GPU-accelerated Double-Stage Delay-Multiply-and-Sum Algorithm for Fast Photoacoustic Tomography Using LED Excitation and Linear Arrays

Miri Rostami, Seyyed Reza; Mozaffarzadeh, Moein; Ghaffari-Miab, Mohsen; Hariri, Ali; Jokerst, Jesse

DOI
10.1177/0161734619862488

Publication date
2019

Document Version
Final published version

Published in
Ultrasonic Imaging

Technical Article

Ultrasonic Imaging
2019, Vol. 41(5) 301–316
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0161734619862488
journals.sagepub.com/home/uix
SAGE

GPU-accelerated Double-stage Delay-multiply-and-sum Algorithm for Fast Photoacoustic Tomography Using LED Excitation and Linear Arrays

Seyyed Reza Miri Rostami[1], Moein Mozaffarzadeh[2], Mohsen Ghaffari-Miab[1], Ali Hariri[3], and Jesse Jokerst[3,4,5]

Abstract

Double-stage delay-multiply-and-sum (DS-DMAS) is an algorithm proposed for photoacoustic image reconstruction. The DS-DMAS algorithm offers a higher contrast than conventional delay-and-sum and delay-multiply and-sum but at the expense of higher computational complexity. Here, we utilized a compute unified device architecture (CUDA) graphics processing unit (GPU) parallel computation approach to address the high complexity of the DS-DMAS for photoacoustic image reconstruction generated from a commercial light-emitting diode (LED)–based photoacoustic scanner. In comparison with a single-threaded central processing unit (CPU), the GPU approach increased speeds by nearly 140-fold for 1024 × 1024 pixel image; there was no decrease in accuracy. The proposed implementation makes it possible to reconstruct photoacoustic images with frame rates of 250, 125, and 83.3 when the images are 64 × 64, 128 × 128, and 256 × 256, respectively. Thus, DS-DMAS can be efficiently used in clinical devices when coupled with CUDA GPU parallel computation.

Keywords

graphics processing unit (GPU), central processing unit (CPU), compute unified device architecture (CUDA), photoacoustic imaging, beamforming, parallel computing, double-stage delay-multiply-and-sum (DS-DMAS), linear-array imaging

[1]Computational Electromagnetics Laboratory, Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran
[2]Laboratory of Acoustical Wavefield Imaging, Department of Imaging Physics, Delft University of Technology, Delft, The Netherlands
[3]Department of NanoEngineering, University of California, San Diego, La Jolla, CA, USA
[4]Materials Science and Engineering Program, University of California, San Diego, La Jolla, CA, USA
[5]Department of Radiology, University of California, San Diego, La Jolla, CA, USA

Corresponding Author:
Mohsen Ghaffari-Miab, Computational Electromagnetics Laboratory, Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran Province, Al Ahmad Street, Jalal, No. 7, 14115-111 Tehran, Iran.
Email: mghaffari@modares.ac.ir

Introduction

Photoacoustic imaging (PAI) is a promising biomedical imaging modality that provides functional, structural, and molecular information[1-3] after a short laser pulse irradiates the tissue. The photoacoustic waves are generated based on thermoelastic expansion effects. Finally, wide-band ultrasound transducers detect the propagated photoacoustic waves.[4-6] PAI usually has higher contrast than ultrasound imaging because it is based on differences in optical absorption rather than differences in physical impedance. PAI also usually offers higher resolution than pure optical imaging because acoustic pressure waves are scattered 1000-fold less than optical waves.[7] PAI has multiple applications such as tumor detection,[8,9] cancer staging,[10] ocular imaging,[11,12] molecular imaging,[13,14] functional imaging,[15,16] oncology,[17,18] ophthalmology,[19] and cardiology.[20]

There are two methods of PAI: photoacoustic tomography (PAT)[21] and photoacoustic microscopy.[22] The focus of this paper is on the PAT. Ultrasound transducers in different types (circular, linear, and arc) detect photoacoustic waves. Optical absorption maps of the tissue can then be obtained via mathematical transformation.[23-27] Circular detection of the photoacoustic waves is difficult to translate into clinical applications,[28] and thus linear-array transducers are commonly used.[29,30] However, image quality is lower with linear-array PAT because there are only a few angles (about 40°) available for detection. This leads to low-quality image versus circular tomography.[31-33] To address this problem, enhanced image formation algorithms should be used.[34-39] Delay and sum (DAS) is usually used for image formation in linear-array scenario. However, it leads to a low-quality image because DAS considers all detected signals to be identical regardless of the source. While DAS is popular because of its simplicity, it suffers from poor spatial resolution. Thus, delay-multiply-and-sum (DMAS) was introduced to improve the photoacoustic/ultrasound image quality in linear-array PAT.[36,40]

Previous work has improved image quality with higher computational complexity. The higher complexity of the algorithms degrades the temporal resolution and prevents real-time imaging. Multiple-core graphics processing units (GPUs) along with the central processing unit (CPU) can address this issue. Indeed, improvements in hardware, parallelism directives, and parallel processing power[41-44] have led GPUs to be used extensively in PAI systems.[45-54] However, the advantages of double-stage delay-multiply-and-sum (DS-DMAS) have not yet been combined with the power of GPU processing.

We have recently introduced DS-DMAS, providing a higher contrast compared with DMAS.[37,38] It should be noticed that the higher image quality has been obtained at the expense of a higher computational complexity. On the other hand, nowadays, real-time PAI systems are being used in different applications.[29,33,55-61] Thus, this work improves the temporal resolution and computational time of the DS-DMAS algorithm via a GPU implementation. To the best of our knowledge, this is the first use of GPU acceleration to minimize the processing time of photoacoustic data acquired from a light-emitting diode (LED)–based scanner. While this has been shown several times for laser-based systems, there is an increasingly large body of work utilizing LEDs that have not yet completely harnessed the utility of GPU acceleration.

Materials and Methods

In this section, we briefly review the concept of beamforming and describe GPU implementation in the DS-DMAS algorithm.

Image Formation

DAS is a non-adaptive beamformer and considers all of the calculated samples to be the same. Its formula is as follows:

$$y_{\mathrm{DAS}}(k) = \sum_{i=1}^{M} x_i(k - \Delta_i), \tag{1}$$

where $y_{DAS}(k)$ is the output of the beamformer in which k is the time index, M is the number of the array elements, and $x_i(k)$ and Δ_i are the detected signals and the corresponding time delay for detector i, respectively. DAS has low resolution and contrast, and thus DMAS was introduced to address these issues[40]:

$$y_{\mathrm{DMAS}}(k) = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_i(k - \Delta_i) x_j(k - \Delta_j). \tag{2}$$

The following improvements are suggested in the literature[40] to overcome the dimensionally squared problem[2]:

$$\begin{aligned} &\hat{x}_{ij}(k) = \\ &\operatorname{sign}\left[x_i(k - \Delta_i) x_j(k - \Delta_j)\right] \sqrt{| x_i(k - \Delta_i) x_j(k - \Delta_j) |}, \\ &\text{for} \quad 1 \leqslant i \leqslant j \leqslant M. \end{aligned} \tag{3}$$

$$y_{\mathrm{DMAS}}(k) = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \hat{x}_{ij}(k). \tag{4}$$

