J. Parallel Distrib. Comput. ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Design space exploration in many-core processors for sound synthesis of plucked string instruments

Jiwon Choi [a,1], Myeongsu Kang [a,1], Yongmin Kim [a,1], Cheol-Hong Kim [b,2], Jong-Myon Kim [a,*]

[a] School of Electrical Engineering, University of Ulsan, Ulsan, South Korea

[b] School of Electronics and Computer Engineering, Chonnam National University, Gwangju, South Korea

ARTICLE INFO

Article history:
Received 21 May 2011
Received in revised form
4 June 2012
Accepted 30 July 2012
Available online xxxx

Keywords:
Many-core processor architecture
Physics-based sound synthesis
Sample-per-processing-element
Design space exploration
Musical instruments

ABSTRACT

Recent advances in physics-based sound synthesis have unveiled numerous possibilities for the creation of new musical instruments. Despite the fact that research on physics-based sound synthesis has been going on for three decades, its higher computational complexity compared to that of signal modeling has limited its use in real-time applications. This limitation has motivated research on parallel processing architectures that support the physics-based sound synthesis of musical instruments. In this paper, we present analytical results of the design space exploration of many-core processors for the physics-based sound synthesis of plucked-string instruments including acoustic guitar, classical guitar and the *gayageum*, which is representative of a Korean plucked-string instrument. We do so by quantitatively evaluating the significance of a sample-per-processing-element (SPE) ratio – i.e., the amount of sample data directly mapped to each processing element, which is equivalent to varying the number of processing elements for a fixed sample size on system performance and efficiency using architectural and workload simulations. The effect of the sample-to-processor ratio is difficult to analyze because it fundamentally affects both hardware and software design when varied. In addition, the optimal SPE ratio is not typically at either extreme of its range – i.e., one sample per processor or one processor per an entire sample. This paper illustrates the correlation between a fixed problem sample size, SPE ratio and processing element (PE) architecture for a target implementation in 130-nm CMOS technology. Experimental results indicate that an SPE in the range of 5513 to 2756, which is equivalent to 48 to 96 PEs for guitars and 96 to 192 PEs for the *gayageum*, provides the most efficient operation for the synthesis of musical sounds sampled at 44.1 kHz, yielding the highest task throughput per unit area or per unit energy. In addition, the produced synthesized sounds appear to be very similar to the original sounds, and the selected optimal many-core configurations outperform commercial processor architectures including DSPs, FPGAs, and GPUs in terms of area efficiency and energy efficiency.

1. Introduction

Digital synthesis techniques used to synthesize the sounds of musical instruments have been designed using highly developed digital signal processing. Representative examples of sound synthesis techniques include sampling synthesis, additive synthesis, subtractive synthesis, frequency modulation synthesis, spectral modeling synthesis (SMS) and physical modeling (PM) synthesis. These sound synthesis techniques support a creative musical environment that allows the user to simulate sounds according to his or her individual senses and sensitivity. For examples, *Virtual Air Guitar* [11] includes a portable user interface for expressive playing of the synthetic guitar, and *Overtone Violin* [17] has gestural controllers for violin. Recently, mobile phones have also been explored as new interfaces for controlling musical parameters as part of locative performances. *iPhone ocarina* [23] and *Stanford Mobile Phone Orchestra* [16] are representative examples of this trend.

To develop musical interfaces using sensors or mobile phones, physics-based methods of musical sound synthesis have received increasing attention in the last three decades [1,18]. Consequently, musical synthesis algorithms based on physical models are becoming more and more efficient in terms of creating high-quality sounds that imitate natural instruments. Physical modeling can provide useful information for acousticians about the most

* Correspondence to: Bldg. #7, Room #308, 93 Daehak-ro Mugeo-dong Nam-gu, Ulsan 680-749, South Korea.

E-mail addresses: jiwon912@mail.ulsan.ac.kr (J. Choi), ilmareboy@ulsan.ac.kr (M. Kang), jafstar@nate.com (Y. Kim), chkim22@chonnam.ac.kr (C.-H. Kim), jmkim07@ulsan.ac.kr, jongmyon.kim@gmail.com (J.-M. Kim).

[1] Present address: Bldg. #7, Room #308, 93 Daehak-ro Mugeo-dong Nam-gu, Ulsan 680-749, South Korea.

[2] Present address: Bldg. #7, Room#506, 77 Yongbong-ro, Buk-gu, Gwangju 500-757, South Korea.

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.07.013

important phenomena during sound production and how the sound of the instrument would change by varying its physical properties.

Several approaches to physical modeling have been developed for generating rich and high-quality instrument sounds. For example, Karplus et al. introduced a very simple system that contained three components such as a low-pass filter, a delay and a gain in a feedback loop to synthesize the sound of plucked strings [13]. They used commuted waveguide synthesis via physical modeling to implement the sound synthesis of plucked string instruments such as acoustic guitars, banjos and mandolins [19,12]. Note that, with this method, the same wave equation could be applied to any perfectly elastic medium that is displaced along a one-dimensional waveguide with extensions to two or three dimensions. In another study, sounds made by a kantele which is a traditional Finnish instrument and a dan tranh which is a Vietnamese traditional instrument were synthesized using a physical modeling synthesis that represents instrument characteristics [21,9]. While such models are physically relevant abstractions, the required computational complexity owing to many numerical integrations of the wave equation limit their use in real-time applications.

Application-specific integrated circuits (ASICs) can meet the requirements of high performance with low power consumption for such sound synthesis algorithms; however, they provide limited, if any, programmability or flexibility needed for varied applications. General-purpose microprocessors (GPPs) or digital signal processors (DSPs) offer the necessary programmability and flexibility for varied applications; yet their potentials are limited. Neither GPPs nor DSPs will be able to meet the much higher levels of performance required by the emerging sound synthesis algorithms, which are used to synthesize the precise sounds of musical instruments, because they lack the ability to exploit the massive parallelism available in these algorithms.

Among the many computational models available for sound synthesis applications, single instruction multiple data (SIMD) processor array architectures are promising candidates [4]. Whereas instruction-level or thread-level processors use silicon area for large multiport register files, large caches, and deeply pipelined functional units, SIMD-based processor arrays contain simple, low-cost processing elements (PEs) for the same silicon area. As a result, SIMD-based processor arrays often employ thousands of PEs while possibly distributing and co-locating PEs with the data I/O to minimize storage and data communication requirements.

While it is evident that the overall performance improves by increasing the number of PEs [24], no general consensus has been reached regarding which grain sizes of processors and memories on the array system provide the most efficient operation of the aforementioned sound synthesis algorithms with regard to performance and both area efficiency and energy efficiency for sample-per-processing-element (SPE) ratio variations which are variations in the amount of sample data directly mapped to each PE.

This paper presents the effects of varying the SPE ratio in an integrated many-core array and proposes the PE architecture that most efficiently delivers the required processing performance with the lowest cost and longest battery life for the physics-based sound synthesis algorithm of three plucked-string instruments including the acoustic guitar, the classical guitar, and the *gayageum*. Earlier studies have focused primarily on a single processor [9,5], whereas this study quantitatively evaluates the impacts of the SPE ratio on system performance and efficiency for many-core array architectures. Unlike many architectural parameters, performance analysis of the SPE ratio is difficult because, as the SPE ratio is varied, characteristics of the software and hardware implementation change the number of executed instructions needed to complete the task.

This paper systematically evaluates the effects of the SPE ratio using a retargeting many-core simulator combined with both area and energy technology modeling. In addition, this paper illustrates the correlation between problem size, SPE ratio and PE architecture for implementation in 130-nm complementary-symmetry metal-oxide-semiconductor (CMOS) technology. To identify the most efficient SPE ratio, seven different PE configurations were simulated for the sound synthesis of plucked-string instruments with a sampling rate of 44.1 kHz and 16-bit quantization. Experimental results indicate that the most efficient operation is achieved at an SPE ratio in the range of 5513–2756. This paper also compares the performances and efficiencies of the selected optimal many-core configurations to commercial high-performance architectures including Digital Signal Processors (DSP), Graphics Processing Units (GPU), and Field-Programmable Gate Arrays (FPGA) in order to show the potential of the proposed PE architecture.

The rest of this paper is organized as follows: Section 2 introduces background information for the musical instruments and sound synthesis techniques. Section 3 presents the structure of the many-core architecture and design space exploration using SPE ratio variation. Section 4 illustrates the simulation environment for evaluating the effects of different SPE values and parallel implementation of sound synthesis of the plucked-string instruments. Section 5 analyzes the execution performance as well as the area and energy efficiency for each SPE configuration. Section 6 concludes this paper.

2. Background

String instruments produce sound from one or more vibrating strings, and they are usually categorized according to the technique used to make the strings vibrate such as plucked-string instruments and bowed-string instruments. Plucked-string instruments produce sound when one or more strings are plucked, and this plucking is the sole method of playing instruments such as banjos, guitars and mandolins. In contrast, bowed-string instruments, such as the violin, viola and cello, generate sound through the pushing/pulling of one or more strings with a bow. To synthesize the sounds of these instruments, many sound synthesis techniques are used, including sampling synthesis, additive synthesis, subtractive synthesis, frequency modulation synthesis, spectral modeling synthesis (SMS) and physical modeling (PM) synthesis. Among these methods, it is important to choose the proper synthesis technique for each musical instrument.

Currently, SMS and PM synthesis are the dominant synthesis techniques for stringed instruments. SMS is based largely on the perception of sound and it is usually suitable for generating periodic sounds such as those emitted by bowed instruments [22,6]. In contrast, PM synthesis is based on mathematical descriptions of acoustic musical instruments and is useful in synthesizing plucked-string instruments [21]. In this study, we employ a commuted waveguide synthesis, which is a kind of PM synthesis technique, to synthesize the sounds of the plucked-string instruments and then implement the synthesis algorithm on the reference many-core architecture. The target musical instruments are the acoustic guitar, the classical guitar, and the *gayageum*, a traditional Korean plucked-string instrument.

2.1. Target musical instruments

These target musical instruments are all plucked-string instruments whose sounds are synthesized using the specified many-core architecture. Their structures are shown in Fig. 1. As depicted in Fig. 1(a) and (b), the acoustic guitar and classical guitar have the same structure. The acoustic guitar is usually played with the fingers or a pick, and it consists of a body with a rigid neck, to which

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

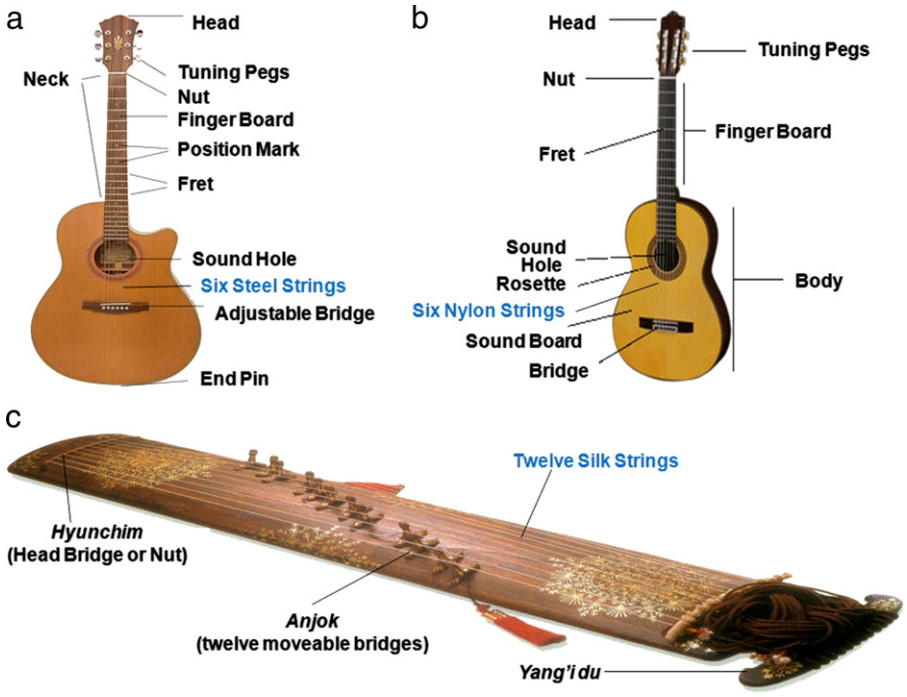

Fig. 1. Structures of target instruments: (a) acoustic guitar, (b) classical guitar and (c) *gayageum*.

