# Acceleration of Ultrasonic Data Compression Using OpenCL on GPU

Boyang Wang, Pramod Govindan, Thomas Gonnot and Jafar Saniie
*Department of Electrical and Computer Engineering*
*Illinois Institute of Technology, Chicago, Illinois, U.S.A.*

***Abstract* – Ultrasonic imaging applications require real-time acquisition and processing of huge volume of data. Subsequently, reducing the data storage becomes an essential requirement for applications involving ultrasound technology. Compression of the acquired data significantly reduces the storage and also helps in faster data processing. Furthermore, computationally efficient data compression enables real-time data transfer to remote locations for expert analysis. In this study, an efficient implementation of ultrasonic data compression based on discrete wavelet transform (DWT) is realized on GeForce GT 750M NVIDIA graphical processing unit (GPU) using Open Computing Language (OpenCL). Execution of OpenCL program on GPU allows parallel computing of the compression algorithm to accelerate the computational performance. This study demonstrates that OpenCL implementation of ultrasonic compression algorithm on GPU requires only 0.29 seconds for compressing 33 Mbytes of ultrasonic data into 6.6 Mbytes (80% compression), which makes it suitable for real-time ultrasonic imaging applications.**

## I. INTRODUCTION

Ultrasonic imaging is one of the most commonly used techniques in medical diagnostics as well as in nondestructive testing applications. Point of care technologies and remote health monitoring have strict requirements for real-time high quality diagnostics. Conventional ultrasound systems are not suitable for real-time processing, which obstructs the system to perform dynamic diagnostics [1]. Advancements in the field of embedded computing have facilitated the development of computationally efficient ultrasound systems for real-time signal processing applications. Handheld imaging systems are commercially feasible, by utilizing the capabilities of powerful embedded processors [2].

Ultrasonic systems impose several challenges. One of them is the large amount of data required for processing, which demands heavy storage. Consequently, this increases the cost and size. The solution for this problem is compression of the acquired data without losing the useful information. Additionally, for real-time applications, the computation time for performing data compression and other algorithms is very significant.

Real-time applications, that involve complex and laborious computational operations, can be efficiently executed by using parallel processing capabilities of graphical processing unit (GPU) [3]. Open Computing Language (OpenCL) allows designers to generate simple C programs that can be interfaced to GPU to enable high level of parallel processing for improved computational performance [4]. Therefore, in this study, OpenCL is used to implement the signal compression on a GPU to accelerate the performance.

Unlike traditional C-programs, which are sequential in nature, OpenCL utilizes the possibilities of parallelism in hardware circuits. The programs written in OpenCL (called kernels) will be executed on several compute devices available within GPU, which is attached to a host processor. Kernels are simple functions that transform input memory objects into output memory objects.

Ultrasonic signal compression is performed by utilizing the high compaction properties of discrete wavelet transform (DWT). To maximize the compression with high quality reconstruction, Daubechies10 (Db10) wavelet is used for the decomposition of ultrasonic signal into multiple sub-bands. The sub-bands with very low energy values are eliminated to achieve signal compression.

In this study, ultrasonic signal compression is implemented by using OpenCL on GeForce GT 750M NVIDIA GPU running at 967 MHz. Additionally, the same compression algorithm is implemented using C++ on Intel (R) Core ™ i7-4500U CPU running at 1.9 GHz. Finally, the computational performance of both GPU and CPU implementations are analyzed and compared.

978-1-4799-8802-0/15/$31.00 ©2015 IEEE

In this paper, Section II explains ultrasonic data compression algorithm using DWT. Section III details the OpenCL architecture. Section IV describes the implementation of ultrasonic data compression algorithm using OpenCL on GPU and analyzes the computational performance of GPU implementation compared to the CPU implementation. Section V concludes the paper.

## II. ULTRASONIC DATA COMPRESSION

Ultrasonic data compression is performed by utilizing the signal compaction properties of Discrete Wavelet Transform (DWT). DWT is based on sub-band decomposition. The input signal samples are decomposed by filtering into lowpass and highpass sub-bands. These sub-bands are further decomposed to get narrower sub-bands which generate smaller size transform coefficients. The sub-bands with very low energy will not carry any required information. Therefore, they can be eliminated to achieve signal compression. The compressed signal can be synthesized to reconstruct the original signal by using inverse DWT.

Due to the similarity between the Daubechies10 (Db10) wavelet function and ultrasonic signal echo, Db10 wavelet is used in this study for maximum compaction of the ultrasonic signal. Furthermore, wavelet packet decomposition method [5] is used as shown in Figure 1 to maximize the compression ratio. By discarding the low energy coefficients (H, LH & LLHL), 80% compression is achieved by compressing this signal.

