한국컴퓨터정보학회 동계학술대회 논문집 제22권 제1호 (2014. 1)

GPU를 이용한 해금의 스펙트럼 모델링

Md Shohidul Islam°, Md Rashedul Islam, Fahmid Al Farid, 김종면*
°*울산대학교 전기공학부
e-mail: : {shohid°, rashedcse}@mail.ulsan.ac.kr, fahmid.farid@gmail.com, jmkim07@ulsan.ac.kr*

Spectral Modeling Synthesis of Haegeum using GPU

Md Shohidul Islam°, Md Rashedul Islam, Fahmid Al Farid, Jong-Myon Kim*
*School of Electrical Engineering, University of Ulsan

● 요 약 ●

This paper presents a parallel approach of formant synthesis method for *haegeum* on graphics processing units (GPU) using spectral modeling. Spectral modeling synthesis (SMS) is a technique that models time-varying spectra as a combination of sinusoids and a time-varying filtered noise component. A second-order digital resonator by the impulse-invariant transform (IIT) is applied to generate deterministic components and the results are band-pass filtered to adjust magnitude. The noise is calculated by first generating the sinusoids with formant synthesis, subtracting them from the original sound, and then removing some harmonics remained. The synthesized sounds are consequently by adding sinusoids, which are shown to be similar to the original *Haegeum* sounds. Furthermore, GPU accelerates the synthesis process enabling- real time music synthesis system development, supporting more sound effect, and multiple musical sound compositions.

키워드: Sound synthesis, Haegeum, IIR Filter, GPU

I. Introduction

Sound synthesis for musical instrument has been studied through a number of methods including sampling, filter, modulation, modeling, etc. Sampling and filtering can synthesize natural sound that uses recorded instrumental sound. But the approach is appropriate for solo play and when the playing style is changed, it requires re-sampling. Modulation can produce new sounds such as electrical sounds, but the synthesized sounds feel like artificial. On the other hands, these issues can be conquered with the method of modeling, which utilizes digital filters based on acoustical characteristics. The tone color can be adjusted by the parameters of the filters and the various playing styles can be described by adding or deleting certain filters, however, the modeling approach requires higher calculation complexity.

In general, modeling is categorized into physical modeling and spectral modeling. Physical modeling analyzes the sound production mechanism of the instrument at first, and then designs the appropriate model for music synthesis. Spectral modeling synthesis (SMS) is introduced by Serra in 1990 to synthesis musical instrumental sounds using spectral modeling technique [1]. The method is attributed with its effectiveness to generate like original sounds, and that is why, sound synthesis of musical instruments has been studied using SMS later on [2-3]. *Haegeum* is a Korean traditional musical instrument consisting of string. Spectral modeling can appropriately synthesize the string and wind instruments having the periodic property of their spectra. GPU based sound synthesis of *haegeum* has not yet studied using spectral modeling.

Sound synthesis using spectral modeling is accomplished by analyzing the spectrum of the instrumental sound as a combination of sinusoid (the "deterministic" part) and noise (the "stochastic" part) component [3-6]. Sinusoid is synthesized by digital resonator using impulse-invariant transform (IIT). For this calculation, pre-determined resonance frequency, bandwidth and magnitude for each note of *haegeum* are utilized by each resonator. Each resonator's output are band pass filtered to adjust the resonance

* Corresponding author.

amplitude. Noise component is generated by subtracting the sinusoids from the original sounds. Consequently, the synthesized single-notes of *haegeum* are resulted from adding the sinusoids with the noise components.

To achieve more augmented reality, the entire synthesis process is accelerated on graphics processing units (GPU). GPU offers massively parallel computing platform and thus can speed up the synthesis techniques enabling- (i) real time music synthesis system development, (ii) supporting more sound effects, and (iii) multiple musical sound composition. We validated our proposed approach using a Compute Unified Device Architecture (CUDA) [7] enabled NVIDIA GeForce GTX 560 graphics card.

The remainder of this paper is organized as follows. Section 2 describes *haegeum* in brief, Section 3 presents sound synthesis approach, Section 4 explains the GPU based implementation, Section 5 presents experimental result analysis and finally Section 6 concludes the paper.

II. Haegeum In Brief

Haegeum is a Korean traditional musical instrument consisting of two strings called *Junghyun* (the thick one) and *Yuhyun* (the thin one). Various sounds are produced by the instrument depending on the playing position. Table 1 shows several notes produced by the instrument along with their playing position and fundamental frequencies. In each playing position, total of eight single notes are possible- *Joong* (Ab3), *Nam* (C4), *Tae* (F4), *Im* (Bb4), *Im* (Bb3), *Hwang* (Eb4), *Joong* (Ab4), and *Nam* (C5). Haegeum sound is characterized by the bow velocity and the length of the string ranged by the finger tips. For example, a sharp sound is produced when the string length is reduced by pressing a finger down on it.

