Journal of The Korea Society of Computer and Information
Vol. 18, No. 8, August 2013
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2013.18.8.001

GPU를 이용한 기타의 음 합성을 위한 효과적인 병렬 구현

강성모*, 김종면*

An Effective Parallel Implementation of Sound Synthesis of Guitar using GPU

Sung-Mo Kang, Jong-Myon Kim

요 약

본 논문에서는 GPU 환경에서 기타의 음 합성을 위한 물리적 모델링의 효율적인 병렬 구현 방법을 제안한다. 물리적 모델링을 이용하여 기타의 개방현(E2, A2, D3, G4, B3, E4)들의 기본음을 합성하기 위해 각 개방현 음 합성을 위한 적절한 필터 계수를 사용하였고, 지연 라인의 길이를 조절하였다. 또한 물리적 모델링 알고리즘을 분석한 결과 지연 라인의 길이만큼 병렬성을 갖는 것을 확인하였다. 따라서 각 개방현의 기타 음을 합성하기 위해 지연 라인의 길이만큼 CUDA 코어를 할당한 후 최적의 성능을 보이도록 알고리즘을 병렬 구현하였다. 모의실험결과, GPU를 이용하여 합성한 기타 음과 원음과의 스펙트럼이 매우 유사하였고, GPU는 기존 고성능 TI DSP보다 68배, CPU보다 3배의 성능 향상을 보였다. 또한, 본 논문에서는 물리적 모델링 알고리즘을 멀티 GPU시스템에서도 구현하고 성능을 분석하였다.

▸ Keywords : 물리적 모델링 합성, 기타 음, 병렬 구현, 그래픽 처리 유닛, OpenCL

Abstract

This paper proposes an effective parallel implementation of a physical modeling synthesis of guitar on the GPU environment. We used appropriate filter coefficients and adjusted the length of delay line for each open string to generate 44,100 six-polyphonic guitar sounds (E2, A2, D3, G4, B3, E4) by using physical modeling synthesis. In addition, we analyzed the physical modeling synthesis algorithm and observed that we can exploit parallelism inherent in the length of delay

•제1저자 : 강성모 •교신저자 : 김종면
•투고일 : 2013. 2. 27, 심사일 : 2012. 3. 23, 게재확정일 : 2013. 3. 29.
* 울산대학교 전기공학부(School of Electrical Engineering, University of Ulsan)
※ 이 논문은 2012년 제46차 한국컴퓨터정보학회 하계학술대회에서 우수논문상을 수상한 논문("물리적 모델링을 통한 실시간 기타 음 합성의 GPU 구현")을 확장한 것임
※ 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2013R1A2A2A05004566).

line. Thus, we assigned CUDA cores as many as the length of delay line and effectively implemented the physical modeling synthesis using GPU to achieve the highest performance. Experimental results indicated that synthetic guitar sounds using GPU were very similar to the original sounds when we compared their spectra. In addition, GPU achieved 68x and 3x better performance than high-performance TI DSP and CPU, respectively. Furthermore, this paper implemented and evaluated the performance of multi-GPU systems for the physical modeling algorithm.

▸ Keywords : Physical modeling synthesis, guitar sound, parallel implementation, graphics processing unit, OpenCL

I. 서 론

최근 디지털 신호처리 기술과 음향 기술의 발전으로 직접 악기를 연주하지 않고 디지털 신호를 합성하여 오디오 신호를 만들어 내는 전자악기들이 개발되어오고 있다[1-3]. 초기의 전자악기들은 음 합성 시 녹음 된 악기의 음을 변조하여 새로운 음을 생성하는 방식을 주로 사용하지만 아날로그 악기의 부드러운 음색을 충분히 재현하지 못하는 단점을 가지고 있어 이를 보완하기 위한 다양한 음 합성 기법들이 연구되고 있다. 대표적인 음 합성 방법으로는 샘플링 합성, 가산 합성, 감산 합성, 주파수 변조 합성, 스펙트럼 모델링 합성, 물리적 모델링 합성 기법 등이 있다. 그 중 물리적 모델링 기반의 음 합성 방법은 고음질의 자연스러운 음을 합성할 수 있어 현악기를 위한 음 합성 알고리즘으로 적합하다[4]. 하지만 물리적 모델링 기반의 음 합성은 많은 양의 산술 연산이 요구되므로 실시간 출력을 만족시키는데 제약을 가진다. 이러한 많은 연산량을 요구하는 멀티미디어 애플리케이션을 위해 GPU와 같은 가속기들의 사용이 증가하고 있다[5].

