Platform Independent GUI for Reconfigurable Ultrasonic System-on-Chip Hardware

Clementine Boulet, Eric Rovira Ricard, Spenser Gilliland, Thomas Gonnot and Jafar Saniie

Department of Electrical and Computer Engineering

Illinois Institute of Technology, Chicago, Illinois, U.S.A.

Abstract—Ultrasonic systems are widely used in industrial and medical imaging applications for diagnosis, nondestructive evaluation (NDE), defect recognition and classification. These applications are big data problems that must be processed in real-time using computationally intensive signal processing algorithms. The Reconfigurable Ultrasonic System-on-Chip Hardware (RUSH) platform, developed for this study, utilizes hardware/software co-design to allow flexibility in system development for these real-time ultrasonic signal processing applications. In this paper, a Graphical User Interface (GUI) is designed for the RUSH platform which allows users to visualize and capture ultrasonic signals in one, two and three dimensions. Furthermore, a TCP based platform-independent client-server protocol has been established for communication between the RUSH system and the user.

Keywords—ultrasounds; graphical user interface (GUI); C++; Qt; QWT; OpenGL.

I. INTRODUCTION

Ultrasonic nondestructive signals have been widely used to determine properties of materials by sending an ultrasonic pulse through it and analyzing the measured reflections. When the pulse reaches a discontinuity within or between materials, e.g. a transition from water to metal or conversely, the wave is partially reflected back, creating an echo. Moreover, if a flaw is present in the object, then the retrieved signal will present additional echoes corresponding to the defects of the material. By analyzing various parameters of these echoes we can deduce the physical properties of the defects. In our application, we essentially detect the material properties of an object by resorting to the ultrasonic measurements. Our ultrasonic measurement system is composed of a transducer that sends ultrasonic signals into a water tank that contain a submerged object for testing and inspection. The system is designed to concurrently process an ultrasonic range of waves from 20 KHz to 20 MHz.

The retrieved signal, which is a collection of echoes, is forwarded to the Reconfigurable Ultrasonic System-on-Chip Hardware (RUSH) platform. This platform is designed to be an easy-to-use extensible System-on-Chip (SoC) for developing low power, inexpensive, real-time processing algorithms. It relies on existing standards for signal analysis and image processing [1]. The RUSH platform is a combination of firmware and software using reconfigurable FPGA, specifically designed for processing ultrasonic signals and providing internet access for remote inspections.

The ultrasonic measurements can be performed in three different scanning approaches: A-scan, B-scan and C-scan. An A-scan is a one-dimensional measurement, oriented in the depth direction (following the z-axis). A B-scan is a two dimensional capture, and thus a collection of A-scans following the y-axis. With the same logic, a C-scan (Constant depth scan) is generated from a three-dimensional measurement, as a collection of B-scan following the x-axis.

The major components of the RUSH platform are the Xilinx FPGA and the embedded microprocessor within the FPGA. The modular design structure of this platform can be easily upgraded with the latest high-performance FPGA devices as they become available. The system use Xilinx Embedded Development Kit (EDK) where the major feature of this EDK tool suite is the integration of the Microblaze processor, capable of running a Linux based operating system . All these tools allow the reuse of substantial portion of available software and drivers, such as TCP/IP protocol or IP cores. Especially, the RUSH system is theoretically capable of providing wired gigabit Ethernet communication at 125 MB/s.

Discrete Wavelet Transform (DWT) is one of the most efficient methods for compressing ultrasonic data and thus has been selected to process the retrieved RUSH data[2]. The data is oversampled to retain very fine features of the ultrasonic signal. Each A-scan is a sample of 2048 points, a B-Scan is a sample of 128 A-scans taken per line along the y-axis. Similarly, C-Scans are obtained by taking B-scans on 128 lines along the x-axis. 3D ultrasonic data compression is carried out by applying 1D-DWT for z, y, and x coordinates.

As an A-scan has significant amount of redundant information due to oversampling, in order to extensively reduce the size of A-scans, a high-order wavelet, Daubechies-10 (db10), is used to compress the signal in the z-direction. Using the Haar wavelet, some redundancies can be removed by detecting high correlations between neighboring A-scans in the y and x directions. Thus a compression of 79% in z-direction has been achieved, and in the combined y and x directions an additional 75% reduction of the compressed results [3].

978-1-4799-4774-4/14/$31.00 ©2014 IEEE

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on October 02,2020 at 06:23:35 UTC from IEEE Xplore. Restrictions apply.

II. IMPLEMENTATION

This section describes the means and the concepts used for implementing the RUSH interface.