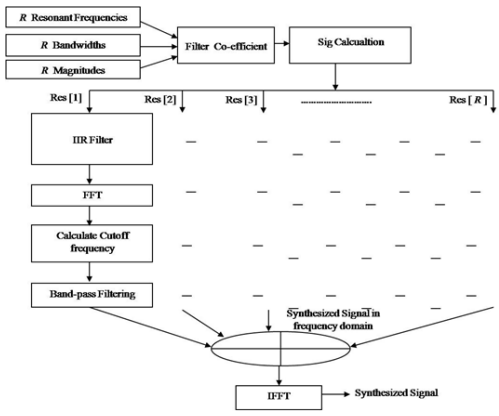

그림 1. 해금 단위음의 GPU 소리 합성 병렬 모델
Fig. 1. Parallel Approach for Sound Synthesis of a Haegeum Single Note on GPU

표 1. 각 음계의 기본 주파수
Table 1. Fundamental Frequencies for Haegeum Notes at Different Playing Positions

Hwangjong position			
Note	Fundamental frequency (Hz)	Note	Fundamental frequency (Hz)
Joong (Ab3)	210	*Im* (Bb3)	228
Nam (C4)	257	*Hwang* (Eb4)	317
Tae (F4)	354	*Joong* (Ab4)	433
Im (Bb4)	478	*Nam* (C5)	524
Jungryeo position			
Note	Fundamental frequency (Hz)	Note	Fundamental frequency (Hz)
Hwang (Eb4)	315	*Tae* (F4)	351
Joong (Ab4)	427	*Im* (Bb4)	474
Nam (C5)	527	*Mu* (Db5)	556
Hwang (Eb5)	611	*Tae* (F5)	668
Chonghwangjong position			
Note	Fundamental frequency (Hz)	Note	Fundamental frequency (Hz)
Joong (Ab4)	408	*Im* (Bb4)	457
Nam (C5)	519	*Hwang* (Eb5)	608
Tae (F5)	687	*Joong* (Ab5)	827
Im (Bb5)	930	*Nam* (C6)	1013

III. Proposed Parallel Model Of Haegeum Sound Synthesis

Synthesizing *haegeum* sounds refers to generating the notes as is mentioned in Table 1 by utilizing a number of characteristics parameters. Figure 1 shows the typical block diagram for the entire synthesis procedure in parallel approach on GPU in which the predetermined parameters such as resonance frequency, bandwidth and magnitudes of the resonance are input. Generating one single note requires a number of resonators. The numbers of resonators are equal to the number of resonances lie in single notes within one cycle. Resonator is implemented by digital filter using impulse-invariant transform (IIT). The input-output of the resonator is characterized by the resonant (formant) frequency , and the resonance bandwidth (ω_{BW}).The samples of the resonat or output, y(n), are resulted from the input, x(n), by using (1)-

$$y(n) = Ax(n) + By(n-1) + Cy(n-2) \quad (1)$$

The co-efficient A, B, and C are calculated using the bandwidth and resonance frequency by the IIT.

$$C = -e^{\omega_{BW}} \qquad (2)$$
$$B = 2e^{\omega_{BW}}\cos(\omega_r)$$
$$A = 1 - B - C$$

Thus the transfer function of the second order difference equation is given by (3).

$$T(z) = \frac{A}{1 - Bz^{-1} Cz^{-2}} \qquad (3)$$

where $z = e^{jw}$.

IV. Implementation on GPU

The synthesis of can be decomposed as (i) filter co-efficient (*A, B,* and *C*) calculation, (ii) synthesized band limited impulse train design, *x(n)*. It is referred to as sig generation later on, (iii) digital filter design by IIT, *y(n)*. It is referred to as infinite impulse response (IIR filter) (iv) generating spectrum, *T(z)*, by FFT, (v) band pass filtering, and (vi) converting back to the time domain by IFFT. In this paper, entire synthesis system is implemented efficiently on GPU in parallel as depicted in Fig. 1.