GPU는 2D 및 3D 그래픽 연산을 처리하기 위한 목적으로 개발된 그래픽 연산전용의 프로세서이다. 오늘날의 GPU는 그래픽 전용 연산장치의 목적뿐만 아니라 산술연산에서 발생하는 작업부하를 줄이기 위한 범용 연산장치로도 사용되며 이를 GPGPU(General-Purpose computing on Graphics Processing Units)라고 부른다. GPU는 높은 부동 소수점 연산속도와 메모리 대역폭 덕분에 다양한 과학 및 공학 계산을 가속화 할 수 있다[6, 7]. 이러한 GPU 프로그래밍은 단순한 산술 반복 연산을 프로세서에 적재된 많은 수의 코어를 통해 효과적으로 병렬 처리할 수 있어 고성능 컴퓨팅에 있어서 가장 적합한 솔루션이라고 평가되고 있다. 이러한 GPU를 범용의 목적으로 사용하기위해서는 OpenCL(Open Computing Language)과 같은 대규모 병렬 프로그래밍 프레임워크가 필요하다. OpenCL은 GPU의 제조사나 하드웨어 아키텍처에서 상관없이 GPU 프로그래밍이 가능하여 GPU 프로그래밍의 유연성을 높이고 광범위한 사용을 가능하게 한다[8].

본 논문에서는 물리적 모델링 기반의 기타 음 합성을 GPU상에서 병렬처리 될 수 있도록 OpenCL을 사용하여 알고리즘을 병렬 구현한다. 인텔 Xeon(R) x5690 @ 3.47GHz와 NVIDIA Geforce GTX 580 상에서 샘플링율이 44.1kMz, 16비트의 양자화 데이터 음을 합성하였으며, 합성음은 원음과의 스펙트럼 비교 결과 원음과 매우 유사함을 확인할 수 있었다. 또한 제안한 방법의 성능 평가를 위해 TI DSP와 실행 시간 측면에서 성능을 비교하여 좋은 결과를 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 실험에 사용되는 GPU와 병렬처리 프레임워크인 OpenCL에 대해 설명한다. 3장에서는 물리적 모델링 알고리즘에 대해 설명하고 이에 대한 병렬처리 방법에 대해서 설명한다. 4장에서는 실험환경 및 음 합성 결과를 보이고, 제안한 병렬 구현 방법과 사용 프로세서의 실행 시간 측면에서 성능을 분석한다. 마지막으로 5장에서 본 논문의 결론을 맺는다.

II. 관련 연구

1. GPU

본 논문에서 사용된 GPU 프로세서는 NVIDIA사의 GeForce GTX 580으로, GTX 580 아키텍처는 그림 1과 같

이 16개의 스트리밍 멀티프로세서(streaming multiprocessor, SM)를 가진다. SM은 명령어를 인출하여 전체 스트리밍 프로세서(streaming processor, SP)에 전송하고, 각 SP들은 명령어를 동시에 수행한다. 하나의 SM은 다음과 같은 특징을 갖는다[9].

