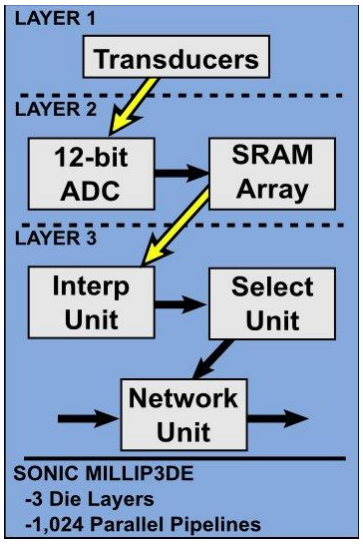
[1] R. Sampson, M. Yang, S. Wei, C. Chakrabarti, and T. F. Wenisch. “Sonic Millip3De: Massively Parallel 3D-Stacked Accelerator for 3D Ultrasound.” Proc. of the 19th IEEE International Symposium on High Performance Computer Architecture, Feb. 2013.



IUS1-D3-5

Lossless Compression with Parallel Decoder for Improving Performance of GPU-based Beamformer

U-Wai Lok[1], Gang-Wei Fan[1], Pai-Chi Li[2]; [1]*National Taiwan University, Taiwan,* [2]*Electrical Engineering, National Taiwan University, Taipei, Taiwan*

Background, Motivation and Objective

The massive data transfer of ultrasound raw data becomes one of the critical limitations for a real time software based beamformer. Lossless compression with sequential decoder can reduce data size of ultrasound raw data but the time-consumed decompression process degrades the performance of data transfer, thus the overall performance is also degraded. Therefore, a fast, parallel decoding scheme is highly desirable. A lossless compression approach associated with parallel decoding on graphics processing units (GPU) is proposed in this study. The hypothesis is that the execution time of transferring compressed data along with decoding is less than transferring the original raw data. Thus, the issue of data transfer is mitigated.

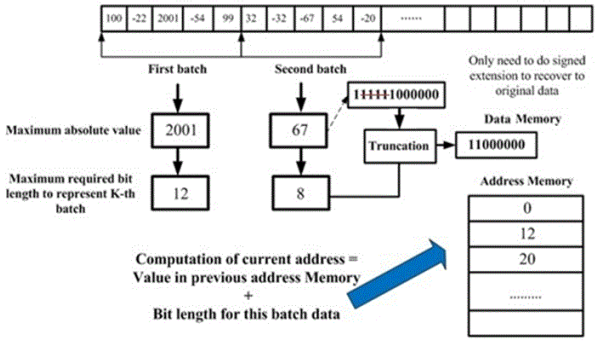
Statement of Contribution/Methods

RF raw data are first demodulated to baseband, and then the baseband data are compressed through our proposed lossless encoder. For the encoder design, baseband data within a channel are grouped into several batches, and the bit length to represent each signal within the same batch is fixed but varied for different batches. The bit length is determined by the sample with the largest absolute value within a batch. In addition, an extra address is utilized to indicate the bit length and the memory location of each batch as shown in figure A. At the decoder side, GPU is only required to compute the bit length and memory position of each sample through address information. Therefore, all samples can be decompressed in parallel in the GPU.

Results/Discussion

The latency to demodulate and compress a frame data in FPGA requires about 4 ms, and the proposed compression ratio reaches 3.5. Hence, the transmission time from front end to the software backend is reduced. At the software backend, the processing time of transferring a compressed frame data from CPU to GPU with the parallel decoder only requires about 5 ms. This is nearly a three-fold improvement. Details of each condition are shown in figure B. The processing time of data transfer from CPU to GPU is reduced by our proposed data compression and parallel decoding process in GPU. Another concomitant benefit of our proposed method is that the data transfer rate between the front end and software backend becomes less demanding.

A



B

Conditions	Time(ms)
Demodulation and encoder in FPGA	3.96
Transmit original RF data from CPU to GPU	16.12
Transmit compressed data from CPU to GPU	3.82
Decoder in GPU	1.21
Transfer compressed data and decoding process	5.04

Figure (A) Illustration diagram of encoder in FPGA and (B) processing time of each condition.

IUS1-D3-6

A Fast Parallelized Eigen-Based Clutter Filter Framework for Ultrasound Color Flow Imaging

Adrian J. Y. Chee[1], Billy Y. S. Yiu[1], Alfred C. H. Yu[1]; [1]*Medical Engineering Program, The University of Hong Kong, Pokfulam, Hong Kong*

Background, Motivation and Objective

Eigen-filters with attenuation response adapted to clutter statistics have emerged as a new class of clutter suppression techniques in color flow imaging (CFI) to achieve high flow detection sensitivity in the presence of tissue motion. There is growing interest to extend this technique beyond laboratory investigations by incorporating it into the CFI processing routine on ultrasound scanners. Nevertheless, bridging such a gap from theory to practice is known to be challenging from a real-time computing perspective because the singular value decomposition (SVD) operation in eigen-filtering is known to pose heavy computational burden. Here we seek to overcome this issue by formulating a new technical framework that enables fast execution of eigen-filtering so as to foster their practical adoption in CFI.

Statement of Contribution/Methods

Our fast eigen-processing framework for CFI was devised by integrating three principles: (1) applying mature eigen-computation algorithms in matrix algebra; (2) leveraging on the single-instruction, multiple-data (SIMD) computing approach that has became prevalent in ultrasound imaging with the advent of graphics processing units (GPU); (3) adopting the single-ensemble-based (Hankel-SVD) eigen-filter paradigm that is algorithmically compatible with SIMD computing. For each CFI pixel, its flow estimates were derived by processing the pixel's slow-time ensemble as follows. First, we computed the SVD of the slow-time ensemble's Hankel matrix using a parallelized algorithm that includes the steps of Householder Transform, QR decomposition, and Givens rotation (implemented through NVidia's CUDA programming tool). After that, the power and frequency of each eigen-component was estimated in parallel using the GPU (via the lag-one autocorrelation method). Adaptive filter order selection was then applied to extract the principal eigen-component corresponding to blood flow. The real-time efficacy of our framework was investigated using a GTX 480 GPU and synthesized CFI datasets with various sizes of pixel counts (P) and slow-time ensemble sizes (N) (5 MHz frequency; 20dB clutter-to-blood signal ratio; 2mm/s max tissue velocity; 38cm/s max flow velocity). Frame processing throughput and the execution time of each computational step were measured and analyzed using CUDA's visual profiler.

Results/Discussion

Using our CFI eigen-processing framework, real-time video-range throughput (24 fps) was achieved for frames with P = 70,000 (128 scanlines, each with 547 depth positions) and N = 10. Flow estimation accuracy was not compromised. Processing time increased exponentially with N: e.g. for P = 10,000, the throughput was >600 fps for N = 4 but was only 23 fps for N = 21. The load only increased linearly with P, implying that our framework is readily scalable for larger frame sizes. To our knowledge, this work is the first demonstration that real-time CFI with eigen-processing can be achieved in practice.