The parallel algorithm running on GPU is called kernel [8-9] and the complete system is implemented by four kernels- (a) *Coeff-Sig* for co-efficient and sig calculation, (b) Filter for generating y(n) in time domain, (c) *FFT* for attaining frequency domain sample *Y(n)*, and (d) BPF-IFFT for band pass filtering and inverse FFT to obtain final synthesized sound in time domain. For each kernel, number of GPU blocks, *gridDim*, and threads per block, *blockDim*, are dynamic that enable the GPU to generate any single notes with any number of resonators.

V. Experimental Results Analysis

This paper aims to implement the *haegeum* sound synthesis on GPU and investigate the achievable speedup over CPU. Therefore, the sequential synthesis approach is implemented on a traditional CPU and the equivalent parallel approach is implemented on the GPU whose system configuration is given in Table II.

표 2. 실험 환경
Table 2. Experimental Environment

Property	CPU	Property	GPU
Processor	Intel(R) Core(TM) i5-3570K	Brand name	NVIDIA GeForce GTX 560
Clock speed	3.40GHz	Processor clock	1620MHz
No. of Cores	4	CUDA core	336
No. of threads	4	Total MP	7
RAM	8.00GB	Max thread per block	1024
Bus/Core ratio	34	Shared memory per MP	49152 Byte
Operating system	Windows 7, 32 bit	Total global memory	1 GB

As described in Section IV, we partition the entire task into four different kernels- Coeff-Sig, Filter, FFT, and BPF-IFFT respectively. Thus, we attempt to observe the achievable speedup at each individual task for the synthesis process. Table III and IV show the CPU execution time, GPU execution time, and speedup information for the synthesis of single notes with various resonator, such as 15 and 19, respectively. As can be seen in the above tables, GPU achieves considerable speedup over CPU for each task segment. One interesting observation is that, when the number of resonator increases, GPU speedup also increases as shown the tables. This is because of the fact that, executing more resonators in CPU requires more time due to serial processing, whereas GPU mitigates the effect by launching equal number of blocks as the resonators.

표 3. 해금 단위 음의 합성 결과 (공명기 = 15)
Table 3. Synthesis Of a Haegeum Single Note (Resonators =15)

	CPU execution time (ms)	GPU execution time (ms)	Speedup
Coeff-Sig	11.39	0.18	63
Filter	32.54	0.34	95
FFT	24.60	0.95	26
BPF-IFFT	8.14	0.09	87

표 4. 해금 단위 음의 합성 결과 (공명기 = 19)
Table 4. Synthesis Of a Haegeum Single Note (Resonators =19)

	CPU execution time (ms)	GPU execution time (ms)	Speedup
Coeff-Sig	14.43	0.20	72
Filter	41.22	0.35	118
FFT	31.16	0.97	32
BPF-IFFT	10.31	0.09	115

V. Conclusions

In this paper, we propose a novel, efficient, and parallel sound synthesis approach of *haegeum* on GPU. Accelerating the synthesis task enables real time music synthesis system development, supporting more sound effects, and multiple musical sound compositions. We compare the performance of the GPU-based approach with the equivalent sequential approach that runs on the traditional CPU. The experimental result

demonstrates that GPU exceedingly outperforms the CPU based approach.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. NRF-2013R1A2A2A05004566).

References

[1] X. Serra and J. O. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition," Comput. Music J., vol. 14, no. 4, pp. 12–24, 1990.

[2] X. Serra and J. O. Smith, "Residual Minimization in a Musical Signal Model based on a Deterministic plus Stochastic Decomposition," J. Acoust. Soc. Am., vol. 95, no. 5–2, pp. 2958-2959, 1994

[3] Spectral Audio Signal Processing, available at http://ccrma.stanford.edu/~jos/sasp/, Online Book, 2007.

[4] H. K. Dennis, "Software for a cascade/parallel formant synthesizer," J. Acoust. Soc. Am., vol. 67, no. 3, pp. 971-995, 1980.

[5] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, Ronald W. Schafer, Discrete-Time Signal Processing, 2nd ed., Prentice Hall, 1989.

[6] X. Serra and J. Bonada,“Sound Transformations Based on the SMS High Level Attributes,” in Proc. Int’l Conf. Digital Audio Effects (DAFX98), 1998

[7] J. Sanders, and E. Kandrot. (2010, Jul. 29). CUDA by Example: An Introduction to General-Purpose GPU Programming. (1stedition).[On-line]. Avaiable: http://www.amazon.com/CUDA-Example-Introduction-General-Purpose-Programming/dp/0131387685

[8] Y. Liu, L. Guo, J. Li, M. Ren, and K. Li, "Parallel Algorithms for Approximate String Matching with k Mismatches on CUDA," in Proc. 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), pp. 2414-2422, May 2012.