DMAS utilizes a correlation process to form a high-quality photoacoustic image.[36,62] However, the performance of the DMAS is degraded at the presence of a high level of imaging noise.[37,38] To address this problem, we have recently introduced the DS-DMAS algorithm:

$$\begin{aligned} y_{\mathrm{DMAS}}(k) &= \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_{id}(k) x_{jd}(k) = \\ &\underbrace{\left[x_{1d}(k)x_{2d}(k) + x_{1d}(k)x_{3d}(k) + \cdots + x_{1d}(k)x_{Md}(k)\right]}_{\text{first term}} \\ &+ \underbrace{\left[x_{2d}(k)x_{3d}(k) + x_{2d}(k)x_{4d}(k) + \cdots + x_{2d}(k)x_{Md}(k)\right]}_{\text{second term}} \\ &+ \cdots \\ &+ \underbrace{\left[x_{(M-2)d}(k)x_{(M-1)d}(k) + x_{(M-2)d}(k)x_{Md}(k)\right]}_{(M-2)\text{th term}} \\ &+ \underbrace{\left[x_{(M-1)d}(k)x_{Md}(k)\right]}_{(M-1)\text{th term}}, \end{aligned} \tag{5}$$

where $x_{id}(k)$ and $x_{jd}(k)$ are the delayed detected signals for element i and j, respectively. The formula of the DS-DMAS can be written as follows:

$$y_{\text{DS-DMAS}}(k)=\sum_{i=1}^{M-2}\sum_{j=i+1}^{M-1}x_{it}(k)x_{jt}(k), \tag{6}$$

where x_{it} and x_{jt} are the ith and jth terms shown in Jokerst et al.[5]

Experimental Setup for PAI

All the experiments in this study were performed using a commercially available LED-based PAI system from PreXion Corporation (Tokyo, Japan) described previously.[63] LED arrays were used as the excitation source and attached to both sides of the ultrasound transducer. The wavelength was 690 nm, the repetition rate was 4 kHz, and the pulse width of the LEDs was 100 ns. A 128-element linear-array transducer with a central frequency of 10 MHz and bandwidth of 80.9% was used to detect the photoacoustic signal. The data acquisition unit has a dynamic range of 16 bits with 1024 samples per element. The sampling rates of the photoacoustic and ultrasound modalities are 40 and 20 MHz, respectively.

To evaluate this implementation, pencil (graphite) lead (0.5 mm HB, Newell Rubbermaid, Inc., Illinois) served as the optical absorber. These were placed at different depths with an interval distance of 0.5 mm. The samples are scanned at different depths from 20 to 24 mm. These were in 2% intralipid (20%, emulsion, Sigma–Aldrich Co., Missouri) mixed with agar as the scattering media. The B-mode frame rate was 6 Hz.

GPU Implementation

The hardware used for the DS-DMAS algorithm includes an Intel core-i7 4790 consisting of eight logical cores and four physical cores. These include NVIDIA GTX 760 with 1152 compute unified device architecture (CUDA) cores and NVIDIA GTX 1070 GPU consisting of 1920 CUDA cores. The characteristics of the hardware are given in Table 1. To have a fair comparison between the processing times, the CPU and GPU are chosen based on a nearly identical price. The CUDA was first described by NVIDIA in 2006 for scientific general-purpose calculations and implementation on GPUs. CUDA makes different cache memories such as pinned memory, texture memory, and shared memory controllable for users due to their low-level programming language. CUDA lets users employ the massive potential of many-core GPUs for parallel programming. Here, the DS-DMAS algorithm is implemented on GPU via a double precision CUDA.

CUDA Implementation

CUDA is a flexible and scalable programming language for parallel computation on GPU. In CUDA, GPU and CPU simultaneously work together with separate memory spaces. The host code is run on the CPU and it manages data transfer for both the GPU and CPU. In addition, it launches kernels (functions or subroutines performed on the GPU). Figure 1 shows that the GPU and CPU have diverse hardware architectures and communicate with each other via a PCI-Express bus.[64] GPUs have several autonomous computational units called streaming multiprocessors (SMs) along with several memories. The main memory is accessible via all threads implemented on the CUDA SMs. It is comparatively large (1 GB) and much faster than the CPU memory. Each SM has an on-chip shared memory that is local to an SM and quite small (16 KB per SM). The shared memory is very fast compared with the main (global) GPU memory. In addition, GPU has some additional memories, including constant and texture memory. Figure 1

Table 1. Hardware Details.

CPU	Intel Core i7-4790 CPU
Number of cores	4
Number of threads	8
GPU	NVIDIA GTX 760
CUDA cores	1152
Base core clock	980 MHz
Memory speed	6.0 Gbps
GPU memory	2.048 GB
Memory interface	GDDR5
Memory bandwidth	192.2 GB/s
Compute capability	3.0
Number of SMs	6
GPU	NVIDIA GTX 1070
CUDA cores	1920
Base core clock	1506 MHz
Memory speed	8.0 Gbps
GPU memory	8.192 GB
Memory interface	GDDR5
Memory bandwidth	256.3 GB/s
Compute capability	3.0
Number of SMs	15

CPU = central processing unit; GPU = graphics processing unit; CUDA = compute unified device architecture; SMs = streaming multiprocessors.

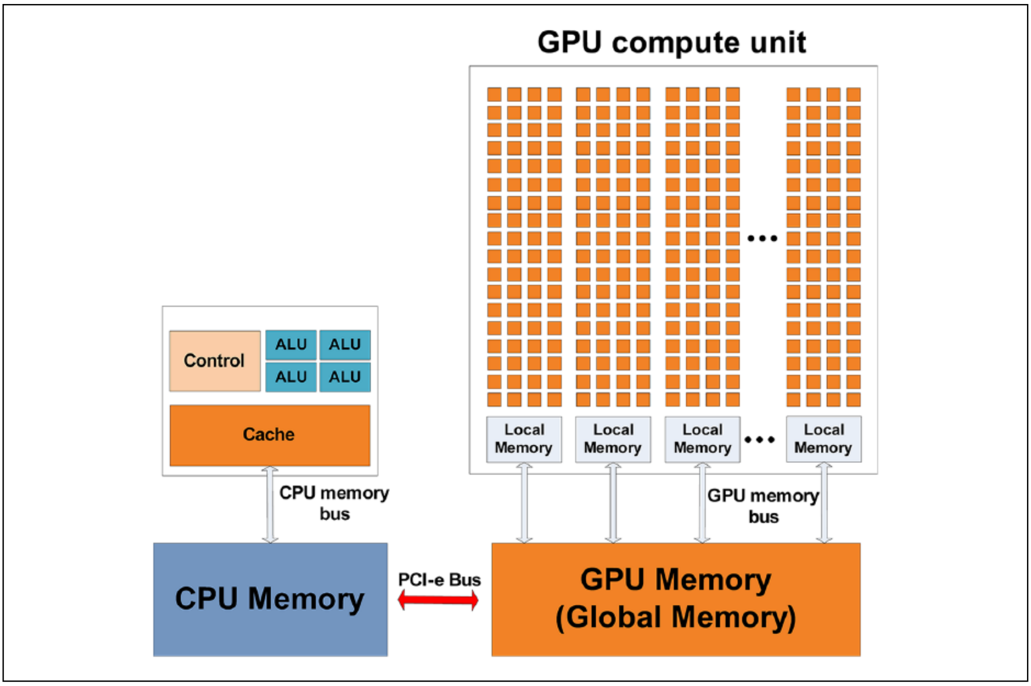

Figure 1. GPU and CPU hardware architectures.[64] GPU = graphics processing unit; CPU = central processing unit.