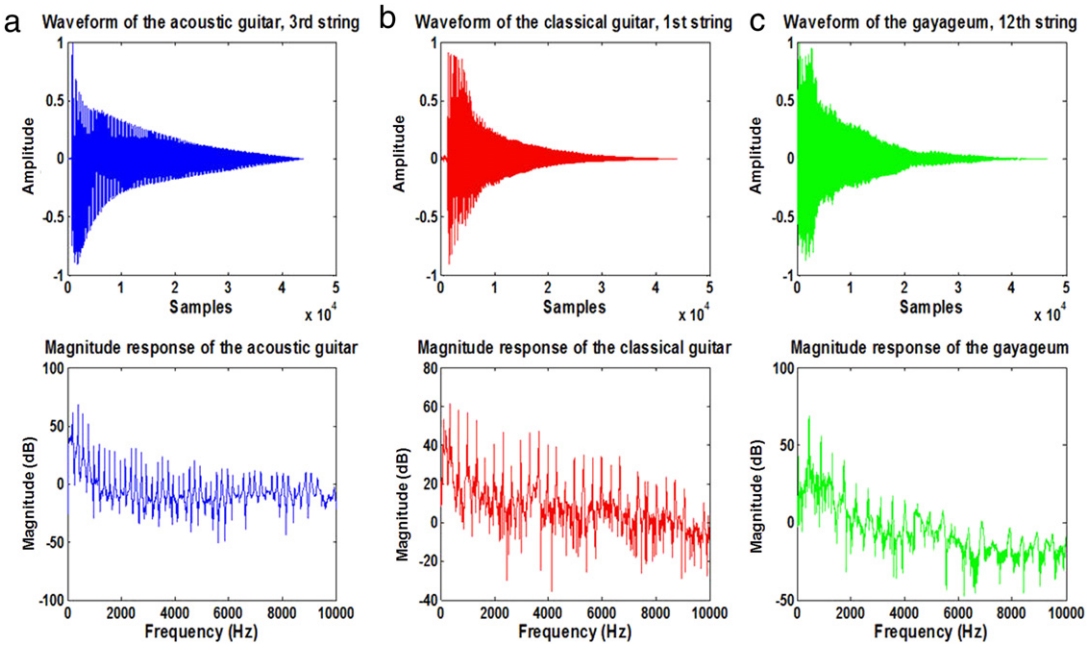

Fig. 2. Waveforms and spectra of target instruments: (a) acoustic guitar (G3, 196 Hz), (b) classical guitar (E4, 329.63 Hz) and (c) *gayageum* (A4, 436.63 Hz).

six steel strings are attached. The classical guitar, on the other hand, is typically strung with six nylon strings and is usually plucked with the fingers. Fig. 1(c) shows the structure of the *gayageum*, consisting of a wooden board made of paulownia wood and *anjok* (movable bridges), which support 12 strings made of twisted silk. *Tolgwae* (pegs for adjusting string tension) are inserted into the other side of the board beside the *yang'i du* (literally "the ear of sheep", to which the strings are attached), which is located at the end of the board. To play the *gayageum*, the left hand presses the strings, while the fingers on that hand make various types of movements such as shaking, bending and vibrating the strings. The right hand is used to pluck or strum the strings. The tone of the *gayageum* is soft, delicate and subtle.

Fig. 2 illustrates the waveforms and spectra of notes played on the three plucked-string instruments: G3 (196 Hz) for the acoustic guitar, E4 (329.63 Hz) for the classical guitar and A4 (463.63 Hz) for the *gayageum*. The decay of each note can be seen on the waveforms, and their decays are similar: a short attack and release followed by a longer sustain. Furthermore, each spectrum illustrates its harmonic structure, and it is possible to detect some of the harmonic differences that are a consequence of the various string materials. For the steel-stringed instrument, the harmonic components regularly occur at multiples of a fundamental frequency, with a very small amount of inharmonicity as shown in Fig. 2(a). However, Fig. 2(b) and (c) illustrate the inharmonicity is comparatively larger for the nylon- and silk-stringed instruments.

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

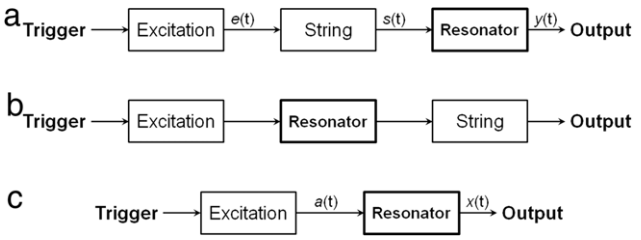

Fig. 3. Block diagrams for synthesizing the plucked-string instruments: (a) schematic diagram of a stringed musical instrument, (b) equivalent structure in the linear, time-invariant case and (c) an aggregate excitation given by the convolution of original excitation with the resonator impulse response.

Consequently, it is more complicated to synthesize the sounds of nylon- and silk-stringed instruments using commuted waveguide synthesis. For example, their synthesis requires a larger amount of memory space to store aggregate excitations, and it is difficult to extract the aggregate excitations in order to re-create the precise sounds.

2.2. Commuted waveguide synthesis

In acoustic stringed musical instruments, the strings couple via a bridge to some resonating structure that is required for efficient transduction of string vibration to acoustic propagation, and the resonator imposes its own characteristic frequency response [20]. Spectral characteristics of the string excitation, the string resonances and the body/soundboard/enclosure resonator are combined multiplicatively as illustrated in Fig. 3. As shown in Fig. 3(a), plucked-string instruments are generally modeled as linear systems. The key idea of commuted synthesis is to commute the string and resonator as depicted in Fig. 3(b), providing a highly efficient way of producing high quality plucked-string sounds.

The body response of a string instrument is complex and generally requires a high-order digital filter to simulate it. However, it is possible to record a body "impulse response" and use it as the string excitation by taking advantage of system commutativity [20]. Consequently, the excitation is convolved with the resonator impulse response, and thus a single aggregate excitation table can be obtained as shown in Fig. 3(c).

The aggregate excitation includes the force exerted by a pick or a finger and the characteristics of the body/soundboard. Therefore, commuted synthesis greatly reduces the complexity of plucked-string instrument implementations because the body filter is replaced by an inexpensive lookup table [19,10,14]. As mentioned previously, however, the inharmonicity changes for different string materials. Consequently, different amounts of excitation are required to generate natural and rich instrument sounds. This complexity can affect the processor grain size determination when we implement the commuted waveguide synthesis algorithm using the reference many-core array architecture. In addition, the number of strings on the target instruments affects the grain size of the many-core architecture. Thus, this paper explores the design space of the many-core array architecture for synthesizing sounds of these plucked-string instruments.

2.3. String model

This paper synthesizes plucked-string instrument sounds using commuted waveguide synthesis. Fig. 4 shows the implementation of the string model.

2.3.1. Delay length

In the string model, the delay length determines the fundamental frequency of the desired sound and can be calculated as:

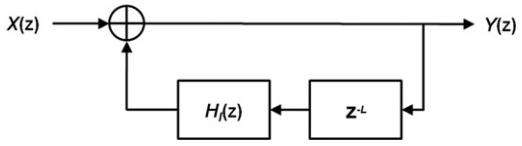

Fig. 4. A block diagram of the string model. z^{-L}: delay line; $F(z)$: fractional delay filter; $H_l(z)$: loop filter.

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

where f_s and f_0 are the sampling rate and the desired fundamental frequency, respectively. The delay length, L, is generally a real number, but there is no way to express any time index between n and $n+1$ in the digital domain such as $n+0.7$. Thus, we round off the delay length to the nearest whole number.

2.3.2. Loop filter

The loop filter represents the frequency-dependent damping of a physical string, and its transfer function is given by

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

where g is the filter gain at its lowest fundamental frequency and a_1 is a filter coefficient that determines the cutoff frequency of the filter. For $H_l(z)$ to be applicable for a stable low-pass transfer function, we require that $-1 < a_1 < 0$ and $|g| \ll 1$ [21]. To design a loop filter, we need to estimate the damping factors, g and a_1, at the harmonic frequencies of the sound and this can be achieved using the short-time Fourier transform (STFT) and tracking the amplitude of each harmonic. The STFT of $y(n)$ is a sequence of discrete Fourier transforms (DFT) defined as

$$Y_m(k) = \sum_{n=0}^{N-1} w(n)y(n+mH)e^{-jw_k n}, \quad m = 0, 1, 2, \ldots \tag{3}$$

with

$$w_k = \frac{2\pi k}{N}, \quad k = 0, 1, 2, \ldots, N-1, \tag{4}$$

where N is the length of the DFT, $w(n)$ is a window function, and H is the hop size (in samples) per frame. To obtain a suitable compromise between time and frequency resolution, it is necessary to use a window length of four times the period length of the signal [21]. The overlap of the windows is 50%, implying that H is 0.5 times the window length. Furthermore, we need to apply zero padding by filling the signal buffer with zeros to reach $N = 4096$ in order to increase the resolution in the frequency domain. The spectral peaks corresponding to harmonics can be found from the magnitude spectrum and the number of harmonics to be detected is typically either $N_h < 20$ for the acoustic and classical guitar or $N_h < 8$ for the *gayageum*, respectively. Spectral peak detection results in a sequence of magnitude and frequency pairs for each harmonic as shown in Fig. 5.

The sequence of magnitude values for one harmonic is called the envelope curve of that harmonic. A straight line is matched to each envelope curve on a decibel scale, since ideally the magnitude of every harmonic should decay exponentially, that is, linearly on a logarithmic scale. As a result, the corresponding loop gain of the string model at the harmonic frequencies is computed as

$$G_k = 10^{\beta_k L/20H}, \quad k = 1, 2, \ldots, N_h, \tag{5}$$

where β_k are the slopes of the envelopes and can be obtained by calculating the slope between the maximum of the envelope curve and the minimum of the envelope curve. H is the hop size and the sequence G_k determines the magnitude response of the loop filter $H_l(w)$ at the harmonic frequencies w_k, $k = 1, 2, \ldots, N_h$.

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

Table 1
Delay lengths and loop filter coefficients used in this study.

Notes	Delay length	Loop filter coefficient		Notes	Delay length	Loop filter coefficient	
		a_1	g			a_1	g
Acoustic guitar				Classical guitar			
E4 (329.63 Hz)	134	−0.0170	0.9908	E4 (329.63 Hz)	134	−0.3400	0.9654
B3 (246.94 Hz)	179	−0.0280	0.9866	B3 (246.94 Hz)	179	−0.1320	0.9929
G3 (196.00 Hz)	225	−0.1190	0.9842	G3 (196.00 Hz)	225	−0.2780	0.9830
D3 (146.83 Hz)	300	−0.3040	0.9787	D3 (146.83 Hz)	300	−0.3580	0.9909
A2 (110.00 Hz)	401	−0.5360	0.9763	A2 (110.00 Hz)	401	−0.2400	0.9762
E2 (82.41 Hz)	535	−0.6520	0.9848	E2 (82.41 Hz)	535	−0.4570	0.9819
Gayageum							
A4 (436.63 Hz)	101	−0.0200	0.9937	G3 (195.13 Hz)	226	−0.1750	0.9816
G4 (386.84 Hz)	114	−0.1300	0.9927	E3 (163.94 Hz)	269	−0.2140	0.9773
E4 (326.67 Hz)	135	−0.0300	0.9903	D3 (146.03 Hz)	302	−0.4740	0.9772
D4 (292.05 Hz)	151	−0.1460	0.9852	A2 (109.38 Hz)	403	−0.2330	0.9738
B3 (246.37 Hz)	179	−0.1810	0.9852	G2 (97.35 Hz)	453	−0.4310	0.9693
A3 (219.40 Hz)	201	−0.1550	0.9817	D2 (72.89 Hz)	605	−0.5310	0.9694

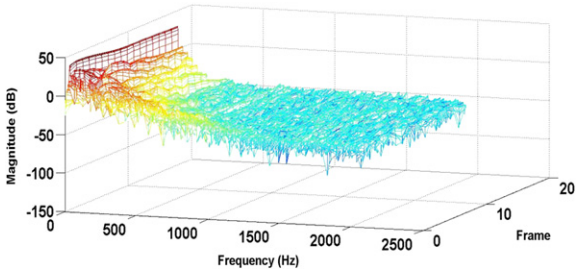

Fig. 5. Spectral peak detection results in a sequence of magnitude and frequency pairs for each harmonic.

