Title	GPU-based beamformer: Fast realization of plane wave compounding and synthetic aperture imaging
Author(s)	Yiu, BYS; Tsang, IKH; Yu, ACH
Citation	IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 2011, v. 58 n. 8, p. 1698-1705
Issued Date	2011
URL	http://hdl.handle.net/10722/139255
Rights	©2011 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Correspondence

GPU-Based Beamformer: Fast Realization of Plane Wave Compounding and Synthetic Aperture Imaging

Billy Y. S. Yiu, *Member, IEEE*, Ivan K. H. Tsang, and Alfred C. H. Yu, *Member, IEEE*

***Abstract*—Although they show potential to improve ultrasound image quality, plane wave (PW) compounding and synthetic aperture (SA) imaging are computationally demanding and are known to be challenging to implement in real-time. In this work, we have developed a novel beamformer architecture with the real-time parallel processing capacity needed to enable fast realization of PW compounding and SA imaging. The beamformer hardware comprises an array of graphics processing units (GPUs) that are hosted within the same computer workstation. Their parallel computational resources are controlled by a pixel-based software processor that includes the operations of analytic signal conversion, delay-and-sum beamforming, and recursive compounding as required to generate images from the channel-domain data samples acquired using PW compounding and SA imaging principles. When using two GTX-480 GPUs for beamforming and one GTX-470 GPU for recursive compounding, the beamformer can compute compounded 512 × 255 pixel PW and SA images at throughputs of over 4700 fps and 3000 fps, respectively, for imaging depths of 5 cm and 15 cm (32 receive channels, 40 MHz sampling rate). Its processing capacity can be further increased if additional GPUs or more advanced models of GPU are used.**

I. Introduction

Ultrasound imaging is conventionally based upon a pulse-echo sensing mechanism that sequentially acquires image data over a group of beamlines [1]. This imaging paradigm typically allows a single transmit axial focus to be defined. If better focusing quality is desired, then the pulse-echo sensing is repeated a few times over each beam-line with different nominal transmit focusing locations. However, in doing so, the overall data acquisition time is lengthened concomitantly, meaning that the imaging frame rate is reduced. To improve the image quality without affecting the frame rate, it is necessary to make use of non-beamline-based imaging paradigms that can form focused broad-view images without requiring additional pulse-echo firings. One approach is to use plane-wave (PW) compounding principles that insonate broad-field pulses with different steering angles [2], [3]. Another non-beamline-based imaging method that has been proposed is the synthetic aperture (SA) imaging technique, which transmits unfocused point-source firings from different lateral positions [4]. For both methods, each firing's pulse echoes are acquired over all array channels; because the transmit firings are essentially broad-field insonations, the received channel-domain raw data can be used to form a low-resolution image (LRI) by performing delay-and-sum beamforming at each pixel position. High-resolution images (HRI) with whole field-of-view focusing may also be obtained computationally via recursive summation of a series of LRIs [5].

Although PW compounding and SA imaging have shown potential to improve image quality, their real-time realization is inherently a nontrivial implementation task. One technical challenge that concerns many system developers is the massive computational demand of these imaging methods compared with conventional ultrasound image formation [6]. Because both PW compounding and SA imaging beamform every pixel directly from the channel-domain data based on different sets of focusing delays [3], [4], their image formation process is inherently more complicated than the conventional approach, which works on a line-by-line basis using the same set of focusing delays. In existing ultrasound scanners, field programmable gate arrays (FPGAs) and digital signal processors (DSPs) are usually used to handle computations related to real-time image formation [7]. Nevertheless, they are merely intended to work with the conventional beamline-based imaging paradigm, so their computational capacity is not sufficient to facilitate all of the computation processes required for advanced ultrasound imaging methods. Thus, it is necessary to develop another real-time computing platform to address this computational bottleneck.

Recently, the emergence of graphics processing units (GPUs) has spurred the pace of development in high-performance computing because they can be readily converted into parallel processors through the use of application programming interfaces provided by the vendors. Leveraging this powerful computational hardware, we present in this paper a GPU-based beamformer architecture that can carry out the image formation steps of PW compounding and SA imaging at real-time processing frame rates. This is in line with our intent to develop a programmable software processor module that can form ultrasound images from channel-domain data samples that are acquired in their RF form using an array transducer.

II. Beamforming Principles for PW Compounding and SA Imaging

A. Overall Description

To facilitate discussion of our beamformer architecture, we first begin by reviewing the mathematical principles

Manuscript received May 9, 2011; accepted May 17, 2011. This work was supported in part by the Hong Kong Innovation and Technology Fund (ITS/492/09, InP/210/10, InP/211/10). All three authors have contributed equally to the preparation of this article.

The authors are with the Medical Engineering Program, The University of Hong Kong, Pokfulam, Hong Kong SAR (e-mail: alfred.yu@hku.hk).