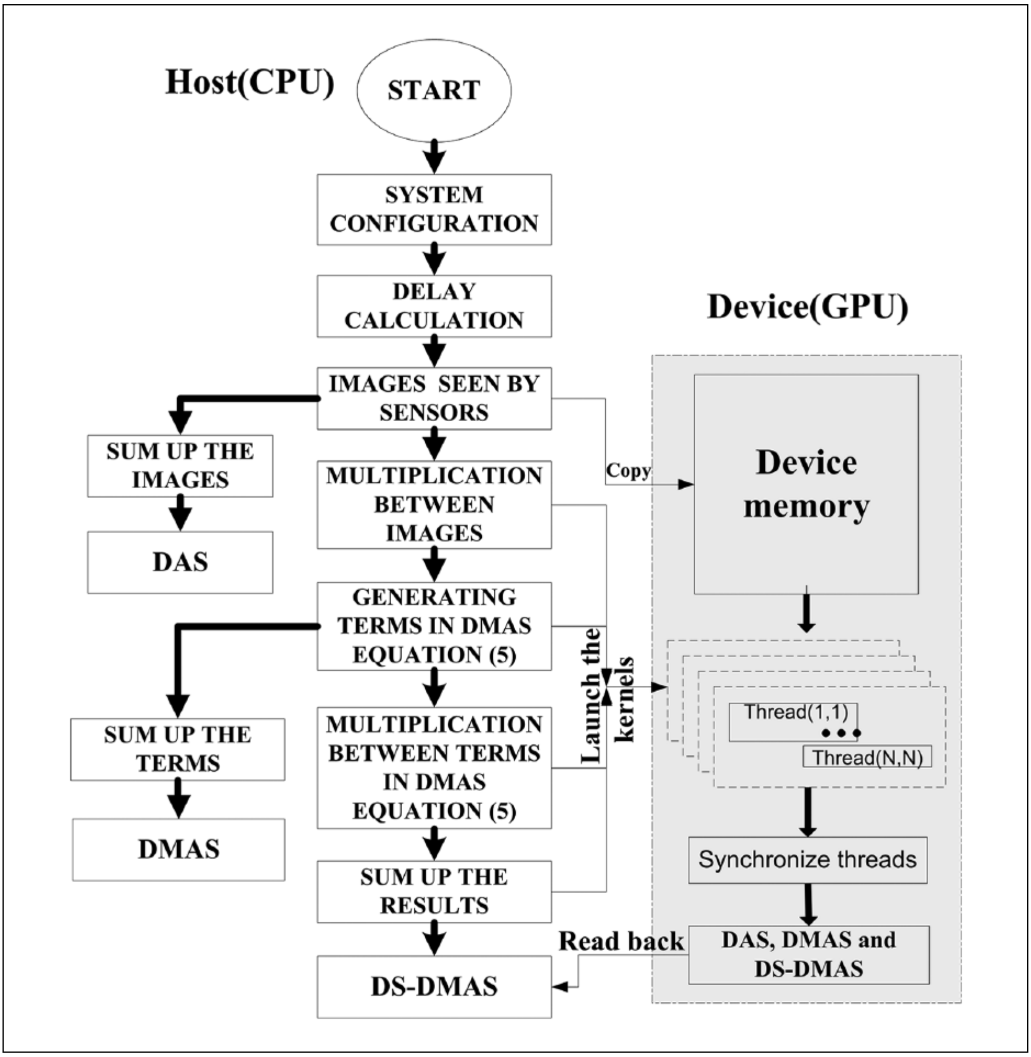

Figure 2. Flowchart of the proposed algorithm (DS-DMAS). CPU = central processing unit; GPU = graphics processing unit; DAS = delay and sum; DMAS = delay-multiply-and-sum; DS-DMAS = double-stage delay-multiply-and-sum.

presents separate memories of CPU and GPU. The user manages the data transfer in both memories to achieve valid results. As the data transferring between GPU and CPU is time-consuming, inessential data transfer should be avoided as much as possible.

Figure 2 shows that the CUDA implementation of the DS-DMAS algorithm contains the following four levels:

Allocating memory on CPU and GPU to configure the desired system,
Transferring required data from CPU to GPU,
Kernel execution for image reconstruction,
Transferring final image (reconstructed by DS-DMAS) from GPU to CPU.

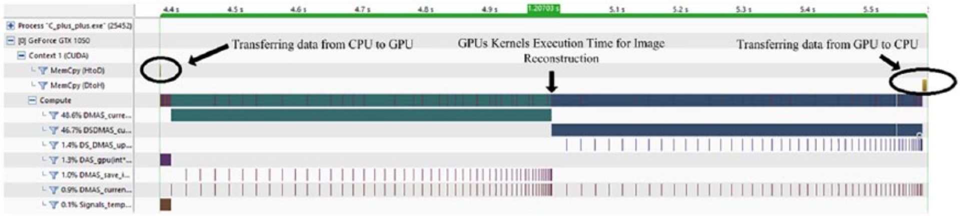

Figure 3. The CUDA profiling result of the DS-DMAS algorithm implementation. The result shows that kernels' execution time for image reconstruction is dominant in the total execution time. CUDA = compute unified device architecture; DS-DMAS = double-stage delay-multiply-and-sum; CPU = central processing unit; GPU = graphics processing unit; DMAS = delay-multiply-and-sum; DAS = delay-and-sum.

In the CUDA implementation, each thread calculates the value of a specific pixel brightness determined by the programmer. Threads are grouped into a three-level hierarchy: thread, block, and grid. Each block contains some threads, and the blocks are mapped into grids. The GPU has some limitations on the number of blocks and grids, resulting from its specifications. In CUDA, the warp is contained in a group of 32 threads that are executed simultaneously. In GPU, each thread has an identity number, where the CUDA user can map it into the GPU using the kernels. A kernel is actually a directive routine that is performed on the GPU. Launching a kernel is executed by subroutine or function call added by the <<>> syntax.[65,66]

Results and Discussion

In this section, the results of the DS-DMAS implementation on GPU are evaluated, and the effects of the key parameters are extracted. Finally, we present the reconstructed photoacoustic image created via the proposed implementation to highlight the superiority of the DS-DMAS algorithm for PAI.

Optimized CUDA Implementation for DS-DMAS

In the CUDA implementation, the GPU's main memory is utilized to store the transferred data via the CPU. Here, the NVIDIA GTX 1070 GPU along with NVIDIA profiler software's (nvprof) results is used for realizing optimum key parameters in the DS-DMAS implementation via the GPU. Figure 3 shows that the major running time of the DS-DMAS code is spent on executing the GPU kernels. The data communication and transfer time (between the CPU and GPU) can be ignored. Moreover, in our case, instructions and functions were executed sequentially in the GPU. The functions have zero overlap with each other due to the nature of the DS-DMAS algorithm. This means that we have several dependent functions. Thus, the only parameter to show the performance of the GPU implementation is the occupancy (number of active warps per SM)—this is reported in the following sections.