As mentioned before, the loop filter represents frequency damping and our hearing is sensitive to the change of decay rate of a sinusoid. Thus, the loop filter should be designed based on an auditory criterion. To match the frequency response of the loop filter $H_l(z)$ to N_h harmonics of the sound signal, we utilize weighted least-squares design and define the error function for the magnitude-only approximation as follows:

$$E = \sum_{k=0}^{N_h-1} W(G_k)\left[|H_l(w_k)| - G_k\right]^2, \tag{6}$$

where N_h is the number of frequency points where the loop gain G_k is approximated, and $W(G_k)$ is a nonnegative error weighting function defined as

$$W(G_k) = \frac{1}{1-G_k}. \tag{7}$$

Let us denote the numerator of Eq. (2) by

$$A = g(1+a_1), \tag{8}$$

and $H_l(w)$ with the numerator removed by

$$\widetilde{H}_l(w_k) = \frac{H_l(w_k)}{A}. \tag{9}$$

Then, Eq. (6) can be expressed as

$$E = \sum_{k=0}^{N_h-1} W(G_k)\left[\left|A\widetilde{H}_l(w_k)\right| - G_k\right]^2. \tag{10}$$

The gain g of the loop filter at low frequencies can be chosen based on the loop gain values of the lowest harmonics and it is good enough to set $g = G_1$. The value for the coefficient a_1 that minimizes E can be found by differentiating Eq. (10) with respect to a_1. This yields

$$\frac{\partial E}{\partial a_1} = 2A_0 \sum_{k=0}^{N_h-1} W(G_k)\frac{\partial\left|\widetilde{H}_l(w_k)\right|}{\partial a_1}\left[\left|A_0\widetilde{H}_l(w_k)\right| - G_k\right]. \tag{11}$$

By substituting Eqs. (2) and (7), we can write $\frac{\partial E}{\partial a_1}$ as given in Box I.

Now, we should find the zero of Eq. (12) and in practice we find a near-optimal solution in the following way. The value of the derivative is evaluated, and depending on the sign of the result, a_1 is changed by a small increment, the derivative is evaluated again, and so on. After the derivative has reached a very small value, the iteration is terminated and the final value for a_1 is used in the sound synthesis. Table 1 shows delay lengths and loop filter coefficients used in this study. Consequently, a differential equation to synthesize musical sounds can be defined as

$$\begin{aligned} Y(z) &= X(z)\times\frac{1}{1-z^{-L}H_l(z)} \\ &= X(z)\times\frac{1}{1-z^{-L}\cdot g\frac{(1+a_1)}{1+a_1z^{-1}}} \\ &= \frac{(1+a_1z^{-1})X(z)}{1+a_1z^{-1}-g(1+a_1)z^{-L}} \end{aligned} \tag{13}$$

$$\left(1+a_1z^{-1}-g(1+a_1)z^{-L}\right)Y(z) = (1+a_1z^{-1})X(z)$$

$$y(n)+a_1y(n-1)-g(1+a_1)y(n-L) = x(n)+a_1x(n-1)$$

$$y(n) = x(n)+a_1x(n-1)-a_1y(n-1)+g(1+a_1)y(n-L).$$

2.4. Single aggregate excitation

The single aggregate excitation can be estimated using an inverse filter via the inverted transfer function of the string model. The transfer function of the string model in Fig. 4, $S(z)$, can be expressed as

$$S(z) = \frac{1}{1-z^{-L}H_l(z)}. \tag{14}$$

Inverting this equation yields

$$S^{-1}(z) = \frac{1+a_1z^{-1}-g(1+a_1z^{-1})z^{-L}}{1+a_1z^{-1}}. \tag{15}$$

The harmonics cannot be not very accurately canceled using Eq. (15) because the order of the loop filter is low. However, the low-order loop filter was chosen because it is efficient for implementation of the synthesis model. Consequently, the resulting

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

$$\frac{\partial E}{\partial a_1} = 2A_0 \sum_{k=0}^{N_h-1} \frac{A_0 \cos w_k (1 + a_1 \cos w_k)^{-3} - G_k \cos w_k (1 + a_1 \cos w_k)^{-1}}{1 - G_k} \tag{12}$$

Box I.

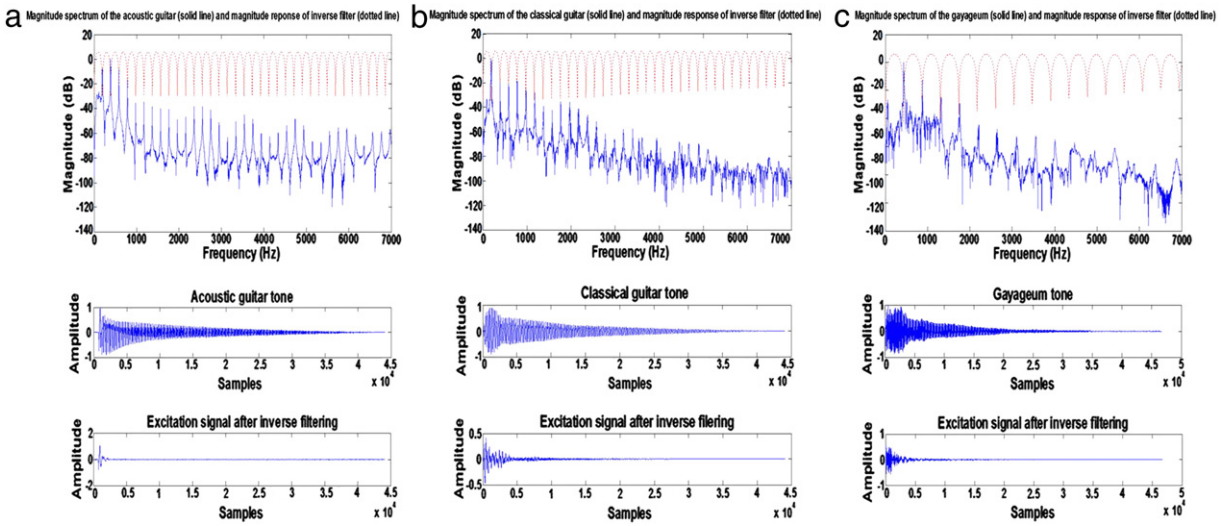

Fig. 6. Magnitude spectra of the target instruments, magnitude response of the inverse filter, and aggregate excitations after inverse filtering for the (a) acoustic guitar (G3, 196 Hz), (b) classical guitar (G3, 196 Hz) and (c) *gayageum* (G4, 386.84 Hz).

excitation suffers from high-frequency noise and inharmonic components. Fig. 6 depicts an example of excitations for each instrument. As illustrated in Fig. 6, the classical guitar results in the largest amount of excitation, followed by that of the *gayageum*. As we expected, the acoustic guitar results in the smallest amount of excitation.

3. Experimental environment

3.1. Many-core array architecture

Fig. 7 shows a reference many-core array along with its interconnection network. This array is a two-dimensional PE type, which contains local memory and an array control unit (ACU). The PEs execute a set of instructions in a lockstep fashion when the sound sample data are distributed. PEs in the array are interconnected in a mesh network so that the ACU controls the PE array, and an autonomous data exchange interface (DEI) transfers data between an addressed memory word in each PE and an array in the I/O unit's memory map. Sound sample data is loaded into the many-core array through a high-bandwidth interface. Each PE has the following minimum characteristics:

- 16 32-bit three-ported general-purpose registers,
- ALU: computes basic arithmetic and logic operations,
- Barrel shifter: performs multi-bit logic/arithmetic shift operations,
- MACC: multiplies 32-bit values and gathers them into a 64-bit accumulator,
- Sleep: activates or deactivates a PE based on local information, and
- Nearest neighbor communications through a north–east–west–south (NEWS) network and serial I/O unit.

3.2. Design space exploration: SPE ratio variation

A key design issue for portable musical systems is determining the impact of having direct access to a sound sample on the processor grain size. To determine the effect of grain size on the reference many-core array architecture in this study, the SPE ratio variation was selected as the design variable. The SPE ratio describes the amount of sample data directly mapped to each processing element. For this paper, a discrete set of SPE mapping is defined (see Table 2), and the local memory size is varied according to the SPE ratio. It is shown that as the SPE ratio is increased, additional local memory for each PE is needed to store the synthesis information (such as filter coefficients, the delay length for each string, and temporary data produced during processing). The local memory size is defined as follows:

$$\mathrm{MEM_{pe}} = \frac{\text{Max Delay Length}}{\mathrm{PE_{str}}} + 2 \quad \text{(words)}, \tag{16}$$

where *Max Delay Length* is the maximum delay length of the desired sounds and $\mathrm{PE_{str}}$ is the number of PEs per each string of the target instruments. For these studies, the *Max Delay Length* was set to 605 because the lowest fundamental frequency of the target musical instruments was 78.89. In addition, the system required a supplemental two-word memory space to store the filter coefficients including a loop filter coefficient (a_1) and a loop gain (g). As a result, the overall system memory was then calculated as $\mathrm{MEM_{sys}} = \mathrm{MEM_{pe}} \times N_{\mathrm{pe}}$ (words), where N_{pe} is the number of PEs.

Fig. 8 pictorially illustrates the assignment of sound samples based on the SPE ratio. Each PE basically synthesizes one sound sample for one of the corresponding instrument strings. However, two sound samples are simultaneously produced when the SPE value is 22,050. For simplicity, seven SPE values were used for the analysis, as defined by $\mathrm{SPE} = f_s/\mathrm{PE_{str}}$. The number of processing elements (N_{pe}) required for any given SPE value to cover the same

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

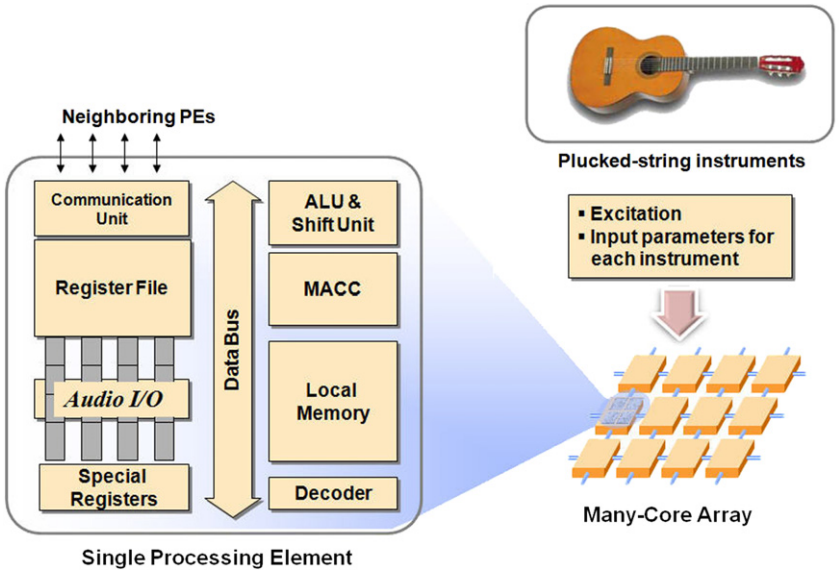

Fig. 7. A block diagram of a many-core array and a processing element.