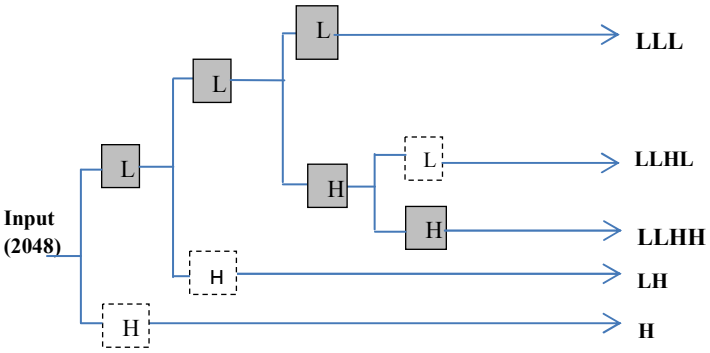


Figure 1. Wavelet packet decomposition (Only LLL and LLHH are retained. Thus, the filters in dashed line are not required)

## III. OpenCL SYSTEM ARCHITECTURE

OpenCL platform model describes a high level representation of any heterogeneous platform [6] as shown in Figure 2. As can be seen from Figure 2, the host is connected to multiple compute devices. The compute device is where the streams of instructions (kernels) execute.

OpenCL compute devices are divided into several processing elements (PEs). Computations on a device occur within the PEs. The kernels are distributed on the PEs by an application programming interface. The number of processing elements and the number of tasks which can be executed simultaneously decides the number of parallel operations, which in turn determine the computational speed.

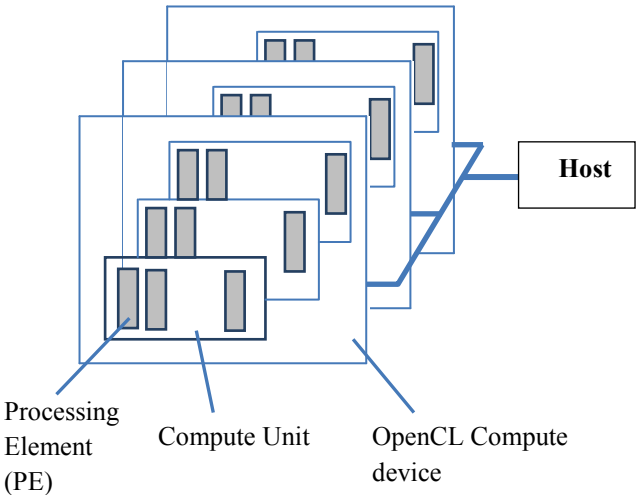


Figure 2. OpenCL compute device with several processing elements

The smallest PE is defined as a work-item. Multiple work-items are grouped into local work groups, as shown in Figure 3. The overall work space is put into an N-dimensioned index space according to the algorithm that the user defines. Each work-item has its own unique global coordinate which can be used to identify the work-item. The work-items that are divided into the same work group will be executed simultaneously.