Effects of using shared memory. To implement the DS-DMAS, active warps per SM are extracted for various shared memory usages. The maximum occupancy would be achieved via the NVIDIA occupancy calculator (Figure 4) without using the shared memory. In the DS-DMAS algorithm, the loops do the exact same computation at each iteration and overwrite the

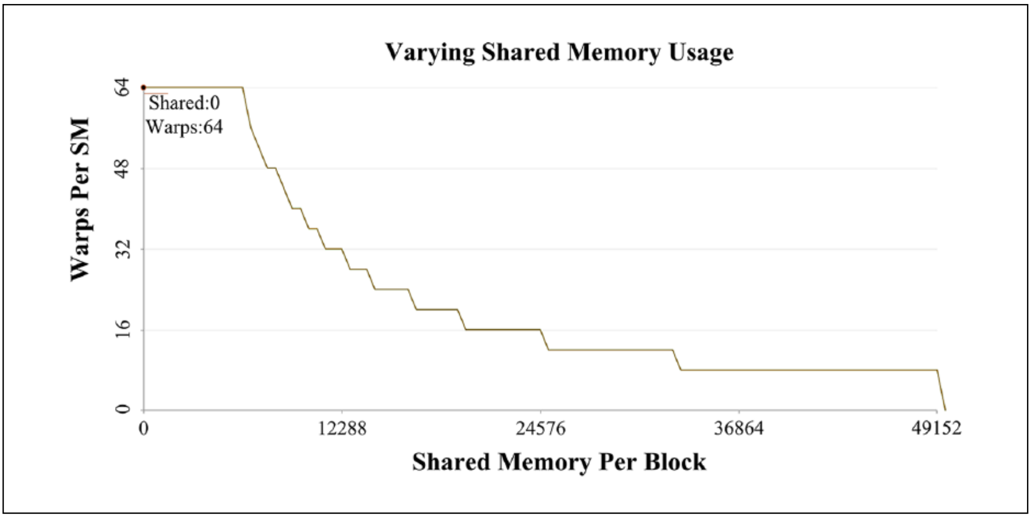

Figure 4. Number of the active warps (GPU occupancy) for various sizes of the shared memory utilized per thread block. Without using the shared memory per thread block, 64 thread warps simultaneously work in GPU, which is the maximum amount. GPU = graphics processing unit; SM = streaming multiprocessor.

result. Thus, shared memory is not needed because we are not using it to share data between threads or to optimize memory access order. Thus, these repetitive data fetching from global memory and data storage in shared memory spaces waste time. Each data point in each pixel is only used once in the implementation of each loop. Hence, storing data and using them in global memory is the best option.

Effects of different block sizes. The number of active warps per SM is also affected by the block size. Hence, changing the number of threads in each block could increase the occupancy without modifying the other parameters. The number of threads in each block is altered from 2 to 1024. The results are shown in Figure 5. The maximum performance is attained when the graph is in the highest value by a block size equal to 128 without using a shared memory.

Total GPU occupancy. Figure 6 shows the achieved occupancy for each SM. The reported values are the average across all the warp schedulers. The line across all the bars is the average—this is the number achieved in the implementation. A block size equal to 128 along with using the main memory led to an occupancy of 88.69% (Figure 6).

Number of used registers. The aim of the implementation is to attain 100% occupancy of multiprocessors. In other words, the application fully employs the available processing potentials. Unfortunately, the amount of shared memory utilized by each block and the number of employed registers bound the occupancy value. Figure 7 shows how the number of registers affects the theoretical occupancy leaving the other parameters constant. The circled point shows that in this implementation, 14 registers are utilized per thread. The current upper limit of the active warps is equal to 64. The results from the NVIDIA occupancy calculator demonstrate that the maximum occupancy could be obtained by a fewer number of employed registers. However, this leads to more usage of the cache memory, which is much slower to access.

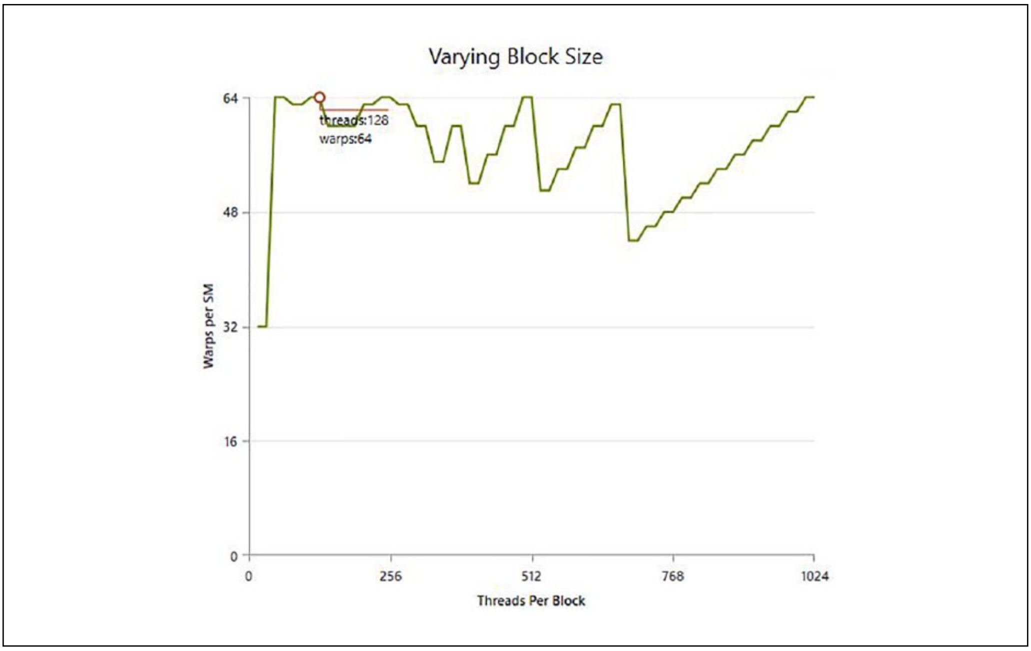

Figure 5. Effect of the different block sizes on the GPU occupancy. The maximum performance and occupancy are attained at a block size of 128. GPU = graphics processing unit; SM = streaming multiprocessor.

