 이 성능 모니터의 장점은 특별한 하드웨어 추가 없이 일관된 모니터링을 제공하며 모니터링에 의해 생기는 오버헤드에 대한 반응을 예상할 수가 있다는 것이다.

- SHRIMP 성능 모니터: SHRIMP는 PC나 워크스테이션을 저비용, 고성능의 다중컴퓨터로 통합하기 위한 시스템이다. SHRIMP에서는 네트워크 인터페이스와 EISA 버스를 통한 이벤트를 관찰 할 수 있는 성능 모니터링 보드를 이용한 하드웨어적 성능 모니터링 방법을 사용한다. 이 성능 모니터의 장점은 다차원 패킷 기반의 히스토그램(histogram), 페이지 태그(page tags), 히스토그램 분류(histogram categories) 와 threshold-driven 인터럽트 등의 측정 메커니즘을 제공하고, 단순하고 유연하다는 것이다.

- MSPARC 성능 모니터: MSPARC은 미시시피 주립 대학의 NSF 공학센터에서 개발한 메쉬 구조의 메시저페이스 다중컴퓨터로 하이브리드 성능 모니터 방법을 사용한다. 하드웨어 구성요소는 하나의 backplane에 8개 노드를 연결한다. 각 노드 MC68302 프로세서와 2개의 보드, 커넥터로 구성된 이루어져 있으며 노드의 상태를 감지한다. 소프트웨어로는 DECIPHER를 사용한다. 이 소프트웨어는 다중프로세서의 성능 향상도, 프로세서 부하, 통신 경로 히스토리 등을 측정할 수 있다.

## III. 성능 모니터 구현

본 논문에서 제시하는 성능 모니터 구성과 구현 방법에 대하여 설명한다.

## 3.1 성능 모니터 구성

TMS320C6711 DSP Starter Kit(DSK)으로 구현한 성능 모니터의 구조는 (그림 1)과 같다.

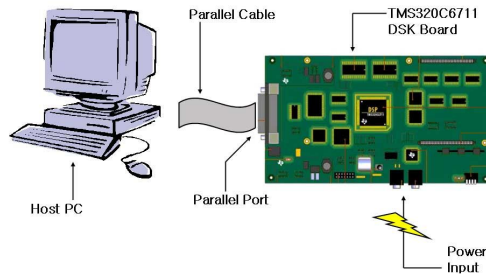


그림 1. 성능 모니터 구조  
Fig.1 Performance Monitor Architecture

(그림 1)에서 Host PC는 모니터링을 위한 코드를 DSK 보드에 다운로드하고 측정된 모니터링 데이터를 전송 받는다. Host PC와 DSK 보드 사이의 통신은 Parallel Cable을 사용한다. 성능 모니터는 DSK 보드의 데이터를 전송받아 분석하고 그 결과를 화면에 출력한다. (그림 1)에서 TMS320C6711 DSK 보드의 핵심 구성요소는 TMS320C6711 마이크로프로세서이다. 이 마이크로프로세서는 TI사의 floating-point DSP 칩으로 최고 150 MHz의 클럭 속도를 가진 고속의 신호 처리용 마이크로프로세서이다. 또한, 10ns의 명령어 사이클을 가지며, 8개의 명령어를 동시에 실행할 수 있는 TMS320C6711 마이크로프로세서의 특징은 <표 1>과 같다.

표 1. TMS320C6711 마이크로프로세서의 특징  
Table 1. Characteristic of TMS320C6711 microprocessor

항목	특징
메모리	72KB On-Chip -4KB 프로그램 cache -4KB 데이터 cache
메모리 인터페이스	32비트 외부 메모리 인터페이스로 SDRAM, SBSRAM, SRAM 지원
통신 포트	2개의 다중 채널 직렬 포트로 고속 및 광대역의 프로세서간 통신 지원
버스	16bit Host 버스로 프로세서 내부 메모리 접근
채널	CPU의 인터럽트를 최소화하면서 지역 메모리와 통신하기 위해 16개의 DMA 채널 사용
타이머	주변기와 동기화를 위해 2개의 32 bit 타이머 사용

### 3.2 성능 모니터 구현 도구

본 논문에서 구현한 성능 모니터는 DSP 코드 컴포저 스튜디오의 API와 MS Visual Basic 6.0을 사용한다. 그리고 운영체제 기능을 하는 DSP/BIOS와 실시간으로 DSP 보드와 Host PC간의 데이터를 전송을 위해 RTDX(Real Time Data Exchange) 기술을 사용한다[9].