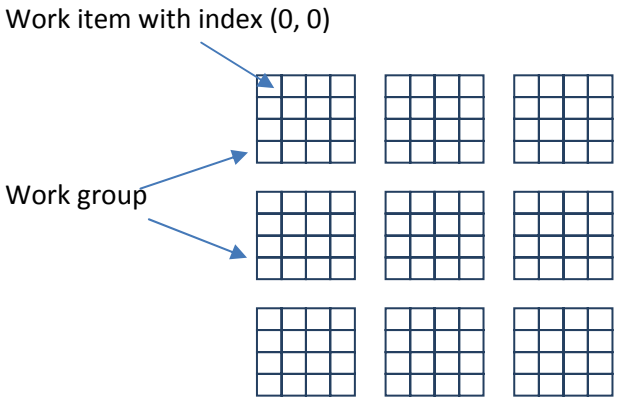


Figure 3. OpenCL work-group consisting of many work-items

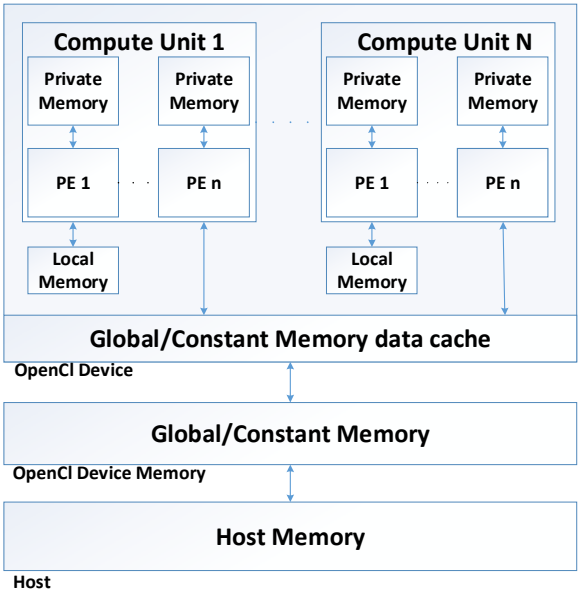


Figure 4. OpenCL memory model

Figure 4 shows OpenCL memory model. The global memory is the memory region that allows read/write access to all work-items in all work-groups. Work-items can read from or write to many elements of a memory object in global memory. Reads and writes to global memory may be cached depending on the capabilities of the device. Constant memory is the memory region of global memory that remains constant during the execution of a kernel. The host allocates and initializes memory objects placed into constant memory. Local memory is the memory region which is local to a work-group. This memory region can be used to allocate variables that are shared by all work-items in that work-group. It may be implemented as dedicated regions of memory on the OpenCL device. Alternatively, the local memory region may be mapped onto sections of the global memory. Private memory is the region of memory which is private to a work-item. Variables defined in one work-item's private memory are not visible to other work-items.

## IV. OpenCL IMPLEMENTATION OF ULTRASONIC DATA COMPRESSION ON GPU

DWT involves filtering and sub-sampling. For maximum compaction of the ultrasonic signal, Db10 wavelet with 20 filter coefficients is used in this study.

At one instance of time, all the PEs within GPU executes the same instruction, which is programmed in the kernel. This indicates that GPU can efficiently employ parallelism for repetitive calculations. Furthermore, by using certain OpenCL APIs, users can define different kernels and let GPU jump between these kernels.

Equation (1) represents an FIR filter. From (1), it can be seen that there are N+1 multiplication operations and N addition operations for each output sample y[n]. If the input signal x[n] has M samples, we need M iterations of similar operations.

$$y[n] = \sum_{i=0}^{N} b_i \cdot x[n-1] \qquad (1)$$

GPU will be able to accelerate the algorithm by breaking down the loops and assigning them to multiple PEs. The OpenCL architecture for filtering and sub-sampling of the ultrasonic signal with 2048 samples is shown in Figure 5. Each work item in Figure 5 represents a PE and it has its own global id. This global id is used to access certain location of the GPU memory.

As shown in Figure 1, the signal compression requires three lowpass filters and two highpass filters. Instead of filtering followed by down-sampling, only alternate samples are chosen and filtered. This reduces the number of filtering operations by 50%, and improves the computational efficiency. Therefore, as shown in Figure 5, during the first stage of decomposition only 1024 work items (PEs) are used to generate 1024 samples.

The second stage of decomposition accepts the 1024 samples from stage 1 output, and generates 512 samples by using 512 work items. This 512 samples are decomposed and transformed into 256 samples in the third stage. This will be further decomposed and transformed into 128 samples in the fourth stage.