- FPU(floating point unit)와 ALU(arithmetic logic unit)로 구성된 32개의 CUDA Core 또는 SP
- 메모리로부터 읽기/쓰기를 실행하는 16개의 LD/ST(lode/store unit)
- 초월함수를 처리하는 SFU(special function unit)
- Warp라 불리는 병렬스레드 그룹을 관리하는 Warp scheduler
- 48KB의 공유 메모리와 L1 캐시(cache)

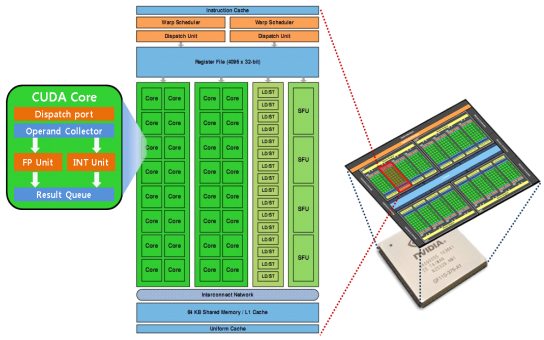

그림 1. GeForce GTX 580 구조
Fig. 1. GeForce GTX 580 Architecture

SM 내의 32개의 SP들은 공유 메모리(shared memory)를 통해 데이터를 공유할 수 있으며, SM들은 전역 메모리(global memory)를 통해 데이터를 공유한다.

2. OpenCL

OpenCL은 애플, AMD, 인텔, NVIDIA, 삼성, NOKIA 등의 프로세서 업체, 시스템 OEM 업체, 응용 프로그램 업체 및 다양한 업계가 참여하여 표준화 작업을 진행한 개방형 범용 병렬 컴퓨팅 프레임워크이다. OpenCL은 CUDA(Compute Unified Device Architecture)와 달리 여러 제조업체의 GPU를 단일 인터페이스를 통해 지원하는 장점이 있다. OpenCL은 여러 개의 스레드(thread)에서 각기 다른 데이터를 사용하여 병렬적으로 실행되는 SIMT(single instruction multiple thread) 방식으로 동작한다.

그림 2는 OpenCL에서의 ND(N차원)-Range의 구성을 나타낸다. 워크아이템(work-item)은 CUDA 스레드와 같은 의미이며 가장 작은 단위의 실행객체이다. 각각의 워크아이템은 동일한 커널 함수를 실행하여 같은 코드로 동작한다. 워크그룹(work-group)은 CUDA thread block과 같은 의미이며, 워크아이템들끼리 동기화가 가능하여 협력 작업을 수행할 수 있다. ND-Range는 워크그룹들로 구성된다.

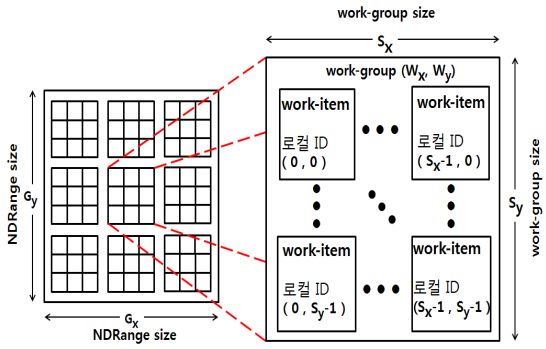

그림 2. ND-Range 구조
Fig. 2. ND-Range architecture

III. 본 론

1. 물리적 모델링 합성

물리적 모델링 합성은 악기가 음을 생성하는 물리적인 현상을 수식적으로 표현한 것이다. 물리적 모델링 합성의 많은 계산량으로 인해 기존에는 음 합성을 위해 해당 음을 녹음하여 재현하는 샘플링 방법이 주로 사용되어왔다. 샘플링 합성의 경우는 전자악기 혹은 스마트 기기와 연동하여 다양한 악기를 연주할 경우 많은 양의 녹음된 음 정보가 요구되어 하드웨어적인 제약을 갖는다. 반면 GPU와 같은 고성능 프로세싱 유닛과 물리적 모델링 합성을 이용할 경우 적은양의 여기 신호와 필터 계수를 이용하여 보다 풍부하고 자연스러운 음 합성이 실시간으로 가능하다. 그림 3은 물리적 모델링을 합성에 기반을 둔 현 모델을 나타낸다.