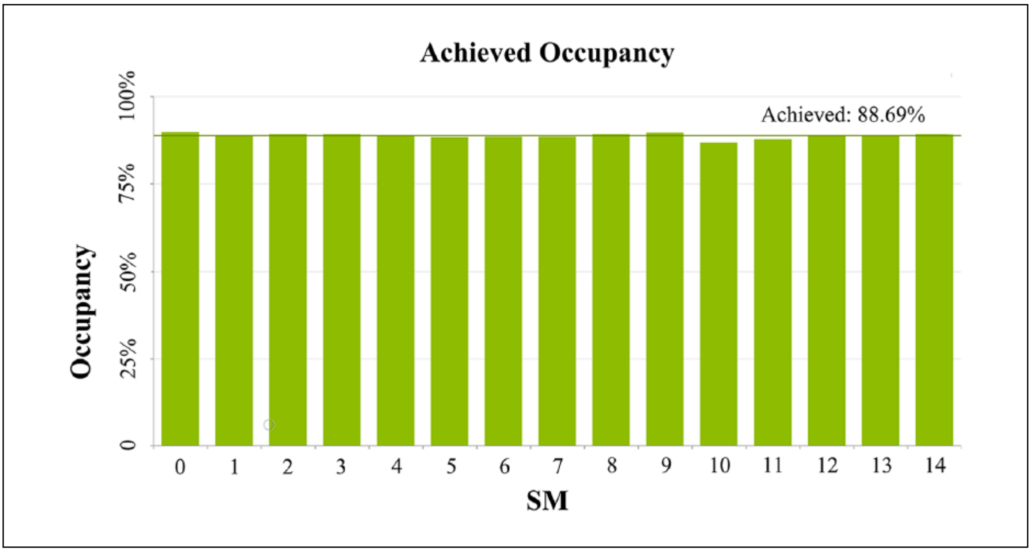

Figure 6. Achieved occupancy with block size equal to 128, without using the shared memory with a DS-DMAS implementation. The overall occupancy of 88.69% is achieved with a GPU CUDA implementation. SM = streaming multiprocessor DS-DMAS = double-stage delay-multiply-and-sum; GPU = graphics processing unit; CUDA = compute unified device architecture.

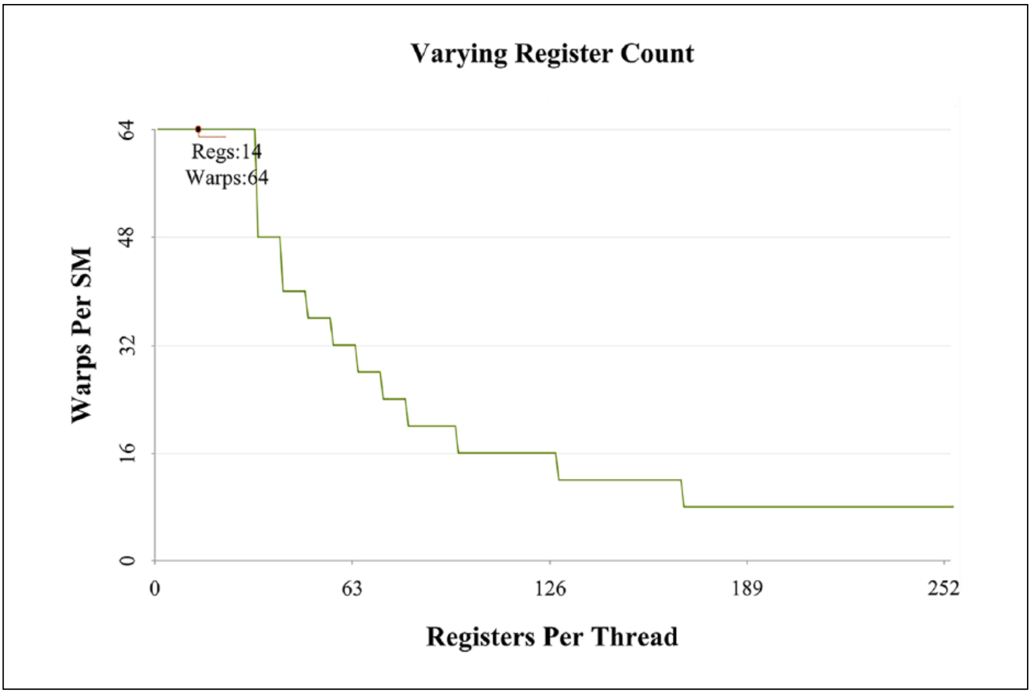

Figure 7. Effect of different numbers of registers per thread on the GPU occupancy. In this implementation, 14 registers are employed for each thread where the circled point is located at the highest amount of the active warps equal to 64. SM = streaming multiprocessor; GPU = graphics processing unit.

Speed-up evaluation. Table 2 shows the speed-up for a photoacoustic image having 1024 × 1024 pixels. This was evaluated for optimized CUDA (described above) on the NVIDIA GTX 1070 GPU and NVIDIA GTX 760 GPU. We note that the performance of the MATLAB code used for comparison is maximized by optimizing the memory access via a matrix operation rather than utilizing nested loops. The time reduction achieved by running optimized CUDA Fortran code is about 114.35-fold that of a single-threaded Fortran code and 133.34-fold that of the optimized MATLAB code with no loss of accuracy.

Figure 8 shows the speed reduction versus the number of pixels. The time gain increases with increasing number of pixels with no loss of accuracy. The relative error for the computed pixels value in CUDA programming model versus the serial implementation on a CPU is about 10^{-13} when double precision parameters are exploited.

DS-DMAS Evaluation

The higher performance of the DS-DMAS algorithm has been extensively evaluated previously.[37,38] We provided reconstructed images to briefly review the improvements offered by DS-DMAS. The reconstruction procedure is performed on a GPU along with a CPU, while the main computation of the reconstruction is performed on the GPU. For quantitative evaluation, the lateral variations of the formed images are presented in Figure 9. DAS leads to high level of sidelobes, and the background noise affects the image. DMAS improves the image quality by higher noise suppression versus DAS, as seen in Figure 9. Figure 9 shows

Table 2. Performance Comparison of the CUDA Optimized DS-DMAS Algorithm Implementation.

Implementation	Processing Time (s)	Speed-Up
CPU MATLAB	20	1
CPU FORTRAN serial	12	1.166
Optimized CUDA NVIDIA 760 GPU	0.3	66.55
Optimized CUDA NVIDIA 1070 GPU	0.15	133.34

The size of image is 1024 × 1024. CUDA = compute unified device architecture; DS-DMAS = double-stage delay-multiply-and-sum; CPU = central processing unit; GPU = graphics processing unit.

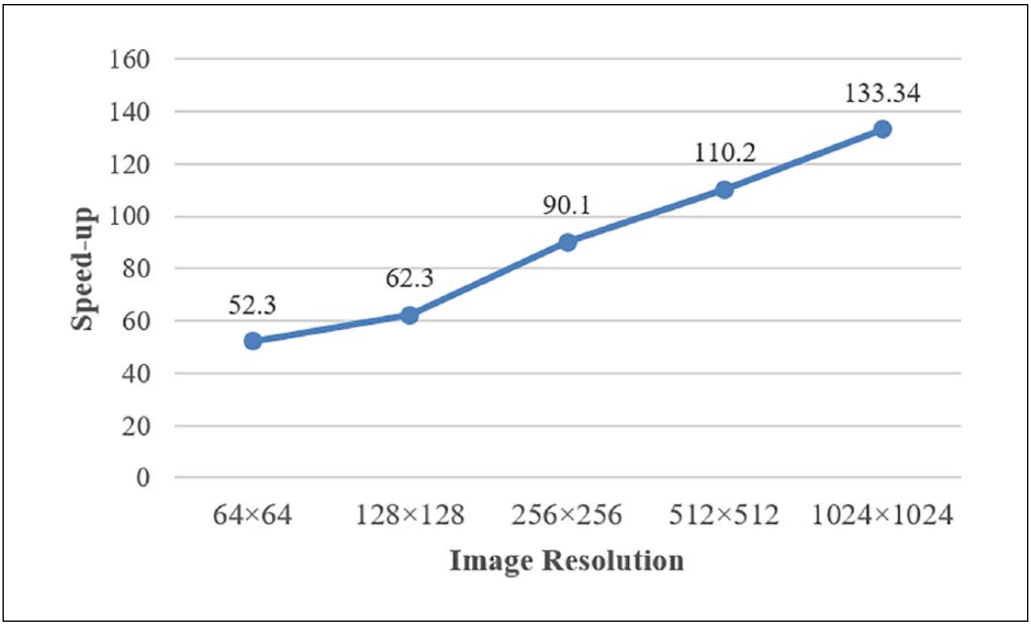

Figure 8. CUDA speed-up for several pixels in the image. CUDA = compute unified device architecture.