A. Development Tools

Three major development tools have been used to built this interface: Qt and its Integrated Development Environment (IDE) , QWT and OpenGL [4-6].

Qt is a Cross-Platform application framework, widely used for developing software with a graphical user interface, and for non-GUI programs such as command-line tools and consoles for servers. The cross-platform feature is an asset to develop and run this GUI regardless of the operating system, desktop platforms (Linux, Mac and Windows), and mobile platforms (Android and iOS). Non-GUI features include network support and a unified cross-platform Application Programming Interface (API) for file handling. In addition, the capability of Qt to support third-party plugins has been used in this project. Qt Creator is a cross-platform IDE using C++, conceived to support Qt libraries natively. This IDE has been used to develop the RUSH Client and server and design the GUI.

QWT is a C++ based third party library and a graphics extension for Qt. This library contains GUI components and utility classes, which are primarily useful for programs with a technical or scientific background. QWT has been used in this project, given that it provides a framework for 1D and 2D plots as well as different layouts (scales, sliders, etc) in order to control and display values, arrays, or ranges of type double. QWT is distributed under an open-source license. Its usage is very diverse, from curve plots to histograms. It is regularly updated to ensure compatibility with the latest Qt version.

OpenGL (Open Graphics Library) is a cross-language, multi-platform API for rendering 2D and 3D computer graphics. The API is typically used to interact with a Graphics Processing Unit (GPU), to achieve hardware-accelerated rendering. In this project, this library has been used as a third party library extension for Qt, in order to render 3D graphics.

B. Implementation Approach

1) TCP/IP Model

The communication between the server and the client is based on TCP/IP, to take advantage of the benefits of using an international standard for communications that runs on most LANs, but also over the Internet. TCP/IP has been chosen over UDP in this project, given that TCP ensures packet retransmissions. We seek data consistency during the transmission, regardless of a possible packet delay introduced during retransmissions. As a layered communication model, we have focused on developing the higher layer (application layer) and taken advantage of Qt libraries to implement the lower layers, as illustrated in Fig. 1.

Fundamentally, the application layer of the RUSH client and server turns a message into a bit stream that is sent to the TCP transport layer by creating a data block. As soon as the information reaches the other side of the connection in the TCP buffer, a signal is transmitted. The size of the message is first retrieved and a subroutine is run to wait until the receiver gets the complete data block. Then this block is processed in order to extract the needed information.

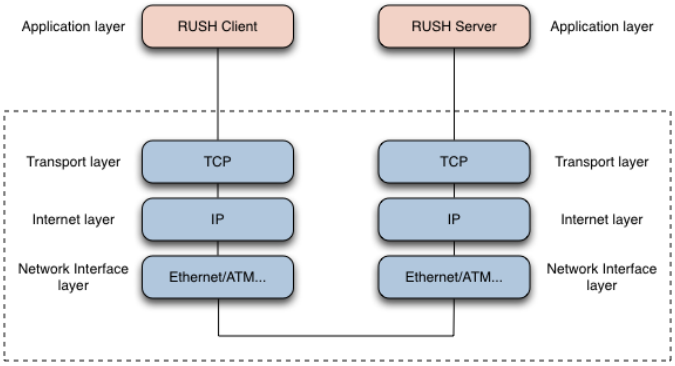

Fig. 1. TCP/IP model used for the RUSH platform

2) Data Serialization of the RUSH application layer

For every transmission or every message, a data block is created. This block is composed of any type of C++ variables to be sent. The block is serialized as a bit stream using Qt libraries, and provided to the TCP/IP transport layer [1]. To do this, when a data block is created, its size (in bytes) is appended at the top of the block. In this way, as soon as the receiver starts receiving the data block, it first retrieves the size of the block, and then, it waits for remaining bytes to be received.