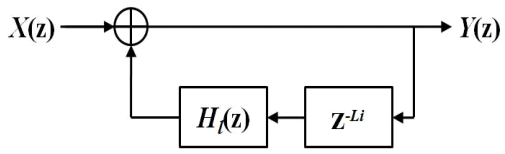

그림 3. 기타의 현 모델 블록 다이어그램
Fig. 3. A block diagram of the string model of Guitar

그림 3에서 X(z)는 현 모델의 입력으로 기타 현의 특성을 제외한 기타 몸체(guitar body)와 브릿지(guitar bridge) 등의 특성을 포함한 여기 신호(excitation signal)이다. Z-Li는 지연 라인(delay line)을 나타내며 지연 라인의 길이 L은 합성음의 샘플링율 f_s과 합성음의 기본 주파수 f_0로 식 (1)과 같이 계산된다.

$$L = f_s / f_0. \tag{1}$$

현 진동의 감쇄현상을 표현하기 위한 루프 필터(loop filter) $H_l(z)$는 저역 통과 필터(low-pass filter)의 특성을 가지며 식 (2)과 같이 표현된다[10].

$$H_l(z) = g\frac{1+a_1}{1+a_1 z^{-1}}, \tag{2}$$

여기서 g는 0Hz에서의 필터 이득, a_1은 차단 주파수를 결정하는 필터 계수이다.

2. 물리적 모델링 합성의 병렬 구현

기타 음 합성을 위한 물리적 모델링 합성은 그림 3에서와 같이 여기 신호 X(z)가 입력으로 들어오면 지연 라인을 거쳐서 루프 필터를 통과하게 되고, 이와 같은 과정이 반복되어 최종 음이 합성된다. 즉 이전의 연산 결과는 지연 라인을 통과 한 후 음 합성을 위한 입력으로 다시 사용되는 데이터 종속성을 갖게 된다. 반면 지연 라인 Z-L에서의 데이터는 데이터 종속성을 갖지 않으므로 지연 라인의 길이만큼 병렬 구현이 가능하다.

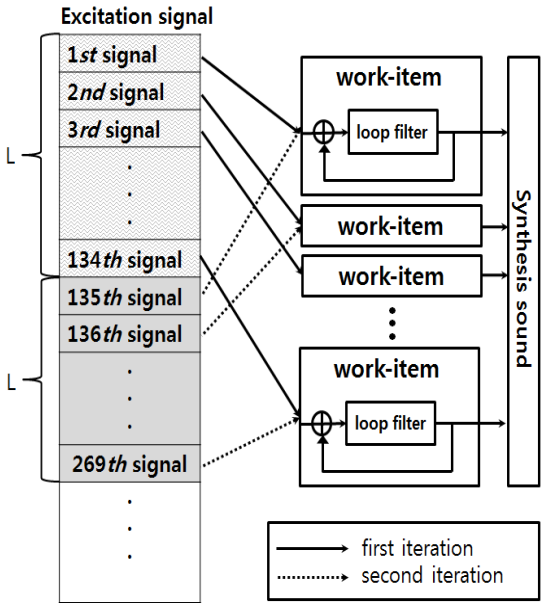

그림 4. 기타의 현 모델 병렬 구현
Fig. 4. A parallel implementation of the string model of Guitar

그림 4는 기타 개방현 1번(E4, 329.63Hz) 음 합성을 위한 병렬 구현 예이다. 먼저 개방현 1번의 병렬 구현을 위해 데이터 독립성을 갖는 지연길이 L(44,100/329.63 ≒ 133)만큼의 워크 아이템을 할당한다. 이때 각 워크 아이템에서 하나의 음 샘플(sample)을 합성하여 한 번에 지연길이 L(133)만큼의 샘플이 동시에 합성된다. 이를 GPU 상에서 맵핑하면 그림 5와 같이 나타낼 수 있다. 하나의 여기 신호 샘플을 하나의 CUDA 코어에 할당 되도록 하여 지연 라인의 길이만큼 CUDA 코어에 맵핑한다. 하나의 SM마다 19개의 CUDA 코어를 사용하여 7개의 SM이 할당되어 한번에 133개의 샘플이 합성된다. 즉 1초 길이의 기타 음 합성을 위해서는 이와 같은 과정을 332번 반복하여 총 44,156개의 샘플을 합성한다.