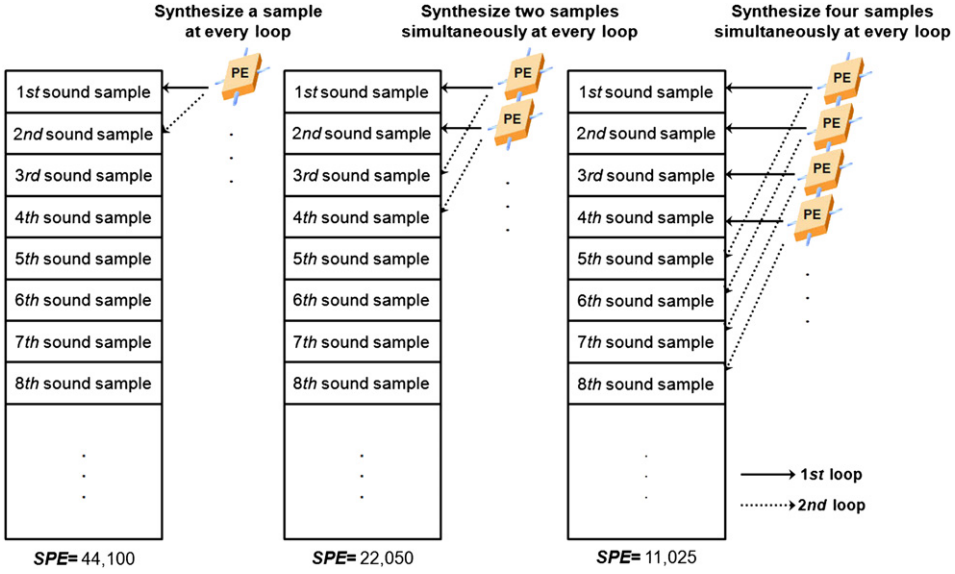

Fig. 8. An example of the assignment of sound samples based on the SPE ratio.

sound sample size is given by $N_{pe} = STR_{num} \times 2^i$ $(i = 0, \ldots, 6)$, where STR_{num} is the number of strings of the target instruments. For all of the configurations, a fixed 44,100 sound data size with a sampling rate of 44.1 kHz and 16-bit quantization was used to synthesize the desired sounds. In addition, all configurations were implemented in 130-nm CMOS technology with a 100 MHz clock frequency for performance analysis. Table 2 describes seven different many-core architecture configurations based on the SPE ratio variations. For a fixed problem sample size, the SPE ratio determines the peak system performance. Consequently, an upper bound on the SPE ratio was set to the minimum system throughput required by the algorithm.

Table 2
SPE configurations used in this study.

Sample per element (SPE)	Memory/PE (words)	PEs/system		System memory (KB)	
		Guitar	*Gayageum*	Guitar	*Gayageum*
44,100	610	6	12	14.64	29.28
22,050	308	12	24	14.79	29.57
11,025	157	24	48	15.08	30.15
5,513	81	48	96	15.52	31.11
2,756	43	96	192	16.52	33.03
1,378	23	192	384	17.67	35.33
689	15	384	768	23.04	46.08

4. Methodology

4.1. Methodology infrastructure

Fig. 9 shows a methodology infrastructure that is divided into three levels: application, architecture and technology. At the application level, an instruction-level many-core simulator was used to profile execution statistics such as cycle count, dynamic instruction frequency and PE utilization for seven different PE configurations by retargeting and optimizing the sound synthesis algorithm of the plucked-string instruments for each configuration based on the architecture and its execution properties. At the architectural level, the architectural modeling

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

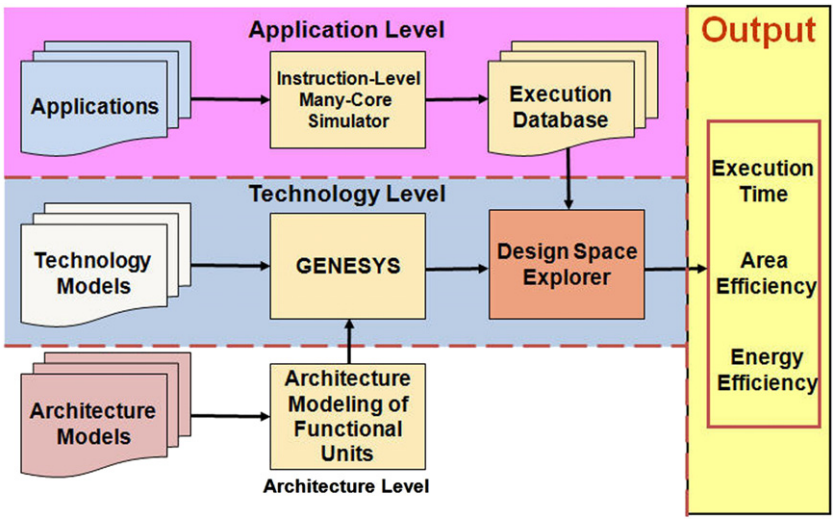

Fig. 9. Simulation methodology infrastructure for the reference many-core processor architecture.

of functional units [3] for many-core arrays was used to calculate the design parameters of each PE configuration. The design parameters were then passed to the technology level. At the technology level, the Generic System Simulator (GENESYS) [15] was used to calculate technology parameters such as latency, area, power and clock frequency for each PE configuration. Finally, a design space analysis tool collected and combined the database information, such as cycle times, instruction latencies, instruction counts, areas and powers of the functional units, obtained from the application, architectural and technology levels in order to determine execution times, area efficiency and energy efficiency for each case. The following describes a summary of simulators used for this study.

4.1.1. Instruction-level many-core simulator

Applications for the reference many-core processors can be programmed using the instruction-level many-core simulator. The simulator allows editing, assembling, executing, and debugging parallel applications. Fig. 10 gives a screenshot of the simulator.

The assembly language is composed by scalar and vector instructions. Scalar instructions execute in the system controller, while vector instructions are broadcast to each PE. All instructions execute in a lockstep fashion to follow traditional single instruction multiple data (SIMD) operation. In addition, constants are used to define architectural parameters such as local memory size in each PE, processor array size, and network topology such as NEWS or Torus-NEWS network. The state of registers, memory, and detector values is displayed in the processor monitor window, as illustrated in Fig. 10. A similar window is available to inspect the state of the system controller. Overall system activity can be monitored through the system view allowing for quick PE selection. The simulator is also instrumented to measure the concurrency level and instruction distribution during the execution of the application. Using this simulator, we can measure important execution parameters such as dynamic instruction counts, operand size resolution, system utilization, and memory usage.

4.1.2. Generic system simulator

GENESYS is a hierarchical tool that explores future ASIC technology and architecture [15] and it is composed of five distinct modules where each module represents a level of the hierarchy (see Fig. 11). The system module consists of three components: architecture, interconnections, and packaging. A set of parameters that describes the many characteristics of the constituent materials, device structures, circuit configurations, system architecture, interconnection networks and packaging of a microelectronics chip is utilized as inputs of the GENESYS models. The GENESYS analytical models are derived from well-known physical principles and these model equations are used to compute key system performance such as power, clock frequency, die size, and throughput.

- *Fundamental*. This module includes a number of constants including Boltzmann's constant, electron charge, the speed of light, and the permittivity of free space.
- *Material*. The material module provides transistor parameters to the device module including the channel and bulk semiconductor materials, the gate electrode material, and the gate insulator material. GENESYS assumes silicon for the channel and bulk material, p^+ or n^+ polysilicon for the gate electrode, and silicon dioxide for the gate insulator. Based on these material assumptions, the temperature-dependent properties such as the semiconductor energy bandgap and low-field mobility, intrinsic carrier concentration, and critical field are determined. Likewise, key characteristics of interconnect metal (Al, Cu) and interlevel dielectric (oxide, polymide) materials, maximum current density and permittivity are specified in GENESYS.
- *Device*. The device module calculates drain current, leakage current, and parasitic capacitances per device width for a uniformly doped bulk silicon metal-oxide-semiconductor field-effect transistor (MOSFET). These parameters are utilized by the circuit module. The key device module input parameters are minimum feature size, supply voltage, gate insulator thickness, junction depth, zero-bias threshold voltage, and device model. Two possible device models can be selected in GENESYS. One is accurate transregional drain current models for deep-submicron MOSFETs that include the effects of high lateral and vertical fields and that provide smooth transitions across subthreshold, saturation, and linear operating regions. These models ensure accurate prediction of leakage and saturation currents as well as propagation delay, particularly for low-power and low-voltage circuits. Another is the alpha-power law model that provides several modifications to estimate process dependent parameters.
- *Circuit*. Many of the key convergent relationships occur at the circuit level where the performance, power dissipation, and area of critical path circuits are considered. The majority of circuit module input parameters are empirically derived gate layout parameters: the pMOS to nMOS width ratio,

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

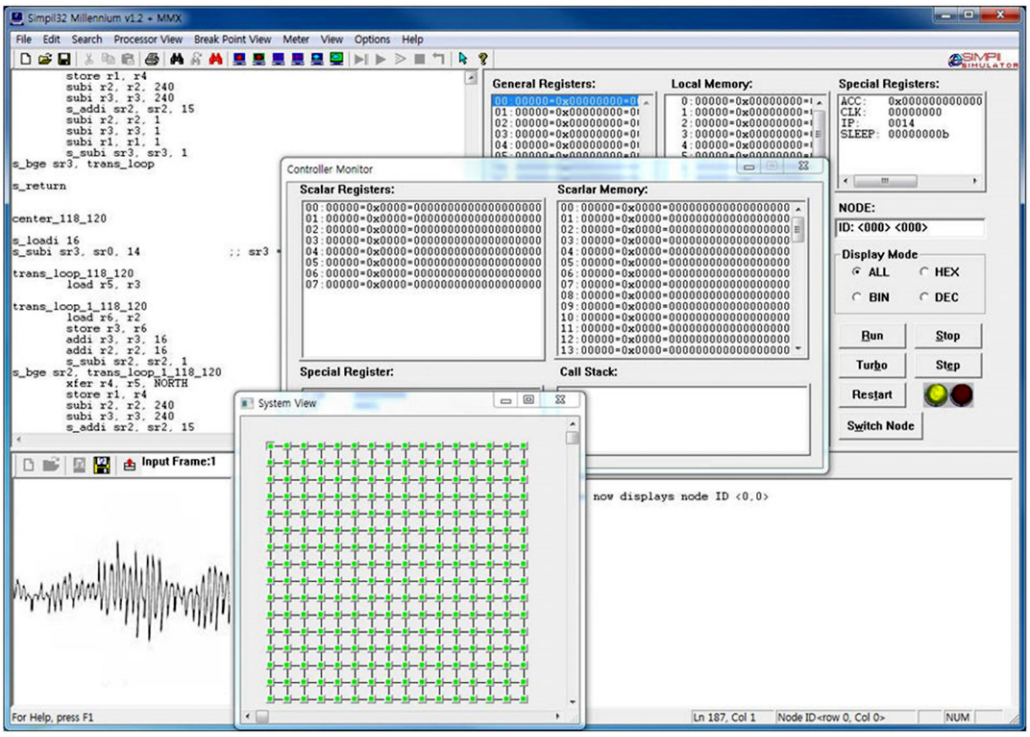

Fig. 10. A screenshot of the instruction-level many-core simulator while executing the physics-based sound synthesis algorithm of plucked-string instruments.

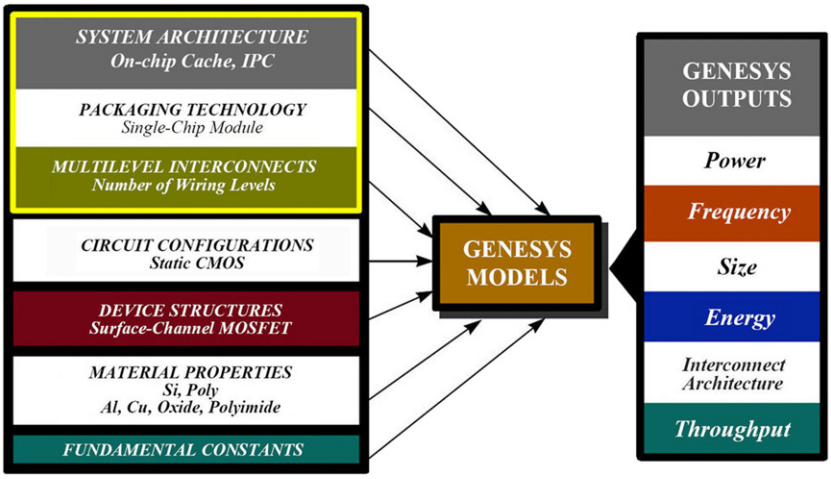

Fig. 11. Organization of GENESYS.

which is nominally set to two for approximately equal rise and fall times, the area of a minimum-sized CMOS inverter in minimum feature squares, and the aspect ratio of this inverter. Furthermore, the driving scheme used to drive long interconnects, either a single driver, an optimized cascaded driver, or optimal repeaters, and the fraction of gates that are in critical paths are also inputs.

