10.1109/ULTSYM.2014.0481

Multidimensional Representation of Ultrasonic Data Processed by Reconfigurable Ultrasonic System-on-Chip Using OpenCL High-Level Synthesis

Spenser Gilliland, Clementine Boulet, Thomas Gonnot and Jafar Saniie
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, Illinois, USA

***Abstract*—Ultrasonic non-destructive testing has been widely used to determine the properties of materials, and more importantly their integrity. However, these techniques often require capturing massive amounts of data and intensive signal processing for processes such as image formation, analysis, characterization, classification and diagnosis. Our study is focused around utilizing a Reconfigurable Ultrasonic System-on-Chip Hardware (RUSH) platform to process ultrasonic signals in real-time. To this end, OpenCL has been utilized in the RUSH platform to provide a means to accelerate computation using the FPGA fabric while providing consistent memory and execution models to enable portability. To visualize the ultrasound data, a Graphical User Interface (GUI) for the Analysis of Multidimensional Ultrasonic data on RUSH (GAMUR) has been incorporated. GAMUR utilizes C++/QT and OpenGL to enable enhanced visualization and control features within the RUSH platform. The interface not only features the ability to view the ultrasound signals in one dimension, two dimensions, or three dimensions; but also to command and configure the hardware accelerators built using OpenCL. Therefore, the system provides a means for analyzing, visualizing and accelerating the extraction of information from multi-dimensional ultrasound data.**

I. Introduction

In industry, material quality can be of critical importance to safety and efficiency. Small cracks or fatigue in airplanes or submarines can lead to significant damage when placed under stress. Therefore, the testing of the materials from the inside out is of great importance to guarantee the reliability of the structure. However, it is neither practical nor reliable to cut a material in order to test it's integrity. So, industry relies on ultrasonic probing, a non-invasive, non-destructive technique to test materials. Submerged ultrasonic testing is commonly used to maximize the energy transmitted to the material under test. However, the principle of submerged ultrasonic probing utilizes ever increasing ultrasonic frequencies to penetrate deep into the material [1]. Therefore, these tests require high speed data acquisition techniques and signal processing that can only reasonably be achieved using System-on-Chip (SOC) design with FPGAs or ASICs.

The use of hardware design for signal acquisition and processing normally requires knowledge of languages such as VHDL or Verilog. However, recent developments in high-level synthesis let us use languages such as C and C++ to design hardware. OpenCL is a vendor neutral standard maintained by the Khronos Foundation. It was originally created as the alternative to the NVIDIA CUDA programming environment for AMD based GPUs; however, it has quickly been adopted by other vendors including Intel, Qualcomm, and Texas Instruments. Originality created for use by GPUs, OpenCL has recently started to receive attention on FPGAs by Altera [2] and Xilinx [3]. OpenCL allows a designer to portably create C programs which target various types of accelerators.

OpenCL solves a unique problem with respect to the currently available heterogeneous parallel programming APIs. At its core, the standard provides common methods for memory management, and execution control. This alleviates the need to produce a custom kernel driver for each accelerator and ensures that memory management occurs in a consistent manner across devices from many different vendors. This greatly increases the portability of the platform from one generation to the next and alleviates the need to implement several of the plumbing layers commonly required in accelerator development.

QT and OpenGL are commonly used together to produce graphical user interfaces (GUI). QT provides common two dimensional layout and windowing support while OpenGL adds hardware accelerated three dimensional rendering capabilities. Using these libraries it is possible to build advanced user interfaces that accurately portray data in both two and three dimensions.

In section II, we present the base system for the submerged ultrasonic testing, using a reconfigurable architecture for signal processing. Section III and IV detail the implementation of these algorithms and their interfacing with the microprocessor using OpenCL High-Level Synthesis (HLS). Then section V presents the GUI in charge of presenting the acquired data using multidimensional graphics. Finally, section VI concludes this paper.