- DSP 코드 컴포저 스튜디오: TI사에서 제작된 통합 환경의 DSP 소프트웨어 개발 툴로서 사용하는 프로세서의 종류에 따라 다른 플랫폼을 제공하고 있다. 본 논문에서는 TMS320C6711 DSP를 사용하기 때문에 C6000 플랫폼을 이용한다[7]. 또 용도에 따라 DSP 보드의 사용 없이 DSP 프로그램을 수행할 수 있는 시뮬레이터와 실제 DSP 보드를 장착하여 보드 상으로 코드를 다운로드 한 후 그 결과를 확인하는 에뮬레이터로 사용한다[8].
- DSP/BIOS: 시스템의 자원을 모니터링 하기 위해 임베디드 실시간 소프트웨어를 개발하고 분석하는데 사용한다. DSP/BIOS는 22개의 모듈로 구성되어 있으며 크게 4 그룹으로 분류할 수 있다[9][10].
  - ① 시스템 환경 설정 : 하드웨어 및 시스템의 환경을 설정한다.
  - ② 실시간 모니터링 및 제어 : target 시스템에서 프로그램을 실행하는 동안 호스트에 정보를 전송하는 기능을 제공한다.
  - ③ 실시간 스케줄링 : 실시간에 특정 태스크를 스케줄링한다.
  - ④ 실시간 통신 : 쓰레드(thread)간 또는 target 시스템-호스트간의 통신 채널을 제공한다.

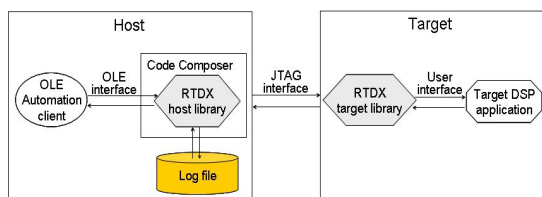


그림 2. RTDX 블록 다이어그램  
Fig. 2. RTDX Block Diagram

- RTDX: (그림 2)와 같이 Host PC와 target 시스템의 DSP 보드 사이에서 실시간으로 데이터를 전송하는 기능을 제공한다. Host PC로 전송된 데이터는 호스트

응용프로그램인 VB, VC++, Microsoft Excel 등으로 분석하고 시각화한다[11].

### 3.3 성능 모니터 구현

본 논문에서 구현한 성능 모니터는 작업부하, 메모리 이용률, 브릿지 트래픽 등을 측정할 수 있다.

- 작업부하(Workload): 성능 모니터에서 측정된 작업부하는 (그림 3)과 같다.

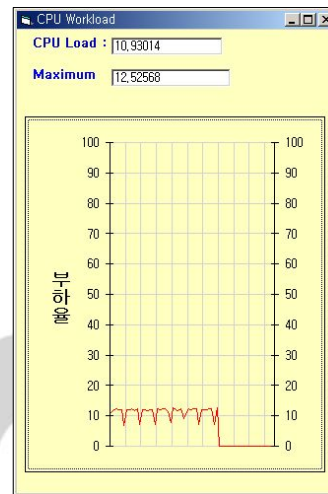


그림 3. 작업부하 측정 결과  
Fig. 3. Measurement result of Workload

작업부하는 DSP 칩의 부하율을 의미하며 아래의 식(1)로 구한다.

$$\begin{aligned} \text{작업부하(\%)} &= Tw / (Tw + Ti) * 100 \\ &= Cw / (Cw + Ci) * 100 \dots\dots\dots (1) \end{aligned}$$