- *Interconnect*. To accurately estimate the interconnection requirement of a single-chip ASIC as dictated by its architectural and physical characteristics, a complete stochastic wirelength distribution model is employed in GENESYS, and three different long-distance interconnect driving schemes are supported: optimized single driver, optimal cascaded driver, and optimized repeaters.
- *System architecture and packaging*. To determine on-chip clock frequency, GENESYS utilizes a critical path model characterized by logic depth of the architecture and a region of synchrony. GENESYS also includes contributions to die size from the minimum pad pitch allowed by the bonding technology. Moreover, a first-order architectural throughput model is engaged in GENESYS, which estimates the cycles-per-instruction performance of a processor as dictated by its pipeline depth, superscalar properties, stalls per instruction and instruction latency.

4.1.3. Architecture modeling of functional units

An architectural modeling simulator of functional units [3] was used to calculate the design parameters of the reference many-core architecture such as gate count, gate depth, Rent's parameters, and average activity factor. Fig. 12 shows the setup for simulations of the many-core architecture models. ARMOR stands for "Architecture Modeling and Analysis for Many-core System-on-Chip", and is a program module that incorporates the architecture models [2]. Information from the architecture model, such as

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

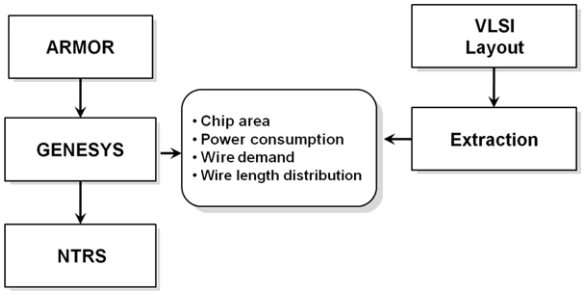

Fig. 12. Simulation setup for many-core architecture models.

interconnect and system parameters, are given to the GENESYS simulator to project performance. Technology parameters from the National Technology Roadmap for Semiconductors (NTRS) [7] are used. For verification, very-large-scale integration (VLSI) layouts are extracted to compare against projected performance data from ARMOR and GENESYS. Published performance data can also be used for verification if proper technology parameters are available for GENESYS.

ARMOR is a program module written in C++ as an enhancement to the existing GENESYS tool. It builds upon the GNENSYS tool, and serves as an alternative system module with many-core modeling features. To compile and use ARMOR, the complete GENESYS modules are required to form an executable program. These GENESYS modules form a hierarchy of physical and practical limits, classified as fundamental, material, device, circuit, and system. The purpose for ARMOR is to define and verify new architecture models that build on GENESYS, investigate design tradeoffs with architecture and wiring models, and promote GENESYS as a tool for system design.

The existing system module incorporates an empirical model of uni-processors to predict future performance. The module assumes single random logic block for the processor core. Constant Rent's parameters (K and P) are assumed for the logic block. Changes in the execution core do not vary wire demand or Rent's parameters. Performance is projected based on an empirical power-law model [15]. Key features of ARMOR are listed as follows:

- Utilize many-core models for multiprocessor systems with an SoC view.
- Provide accurate Rent's parameters for each functional unit for use with the wire models.
- Fully utilize wire models to interconnect functional units and processors in an array.
- Increase the ability to view key performance of individual units.
- Categorize wire demands into multiple length tiers: local, intermediate, and global as defined by connectivity of units for detailed views on wire demands.

Fig. 13 illustrates the design of the ARMOR system module. Each of the blocks represents a program function within ARMOR. These functions are classified into three categories according to the wire demand. Functional unit models exist at the local wire demand level because their wire demands are localized within the unit. At the intermediate wire demand level, two separate models are created to characterize different system architectures. The *Single Node* module characterizes the connectivity of a non-pipelined processing element in a many-core array. At the global wire demand level, the *Tiled Array* module characterizes the interconnection network for a many-core array. All specifications are collected in the *System Model*. Along with architectural parameters provided by the user, system and interconnection parameters are provided to the GENESYS modules to project performance.

Table 3
Summary of evaluation metrics.

Execution time	$t_{\text{exec}} = \frac{C}{f_{\text{clk}}}$
Area efficiency	$\eta_A = \frac{1}{t_{\text{exec}} \times \text{Area}} \left[\frac{1}{\text{s mm}^2}\right]$
Energy efficiency	$\eta_E = \frac{1}{t_{\text{exec}} \times \text{Energy}} \left[\frac{1}{\text{s J}}\right]$

C is the cycle count, f_{clk} is the clock frequency, t_{exec} is the cycle time, *Area* is the PE array area (mm^2), and *Energy* is the system energy required to complete the sound synthesis algorithm in 130-nm CMOS technology.

4.2. Evaluation metrics

This section presents evaluation metrics to determine optimal points in terms of the selected cost criterion. To determine the performance and efficiency characteristics of the sound synthesis algorithm for each PE configuration, we used cycle-accurate simulation and technology modeling. We developed the sound synthesis algorithms of each musical instrument in their respective assembly languages for the many-core array system, in which all of the algorithms for each PE configuration have the same parameters, data sets and calling sequences. We then combined their execution statistics such as instruction count, execution cycle and PE utilization with the GENESYS predictions to evaluate each benchmark's execution time, area efficiency and energy efficiency, which form the basis of the study comparison and are defined in Table 3.

4.3. Parallel implementation of sound synthesis

This section describes a parallel implementation of the sound synthesis algorithm. Fig. 14 presents a flow diagram that depicts the implementation of the sound synthesis algorithm using the reference many-core array. Each row represents one of the strings of the target instruments. For the acoustic and classical guitars, $6 \times N$ PEs are required to simultaneously generate N polyphonic sound samples, while the *gayageum* requires $12 \times N$ PEs. Synthesizing sound samples of the target instruments requires numerous synthesis parameters, such as aggregate excitations, delay lengths, and filter coefficients for each string of each instrument. The ACU distributes these synthesis parameters to each PE, which then creates a sound sample using the distributed parameters.

Fig. 15 shows a sound synthesis example of the sixth string of the classical guitar when the SPE ratio was 11,025. As depicted in Fig. 15, we considered two methods for synthesizing sound samples with a fixed number of PEs: (1) successively generating sound samples from the first PE to the last PE without using any conditional executions (e.g., using masking or "sleep" instructions) as shown in Fig. 15(a), and (2) creating sound samples from the first PE to the last PE using sleep instructions as shown in Fig. 15(b). For the former method, two additional processes were needed because each string had irregular delay lengths. First, it was necessary to update index values to indicate an appropriate memory address in order to obtain sound samples in the local memory of each PE, and second, it was necessary to establish communication between the PEs in order to obtain the previously synthesized samples in the local memory of the other PEs.

To synthesize the musical sounds of the target instruments, we used the commuted waveguide synthesis, which is a loop-back algorithm using a delay line. As shown in Fig. 15(a), PE4 synthesizes the 536th sound sample at the 134th step, after which it is necessary to use a sound sample that is synthesized by PE1 at the first step. Consequently, PE1 sends the sound sample to PE4 by communicating with PE2, PE3 and PE4. In contrast to the former method, the latter method only requires sleep instructions, slightly decreasing system utilization because increasing the number of

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

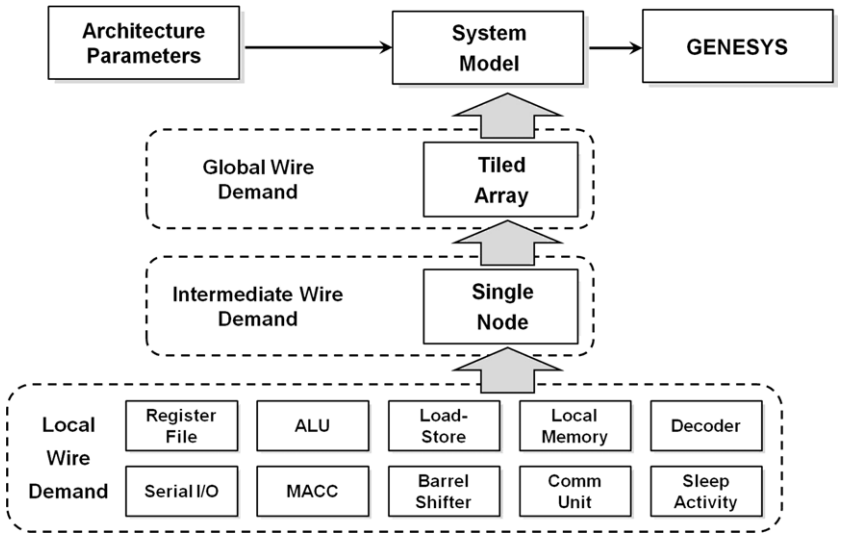

Fig. 13. ARMOR simulation module.

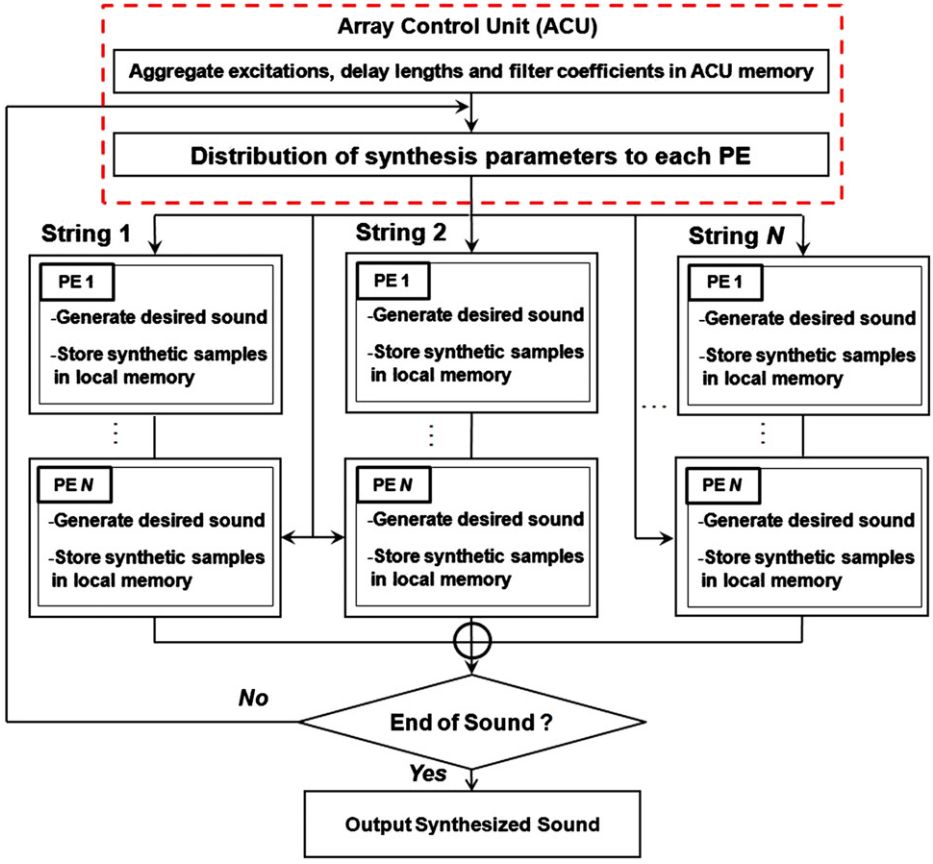

Fig. 14. A flow diagram of parallel implementation for the plucked instrument using commuted waveguide synthesis.

PEs per string results in putting more PEs to sleep. However, it is not necessary to update the index value or to communicate between PEs. For simplicity, we used the latter method in this study.

To illustrate how each PE synthesizes the desired sound in more detail, a pictorial representation of the full synthesis mechanism with the reference many-core array is depicted in Fig. 16. The many-core array synthesizes one-second-long, six-note polyphonic classical guitar sounds with a sampling rate of 44.1 kHz in which 44,100 six-note polyphonic sound samples are synthesized for one second.