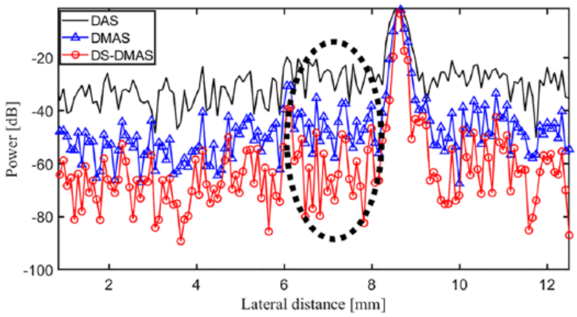

Figure 9. The lateral variations of the images are shown. The dotted circle shows the lower sidelobes of DS-DMAS (about 39 and 20 dB compared with DAS and DMAS, respectively). DS-DMAS = double-stage delay-multiply-and-sum; DAS = delay-and-sum; DMAS = delay-multiply-and-sum.

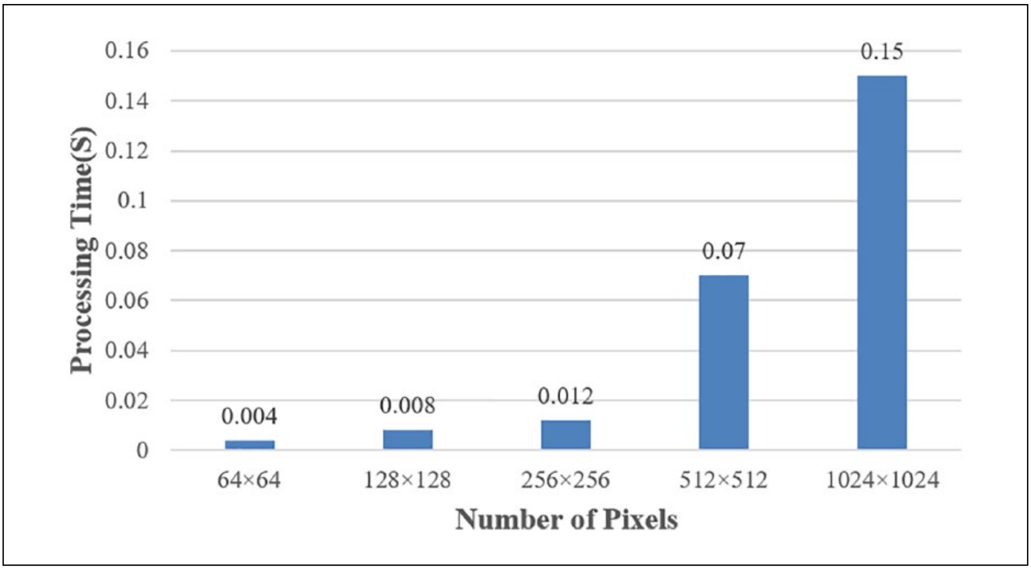

Figure 10. The processing time versus number of pixels. A higher time is needed to reconstruct the photoacoustic image with a larger number of pixels.

that DS-DMAS outperforms other methods in terms of noise suppression and sidelobe reduction. The DS-DMAS results in a 39- and 20-dB reduction in sidelobes versus DAS and DMAS, respectively.

Frame Rate

We have evaluated the performance of the GPU implementation when different numbers of pixels are used for the reconstruction. Figure 10 shows that the processing time would increase as the number of pixels is increased. Frame rates of 250, 125, and 83.3 are achieved when a size of 64 × 64, 128 × 128, and 256 × 256 are used chosen for the reconstructed photoacoustic image, respectively. A 512 × 512 images takes 0.07 s to be reconstructed using DS-DMAS.

Conclusion

In this paper, we implemented the DS-DMAS algorithm in a parallel approach on GPU for photoacoustic image reconstruction. The implementation used a NVIDIA GTX 1070 and NVIDIA GTX 760 GPUs with the CUDA programming model. After optimization, the CUDA programming model implemented on GPU offered a speed-up of nearly 133.34 × versus the CPU for a 1024 × 1024 pixel image. In addition, a higher speed-up was attained for a larger number of pixels. Using the proposed GPU implementation, it is possible to reconstruct photoacoustic images that were 64 × 64, 128 × 128, and 256 × 256 with a frame rate of 250, 125, and 83.3, respectively.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Seyyed Reza Miri Rostami https://orcid.org/0000-0002-6221-9257

Mohsen Ghaffari-Miab https://orcid.org/0000-0002-8638-2753

Supplemental Material

Supplemental material for this article is available online.