식(1)에서 Tw와 Ti는 각각 작업시간과 유휴 시간을 의미한다. 그리고 Cw와 Ci는 작업 주기와 유휴 주기를 나타낸다. 작업시간과 유휴 시간을 측정하는 방법은 시간 및 명령어 사이클을 이용하는 방법이 있다. 본 논문에서는 시간을 이용하는 방법을 사용하였으며 DSP의 유휴 루프(idle loop)를 이용하여 구한다. 구현 방법으로는 태스크 실행 상태와 스케줄링을 이용한다. 태스크 실행 상태는 (그림 4)의 태스크 스위칭 다이어그램에서 알 수 있다. Tw는 TSK\_RUNNIG 상태에서

소요되는 시간으로 TSK\_BLOCKED 상태와 TSK\_READY 상태의 시간차로 구한다. TSK\_TERMINATED 상태에서 TSK\_READY 상태로 전환하는데 소요되는 시간차로 Tt를 구한다.

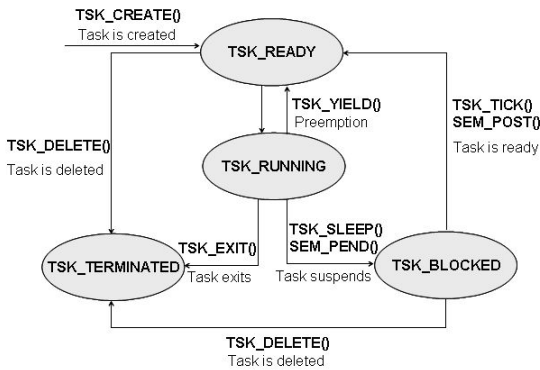


그림 4. 태스크 스위칭 다이어그램  
Fig. 4. Task Switching Diagram

- 메모리 이용률(memory usage rate): 기존의 코드 컴포저 스튜디오에서는 메모리 이용률을 측정할 수 없다는 단점이 있다. 이러한 단점을 해결하기 위해 본 논문에서는 DSP /BIOS의 MEM 모듈을 이용하여 메모리 이용률을 측정할 수 있도록 구현한다.

메모리 이용률은 응용 프로그램을 수행하기 위해 사용된 메모리 양의 비율을 식(2)로 구한다.

$$\text{메모리 이용률} = Ma / Mt * 100 \dots\dots\dots (2)$$

식(2)에서 Ma와 Mt는 각각 할당 메모리 크기와 전체 메모리 크기를 의미한다. Ma와 Mt는 메모리 할당 시 할당된 전체 메모리 크기를 계산하고, 해제 시 전체 할당된 메모리 크기에서 해제된 메모리 크기의 차를 구하면 각 시간 별로 할당된 메모리 크기를 구할 수 있다. 성능 모니터에서 측정된 메모리 이용률은 (그림 5)와 같다.

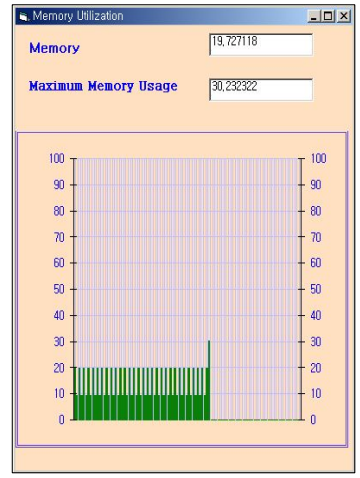


그림 5. 메모리 이용률 측정 결과  
Fig. 5. Measurement result of Memory Usage Rate

- 브릿지 트래픽(Bridge Traffic): 성능 모니터에서 측정된 브릿지 트래픽은 (그림 6)과 같다.

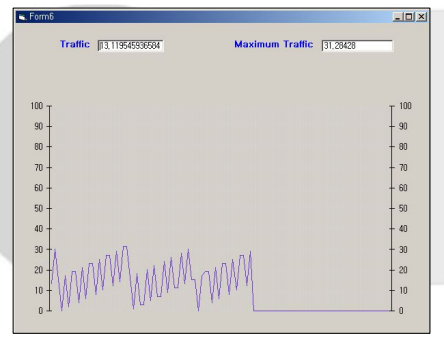


그림 6. 브릿지 트래픽 측정 결과  
Fig. 6. Measurement result of Bridge Traffic

브릿지 트래픽은 브릿지의 데이터 전송량을 의미한다. PCI 버스의 최대 데이터 전송량이 132MB이므로 브릿지 트래픽은 식(3)으로 구한다.