Digital Object Identifier 10.1109/TUFFC.2011.1999

of PW compounding and SA imaging. The overall goal of these two imaging paradigms is to derive HRIs that are compounded recursively from a set of LRIs. For the ith HRI and a compound frame size of M, the image value for pixel P_o can be denoted mathematically as:

$$H_i(P_o) = \sum_{m=i-M+1}^{i} L_m(P_o), \tag{1a}$$

where $L_m(P_o)$ represents the corresponding pixel value in the mth LRI. By inspection, the recursive form of (1a) is given by

$$H_i(P_o) = H_{i-1}(P_o) + L_i(P_o) - L_{i-M}(P_o), \tag{1b}$$

where $H_{i-1}(P_o)$, $L_i(P_o)$, and $L_{i-M}(P_o)$ are, respectively, the pixel value for the previous HRI, the latest LRI, and the earliest LRI in the compounding group. In general, $L_m(P_o)$ can be computed from the channel-domain RF data received for a particular transmit firing (steered planar pulses or point-source excitations, depending on whether PW compounding or SA imaging is performed). This computation process involves multiple stages which are described in the next two subsections.

B. Analytic Signal Conversion

As a precursor step in LRI formation, the analytic signal is first computed for each channel-domain RF data vector (with K depth samples). This quantity requires computation of the RF signal's Hilbert transform, which can, in practice, be found via a finite-impulse response (FIR) filtering operation whose impulse response is set equal to the following definition of the Hilbert transform:

$$h(l) = \begin{cases} 0 & \text{for even } l \\ 2/(\pi l) & \text{for odd } l. \end{cases} \tag{2}$$

As such, for the kth depth sample in the nth receive channel for the mth transmit firing, each Hilbert-transformed data sample is essentially given by the following convolution output:

$$\rho_{n,m}(k) = \sum_{l=k}^{k+D-1} h(l) r_{n,m}(l), \tag{3}$$

where $r_{n,m}(l)$ denotes the corresponding channel-domain RF sample and D is the number of taps in the FIR filter kernel.

C. Delay-and-Sum Procedure

Given the analytic channel-domain signals, delay-and-sum beamforming is performed to compute the LRI of the mth transmit firing. This procedure can essentially be considered as a weighted summation of interpolated channel-domain samples $\alpha_{n,m}(P_o)$ for all N array channels. For the pixel P_o, its value can be expressed as follows for an apodization weight w_n in the nth channel:

$$L_m(P_o) = \sum_{n=1}^{N} w_n \cdot \alpha_{n,m}(P_o). \tag{4}$$

In the simplest case, $\alpha_{n,m}(P_o)$ can be found via a linear interpolation of the two adjacent analytic data samples $a_{n,m}(\kappa)$ and $a_{n,m}(\kappa+1)$ that correspond most closely with the pixel's focusing delay in the nth channel. This is mathematically given by

$$\alpha_{n,m}(P_o) = \lambda \cdot a_{n,m}(\kappa) + [1-\lambda] \cdot a_{n,m}(\kappa+1), \tag{5a}$$

where κ is the proximal depth sample number corresponding to the focusing delay $\tau_{n,m}(P_o)$, and λ is the interpolation weight (between 0 and 1). For a given RF sampling rate f_s, these two quantities can be found as

$$\kappa = \lfloor f_s \cdot \tau_{n,m}(P_o) \rfloor, \tag{5b}$$

$$\lambda = 1 + \kappa - f_s \cdot \tau_{n,m}(P_o), \tag{5c}$$

where $\lfloor \cdot \rfloor$ represents a floor operator that gives the largest integer not greater than the operand.

In PW compounding and SA imaging, the beamforming delay to be used for each channel corresponds to the two-way time-of-flight from the transmit source to the pixel position and back to receive element position [3]–[5]. For a pixel with lateral-depth position coordinates $\{x_o, z_o\}$, this is equal to the following for the nth channel at the mth firing:

$$\tau_{n,m}(P_o) = \frac{d_T(P_o;m) + d_R(P_o;n)}{c_o}, \tag{6a}$$

where $d_T(P_o;\ m)$ and $d_R(P_o;\ n)$ respectively denote the transmit propagation distance for the mth firing and the receive propagation distance for the nth channel with respect to the pixel of interest. These two distance quantities can be found from geometrical principles as

$$d_T(P_o;m) = \sqrt{|x_o - x_T(m)|^2 + |z_o - z_T(m)|^2}, \tag{6b}$$

$$d_R(P_o;n) = \sqrt{|x_o - x_R(n)|^2 + |z_o - z_R(n)|^2}, \tag{6c}$$

for a given transmit-center position $\{x_T(m),\ z_T(m)\}$ and a receiver position $\{x_R(n),\ z_R(n)\}$.