References

1. Wang X, Pang Y, Ku G, Xie X, Stoica G, Wang LV. Noninvasive laser-induced photoacoustic tomography for structural and functional in vivo imaging of the brain. Nat Biotechnol. 2003;21(7):803-6.
2. Wang X, Xie X, Ku G, Wang LV, Stoica G. Noninvasive imaging of hemoglobin concentration and oxygenation in the rat brain using high-resolution photoacoustic tomography. J Biomed Opt. 2006;11(2):024015.
3. Wang J, Lin CY, Moore C, Jhunjhunwala A, Jokerst JV. Switchable photoacoustic intensity of methylene blue via sodium dodecyl sulfate micellization. Langmuir. 2018;34(1):359-65.
4. Xu M, Wang LV. Photoacoustic imaging in biomedicine. Rev Sci Instrum. 2006;77(4):041101.
5. Jokerst JV, Thangaraj M, Kempen PJ, Sinclair R, Gambhir SS. Photoacoustic imaging of mesenchymal stem cells in living mice via silica-coated gold nanorods. ACS Nano. 2012;6(7):5920-30.
6. Kim T, Lemaster JE, Chen F, Li J, Jokerst JV. Photoacoustic imaging of human mesenchymal stem cells labeled with Prussian blue-poly(l-lysine) nanocomplexes. ACS Nano. 2017;11(9):9022-32.
7. Wang LV, Hu S. Photoacoustic tomography: in vivo imaging from organelles to organs. Science. 2012;335(6075):1458-62.
8. Guo B, Li J, Zmuda H, Sheplak M. Multifrequency microwave-induced thermal acoustic imaging for breast cancer detection. IEEE Trans Biomed Eng. 2007;54(11):2000-10.
9. Pramanik M, Ku G, Li C, Wang LV. Design and evaluation of a novel breast cancer detection system combining both thermoacoustic (TA) and photoacoustic (PA) tomography. Med Phys. 2008;35(6 Pt. 1):2218-23.
10. Mehrmohammadi M, Joon Yoon S, Yeager D, Emelianov SY. Photoacoustic imaging for cancer detection and staging. Curr Mol Imaging. 2013;2(1):89-105.
11. Hariri A, Wang J, Kim Y, Jhunjhunwala A, Chao DL, Jokerst JV. In vivo photoacoustic imaging of chorioretinal oxygen gradients. J Biomed Opt. 2018;23(3):036005.
12. de La Zerda A, Paulus YM, Teed R, Bodapati S, Dollberg Y, Khuri-Yakub BT, Blumenkranz MS, Moshfeghi DM, Gambhir SS. Photoacoustic ocular imaging. Opt Lett. 2010;35(3):270-2.
13. Pu K, Shuhendler AJ, Jokerst JV, Mei J, Gambhir SS, Bao Z, Rao J. Semiconducting polymer nanoparticles as photoacoustic molecular imaging probes in living mice. Nat Nanotechnol. 2014;9(3):233-9.
14. Turani Z, Fatemizadeh E, Blumetti T, Daveluy S, Moraes AF, Chen W, Mehregan D, Andersen PE, Nasiriavanaki M. Optical Radiomic Signatures Derived from Optical Coherence Tomography Images Improve Identification of Melanoma. Cancer Research. 2019;79(8):2021-2030.
15. Yao J, Xia J, Maslov KI, Nasiriavanaki M, Tsytsarev V, Demchenko AV, Wang LV. Noninvasive photoacoustic computed tomography of mouse brain metabolism in vivo. NeuroImage. 2013;64:257-66.
16. Nasiriavanaki M, Xia J, Wan H, Bauer AQ, Culver JP, Wang LV. High-resolution photoacoustic tomography of resting-state functional connectivity in the mouse brain. Proc Natl Acad Sci U S A. 2014;111(1):21-6.
17. Li ML, Oh JT, Xie X, Ku G, Wang W, Li C, Lungu G, Stoica G, Wang LV. Simultaneous molecular and hypoxia imaging of brain tumors in vivo using spectroscopic photoacoustic tomography. Proc IEEE. 2008;96(3):481-9.
18. Valluru KS, Wilson KE, Willmann JK. Photoacoustic imaging in oncology: translational preclinical and early clinical experience. Radiology. 2016;280(2):332-49.

19. Song W, Wei Q, Liu W, Yi J, Sheibani N, Fawzi AA, Linsenmeier RA, Jiao S, Zhang HF. A combined method to quantify the retinal metabolic rate of oxygen using photoacoustic ophthalmoscopy and optical coherence tomography. Sci Rep. 2014;4:6525.
20. Taruttis A, Herzog E, Razansky D, Ntziachristos V. Real-time imaging of cardiovascular dynamics and circulating gold nanorods with multispectral optoacoustic tomography. Opt Express. 2010;18(19):19592-602.
21. Van de Sompel D, Sasportas LS, Jokerst JV, Gambhir SS. Comparison of deconvolution filters for photoacoustic tomography. PLoS ONE. 2016;11(3):e0152597.
22. Wang LV. Multiscale photoacoustic microscopy and computed tomography. Nat Photonics. 2009;3(9):503-9.
23. Mastanduno MA, Gambhir SS. Quantitative photoacoustic image reconstruction improves accuracy in deep tissue structures. Biomed Opt Express. 2016;7(10):3811-25.
24. Mozaffarzadeh M, Yan Y, Mehrmohammadi M, Makkiabadi B. Enhanced linear-array photoacoustic beamforming using modified coherence factor. J Biomed Opt. 2018;23(2):026005.
25. Paltauf G, Viator J, Prahl S, Jacques SL. Iterative reconstruction algorithm for optoacoustic imaging. J Acoust Soc Am. 2002;112(4):1536-44.
26. Xu M, Xu Y, Wang LV. Time-domain reconstruction algorithms and numerical simulations for thermoacoustic tomography in various geometries. IEEE Trans Biomed Eng. 2003;50(9):1086-99.
27. Mozaffarzadeh M, Periyasamy V, Pramanik M, Makkiabadi B. An efficient nonlinear beamformer based on p^{th} root of detected signals for linear-array photoacoustic tomography: application to sentinel lymph node imaging, Arxiv:1805.09913, 2018. Available from https://arxiv.org/abs/1805.09913.
28. Upputuri PK, Pramanik M. Recent advances toward preclinical and clinical translation of photoacoustic tomography: a review. J Biomed Opt. 2016;22(4):041006.
29. Sivasubramanian K, Periyasamy V, Dienzo RA, Pramanik M. Hand-held, clinical dual mode ultrasound—photoacoustic imaging of rat urinary bladder and its applications. J Biophotonics. 2018;11(5):e201700317.
30. Zeng Y, Xing D, Wang Y, Yin B, Chen Q. Photoacoustic and ultrasonic coimage with a linear transducer array. Opt Lett. 2004;29(15):1760-2.
31. Mercep E, Jeng G, Morscher S, Li PC, Razansky D. Hybrid optoacoustic tomography and pulse-echo ultrasonography using concave arrays. IEEE T Ultrason Ferr. 2015;62(9):1651-61.
32. Yin B, Xing D, Wang Y, Zeng Y, Tan Y, Chen Q. Fast photoacoustic imaging system based on 320-element linear transducer array. Phys Med Biol. 2004;49(7):1339-46.
33. Zhou Q, Ji X, Xing D. Full-field 3D photoacoustic imaging based on plane transducer array and spatial phase-controlled algorithm. Med Phys. 2011;38(3):1561-6.
34. Mozaffarzadeh M, Mahloojifar A, Orooji M, Kratkiewicz K, Adabi S, Nasiriavanaki M. Linear-array photoacoustic imaging using minimum variance-based delay multiply and sum adaptive beamforming algorithm. J Biomed Opt. 2018;23(2):026002.
35. Park S, Karpiouk AB, Aglyamov SR, Emelianov SY. Adaptive beamforming for photoacoustic imaging using linear array transducer. In: Ultrasonics Symposium, Beijing, China, 2-5 November 2008-21 March 2009, pp. 1088-91.
36. Alshaya A, Harput S, Moubark AM, Cowell DMJ, McLaughlan J, Freear S. Spatial resolution and contrast enhancement in photoacoustic imaging with filter delay multiply and sum beamforming technique. In: Ultrasonics Symposium, Tours, France, 18-21 September-3 November 2016, pp. 1-4.
37. Mozaffarzadeh M, Sadeghi M, Mahloojifar A, Orooji M. Double-stage delay multiply and sum beamforming algorithm applied to ultrasound medical imaging. Ultrasound Med Biol. 2018;44(3):677-86.
38. Mozaffarzadeh M, Mahloojifar A, Orooji M, Adabi S, Nasiriavanaki M. Double-stage delay multiply and sum beamforming algorithm: application to linear-array photoacoustic imaging. IEEE Trans Biomed Eng. 2018;65(1):31-42.
39. Paridar R, Mozaffarzadeh M, Mehrmohammadi M, Orooji M. Photoacoustic image formation based on sparse regularization of minimum variance beamformer. Biomed Opt Express. 2018;9(6):2544-61.
40. Matrone G, Savoia AS, Caliano G, Magenes M. The delay multiply and sum beamforming algorithm in ultrasound B-mode medical imaging. IEEE Trans Med Imaging. 2015;34(4):940-9.