$$\text{브릿지 트래픽(\%)} = BWPCI / 132 * 100 \dots\dots\dots (3)$$

식(3)에서 BWPCI는 초당 PCI 버스의 전송량을 의미한다. 그리고 132는 PCI 버스의 최대 데이터 전송량을 의미하며 33 MHz x 32 bit로 구한다. 브릿지 트래픽을 구하기 위해 응용 프로그램에서는 브릿지를 경유한 시작 시간과

데이터 크기를 측정하는 API를 사용한다. DSP 보드에서 데이터 전송을 요청하면 이 API를 호출하여 브릿지 트래픽을 구할 수 있다.

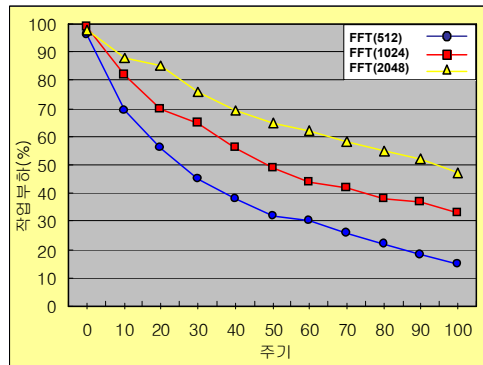
### IV. 실험 및 결과 분석

본 논문에서는 다양한 DSP 응용알고리즘을 실행하면서 TI사의 코드 컴포저 스튜디오와 성능 모니터로 작업부하를 측정한다. 그리고 그 결과가 서로 일치한다는 것을 보임으로써 본 논문에서 구현한 성능 모니터의 정확성을 검증한다. 또한 성능 모니터를 사용하여 메모리 이용률, 브릿지 트래픽을 측정하여 신호처리시스템의 성능을 평가한다.

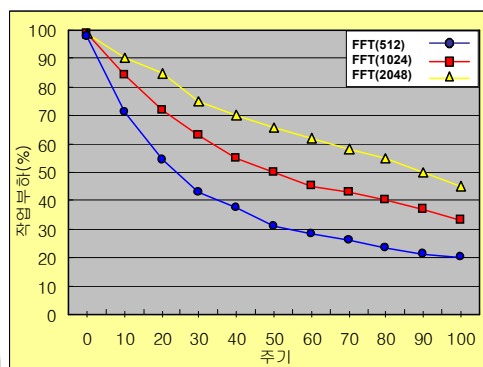
#### 4.1 작업부하 결과분석

첫 번째 실험은 본 논문에서 구현한 병렬신호처리시스템용 성능 모니터의 정확성을 검증하는 것이다. FFT, 2D FFT, Matrix Multiplication, Fir Filter 등의 대표적인 DSP 알고리즘을 수행하면서 성능 모니터와 코드 컴포저 스튜디오로 작업부하를 측정하고 그 결과를 비교 분석한다. DSP 알고리즘을 수행하기 위해 하나의 wave 파일에서 데이터를 주기적으로 입력한다. 하나의 명령어를 수행시키기 위해서는  $1000 * 10^{-6}$ 의 시간이 필요하므로 주기는 1 msec로 한다. 주기당 읽어들이는 데이터 크기는 512, 1024, 2048로 제한한다.

- FFT 와 2D FFT: FFT는 이산 데이터 값들의 푸리에 변환 계산을 위한 알고리즘으로 실세계 신호로부터 주기적으로 얻어지는 데이터들을 그 요소 주파수들의 형태로 변환하는 알고리즘이다[12]. 2D FFT는 2차원 데이터에 대해서 FFT를 수행하는 연산으로 유체역학에서 속력벡터, 기울기(gradient) 계산 그리고 이미지 처리에서 특징 추출 등에 많이 사용된다. 본 논문에서 적용한 FFT 및 2D FFT 알고리즘은 TI사에서 제공하는 1D FFT를 사용하였다. (그림 7)은 FFT와 2D FFT를 수행하면서 주기 및 데이터 크기에 따른 작업부하의 변화를 측정한 결과이다.