5. Experimental results

In this section, seven different PE configurations of the sound synthesis algorithm are examined for each musical instrument using the Design Space Explorer, and the best performing candidate for each evaluation metric is identified. In addition, the performance and efficiency of the best many-core architecture is compared with that of other commercial processor architectures including DSPs, FPGAs, and GPUs to demonstrate the potential of the proposed PE architecture. Moreover, since the synthesized sound quality is as important as the performance and efficiency,

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

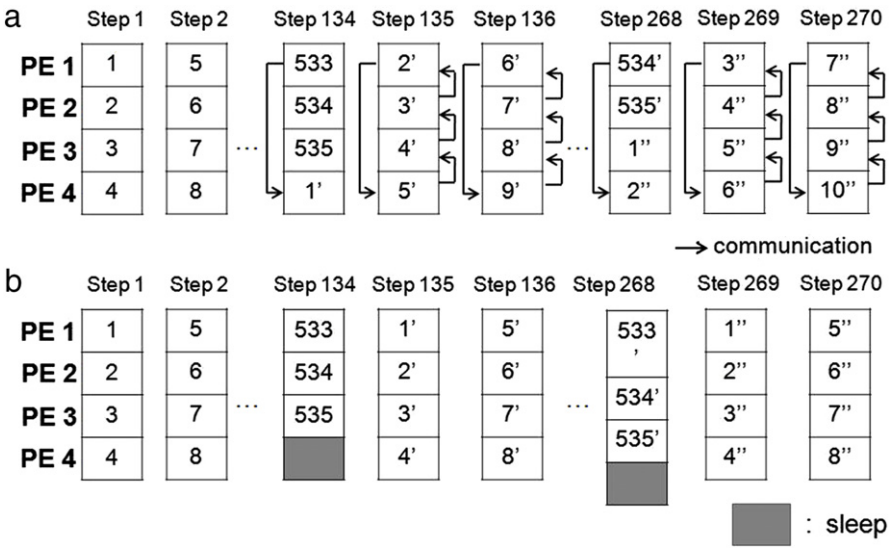

Fig. 15. Process of sound synthesis: sixth string of the classical guitar (delay length is 535). (a) Method 1 for generating sound samples for SPE ratios, and (b) Method 2 for producing sound samples for various SPE ratios.

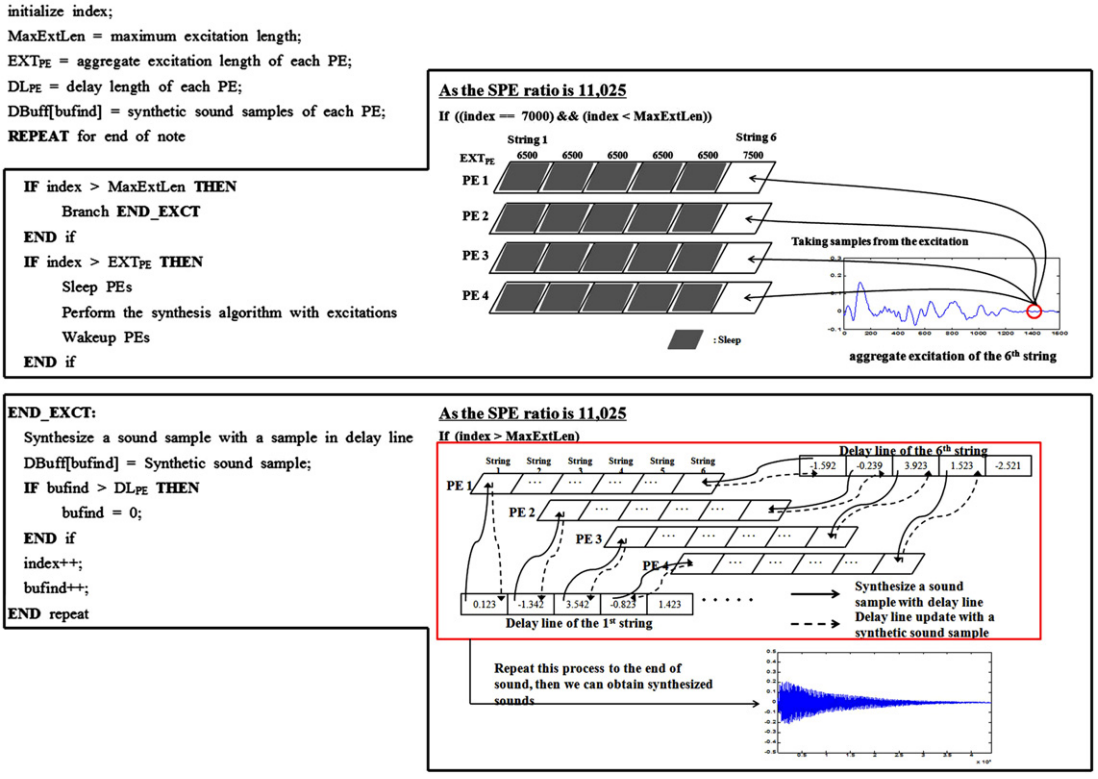

Fig. 16. Full synthesis mechanism with the many-core processor for synthesizing musical sounds of the plucked-string instruments.

the following subsection presents the synthesized sound results of the three plucked-string instruments using the reference many-core architecture.

5.1. Synthetic sounds

For this study, we synthesized sounds of three plucked-string instruments using commuted waveguide synthesis on seven different PE architectures as a function of SPE ratio. Fig. 17 presents the sound spectra, convincingly illustrating that the synthetic sounds produced via the reference many-core architecture are very similar to the original sounds of each target instrument.

5.2. Subjective listening test: MUSHRA

We also evaluate the quality of synthesized musical sounds using the ITU-R BS. 1534 standard [8] known as MUSHRA (Multiple Stimuli with Hidden Reference and Anchors). The assessment consists of rating the quality degradation of test sounds including synthesized sounds and anchors relative to those of known references on a scale from 0 to 100. This rating scale is further broken into five intervals labeled "Excellent (81–100)", "Good (61–80)", "Fair (41–60)", Poor (21–40)" and "Bad (0–20)". In MUSHRA, anchors are used to measure absolute degradation, and they are defined as reference signals low-pass filtered at 3.5 kHz.

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

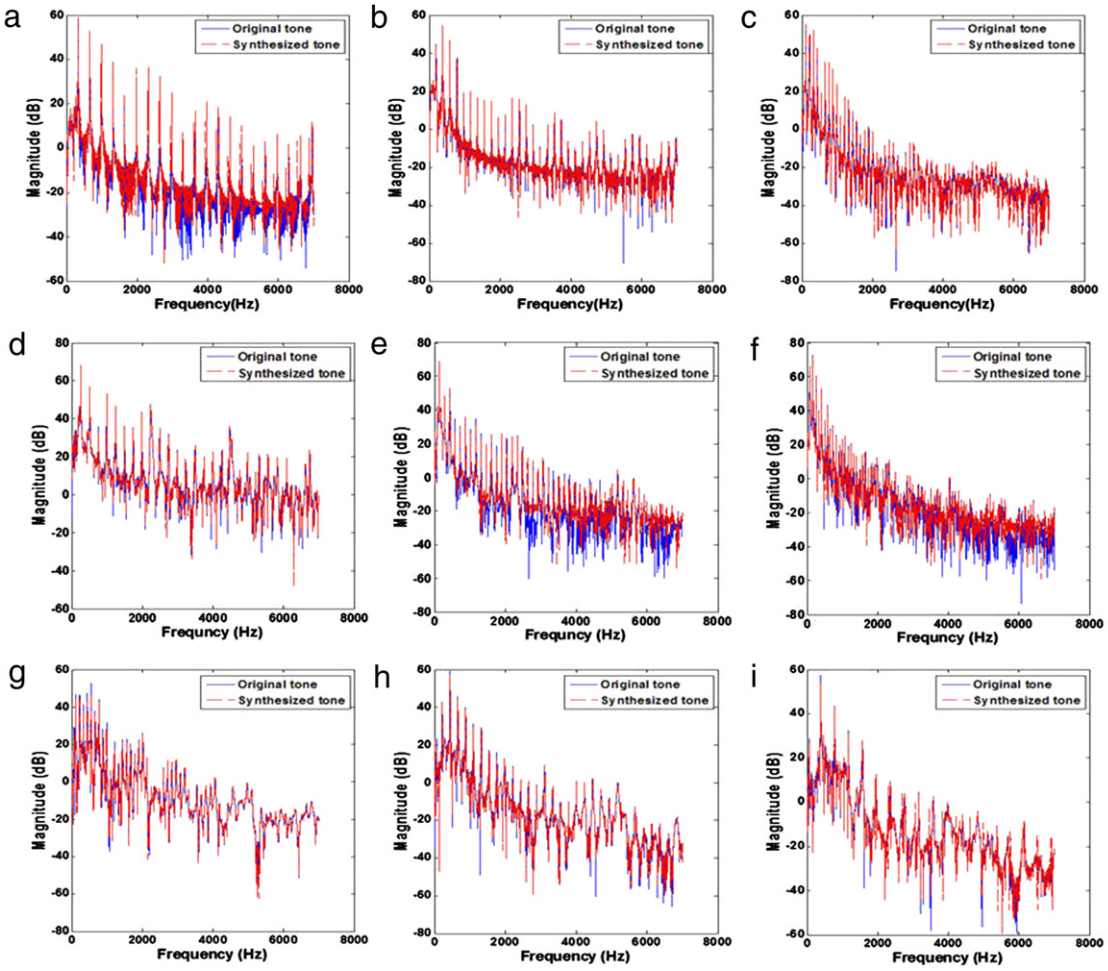

Fig. 17. Spectra of the original sounds (solid line) and synthesized sounds using the reference many-core processor (dotted line): (a) acoustic guitar (E4, 329.63 Hz), (b) acoustic guitar (G3, 196 Hz), (c) acoustic guitar (A2, 110 Hz), (d) classical guitar (B3, 246.94 Hz), (e) classical guitar (D3, 146.83 Hz), (f) classical guitar (E2, 82.41 Hz), (g) *gayageum* (D3, 146.03 Hz), (h) *gayageum* (A3, 219.4 Hz), and (i) *gayageum* (G4, 386.84 Hz).

Consequently, we evaluate the quality of synthesized musical sounds relative to original and low-pass filtered sounds, and the subjective ratings obtained from ten listeners are shown in Fig. 18, which presents synthesized musical sounds are similar to their originals with high ratings of the subjective listening test except some notes such as E4 and B3 for the acoustic guitar, and E4 for the classical guitar.

5.3. Execution time

Fig. 19 shows the execution times for variable SPE configurations. As expected, the execution time decreases as the number of PEs increases (or the SPE ratio values decrease) owing to increased parallelism. However, the slopes, which indicate the speedup efficiency, are not equal. For the SPE ratios below 22,050, the execution time decreased by half when the number of PEs doubled. Yet, the execution time reduction between SPE $=$ 44,100 and SPE $=$ 22,050 was less than those of other SPE configurations because PE activation instructions were considerably involved for an SPE value equal to or less than 22,050.

We also observed that the execution time of the sound synthesis algorithm using variable SPE configurations was correlated with the sampling rate. To achieve CD-quality sound, a sound sample should be sampled at 44.1 kHz, where a sound sample must be synthesized within $1/44\,100 = 0.2$ ms. As shown in Fig. 19, the *gayageum* required the longest execution time of all the configurations. Consequently, it was necessary to determine whether or not the *gayageum* had a sampling rate of 44.1 kHz. It took 15.9 ms and 0.5 ms to synthesize 44,100 12-note polyphonic sound samples when the SPE ratio was 44,100 and 689, respectively. These values indicated that a sound sample for each *gayageum* string could be synthesized within 0.36 μs in which a PE for each string synthesizes a sound sample within $15.9\ \text{ms}/44,100 = 0.36\ \mu\text{s}$ for an SPE ratio of 44,100 and 0.01 μs in which 64 PEs for each string simultaneously synthesizes a sound sample within $0.5\ \text{ms}/44,100 = 0.01\ \mu\text{s}$ for an SPE ratio of 689, respectively. Thus, all of the SPE configurations were sufficiently fast to guarantee CD-quality sound.

5.4. System utilization

To achieve high performance and efficiency in a many-core array system, it is critical to maintain a high system utilization, which scales with the average number of active processing elements. Fig. 20 presents the system utilization of seven different SPE configurations for each target instrument. For an SPE equal to or greater than 22,050, system utilization increased when the SPE value decreased because more PEs were put to sleep in order to map irregular delay lengths into the fixed number of PEs, as illustrated in Fig. 15(b). For an SPE less than 22,050, the system utilization approached a constant that was greater than 80%.