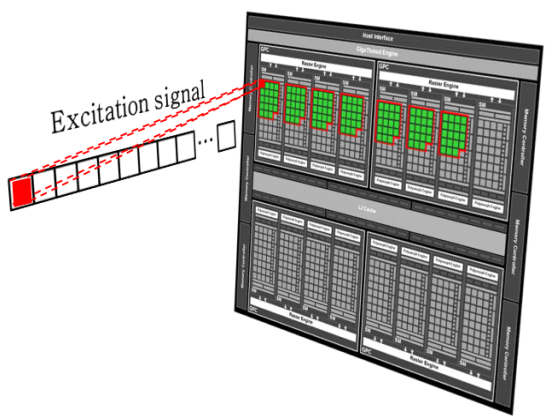

그림 5. 기타의 1번 현에 대한 워크 아이템의 GPU 맵핑
Fig. 5. GPU Mapping of work-item on 1st string of guitar

IV. 실험 결과

1. 실험 환경

본 실험에서 사용된 CPU는 인텔사의 Xeon x5690이며, 연산에 사용된 GPU는 NVIDIA의 GeForce GTX 580이다. 표 1은 물리적 모델링 합성을 이용한 기타 음 합성을 위해 사

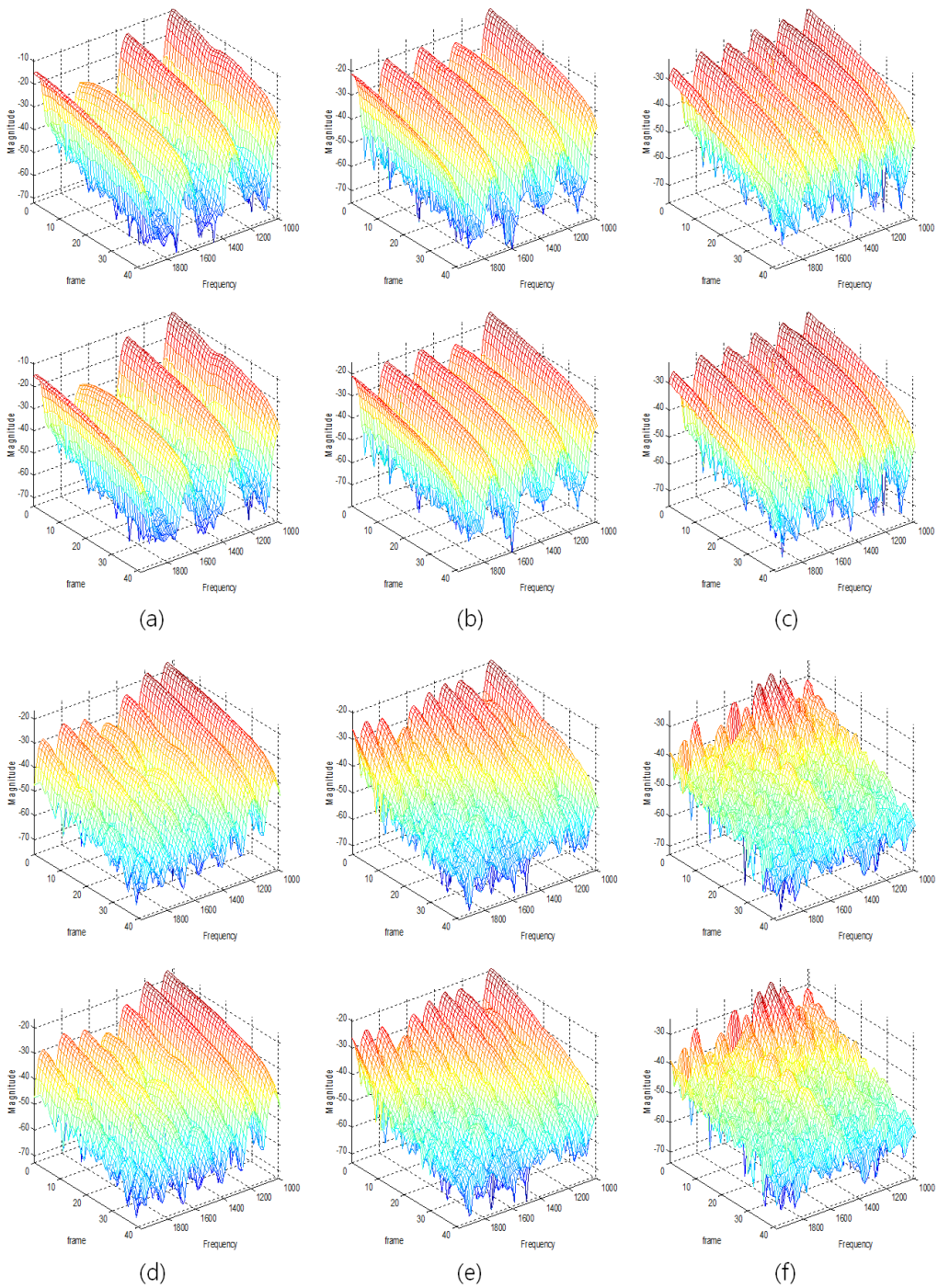

그림 6.악기의 원음(상)과 GPU를 이용한 합성 음(하)에 대한 6가지 현의 스펙트럼 : (a) 1번 현, (b) 2번 현, (c) 3번 현, (d) 4번현, (e) 5번 현, (f) 6번 현

Fig. 6. Spectra of six strings for original sounds(top) and synthesized sounds(bottom) using GPU : (a) 1st string, (b) 2nd string, (c) 3rd string, (d) 4th string , (e) 5th string, (f) 6th string

용된 전체 시스템의 실험 환경을 보여준다. 사용된 GPU는 512개의 CUDA 코어를 가지고 있으며 GPU 장치의 자세한 사양은 표 2와 같다.

표 1. 시스템 환경
Table 1. System Environment

실험 요소	장치 명
CPU (수)	Intel Xeon x5690 3.46GHz (2)
Memory	144 (GB)
OS	CentOS 5.8