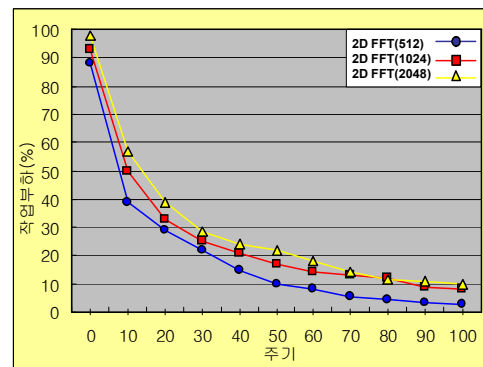


(a)코드 컴포저 스튜디오(code composer studio)

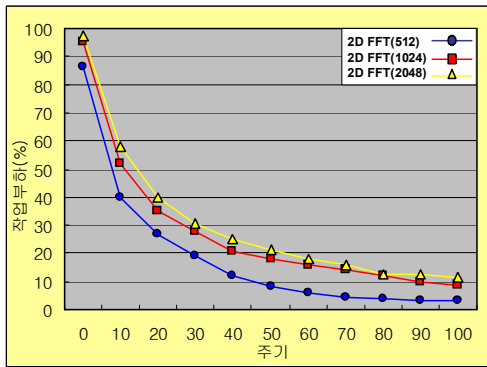


(b) 성능 모니터(performance monitor)

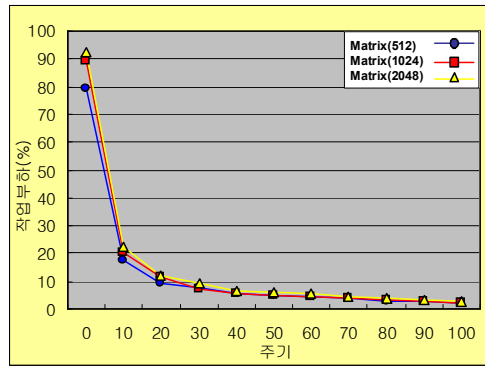
그림 7. 1D FFT 수행 결과  
Fig. 7. Execution result of 1D FFT



(a)코드 컴포저 스튜디오(code composer studio)



(b) 성능 모니터(performance monitor)

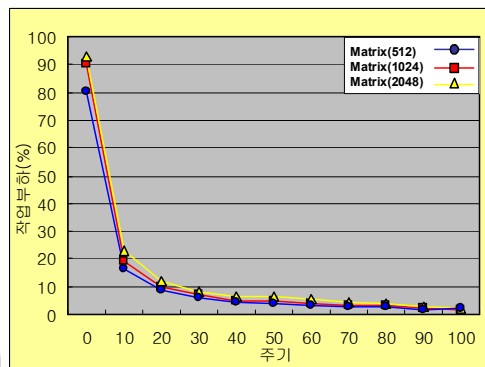


(a)코드 컴포저 스튜디오(code composer studio)

그림 8. 2D FFT 수행 결과  
Fig. 8. Execution result of 2D FFT

(그림 7)과 (그림 8)을 살펴보면 주기가 증가함에 따라 DSP 작업부하율이 감소하는 것을 알 수 있다. 그리고 데이터 크기가 작을수록 DSP의 부하율이 낮아지는 것을 알 수 있다. 이것은 FFT 알고리즘 수행시 한번에 읽어 들이는 데이터의 크기와 처리하는 양이 비례하기 때문에 데이터의 크기가 2048인 경우의 부하율이 가장 높고 512인 경우가 가장 낮다는 것을 알 수 있다. 또한 데이터의 입력 주기가 클수록 받아들이는 데이터양이 많아 DSP 부하율이 증가한다. (그림 7)과 (그림 8)에서는 본 논문에서 구현한 성능 모니터와 코드 컴포저 스튜디오에서 측정된 결과가 유사하다는 것을 확인할 수 있다. 따라서 본 논문에서 구현한 성능 모니터의 측정 결과가 정확함을 알 수 있다.