II. System description

The system is composed of two distinct parts: the ultrasonic platform, RUSH, and the Graphical User Interface, GAMUR. RUSH is designed for ultrasonic testing using submerged transducers. In the experiment, shown in figure 1, the ultrasound transducer is connected to a pulsar-receiver that sends an ultrasonic burst into the material under test, and listens for echoes. In order to reliably measure the echoes, the signal from

978-1-4799-7049-0/14/$31.00 ©2014 IEEE

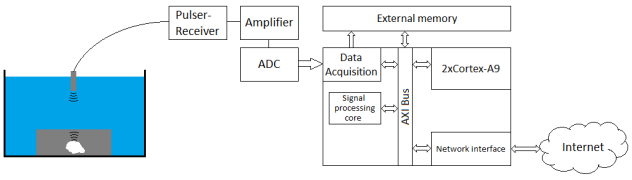

Figure 1. RUSH Experimental Setup for Submerged Ultrasound

the pulsar-receiver is amplified and fed to an analog-to-digital converter which is read by the RUSH platform [4].

The Zynq SoC [5], the center of the RUSH system, acquires the sampled data. The Zynq SoC contains a Xilinx Kintex 7 series FPGA fabric integrated with a dual-core ARM Cortex-A9 microprocessor. In our system, the signal is acquired through the FPGA fabric and processed in either the CPUs or the remaining FPGA fabric. The microprocessor runs a Linux-based OS and integrates a server for the GUI to control data sampling and retrieval. This server is custom built for each platform and sends the available options to the user interface using a custom protocol.

While most of the ultrasonic devices have an integrated display, or a separate display, this project utilizes the network to provide data to a remote or local operator. The GAMUR GUI connects to the server via TCP/IP and retrieves the platform information including the options available on the connected platform. Consequently, the data collection can be measured and managed from anywhere on the Internet.

The user is able to alter the configuration using GAMUR to select the appropriate parameters for their experiment. Once the platform is fully configured to the user's requirements, the user can perform a capture operation which will initiate the pulsar-transducer and begin the experiment. When the capture is complete, the OpenCL based accelerator can begin processing the data. Finally, the processed data is transferred and displayed using a multidimensional representation in either one, two or three dimensional formats.

III. OpenCL based High-Level Synthesis

The OpenCL kernel language is a subset of the C programming language. In general, the programming model is based on a single instruction multiple data (SIMD) execution model. In this model, a single program will be loaded by the target device. The program is then run to completion across an array of processors. In OpenCL terms this "array of processors" is called a work group, and the executable code run on the "array of processors" is called a kernel.

The basic run time cycle for an accelerator from the host's perspective is "allocate memory", "copy memory to device", "execute kernel", and "copy memory from device". The "allocate memory" method, *clCreateBuffer*, allocates a device accessible buffer in the accelerator to hold the data to be processed. The "copy memory to device" method, *clEnqueueWriteBuffer*, then copies the data from host memory to device memory. The "execute kernel" method, *clEnqueueNDRangeKernel*, then triggers execution of the accelerator. Finally, the "copy memory from device" method, *clEnqueueReadBuffer*, copies the results from the accelerator back to the host processor.

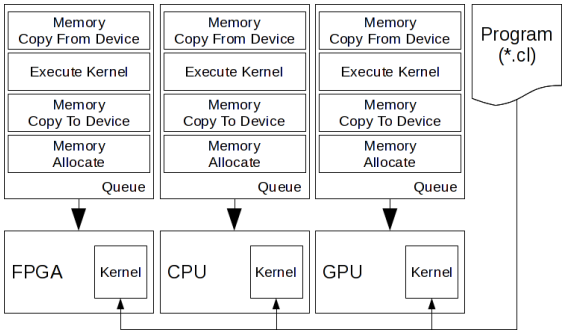

Figure 2. An example OpenCL Context with FPGA, CPU, and GPU

In order to attain the highest possible throughput, OpenCL implements the concept of execution queues as shown in figure 2. The language allows a user to create queues for each OpenCL capable device in the system. These queues are used to order operations such as memory allocation, memory copies, and kernel initiation. When a kernel is en-queued, the number of compute units requested may be larger than the number of compute units in the work group. OpenCL hides the complexity of multiplexing the range of kernel to the range of the work group from the user.

It is fairly obvious to see how OpenCL maps to common GPU based systems which use SIMD based processors. However for FPGA's, the system must map the logic defined in the OpenCL kernel to an FPGA configuration. To this end, high-level synthesis is used to transform the OpenCL program code into an RTL representation. This RTL can then be synthesized and instantiated in the FPGA using either dynamic reconfiguration or JTAG programming.

High level synthesis is an umbrella term for current technologies which will translate higher level languages such as C, and C++ to synthesizable RTL netlists in VHDL or Verilog. The authors of this paper are currently using an early access version of an OpenCL solution from Xilinx. The Xilinx OpenCL system utilizes Vivado HLS in the backend to generate RTL level representations of OpenCL program code which are synthesized, placed, and routed using the Vivado tools.