III. GPU Beamformer Architecture

A. Hardware Setup

Fig. 1 shows a high-level illustration of the hardware organization for our GPU-based beamformer that is intended to facilitate real-time realization of PW compounding and SA imaging. As can be seen, the GPUs are

housed inside a PC workstation. They are connected as expansion boards through PCI-Express buses with real-time data-transfer bandwidth (maximum of 8 GB/s for 16-lane buses). Their parallel processing resources are managed through a software-based application programming interface known as CUDA (compute unified device architecture; NVIDIA, Santa Clara, CA). Note that the computational hardware involved in this beamformer is inherently different from those seen in a few ultrasound research platforms that are based on computer clusters [8], pipelined DSP networks [9], distributed groups of FPGAs [10], and multi-core CPUs [11].

During operation, our GPU-based beamformer takes in channel-domain RF data acquired from an array transducer and calculates HRIs recursively in real-time. In approaching this task, the beamformer's computational resources have been partitioned into two sectors in which one of the GPUs in the group is designated for HRI processing and the rest are responsible for LRI processing (as shown in Fig. 1). The channel-domain data samples are presumed to be first streamed from the scanner's front-end electronics to the PC's RAM. They are transferred into each LRI-processing GPU in frame batches (controlled by a master CPU) to compute a set of consecutive LRIs, which are subsequently fed into the HRI-processing GPU for recursive compounding with other LRIs. The HRIs are then shown on the PC display in real-time and are sent to the storage device for archival.

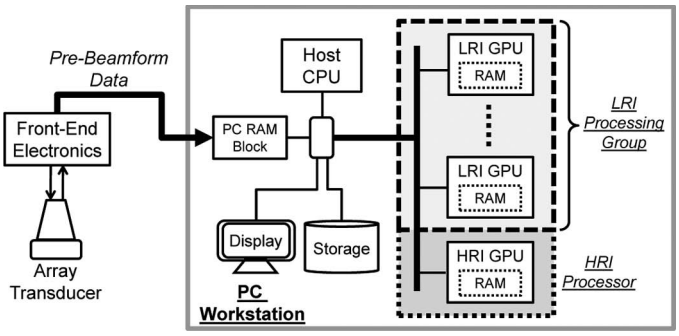

Fig. 1. System-level overview of the hardware setup for the GPU-based beamformer. During operation, each frame of channel-domain data is fed into an idle GPU in the LRI processing group to facilitate beamforming. The HRI-processing GPU is then used to perform recursive compounding of multiple LRIs.

B. Computation of Low-Resolution Images

To facilitate computation of LRIs in our beamformer, a batch-based pipelining approach has been adopted. Each available GPU in the LRI-processing group is assigned to handle one batch of channel-domain data that are acquired from a set of transmit firings. As described in Section II, for the processing of each LRI, the beamformer must perform the operations of analytic signal conversion and delay-and-sum; the GPU processing architecture for this is described as follows.

1) Analytic Signal Conversion: Fig. 2(a) gives a conceptual illustration of how this stage is implemented on the GPU. As can be seen, one block of threads in the GPU is assigned to compute the analytic signal for one channel of pre-beamform RF data. In turn, each thread in the block is instructed to carry out a Hilbert transform operation [see (2) and (3)] and derive the analytic signal samples for a consecutive group of depth samples in the same channel (group size is kept down to a few samples to achieve a high parallel processing efficiency).

2) Delay-and-Sum Operation: Fig. 2(b) illustrates how delay-and-sum beamforming is performed in the GPU-based beamformer to obtain the LRI pixel values. In this processing stage, each block of threads is allocated to compute the LRI pixel values within a two-dimensional grid. For each individual thread, it is instructed to calculate a single LRI pixel value via the following four-step procedure:

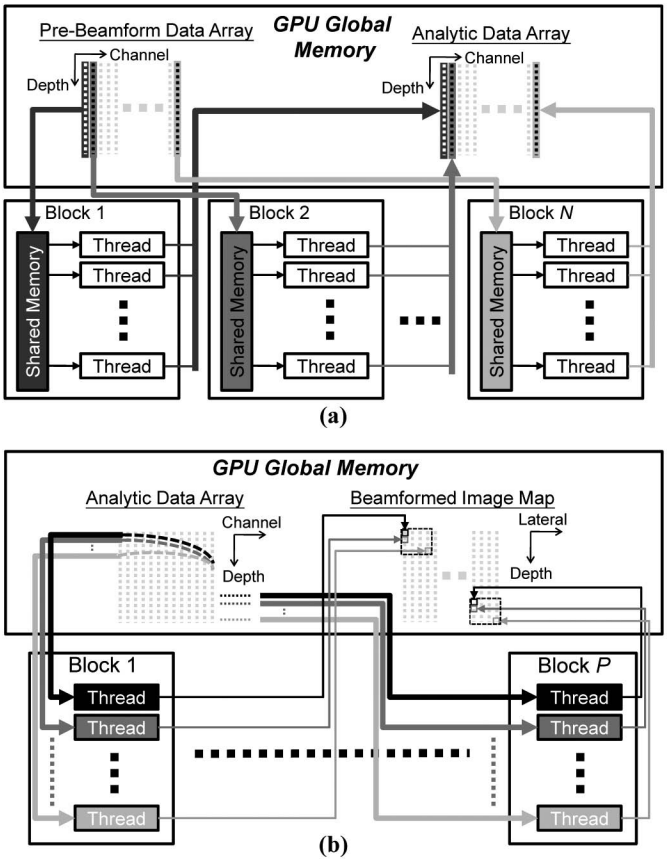

Fig. 2. Multi-thread processing architecture for LRI computation. During analytic signal conversion (a), latency is reduced by copying an entire channel of RF data to the thread block's shared memory. For the delay-and-sum stage (b), the position of sample in each channel to be beam-summed in a thread is denoted as a dashed curve in the analytic data array.