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

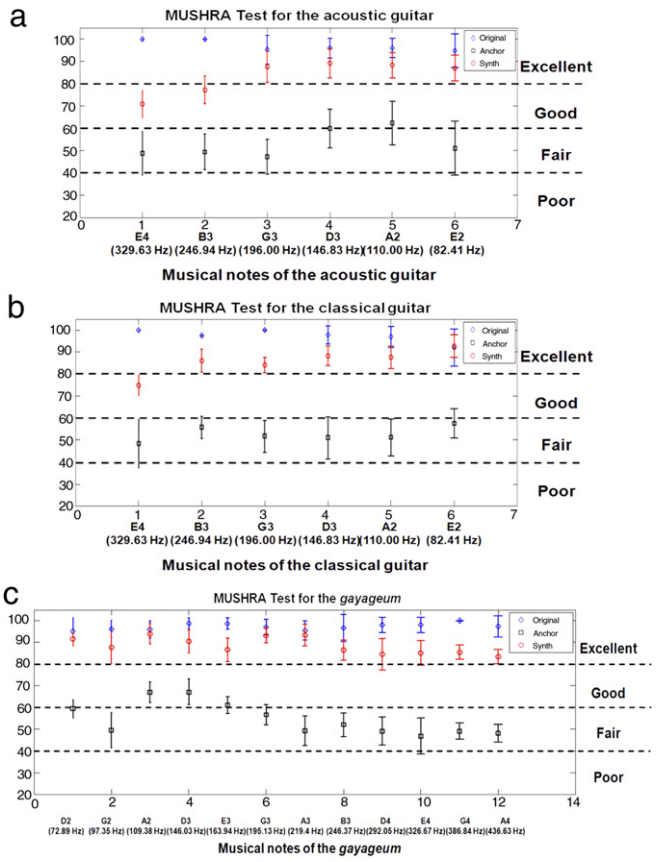

Fig. 18. Subjective ratings of the synthesized sounds. (a) Acoustic guitar, (b) classical guitar, and (c) *gayageum*. Bars indicate 95% confidence intervals.

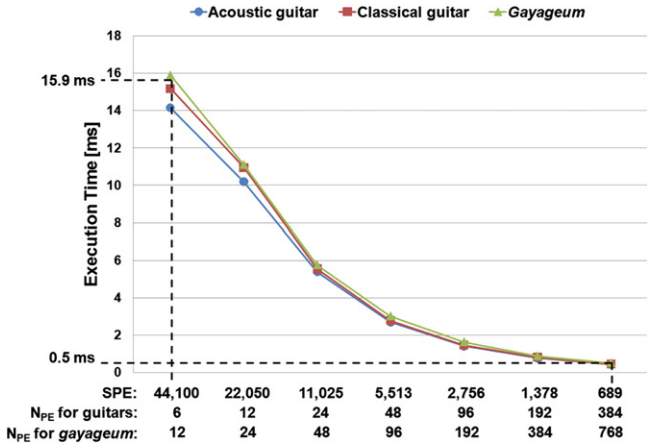

Fig. 19. Execution times for different SPE ratio configurations.

5.5. System area and power evaluation

One key characteristic of the reference many-core array system is its suitability for portable, battery-operated hand-held devices. To implement this system in portable musical devices, the following two limits must be recognized: (1) power consumption is limited by the battery energy, and (2) the specific implementation technology poses a hard bound on the maximum allowable silicon area for monolithic integration. To conquer these constraints, the system area must be limited to 140 mm^2 and the power should not exceed 3 W [7]. Fig. 21 shows the curve of power versus area plane with variable SPE configurations to produce musical sounds of the target instruments. The most apparent result is that all

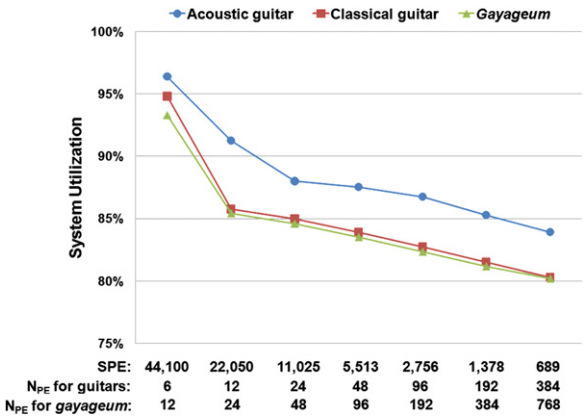

Fig. 20. System utilization with SPE ratio variation.

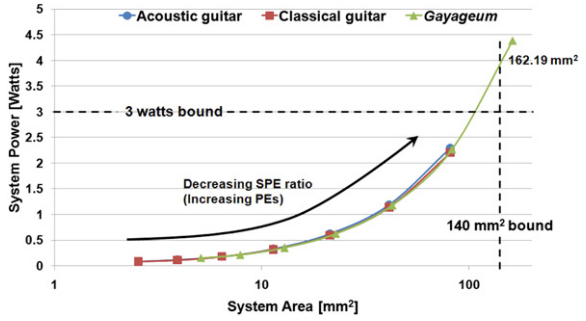

Fig. 21. System power versus system area for different SPE ratio configurations.

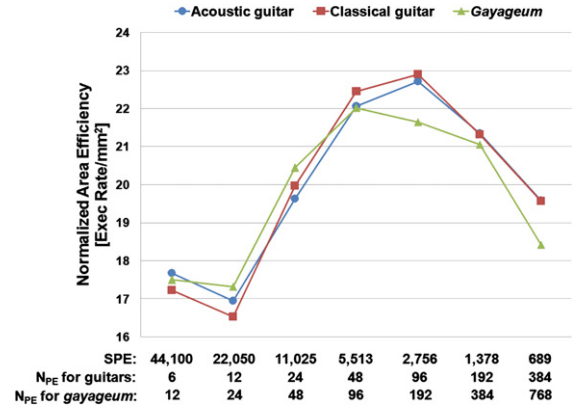

Fig. 22. Area efficiencies for different SPE ratios.

configurations for the musical instruments are feasible. It means that all configurations have a power consumption less than 3 W and a system area less than 140 mm^2 except for configurations of the *gayageum* at SPE = 1378.

5.6. Area efficiency

Area efficiency is the amount of task throughput per unit of area, and increasing area efficiency improves component utilization for given system capabilities. Fig. 22 shows the area efficiencies for each SPE value. The efficiencies for each sound synthesis task of the musical instruments have been normalized relative to the task average to allow comparisons of different musical instruments across the sound synthesis algorithms. Thus, the horizontal offset of the curve is not significant, but the shape of the curve is significant.

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

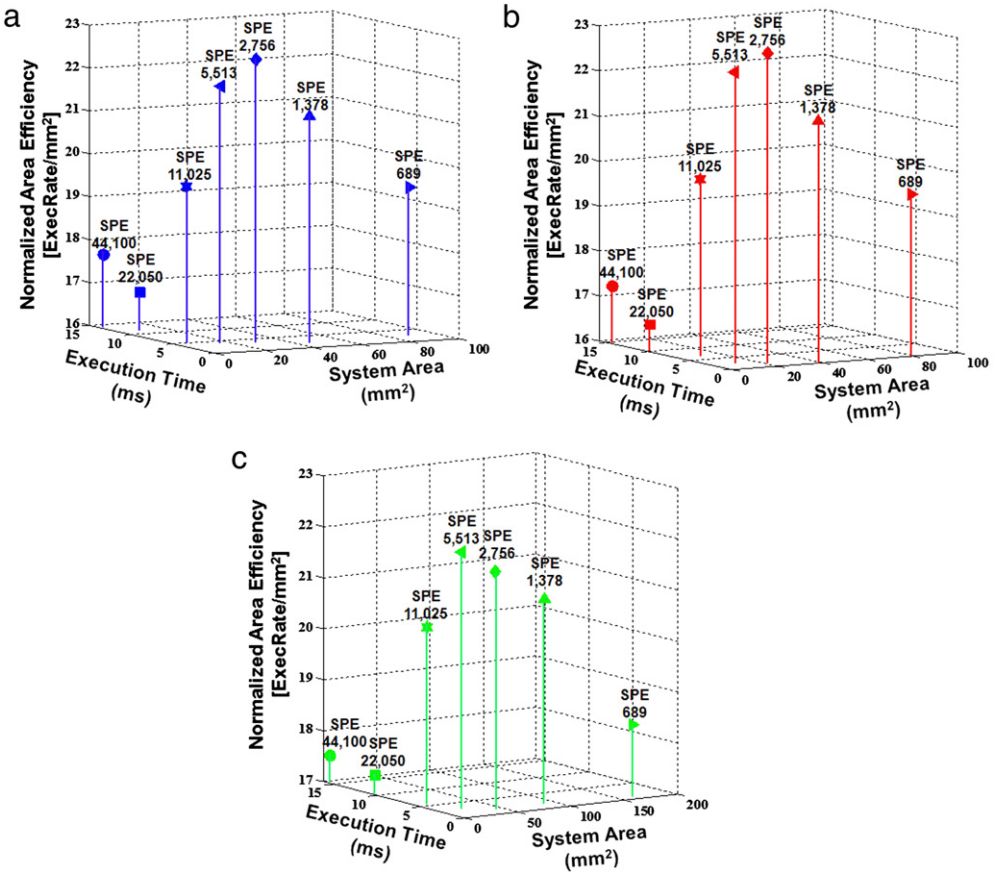

Fig. 23. System areas versus execution time for different SPE ratio configurations. (a) Acoustic guitar, (b) classical guitar and (c) *gayageum*.

The *gayageum* achieved the highest area efficiency at SPE = 5513, and both the acoustic and classical guitars achieved the highest area efficiency at SPE = 2756. These results correlate with the combination of execution time and system area. As shown in Fig. 23, an SPE of 5513 for the *gayageum* and an SPE of 2756 for the guitars achieved smaller execution times (or higher sustained throughputs) with smaller area overhead. In addition, we observed that the minimum area efficiencies of each instrument were achieved at an SPE ratio of 22,050. This is because more PEs are put to sleep in order to map irregular delay lengths into the fixed number of PEs, sharply decreasing system utilization between SPE ratio = 44,100 and SPE ratio = 22,050, as depicted in Fig. 20.

5.7. Energy efficiency

Energy efficiency is the task throughput achieved per Joule, and increasing energy efficiency improves the sustained battery life for given system capabilities. Fig. 24 shows the energy efficiency versus SPE ratio. The maximum area efficiency was achieved at an SPE ratio of 5513 (for the *gayageum*) and an SPE ratio of 2756 (for the acoustic and classical guitars). These results correlate well with the combination of execution time and energy consumption. As shown in Fig. 25, SPE = 5513 for the *gayageum* and SPE = 2756 for the guitars achieved smaller execution times (or higher sustained throughputs) with a small increase in the energy consumption.

Overall, the analysis indicates that an SPE ratio in the range of 5513 and 2756, which is equivalent to PEs between 48 and 96 for both the acoustic and classical guitars and between 96 and 192 for the *gayageum*, provides the most efficient operation for sound synthesis of the plucked-string instruments and reference many-core architecture (see Fig. 26).

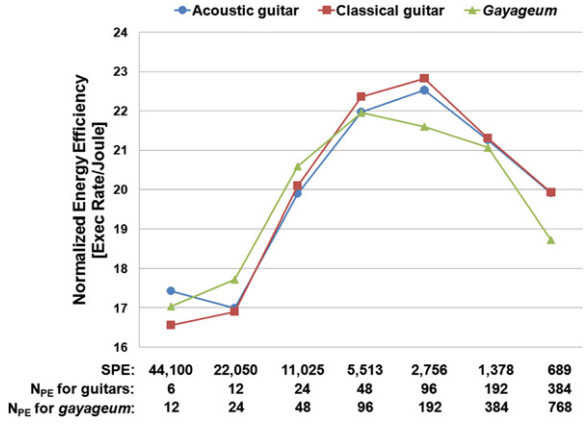

Fig. 24. Energy efficiencies for different SPE ratio configurations.