IV. OpenCL Implementations

Several applications were implemented using OpenCL. Unfortunately, no performance results can be presented until the OpenCL product is publicly released. Therefore, this section will describe the processes used in the implementations. For additional, information on the algorithms described below and performance estimates for execution on ARM, please see [6].

A. Split Spectrum Processing

Split spectrum processing was implemented on the device using a single kernel. This kernel is called multiple times by the host and performs both the filter and inverse FFT portions of the algorithm. Additionally, it offers a secondary mode that performs the initial FFT without windowing. The decision to make a secondary execution mode for the kernel instead of implementing an additional kernel reduces the time

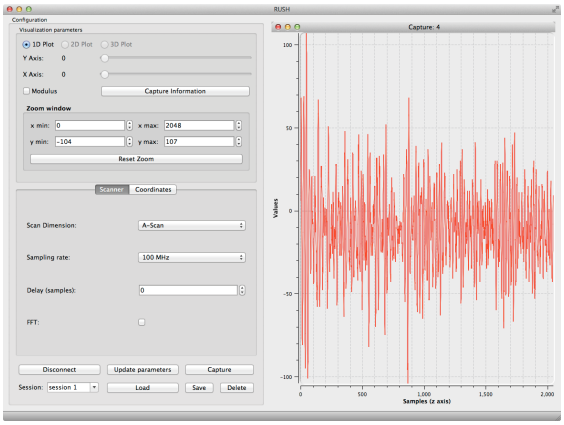

Figure 3. GUI showing one dimensional data

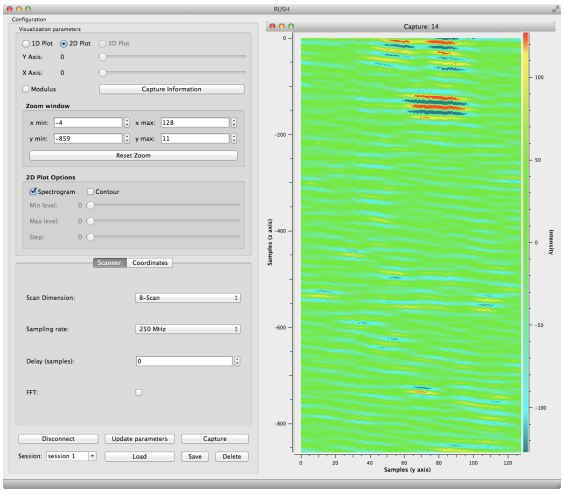

Figure 4. GUI showing two dimensional data

spent performing FPGA reconfiguration while allowing the maximum possible FFT size in the fabric.

B. Chirplet Signal Decomposition

The chirplet signal decomposition algorithm was implemented on the device using a single kernel for similar reasons as the FFT. This kernel simply generates a chirplet with specified parameters and compares the result to the original signal. The host is then responsible for altering the chirplet parameters to attain a higher correlation.

V. Multidimensional Data Representation

The GUI receives data directly from the RUSH platform via a network connection. The data can be of one, two or three dimensional and is represented accordingly. For instance, the two dimensional data can be displayed using a two dimensional plot, or the user can select a one dimensional waveform by fixing a value in the x direction. This same principle applies for three dimensional data that can be displayed as one, two or three dimensional data.

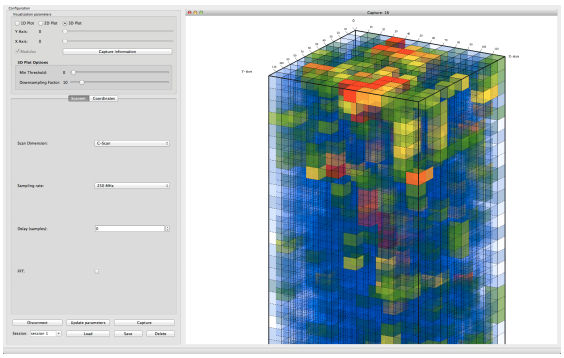

Figure 5. GUI showing three dimensional data

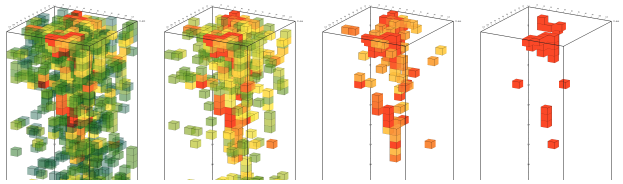

Figure 6. 3D representation of the data for different threshold values. From right to left: T=53, 71, 99, 127

A. One-Dimensional Representation

The one dimensional data usually represents the acquisition of the signal corresponding to a depth of an ultrasonic echo, called an A-scan. The result is a curve containing the amplitude of different echoes representing potential flaws along the depth of the material.