41. Mokdad A, Azmi P, Mokari N, Moltafet M, Ghaffari-Miab M. Cross-layer energy efficient resource allocation in PD-NOMA based H-CRANs: implementation via GPU. IEEE Trans Mob Computing. 2019;18:1246-59. doi:10.1109/TMC.2018.2860985.
42. Masumnia-Bisheh K, Ghaffari-Miab M, Zakeri B. Evaluation of different approximations for correlation coefficients in stochastic FDTD to estimate SAR variance in a human head model. IEEE Trans Electromagn C. 2017;59(2):509-17. doi:10.1109/TEMC.2016.2614128.
43. Pratx G, Xing L. GPU computing in medical physics: a review. Med Phys. 2011;38(5):2685-97.
44. Rostami SRM, Ghaffari-Miab M. Finite difference generated transient potentials of open-layered media by parallel computing using OPENMP, MPI, OPENACC, and CUDA. IEEE Trans Antenn Propag. 2019:doi:10.1109/TAP.2019.2920253 (epub ahead of print).
45. Kang H, Lee SW, Lee ES, Kim SH, Lee TG. Real-time GPU-accelerated processing and volumetric display for wide-field laser-scanning optical-resolution photoacoustic microscopy. Biomed Opt Express. 2015;6(12):4650-60.
46. Shan T, Qi J, Jiang M, Jiang H. GPU-based acceleration and mesh optimization of finite-element-method-based quantitative photoacoustic tomography: a step towards clinical applications. Appl Opt. 2017;56(15):4426-32.
47. Ding L, Deán-Ben XL, Razansky D. Real-time model-based inversion in cross-sectional optoacoustic tomography. IEEE Trans Med Imaging. 2016;35(8):1883-91.
48. Shan T. Accelerated time domain quantitative photoacoustic tomography (TD-qPAT) based on graphic processing units (GPU) for clinical application of breast cancer imaging. In: Optical Tomography and Spectroscopy, paper OW4D.3, Optical Society of America, 3-6 April 2018. Available from https://www.osapublishing.org/abstract.cfm?uri=OTS-2018-OW4D.3.
49. Liu S, Feng X, Gao F, Jin H, Zhang R, Luo Y, Zheng Y. GPU-accelerated two dimensional synthetic aperture focusing for photoacoustic microscopy. APL Photon. 2018;3(2):026101.
50. Wang K, Huang C, Kao YJ, Chou CY, Oraevsky AA, Anastasio MA. Accelerating image reconstruction in three-dimensional optoacoustic tomography on graphics processing units. Med Phys. 2013;40(2): 023301.
51. Lutzweiler C, Deán-Ben XL, Razansky D. Expediting model-based optoacoustic reconstructions with tomographic symmetries. Med Phys. 2014;41(1):013302.
52. Arridge S, Beard P, Betcke M, Cox B, Huynh N, Lucka F, Ogunlade O, Zhang E. Accelerated high-resolution photoacoustic tomography via compressed sensing. Phys Med Biol. 2016;61(24):8908.
53. Wen T, Li L, Zhu Q, Qin W, Gu J, Yang F, Xie Y. GPU-accelerated kernel regression reconstruction for freehand 3D ultrasound imaging. Ultrason Imaging. 2017;39(4):240-59.
54. Rostami SRM, Mozaffarzadeh M, Hariri A, Jokerst JV, Ghaffari-Miab M. OpenACC GPU implementation of double-stage delay-multiply-and-sum algorithm: toward enhanced real-time linear-array photoacoustic tomography. In: Photons Plus Ultrasound: imaging and Sensing 2019, International Society for Optics and Photonics, volume 10878, p. 108785C. Available from https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10878/108785C/OpenACC-GPU-implementation-of-double-stage-delay-multiply-and-sum/10.1117/12.2511115.short?SSO=1.
55. Gamelin J, Maurudis A, Aguirre A, Huang F, Guo P, Wang LV, Zhu Q. A real-time photoacoustic tomography system for small animals. Opt Express. 2009;17(13):10489-98.
56. Taruttis A, Ntziachristos V. Advances in real-time multispectral optoacoustic imaging and its applications. Nat Photon. 2015;9(4):219-27.
57. Li M, Liu C, Gong X, Zheng R, Bai Y, Xing M, Du X, Liu X, Zeng J, Lin R, Zhou H, Wang S, Lu G, Zhu W, Fang C, Song L. Linear array-based real-time photoacoustic imaging system with a compact coaxial excitation handheld probe for noninvasive sentinel lymph node mapping. Biomed Opt Express. 2018;9(4):1408-22.
58. Arnal B, Wei CW, Perez C, Nguyen TM, Lombardo M, Pelivanov I, Pozzo LD, O'Donnell M. Sono-photoacoustic imaging of gold nanoemulsions: part II. real time imaging. Photoacoustics. 2015;3(1):11-9.
59. O'Donnell M, Nguyen TM, Wei CW, Xia J. Real-time photoacoustic and ultrasound imaging system and method, US Patent App. 15/308,828, 2017. Available from http://www.freepatentsonline.com/y2017/0079622.html.

60. Cui H, Yang X. Real-time monitoring of high-intensity focused ultrasound ablations with photoacoustic technique: an in vitro study. Med Phys. 2011;38(10):5345-50.
61. Song L, Kim C, Maslov K, Shung KK, Wang LV. High-speed dynamic 3D photoacoustic imaging of sentinel lymph node in a murine model using an ultrasound array. Med Phys. 2009;36(8):3724-9.
62. Park J, Jeon S, Meng J, Song L, Lee JS, Kim C. Delay-multiply-and-sum-based synthetic aperture focusing in photoacoustic microscopy. J Biomed Opt. 2016;21(3):036010.
63. Hariri A, Lemaster J, Wang J, Jeevarathinam AS, Chao DL, Jokerst JV. The characterization of an economic and portable LED-based photoacoustic imaging system to facilitate molecular imaging. Photoacoustics. 2018;9:10-20.
64. Rostami SRM, Ghaffari-Miab M. Fast computation of finite difference generated time-domain green's functions of layered media using OpenAcc on graphics processors. In: 2017 Iranian Conference on Electrical Engineering, pp. 1596-99. doi:10.1109/IranianCEE.2017.7985300.
65. Ruetsch G, Fatica M. CUDA FORTRAN for Scientists and Engineers. Santa Clara, CA: NVIDIA Corporation; 2011.
66. NVIDIA CUDA C Programming guide, 2010. Available from http://developer.download.nvidia.com/compute/cuda/3_1/toolkit/docs/NVIDIA_CUDA_C_ProgrammingGuide_3.1.pdf.