1) Estimate the focusing delays for all channels with respect to the pixel position for that thread [see (6a)–(6c)];
2) Retrieve each channel's analytic data samples whose index position corresponds most closely with the pixel's focusing delay for that channel;

3) Interpolate each channel's signal value to be used for LRI pixel summation based on the retrieved analytic samples [see (5a)–(5c)];
4) Obtain the LRI value by multiplying an apodization weight to each of the interpolated signal values and summing the apodized values [see (4)].

C. Computation of High-Resolution Images

Once a batch of LRIs has been computed, it is transferred to the GPU designated for HRI processing to carry out recursive compounding with other LRIs. As noted in (1a) and (1b), each HRI pixel value is the sum of multiple LRI values at the same pixel position, and it can be calculated recursively from the previous HRI, latest LRI, and earliest LRI in the compounding group. In our beamformer, each GPU thread in the HRI processor has been assigned to handle one pixel of recursive summation. This thread assignment scheme is quite similar to the one used for the delay-and-sum operation, except for the trivial difference of retrieving values from HRI/LRI frames rather than from the analytic data array.

D. Processing Speed Enhancement Strategies

To reduce latency overheads in GPU-based parallel processing, efficient management of memory access is known to be crucial, given that each memory read-write operation may require up to several hundred clock cycles. In general, this task can be facilitated through agile use of GPU's two-tier memory structure, which comprises: shared memory for each thread block (small in size, but with fast access speed); texture and global memory residing in the GPU's device core (slower access speed, which may improve if cached). As a general rule of thumb, it is desirable to exploit the shared memory to store data values that are repeatedly accessed by the beamformer.

In the first stage of LRI computation [Fig. 2(a)], processing latency is lowered by using the shared memory to store an entire channel of RF data and thereby facilitating fast data access by each thread in a block. The Hilbert transform filter coefficients are also stored in the shared memory to accelerate the analytic signal computation process. For the delay-and-sum stage [Fig. 2(b)], latency is kept low by either: 1) creating texture memory pointers to cache analytic data samples that are repeatedly fetched to different threads (for Tesla-class GPUs), or 2) simply exploiting the global memory cache (for Fermi-class GPUs). Note that the analytic data samples are not transferred to the shared memory during delay-and-sum because the size of this fast-access memory in currently available GPUs is not large enough to store the entire data array. Instead, the shared memory is used in this stage to store the apodization weights and the set of pre-calculated channel-domain receive delays for the corresponding scanline (only transmit delays must be computed during operation because the transmit-center position is different for each firing). For the HRI compounding operation, latency overhead is rather nominal because each thread only requires retrieval of a few data samples from memory to calculate an HRI pixel.

IV. Prototype Implementation

A. PC Backbone

Based on our beamformer architecture, we have assembled a prototype PC platform that can support different combinations of GPU devices. This prototype operates on a motherboard with seven PCI-Express 16-lane expansion slots (X58; EVGA Corporation, Brea, CA), and it uses a quad-core, 2.66-GHz CPU as the host controller (i7–920; Intel Corporation, Santa Clara, CA); 6 GB of DDR3 RAM is included in the prototype PC as a data buffer for channel-domain samples.

B. GPU Computational Platform

In this work, the prototype PC has been used to test the efficiency of various multi-GPU combinations in carrying out beamforming for PW compounding and SA imaging. In all of the multi-GPU combinations, HRI processing is performed using a Fermi-class GTX-470 GPU (with 448 cores, 1.215 GHz clock speed, 1280 MB of global memory, and a 768 kB level-two cache). For LRI processing, different dual-GPU configurations have been considered based on the Tesla-class GTX-275 GPU and the Fermi-class GTX-480 GPU (three possible combinations of these have been attempted). Specifications for these GPU models are readily available from the manufacturer, so they will not be repeated here. Nevertheless, a few important differences should be noted between them. First, the GTX-480 has distinctly more processor cores than the GTX-275 (480 versus 240), although their processor clock is similar (1.401 GHz versus 1.404 GHz). Second, the GTX-480 includes a level-two global memory cache (768 kB) that is not found in the GTX-275, but its texture memory filling rate is slower (42 billion/sec versus 50.6 billion/sec). Third, the GTX-480 allocates 48 kB of shared memory for every 32 cores, in contrast to the GTX-275, which assigns 16 kB for every 8 cores. Note that to facilitate comparative analysis, our work has also included the evaluation of single-GPU configurations that involve the use of a GTX-275 GPU or a GTX-480 GPU. For these configurations, the LRI computation and HRI compounding operations were performed on the same GPU.