GPU	NVIDIA GeForce GTX 580

표 2. NVIDIA GeForce GTX 580 사양
Table 2. Specification of the NVIDIA GeForce GTX 580

항목	값
Global memory	3GB
local memory	48KB
Clock Frequency	1.51GHz
Max workGroup	1024
Max Compute Units	16
CUDA version	4.2.1
OpenCL version	1.1

2. 음 합성 결과

그림 6은 본 논문에서 제안한 물리적 모델링 합성의 병렬 구현 방법을 이용하여 합성한 개방현 음과 원음과의 스펙트럼 비교 결과를 보여준다. 그림 6에서 보는 바와 같이 합성음과 원음의 스펙트럼은 매우 유사한 결과를 보였고, 청취 결과에서도 거의 동일한 결과를 보였다. 합성한 음은 아래 웹사이트를 통해 확인할 수 있다.

http://eucs.ulsan.ac.kr/KSCI/2013/01/GPU-PM.

3. CPU와 GPU간의 성능 비교

GPU를 이용한 기타 음 합성 알고리즘의 성능을 평가하기 위해 GPU와 CPU의 성능을 비교하였다. 그림 7은 CPU와 GPU에서 6개 현의 음을 각각 합성하는데 걸리는 시간을 보여준다. 순차처리를 하는 CPU는 6개의 현에 대해 같은 수준의 시간을 소비한다. 하지만 병렬처리 알고리즘을 GPU에서 수행하게 되면 각 현의 지연길이 E4(133), B3(178), G3(226), D3(300), A2(402), E2(535)에 따라 병렬성이 달라지고, 병렬성이 높은 현의 실행시간은 낮은 현에 비해 실행시간이 더 빠른 것을 알 수 있다.