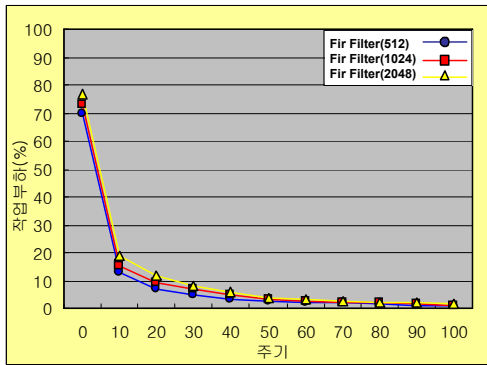
•행렬 곱셈(matrix multiplication): 그래프이론 및 수치 알고리즘, 신호처리 등에 많이 사용되는 연산이다. 본 논문에서 적용한 행렬곱셈은 TI사에서 제공하는 알고리즘을 사용하였으며 본 논문에서는 곱셈에 사용할 두 개의 행렬은 정방행렬로 가정한다[13][14]. (그림 9)를 살펴보면 주기가 증가하면서 작업부하는 감소하는 것을 확인할 수 있다.



(b) 성능 모니터(performance monitor)

그림 9. 행렬곱셈 수행 결과  
Fig. 9. Execution result of matrix multiplication

•Fir Filter: 입력 신호의 상태를 향상시키거나, 신호로부터 정보를 알아내거나, 합쳐진 신호를 다시 각각의 신호로 분리할 때 사용되는 알고리즘이다. 저주파 대역에서 그 특성이 매우 양호하여 음성신호 처리 분야에서 활용도가 높다. 본 논문에 적용한 알고리즘은 TI사에서 제공하는 일반적인 목적의 FIR Filter 알고리즘으로 입력신호는 행렬 형태로 입력한다. (그림 10)을 살펴보면 주기가 증가하면서 작업부하는 감소하는 것을 확인할 수 있다.



(a) 코드 컴포저 스튜디오(code composer studio)

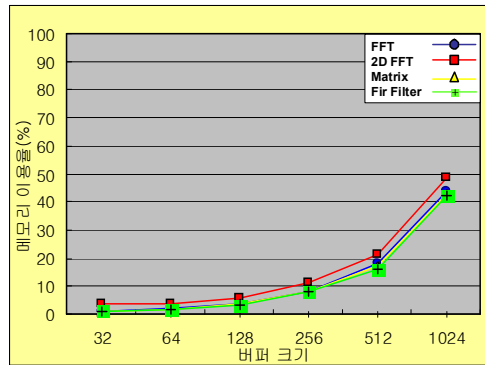
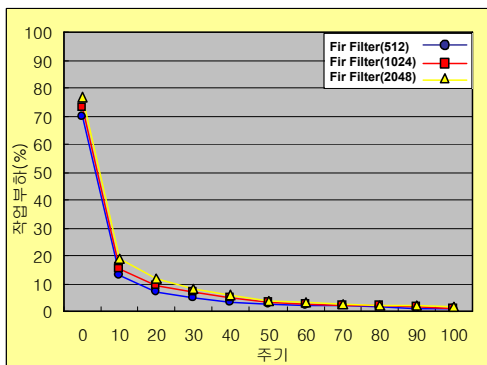


그림 11. 버퍼 크기에 따른 메모리 이용률  
Fig. 11. Memory Usage Rate vs. Buffer Size



(b) 성능 모니터(performance monitor)

그림 10. Fir Filter 수행 결과  
Fig. 10. Execution result of Fir Filter

#### 4.2 메모리 이용률 결과분석

메모리 이용률은 본 논문에서 구현한 성능 모니터를 이용하여 측정한다. (그림 11)은 FFT, 2D FFT, 행렬 곱셈과 Fir Filter 수행시 버퍼 크기에 따른 메모리 이용률을 측정한 결과이다. (그림 11)에서 각 알고리즘 별로 차이는 있지만 버퍼 크기가 증가하면서 메모리 이용률이 증가하는 것을 볼 수 있다. 메모리 이용률은 DSP 작업부하와는 달리 주기 및 데이터 크기에 대한 영향을 받지 않았다. 반면에 응용 프로그램 수행시 메모리 할당 및 해제 단위가 되는 버퍼의 크기에 따라 달라진다. 메모리의 경우 한 번 할당된 후에는 DSP 알고리즘의 규칙적인 수행이 메모리 이용률에 거의 영향을 미치지 않음을 알 수 있다. 응용 프로그램을 수행할 때, 한번에 할당되는 메모리 크기가 클수록 메모리 이용률이 증가하는 것을 알 수 있다.