C. Beamformer Software

The GPU-based beamformer is coded in C++ via a functional programming approach, and various CUDA syntaxes and functions (ver. 3.0) are invoked to realize multi-thread processing on the GPUs. In the current version of the beamformer software, LRI processing is car-

ried out in batches of 50 frames, and this parameter is chosen to maintain balance between processing overheads and data transfer bandwidth (confirmed via low-level performance tests, data not shown). For the analytic signal conversion stage of the beamfomer, a 51-tap FIR filter is implemented for the Hilbert transform, and its coefficients are computed during beamformer initialization. This filter order is empirically chosen to achieve consistent Hilbert transform results. For the delay-and-sum stage, the grid size responsible by each thread block is empirically tuned to be 16 × 16 pixels for GTX-275 GPUs and 64 × 8 pixels for GTX-480 GPUs (because the two GPU models had different memory caching performance). Also, the beamforming delays are computed as the two-way pulse-echo propagation times for a nominal acoustic speed of 1540 m/s, and a Hanning window is used as the apodization weight.

TABLE I. Parameters for the Imaging Example.

General imaging parameters	
Acoustic speed of tissue layer	1540 m/s
Attenuation coefficient	0.5 dB/(cm·MHz)
Array size	128 elements
Array pitch	0.3048 mm
Imaging frequency	10 MHz
Pulse repetition frequency	5 kHz
RF sampling frequency	40 MHz
Number of RF samples per channel	520 for every 1 cm
Transmit waveform	two-cycle sinusoid
Plane wave compounding parameters	
Steering angle range	−12° to +12°
Number of steering angles	49 (in 0.5° increments)
Plane wave transmit aperture	128 channels
Receive aperture	32 channels (spanned over array)
Synthetic aperture imaging parameters	
Virtual source axial position	−2 cm
Point source lateral spacing	0.6 mm
Number of virtual sources	49
Virtual source transmit aperture	64 channels
Receive aperture	32 channels (centered upon Tx aperture)
Beamline imaging parameters	
Transmit focus	3 cm
Transmit aperture	64 channels
Receive aperture	32 channels
Number of beamlines	127

V. Performance Assessment

A. Overview of Study

To assess our beamformer's efficacy in executing image formation operations in real-time, we have performed a series of imaging experiments using typical data acquisition parameters for PW compounding and SA imaging (see Table I). In these studies, channel-domain RF data were synthesized for a field-of-view that encompasses a group of point targets located at different depths. We have used our beamformer to compute HRIs for various imaging depths (between 5 and 15 cm, in 1-cm increments) that essentially correspond to different RF data sizes in each channel (with a 40 MHz RF sampling rate, there are 520 additional RF samples acquired per channel for every 1-cm depth increase). Each HRI is 512 × 255 pixels (for all imaging depths examined), and it is derived from recursive compounding of 49 LRIs that correspond to the total number of independent firing positions in SA imaging or planar steering directions in PW compounding.

Fig. 3 shows an example of HRIs produced by using the GPU-based beamformer to process channel-domain RF data obtained from the field-of-view. As can be seen, the images for both PW compounding and SA imaging gave a sharper visualization of the point targets than those for beamline-based imaging (generated from another codec that we have written). In particular, the defocusing effects that appeared in the beamline-based images are less apparent in the compounded PW and SA images. Such an observation is consistent with previous reports on the anticipated benefits of PW compounding and SA imaging [3]–[5].

To quantitatively analyze the GPU beamformer's performance, we have measured the processing throughput by counting the mean number of HRIs that can be computed in 1 s. This was averaged over a 10-s observation period (with synchronized availability of channel-domain RF data samples). Such a performance measure provides a practical account of the processing capacity that includes various overhead sources like memory transfer from RAM (for loading channel-domain RF samples) and between GPUs (for transferring batches of LRI data into the HRI-processing GPU).

B. HRI Processing Throughput

1) Overview of Results: As an indicator of whether our GPU-based beamformer is capable of producing HRIs in real-time, Fig. 4 plots the processing frame rate as a function of imaging depth for different LRI-computation array compositions and beamformer sizes. In general, it should be noted that the HRI processing throughput (inclusive of LRI computation, recursive compounding, and latency overheads) decreases at greater depths, as expected, given the increased amount of RF data samples that must be processed over each channel. Nevertheless, the HRI throughput for all the multi-GPU configurations is still greater than 1500 fps even at an imaging depth of 15 cm that is used in cardiac imaging scenarios. This is much faster than the real-time display frame rate requirement (usually 100 fps, at most), and it makes instant replay of the HRI cineloop at higher frame rates easily possible. In general, from a real-time data streaming perspective, the processing throughput should be faster than the data acquisition frame rate (i.e., pulse repetition frequency (PRF) in PW compounding and SA imaging [5]) to avoid dropping raw data frames in the beamformer. This implies that, in our example, the PRF may need to be adjusted