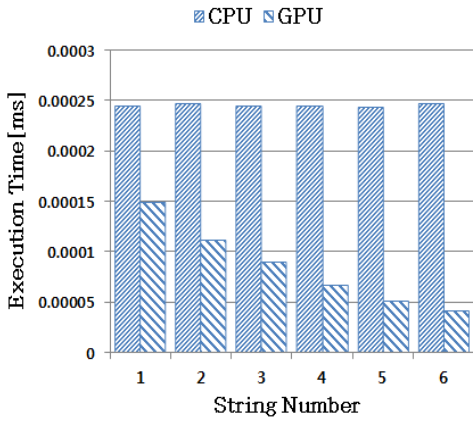

그림 7. 기타의 각 현에 대한 합성시간
Fig. 7. Synthesis Time for each string on guitar

본 논문에서는 1개의 GPU와 CPU 간의 성능 비교 뿐만 아니라 다수의 GPU를 사용하였을 경우 음 합성 알고리즘의 성능 추이 변화도 분석하였다. 이러한 다수의 GPU 환경에서 성능을 분석하기 위해 NVIDIA에서 제공하는 Visual Profiler을 이용하였다. Visual Profiler는 CUDA나 OpenCL 실행 파일을 실행하여 타임라인을 통하여 GPU상에서 알고리즘의 성능 분석과 병목 현상에 대한 정보를 볼 수 있는 크로스 플랫폼 성능 프로파일링 도구이다[11]. Visual Profiler를 이용하여 여기신호를 호스트(Host)에서 디바이스(Device)로 전송하는데 필요한 시간 HtoD, 합성 음을 디바이스에서 호스트로 읽어오는데 필요한 시간 DtoH, 실제 음 합성에 사용되는 시간을 측정하였다.

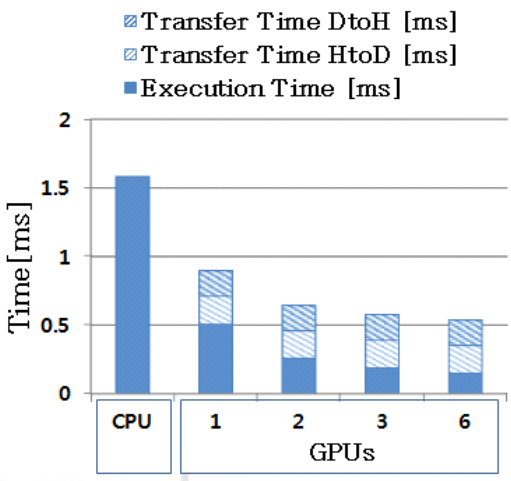

그림 8. 멀티 GPU의 성능 비교
Fig. 8. Performance comparison between multi-GPU

4. 상용 프로세서와의 성능 비교

본 논문에서는 네 가지의 형태의 GPU 시스템에서 기타 음 합성 알고리즘의 성능을 평가하였을 뿐만아니라 기존 고성능 TI DSP(TI TMS320C6416)와 CPU(Xeon x5690)와의 성능도 비교하였다. 표 3은 GPU 장치의 수(1, 2, 3, 6)를 변화시키며 측정한 성능을 상용 프로세서들과 비교한 결과를 보여준다. 성능 비교 결과, 신호처리 분야에서 많이 사용되는 TI DSP보다 최소 68배에서 최대 232배 높은 성능을 보였으며, CPU보다 최소 3배에서 최대 10배 높은 성능 향상을 보였다.

표 3. 상용 프로세서 아키텍처 간의 성능 비교
Table 3. Performance comparison between commercial processor architectures

항목	TI DSP	CPU	GPU			
Number of Device	1	1	1	2	3	6
Core	1	6	512	1024	1536	3072
Clock Frequency	720 MHz	3.46 GHz	1.51 GHz	1.51 GHz	1.51 GHz	1.51 GHz
Execution Time(ms)	34.5	1.58	0.90	0.64	0.58	0.54