In figure 3, we can see the raw signal provided by the acquisition board. Additionally, we can see the interface with the signal processing and acquisition options of the platform. Note that only the one dimensional plot is available for one dimensional data.

B. Two-Dimensional Representation

In two dimensions, there are two ways of observing the data, a B-Scan or a C-Scan. For a B-Scan, the two dimensional data represents the echoes from different pulses along a x or y axis where the z vector represents the depth of the material. The B-scan is obtained by combining many single axis A-scans. In a C-scan, samples are shown at constant depth across both x and y axis. The C-scan requires a scan in both the x and y directions using either a two dimensional pulsar transducer array or a two-axis CNC scanner.

In figure 4, we can see an example of two dimensional plot, more specifically a B-scan. We can see the echoes in red and blue colors representing the positive and negative extremes of the signal.

C. Three-Dimensional Representation

The three dimensional data representation is designed to show the localization and intensity of the different echoes measured from the material. Arising from this representation, there is the possibility that echoes will occlude each other from the view of the user. To minimize this byproduct, the GUI makes use of the OpenGL transparency properties. Therefore,

if an echo is strong, it will show as solid red. However, if the echo is faint, it will be shown in a light blue color with representative transparency. All echoes between these extremes will be represented using the full color spectrum from red to blue, and the full range from solid to completely transparent.

In order to optimize the representation, the information needs to be simplified. This simplification requires two main operations to take place. First the data undergoes a sub-sampling process. Among the thousands of samples in an A-scan for example, only about a hundred discrete positions are displayed. The second operation is to determine the peak energy of the echo and display it as cubes, representing the energy in that location. In addition, a threshold parameter allows the user to limit the display to the echoes with sufficient energy to make the flaws more visible.

Figure 5 shows the user interface when displaying data in three dimensions. We can clearly distinguish the high echo energy areas in solid red from the lower ones in translucent blue. We can see in the interface a selection slider to change the threshold for displaying echoes. Figure 6 shows the representation of the same data for different values of the threshold.

VI. Conclusion

In this paper, we presented a complete framework for submerged ultrasonic testing. The design relies on OpenCL based high-level synthesis for the implementation and utilization of hardware signal processing algorithms and is complemented by a networked GUI application that allows multidimensional representation of the acquired data and control of the capture parameters.

References

[1] S. MacIntosh, D. N. Sinha, G. Kaduchak, "Noninvasive noncontact fluid detection in submerged containers using swept frequency ultrasonic technique," *Ultrasonics Symposium, 2001 IEEE*, vol. 1, pp. 689-692 vol. 1, 2001

[2] Altera, "Altera Opens the World of FPGAs to Software Programmers with Broad Availability of SDK and Off-the-Shelf Boards for OpenCL," available at http://newsroom.altera.com/press-releases/nr-altera-sdk-opencl-conformance.pdf

[3] Xilinx, "Software-based system realization with C/C++ and OpenCL", [Online]. http://www.xilinx.com/content/xilinx/en/products/design-tools/all-programmable-abstractions/#software-based

[4] Gilliland, S.; Saniie, J.; Aslan, S., "Linux based reconfigurable platform for high speed ultrasonic imaging," Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on , vol., no., pp.486,489, 5-8 Aug. 2012

[5] Xilinx, "Zynq-7000 All Programmable SoC Technical Reference Manual", UG585 (v1.5) March 7, 2013 [Online]. Available: http://www.xilinx.com/support/documentation/user-guides/ug585-Zynq-7000-TRM.pdf

[6] Gilliland, S.; Govindan, P.; Gonnot, T.; Saniie, J., "Performance evaluation of FPGA based embedded ARM processor for ultrasonic imaging," Ultrasonics Symposium (IUS), 2013 IEEE International , vol., no., pp.519,522, 21-25 July 2013