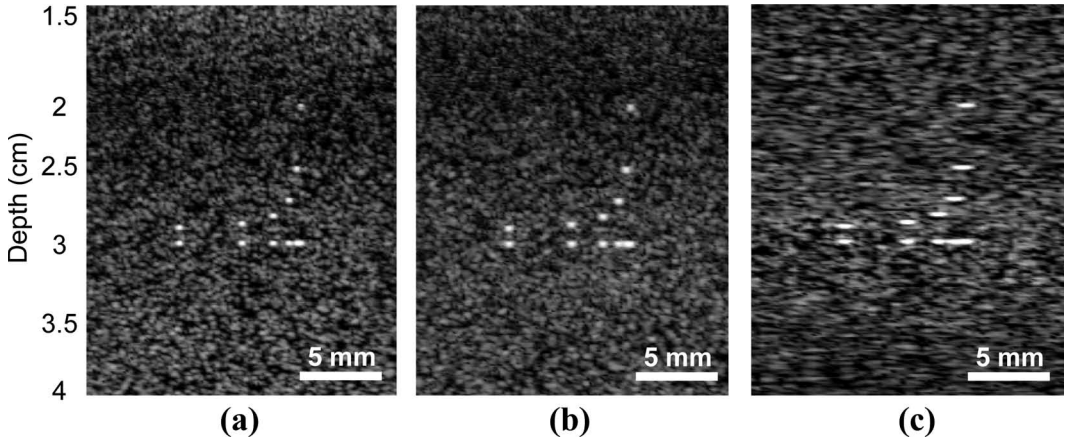

Fig. 3. Point-target image examples for three imaging paradigms: (a) SA imaging (49 virtual point sources); (b) PW compounding (49 steered plane waves); (c) conventional beamline-based imaging (3 cm axial focus). All images were obtained using a 32-channel beamformer on receive. See Table I for data acquisition parameters.

depending on the beamformer's processing capacity and the size of the raw data array.

2) Scaling of Processing Power Using Multiple GPUs and Advanced GPU Models: Among the GPU array compositions [Fig. 4(a)], the one that uses two GTX-480s for LRI computations and one GTX-470 for HRI compounding has achieved the highest HRI processing throughput (see black curve). For 32 receive beamforming channels, it is capable of computing more than 4700 and 3000 HRIs per second, respectively, for imaging depths of 5 cm (used in carotid imaging) and 15 cm (needed for cardiac imaging). This shows that multi-GPU configurations can effectively boost the throughput beyond that achievable with the use of a single GPU (the two light-gray curves). Note that the throughput for the GPU array involving two GTX-275s for LRI processing (dark-gray dashed curve) is similar to that for the one that uses a single GTX-480 for LRI processing (light-gray solid curve). This is an expected result because the total number of parallel computation cores available for LRI processing is the same for the two configurations (both have 480 cores in total). This indicates that our beamformer's HRI processing throughput scales directly with the number of computation cores in the platform. Another point of interest is that for the hybrid configuration (GTX-275 & GTX-480 for LRI computation plus GTX-470 for HRI compounding), the HRI processing throughput (dark-gray solid curve) is between that for two GTX-275s and two GTX-480s, as expected. This shows that the combined use of both Tesla-class and Fermi-class GPUs is possible in our beamformer architecture.

3) Balancing Between Processing Throughput and Beamformer Size: When more receive channels are used in the GPU-based beamformer, a reduction in HRI processing throughput can be noticed [see Fig. 4(b)]. In particular, when using two GTX-480s for LRI computation and a GTX-470 for HRI compounding, the throughput for 128 channels (light gray curve) is, respectively, less than two-thirds and one-third of those for 64 channels (dark gray