5.8. Performance comparison between the optimal many-core array and the commercial processor architectures

Table 4 shows a performance comparison between the optimal many-core configuration with an SPE ratio of 2756 and those of TI DSP, NVIDIA Graphics Processing Unit (GPU), and Field-Programmable Gate Array (FPGA) for synthesizing the acoustic guitar sounds. Although a comparison between the many-core array and the commercial processor architectures involves unavoidable errors, the goal of this study was to show the potential for improved performance of the optimal many-core architecture for sound synthesis of plucked-string instruments rather than to conduct a precise performance comparison. Experimental results show that the optimal many-core architecture with SPE = 2756

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

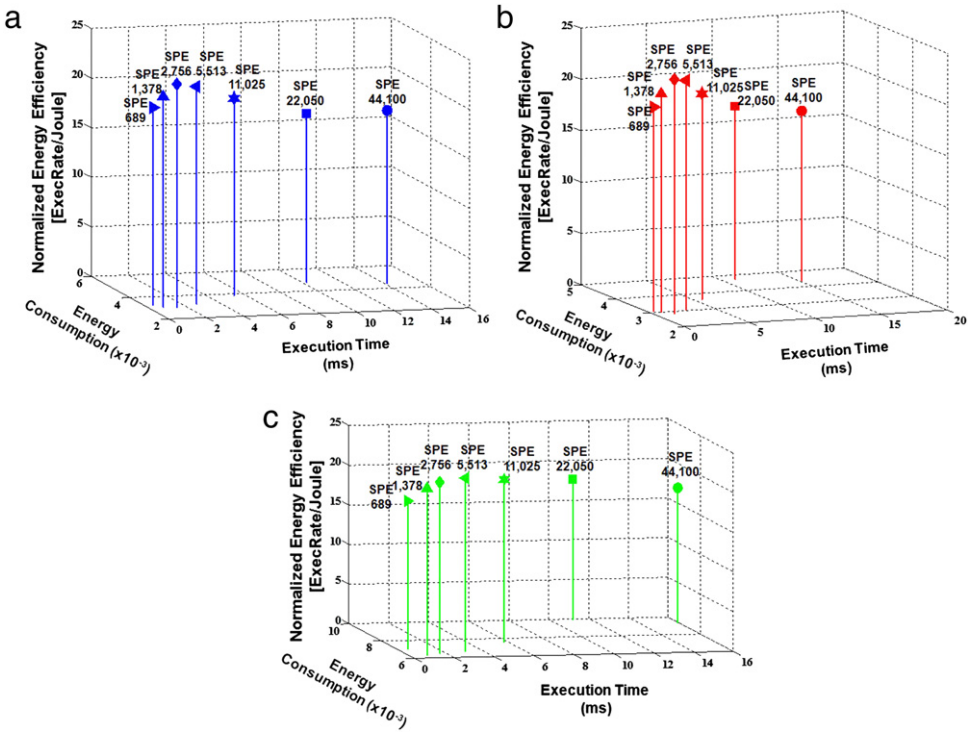

Fig. 25. Energy consumptions versus execution time for different SPE ratio configurations. (a) Acoustic guitar, (b) classical guitar and (c) *gayageum*.

Table 4
Performance comparison between the optimal many-core configuration and commercial processor architectures.

Parameter (unit)	Many-core (SPE:2756)	TI DSP	Six NVIDIA GPUs, each including 512 cores	FPGA
Technology (nm)	130	130	–	–
Clock frequency	100 MHz	720 MHz	1.51 GHz	100 MHz
System area (mm^2)	21.3	529	520	–
Average power (mW)	2101	950	–	1560.97
Average throughput (MIPS)	8449	1210	–	–
Execution time (ms)	1.4531	34.54	0.174	4.85
Energy (mJ)	2.84	32.8	–	7.57
Area efficiency ($1/(s\ mm^2)$)	32.3	0.055	11.052	–
Energy efficiency ($1/(s\ J)$)	242,100	882	–	27,163

outperforms the commercial processor architectures in terms of area and energy efficiencies. For area efficiency, the SPE = 2756 configuration is 587-fold and 2.9-fold better than both TMS320C6416 and six NVIDIA GPUs, each including 512 cores, respectively, for executing the physics-based sound synthesis algorithm. Likewise, the optimal configuration yields 274-fold and 8.9-fold better energy efficiencies compared to those of TMS320C6416 and FPGA, respectively.

6. Conclusions

The demand for real-time sound synthesis algorithms of musical instruments for portable hand-held devices has grown rapidly in recent years, and it has motivated research of parallel processing architectures that support the tremendous amounts of sound data inherent in musical instruments. In other words, hand-held musical systems need to possess the computational power necessary to operate as the most efficient vehicle for portable, battery-operated deployment of sound synthesis. In this paper, the design space of many-core array architectures for the physics-based sound synthesis of three plucked-string instruments including the classical guitar, the acoustic guitar and the *gayageum* was explored by quantitatively evaluating the influence of the SPE ratio on system performance and efficiency

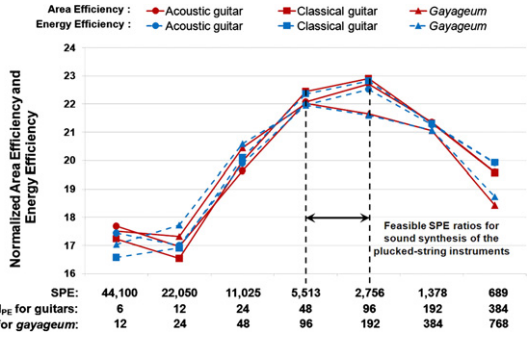

Fig. 26. Feasible SPE ratio configurations for the plucked-string instruments.

using architectural and workload simulations. In addition, the correlation between a fixed problem sample size, SPE ratio, and PE architecture was illustrated for a target implementation in 130-nm CMOS technology.

The analysis presented indicates that an SPE in the range of 5513 and 2756 provides the most efficient operation – i.e., one that maximizes performance per unit cost and per unit energy for synthesizing the musical sounds of three plucked-string instruments sampled at 44.1 kHz with 16-bit quantization. In addition, the selected optimal many-core configurations outperform commercial

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013

high-performance processor architectures including DSPs, FPGAs and GPUs in terms of either area efficiency or energy efficiency.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea (MEST) (No. 2012-0004962).

References

[1] S. Bilbao, Robust physical modeling sound synthesis for nonlinear systems, IEEE Signal Process. Mag. 24 (2) (2007) 32–41.
[2] S.M. Chai, Real time image processing on parallel arrays for gigascale integration, Ph.D. Thesis, Georgia Institute of Technology, 1999.
[3] S.M. Chai, T. Taha, D.S. Wills, J.D. Meindl, Heterogeneous architecture models for interconnect-motivated system design, IEEE Trans. Very Large Scale Integr. (VLSI) Syst 8 (6) (2000) 660–670.
[4] J.-C. Chiu, Y.-L. Chou, H.-Y. Tzeng, A multi-streaming SIMD architecture for multimedia applications, in: Proceedings of the 6th ACM Conference on Computing Frontiers, CF'09, 2009, pp. 51–60.
[5] S. Cho, U. Chong, S. Cho, Synthesis of the Dan Trahn based on a parameter extraction system, J. Audio Eng. Soc. 58 (6) (2010) 498–507.
[6] M. Coodwin, Residual modeling in music analysis-synthesis, in: Acoustics, Speech, and Signal Processing, 1996 IEEE International Conference, ICASSP-96, 1996, pp. 1005–1008.
[7] International technology roadmap for semiconductors 2009 edition. http://www.itrs.net/Links/2009ITRS/Home2009.htm.
[8] ITU, ITU-R BS.1534-1: method for the subjective assessment of intermediate quality levels of coding systems, 2003.
[9] M. Karjalainen, J. Backman, J. Polkki, Analysis, modeling, and real-time sound synthesis of the kantele, a traditional finnish string instrument, in: Acoustics, Speech, and Signal Processing, 1993 IEEE International Conference, ICASSP-93, 1993, pp. 229–232.
[10] M. Karjalainen, U.K. Laine, T.I. Laakso, V. Valimaki, Transmission-line modeling and real-time synthesis of string and wind instruments, in: Proceedings of the 1991 International Computer Music Conference, ICMC'91, 1991, pp. 293–296.
[11] M. Karjalainen, T. Maki-patola, A. Kanerva, A. Huovilainen, P. Janis, Virtual air guitar, J. Audio Eng. Soc. 54 (10) (2006) 964–980.
[12] M. Karjalainen, V. Valimaki, T. Tolonen, Plucked-string models: from the Karplus–Strong algorithm to digital waveguides and beyond, Comput. Music J. 22 (3) (1998) 17–32.
[13] K. Karplus, A. Strong, Digital synthesis of plucked string and drum timbres, Comput. Music J. 7 (2) (1983) 43–55.
[14] T.I. Laakso, V. Valimaki, M. Karjalainen, U.K. Laine, Splitting the unit delay—tools for fractional delay filter design, IEEE Signal Process. Mag. 13 (1) (1996) 30–60.
[15] S. Nugent, D.S. Willis, J.D. Meindl, A hierarchical block-based modeling methodology for SoC in GENESYS, in: 15th Annual IEEE International ASIC/SOC Conference, 2002, pp. 239–243.
[16] J. Oh, J. Herrera, J.N. Bryan, L. Dahl, G. Wang, Evolving the mobile phone orchestra, in: Proceedings of the 2010 International Conference on New Interfaces for Musical Expression, NIME 2010, 2010, pp. 82–87.
[17] D. Overholt, The overtone violin, in: Proceedings of the 2005 International Conference on New Interfaces for Musical Expression, NIME 05, 2005, pp. 34–37.
[18] X. Serra, State of the art and future directions in musical sound synthesis, in: Proceedings of the IEEE 9th Workshop on Multimedia Signal Processing, MMSP 2007, 2007, pp. 9–12.
[19] J.O. Smith, Efficient synthesis of stringed musical instrument, in: Proceedings of International Computer Music Conference, ICMC-93, 1993, pp. 64–71.
[20] V. Valimaki, C. Erkut, Commuted waveguide synthesis of the clavichord, Comput. Music J. 27 (1) (2006) 71–82.
[21] V. Valimaki, J. Huopaniemi, M. Karjalainen, Z. Janosy, Physical modeling of plucked string instruments with application to real-time sound synthesis, J. Audio Eng. Soc. 44 (5) (1996) 331–353.
[22] S.T. Verma, T.H.Y. Meng, Extending spectral modeling synthesis with transient modeling synthesis, Comput. Music J. 24 (2) (2006) 47–59.
[23] G. Wang, Designing smule's ocarina: the iPhone's magic flute, in: Proceedings of the 2009 International Conference on New Interfaces for Musical Expression, NIME 09, 2009, pp. 303–309.
[24] D.-H. Woo, H.S. Lee, Extending Amdahl's law for energy-efficient computing in the many-core era, IEEE Comput. 41 (12) (2008) 24–31.

Jiwon Choi received a B.S. degree from the Department of Computer Engineering and Information Technology at the University of Ulsan, Ulsan, Korea in 2009. He is a current candidate for an M.S. degree. His research interests include multimedia-specific processor architecture, parallel processing and embedded systems.

Myeongsu Kang received a B.S. and an M.S. in computer engineering and information technology from the University of Ulsan, Ulsan, Korea, in 2008 and 2010, respectively. His current research interests include sound synthesis of musical instruments and image processing using digital signal processing techniques, and implementing multimedia applications on parallel processors.

Yongmin Kim received a B.S. in computer engineering and information technology from the University of Ulsan, Ulsan, Korea in 2009 and is currently an M.S. candidate. His research interests include multimedia specific processor architecture, parallel processing, and embedded systems.

Cheol-Hong Kim received a B.S., an M.S., and a Ph.D. in computer engineering from Seoul National University, Seoul, Korea, in 1998, 2000, and 2006, respectively. His current research interests include embedded systems, mobile systems, system on chip design, and parallel processing.

Jong-Myon Kim received a B.S. in electrical engineering from Myongji University, Yongin, Korea, in 1995, an M.S. in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2000, and a Ph.D. in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2005. He is currently an assistant professor of Computer Engineering and Information Technology at the University of Ulsan, Ulsan, Korea. His research interests include multimedia processing, digital watermarking, multimedia specific processor architecture, parallel processing, and embedded systems. He is a member of IEEE and the IEEE Computer Society.

Please cite this article in press as: J. Choi, et al., Design space exploration in many-core processors for sound synthesis of plucked string instruments, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.07.013