V. 결 론

본 논문에서는 물리적 모델링 기반의 기타 음 합성 알고리즘을 병렬화하였으며, OpenCL을 사용하여 GPU 상에서 효율적으로 실행되도록 맵핑(mapping)하였다. 병렬 구현된 알고리즘의 결과를 확인하기 위해 44.1kHz의 샘플링율로 합성된 음과 원음의 청취 및 스펙트럼 분석 결과 거의 동일함을 알 수 있었다. 또한 음 합성에 사용되는 GPU의 수를 변화시키며 실험한 결과 GPU가 1개에서 2개로 늘어날 때 실행시간은 약 1.97배, 3개일 때 약 2.69배, 6개일 때 약 3.4배의 성능 향상을 보였다. 하지만 데이터 전송을 위한 시간을 포함한 총 실행시간은 2개 일 때 약 1.39배, 3개일 때 약 1.55배, 약 1.67배의 성능 향상을 보였다. 또한 1개의 GPU를 사용한 기타 음 합성은 TI DSP보다 실행시간에서 약 68배 높은 성능을 보였으며, CPU보다는 약 3배의 성능 향상을 보였다. 향후에는 기타 음 합성 알고리즘을 위해 성능 및 에너지 효율 측면에서 GPU 내 최적의 코어 개수를 결정하는 연구를 수행할 예정이다.

참고문헌

[1] M. karjalainen, T.Maki-Patola, A. Kanerva, A. Huovilainen, and P. janis, "Virtual air guitar," in Proc. of AES Convention, pp. 2-19, Oct. 2004.

[2] L. Kessous, J. Castet, and D. Arfib, "GXtar, an interface using guitar techniques," in Proc. of International Conference on New Interfaces for Musical Expression, pp. 192-195, June 2006.

[3] J. Kanebako, J. Gibson, and L. Mignonneau, "Mountain guitar : a musical instrument for everyone," in Proc. of International Conference on New Interfaces for Musical Expression, pp. 396-397, June 2007.

[4] M. Kang, S. Cho, and U. Chong, "Implementation of Non-Stringed Guitar Based on Physical Modeling Synthesis," Journal of the Acoustical Society of Korea, Vol. 28, No. 8, pp.119-126, Feb. 2009.

[5] J. E .Stone, D. Gohara, G. Shi, "OpenCL : A parallel programming standard for heterogeneous computing systems", Computing in Science and Eng, Vol. 12, No. 3, pp. 66-73, May 2010.

[6] H.-G. Jeon, J.-W. Ahn, J.-M. Kim, and C.-H. Kim, "Memory Delay Comparison between 2D GPU and 3D GPU," Journal of The Korea Society of Computer and Information, Vol. 17, No. 7, pp. 1-11, July 2012.

[7] H.-J. Choi, S.-G. Kang, J.-M. Kim, and C.-H. Kim, "Analysis of the CPU/GPU Temperature and Energy Efficiency depending on Executed Applications," Journal of The Korea Society of Computer and Information, Vol. 17, No. 5, pp. 9-19. May 2012.

[8] O. E. Albayrak, I. Akturk, O. Ozturk, "Effective Kernel Mapping for OpenCL Applications in Heterogeneous Platforms," Proc. of International Conference on Parallel Processing Workshop, pp. 81-88, Sept. 2012.

[9] NVIDIA Fermi Compute Architecture Whitepaper, http://www.nvidia.com/content/PDF/fermi

_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf
〔10〕 V. Valimaki, J. Juopaniemi, M. Karjalainen, and Z. Janosy, "Physical Modeling of Plucked String Instruments with Application to Real-time Sound Synthesis," J. Audio Eng. Soc., Vol. 44, No. 5, pp. 331-353, May 1996.
〔11〕 NVIDIA Visual Profiler, https://developer.nvidia .com/nvidia-visual-profiler

저 자 소 개

강 성 모

2011: 울산대학교
컴퓨터정보통신공학부 공학사.
현 재: 울산대학교
전기공학부 석사과정.
관심분야: 멀티미디어 신호처리,
임베디드 소프트웨어,
SoC설계
Email : dryujinwolf@mail.ulsan.ac.kr

김 종 면

1989: 명지대학교 전기공학과 공학사.
2000: University of Florida
전기컴퓨터공학과 공학석사.
2004: Georgia Tech.
전기컴퓨터공학과 공학박사.
현 재: 울산대학교 전기공학부 교수
관심분야: 컴퓨터구조, 병렬프로세서,
멀티미디어 신호처리,
SoC설계
Email : jongmyon.kim@gmail.com