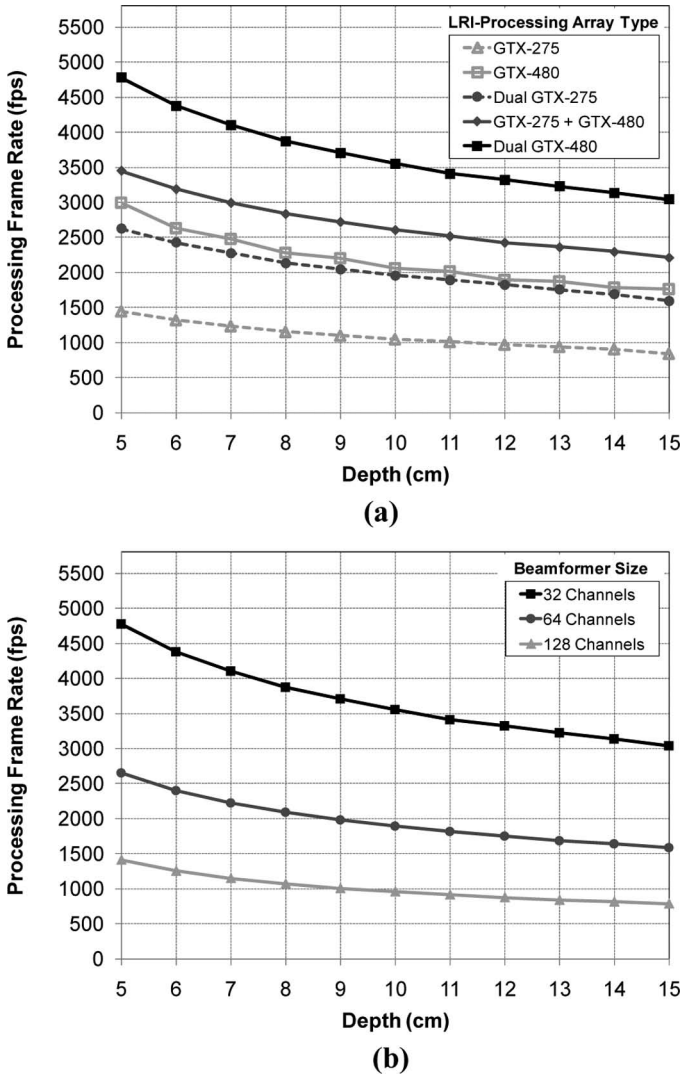

Fig. 4. HRI processing throughput as a function of imaging depth for (a) various LRI-computation array types with a 32-channel beamformer size; (b) various beamformer channel sizes with a dual GTX-480 array for LRI-computation. For each 1 cm increase in imaging depth, the RF data size is raised by 520 samples for each channel (in accordance with the sampling parameters). A GTX-470 GPU is used for HRI processing in all multi-GPU configurations.

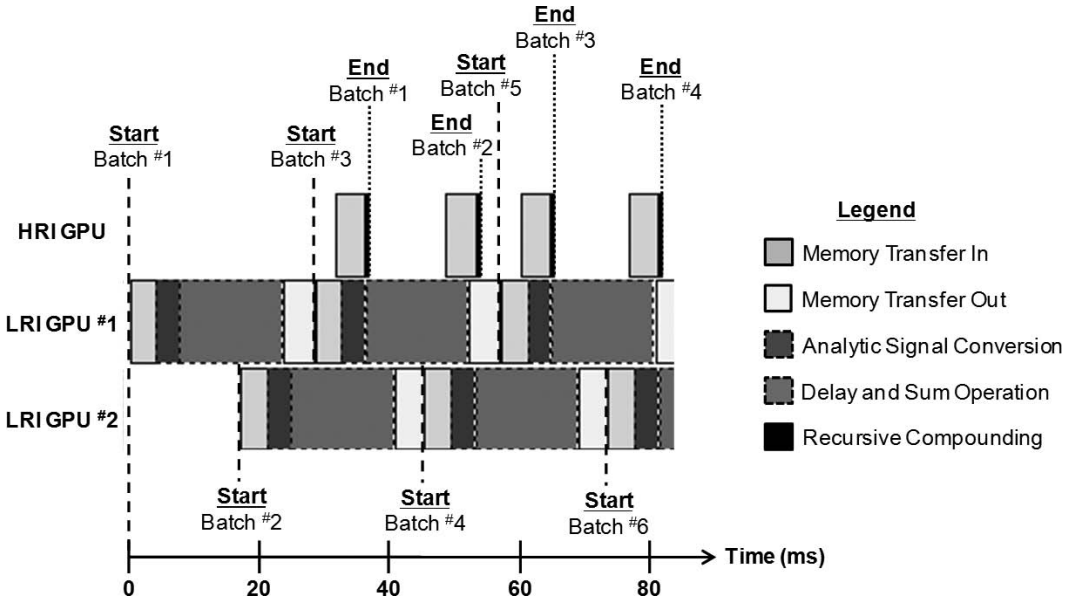

Fig. 5. Individual process timings (obtained from actual measurements) for a GPU array with two GTX-480s for LRI computation and one GTX-470 for HRI compounding. Results are shown over a time window encompassing a few batch-processing cycles, and they correspond to the case with 32 receive channels and 5 cm imaging depth. The processing is carried out in batches of 50 frames.

curve) and 32 channels (black curve). Such a performance gap remains apparent at all imaging depths examined in our example. As such, depending on image quality requirements, it may be beneficial to tune the beamformer channel size to ensure that the gain in image contrast is commensurate with the increase in processing demand.