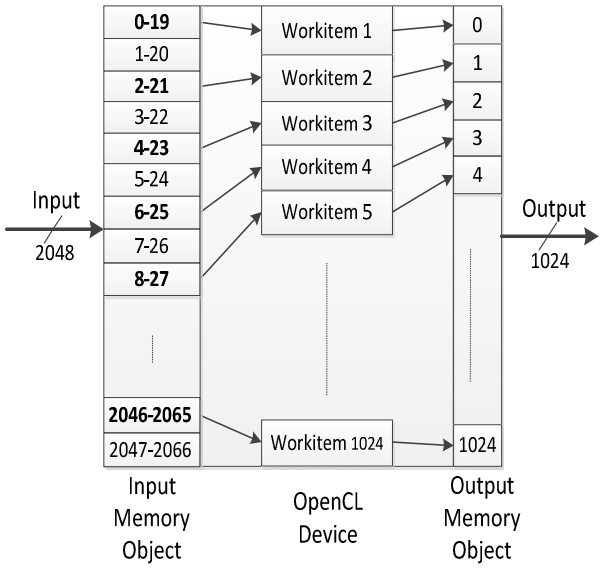


Figure 5. OpenCL filter structure

In this study, the ultrasonic compression algorithm is implemented on GeForce GT 750M NVIDIA GPU running at 967 MHz and also on Intel (R) Core ™ i7-4500U CPU running at 1.9 GHz. While GPU executes OpenCL program, CPU implementation uses C++. The program initialization routine for both GPU and CPU implementations is performed by CPU. The OpenCL program execution on GPU requires establishing communication between CPU and GPU, which includes reading the OpenCL kernel file and building the binary executable for GPU. Furthermore, CPU needs to copy the input data and filter coefficients to the GPU memory for processing, which requires additional time. For the CPU implementation, the initialization routine does not require considerable time as compared to GPU. However, the possibility of parallel processing is much higher in case of GPU implementation. The time consumed by various sub-tasks for GPU and CPU implementations for compressing 33 Mbytes of ultrasonic data into 6.6 Mbytes (80% compression) is detailed in Table I.

From Table I, it can be seen that the initialization requires 0.655 seconds for GPU implementation which is much higher compared to 0.086 seconds required for CPU implementation. However, the total execution time is lower for GPU. Furthermore, if the initialization is excluded, GPU executes 4 times faster than CPU, which is significant. Table I also shows that both GPU and CPU implementations are suitable for real-time ultrasonic imaging applications, since the execution times are very small compared to ultrasonic data acquisition time.

Table I. Execution time (in seconds) for GPU and CPU implementations for compressing 33 Mbytes of ultrasonic data into 6.6 Mbytes (80% compression)

| Task | GT 750M NVIDIA GPU (967 MHz) | Intel (R) Core ™ i7-4500U CPU (1.9 GHz) |
|---|---|---|
| Initialization | 0.655 | 0.086 |
| Reading data | 0.04 | 0.037 |
| Pre-processing | 0.12 | 0.131 |
| Stage 1 low pass decomposition | 0.369 | 0.525 |
| Stage 2 low pass decomposition | 0.193 | 0.239 |
| Stage 3 high pass decomposition | 0.126 | 0.12 |
| Stage 3 low pass decomposition | 0.051 | 0.122 |
| Stage 4 high pass decomposition | 0.045 | 0.062 |
| **Total (including initialization)** | **0.944** | **1.322** |
| **Total (excluding initialization)** | **0.289** | **1.236** |

## V. CONCLUSION

Present day ultrasonic imaging applications require processing huge amount of data in real-time. Therefore, data compression becomes an essential part of ultrasonic applications to reduce the storage. This study demonstrates an efficient implementation of ultrasonic compression algorithm realized on GeForce GT 750M NVIDIA GPU using OpenCL to accelerate the computational performance. Compression is achieved by using the signal compaction properties of DWT. OpenCL program executed on NVIDIA GT 750M GPU completes the compression of 33 MBytes into 6.6 MBytes (80% compression) in 0.289 seconds, whereas the C++ program implemented on Intel (R) Core ™ i7-4500U CPU requires 1.236 seconds for performing the same operation. Furthermore, OpenCL implementation provides greater flexibility as that of C++ design, and higher computational efficiency as that of a hardware design, which makes it a better choice for real-time ultrasonic imaging applications.

## REFERENCES

[1] G. R. Lockwood, J.R. Talman, "Real-Time 3-D Ultrasound Imaging Using Sparse Synthetic Aperture Beamforming", *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 45, no. 4, pp. 980-988, July 1998.

[2] R. Sampson, M. Yang, S. Wei, C. Chakrabarti,T. F. Wenisch, "Sonic Millip3De: An Architecture for Handheld 3D Ultrasound", *IEEE Transactions on Micro*, vol. 34, no. 3, pp. 100-108, June 2014.

[3] T. Gotoh, J. Amagai, T. Hobiger, M. Fujieda, M. Aida, "Development of a GPU-based Two-Way Time Transfer Modem", *IEEE Transactions on Instrumentation and Measurement,* vol. 60, no. 7, pp. 2495 - 2499, July 2011.

[4] S. Gilliland, B. Clementine, T. Gonnot, J. Saniie, "Multidimensional representation of ultrasonic data processed by Reconfigurable Ultrasonic System-on-Chip using OpenCL high-level synthesis", *IEEE International Ultrasonics Symposium (IUS)*, pp. 1936-1939, September 2014.

[5] P. Govindan, J. Saniie, "Performance Evaluation of 3D Compression for Ultrasonic Nondestructive Testing Applications", *IEEE International Ultrasonics Symposium (IUS)*, pp, 437–440, July 2013.

[6] The open standard for parallel programming of heterogeneous systems [Online] Available: https://www.khronos.org/opencl/