#### 4.3 브릿지 트래픽 결과분석

본 논문에서 측정한 결과에서는 각 알고리즘을 적용했을 때 차이가 거의 없음을 알 수 있다. 즉, 다양한 데이터 크기와 주기에서 알고리즘을 수행하더라도 매 시간별 브릿지를 통과하는 데이터의 양이 같다는 것을 알 수 있다.

### V. 결론

기존의 TI사의 코드 컴포저 스튜디오에서는 작업부하만을 측정할 수 있기 때문에 병렬신호처리시스템의 성능을 분석하는데 한계가 있었다. 이러한 단점을 해결하기 위해 본 논문에서는 병렬신호처리시스템의 성능 분석을 위한 새로운 성능 모니터를 구현하고 검증한다.

본 논문에서 구현한 성능 모니터는 작업부하 뿐만 아니라 코드 컴포저 스튜디오로 측정 할 수 없는 메모리 이용률과 브릿지 트래픽 등을 측정할 수 있다. 이 성능 모니터를 사용하여 DSP의 다양한 응용 프로그램을 수행하면서 작업부하를 측정하였다. 그리고 TI사의 코드 컴포저 스튜디오에서 측정한 작업부하와 비교하여 본 논문에서 구현한 성능 모니터의 정확성을 검증하였다.



## 참고문헌

- [1] 金在錫, “고속 DSP 기술 동향,” 전자공학회지 제22권 제2호, pp 78-88, 1995
- [2] <http://focus.ti.com/docs/toolfolder.html?PartNumber=TMDS32006711>.
- [3] Tsai, Jeffery. J. P, Yaodong Bi, and, Steve J. H. Yang, Ross A. W. Smith, “Distributed Real-Time System: Monitoring, Visualization, Debugging and Analysis”, John Wiley & Sons, Apr. 1996.
- [4] Jeffrey J. P. Tsai and Steve J. H. Yang, “Monitoring and debugging of distributed real-time systems”, IEEE Computer Society Press, 1995.
- [5] Margaret Martonosi, Douglas W. Clark, Malena Mesarina, “The SHRIMP Performance Monitor: Design and Applications” ,ACM SIGMETRICS Symposium on Parallel and Distributed Tools, May 1996.
- [6] J. Harden, D. Reese, F. To, D. Linder, C. Borchert, G. Jones, “A performance monitor for the MSPARC multicomputer”, inProc .IEEESoutheastcon’92, Apr.12--15, 1992, pp. 724--729.
- [7] <http://focus.ti.com/docs/toolfolder.html?PartNumber=TMDS324685C-07>.
- [8] Code Composer Studio User’s Guide, Texas Instruments, February. 2000.
- [9] Understanding Basic DSP/BIOS Features, Texas Instrument, Application Report, April. 2000.
- [10] TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide, Texas Instruments, May. 2000.
- [11] TMS320C6000 DSP/BIOS User’s Guide, Texas Instrument, May. 2000.
- [12] <http://www.terms.co.kr/FFT.htm>
- [13] TMS320C62x DSP Library Programmer ‘s Reference, Texas Instruments, Mar. 2000.
- [14] A Parallel Approach for Matrix Multi plication on the TMS320C4x DSP, Texas Instruments, Feb.1994.

## 저자 소개



### 이 원 주

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



### 김 호 남

1988년 홍익대학교 전자계산학과  
(이공학사)  
1990년 홍익대학교 전자계산학과  
(이공학석사)  
2002년 홍익대학교 전자계산학과  
박사수료  
현재 청강문화산업대학  
컴퓨터소프트웨어과 부교수  
<관심분야>Programming  
Language, Object  
Oriented Programming,  
Mobile Programming,  
컴퓨터 보안