C. Individual Process Timings

1) LRI Computation Accounted for a Large Portion of the Process Time: As a more detailed evaluation of the GPU-based beamformer's performance, Fig. 5 shows the timing breakdown analysis for a representative time window in a case that uses a GPU array comprising two GTX-480s (for LRI computation) and one GTX-470 (for HRI compounding). This timing diagram is obtained for an imaging depth of 5 cm and a beamformer size of 32 channels. In general, it has shown that LRI computation occupies more computational resources than HRI compounding. Indeed, the two LRI-processing GPUs are mostly engaged during the beamformer's operation, while there is idle time in the HRI-processing GPU. As such, the beamformer's processing throughput seems to be constrained by LRI computations. On the other hand, the spare resources available on the HRI-processing GPU provide opportunities for implementing various post-processing operations related to the analysis and interpretation of HRIs (e.g., image segmentation algorithms).

2) Delay-and-Sum Operation Dominated the LRI Computation Time: Another observation to be noted from Fig. 5 is that, between the two LRI processing stages, the delay-and-sum operation seems to occupy a significant longer timeframe than analytic signal conversion. Nevertheless, the operational time is still well within real-time constraints (Fig. 5 shows that batch processing of 50 LRI frames requires about 25 ms). This computational time can be expected to decrease further as newer GPU models with more parallel computation cores and faster memory access schemes become available.

VI. Concluding Remarks

High computational demand is known to be a technical hurdle for real-time implementation of advanced imaging methods, like PW compounding and SA imaging, that work with the pre-beamform data of each array element. To address this processing demand, we have designed a GPU-based beamformer architecture that offers real-time processing frame rates of more than 1000 fps when using two Fermi-class GPUs as the LRI-processing core and another GPU as the HRI processor. The advantages of using GPUs as a beamforming processor are two-fold. First, their hardware architecture, comprising hundreds of processor cores, has been specifically developed to facilitate single-instruction, multiple-thread computations. This parallelism can help accelerate the beamforming operation of multiple image pixels without running into power wall issues faced by CPU microprocessors. Second, the software-based programmability of GPUs (via the C++ language) is perhaps a more accessible alternative than FPGAs that must be configured using low-level hardware description languages.

To our knowledge, this work is the first demonstration of using GPUs to develop an ultrafast processor for advanced imaging methods that work with pre-beamform data. We expect that this platform can be readily ex-

tended to facilitate real-time realization of other advanced techniques, such as adaptive beamforming. It is our anticipation that, with the advent of GPUs, high-speed back-end processing of various novel imaging methods can be performed on a stand-alone PC workstation. Such a back-end processor design approach is seemingly a less labor-demanding alternative to others that are based on the use of computing devices such as FPGAs and DSPs, whose programming is more complicated than GPUs.

References

[1] T. A. Whittingham, "Medical diagnostic applications and sources," *Prog. Biophys. Mol. Biol.*, vol. 93, no. 1–3, pp. 84–110, 2007.

[2] J. Y. Lu, J. Cheng, and J. Wang, "High frame rate imaging system for limited diffraction array beam imaging with square-wave aperture weightings," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 53, no. 10, pp. 1796–1812, 2006.

[3] G. Montaldo, M. Tanter, J. Bercoff, N. Benech, and M. Fink, "Coherent plane-wave compounding for very high frame rate ultrasonography and transient elastography," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 56, no. 3, pp. 489–506, 2009.

[4] J. A. Jensen, S. I. Nikolov, K. L. Gammelmark, and M. H. Pedersen, "Synthetic aperture ultrasound imaging," *Ultrasonics*, vol. 44, suppl. 1, pp. e5–e15, 2006.

[5] S. I. Nikolov, K. L. Gammelmark, and J. A. Jensen, "Recursive ultrasound imaging," in *Proc. IEEE Ultrasonics Symp.*, 1999, pp. 1621–1625.

[6] S. I. Nikolov, B. G. Tomov, and J. A. Jensen, "Real-time synthetic aperture imaging: opportunities and challenges," in *Proc. Asilomar Conf. Signals Systems and Computers*, 2006, pp. 1548–1552.

[7] G. York and Y. Kim, "Ultrasound processing and computing: Review and future directions," *Annu. Rev. Biomed. Eng.*, vol. 1, pp. 559–588, 1999.

[8] F. Zhang, A. Bilas, A. Dhanantwari, K. N. Plataniotis, R. Abiprojo, and S. Steriopoulos, "Parallelization and performance of 3D ultrasound imaging beamforming algorithms on modern clusters," in *Proc. ACM Int. Conf. Supercomputing*, 2002, pp. 294–304.

[9] C. R. Hazard and G. R. Lockwood, "Theoretical assessment of a synthetic aperture beamformer for real-time 3-D imaging," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 46, no. 4, pp. 972–980, 1999.

[10] J. A. Jensen, M. Hansen, B. G. Tomov, S. I. Nikolov, and H. Holten-Lund, "System architecture of an experimental synthetic aperture real-time ultrasound system," *Proc. IEEE Ultrasonics Symp.*, 2007, pp. 636–640.

[11] R. Diagle, "Ultrasound imaging system with pixel oriented processing," WIPO Patent Application WO/2006/113445.