C. Software Structure

1) Server and Client Interaction

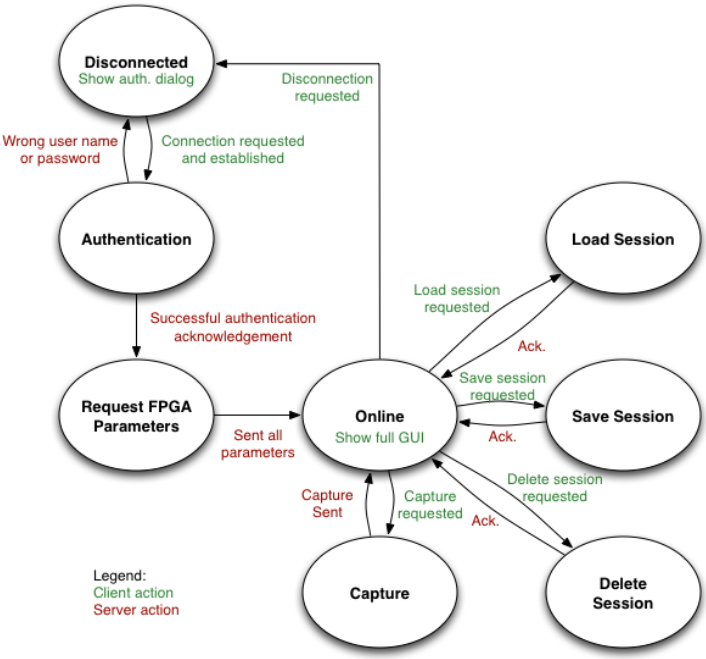

Fig. 2. RUSH GUI State Diagram

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on October 02,2020 at 06:23:35 UTC from IEEE Xplore. Restrictions apply.

The state diagram in Fig. 2 shows the design of the Client-Server interaction. The client, starting at the *Disconnected* state, follows this state diagram and the server provides the client with the information corresponding to its requests.

The client connects to the server using a username, a password, an IP address and a port. Multiple users can connect to the server simultaneously. Once connected and authenticated, the server sends all the specific FPGA parameters from the specific SoC, where the server software is running (dynamic parameters). Then, the client stays in the *Online* state, waiting for the user to load, save, or delete sessions, or to request an ultrasonic data capture. Sessions are a set of configuration values, corresponding to system-specific dynamic parameters.

2) Dynamic Parameters

As it has just been mentioned, the RUSH GUI retrieves the available FPGA parameters defined in the server (on RUSH SoC). This feature introduces the following main advantages:

- The same GUI can be connected to different SoC systems, regardless of their specifics parameters.
- The convenience for upgradeability and scalability is dramatically improved, given that adding a new SoC will not require updating the GUI in every user device.

For convenience, dynamic parameters can be defined by the developer, in a single C++ class, which has to be adapted to the specific SoC drivers. Anything else in the software requires further modifications.

III. GUI FEATURES

Fig. 3, 4, 5 and 6 show the GUI features once the user has succesfully connected and authenticated to the server (RUSH SoC). The following sections introduce the main features developed in the GUI.

A. Dynamic Parameters and Sessions

The dynamic parameters, corresponding to the specific RUSH SoC, are presented to the user in the lower left part of the GUI window, in tabs. In this case, there are 2 tabs: Scanner and Coordinates. The user can set values for the presented parameters, and save them as a session. In the same way, one can retrieve or delete a set of parameter values by just choosing the desired session name. To activate the current dynamic parameter values into the FPGA, the user has to click *Update Parameters*. The buttons *Delete*, *Save*, and *Update Parameters* are enabled only if the user is authenticated as an administrator. Otherwise, the user is only allowed to load sessions and request captures. The visualization options are performed locally and are independent of the communication with the RUSH SoC. Therefore, we have decided to make them manageable by both administrator and non-administrator users.

B. A-Scan

When an A-Scan is captured, a one-dimensional plot is created and displayed using QWT libraries, as shown in Fig. 3. The user is presented with zoom features, so he or she can better focus on a specific region of the signal. The user can choose to display the original signal or to display its absolute value, checking the option *Modulus*. This option can be sometimes more convenient to identify ultrasonic flaws.

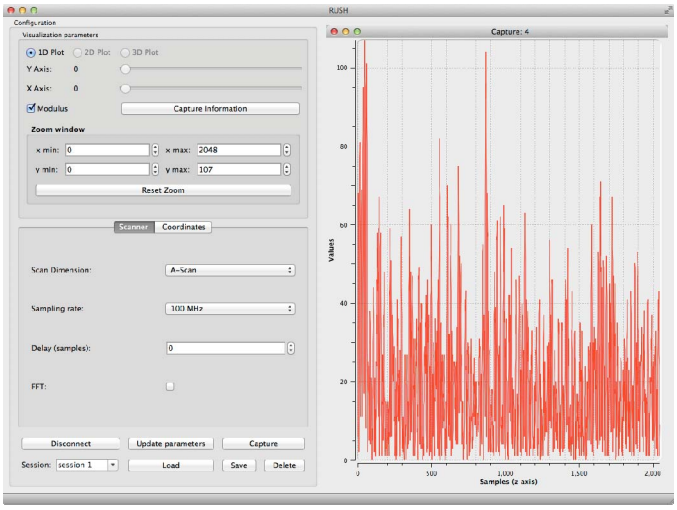

Fig. 3. RUSH GUI displaying an A-Scan

C. B-Scan

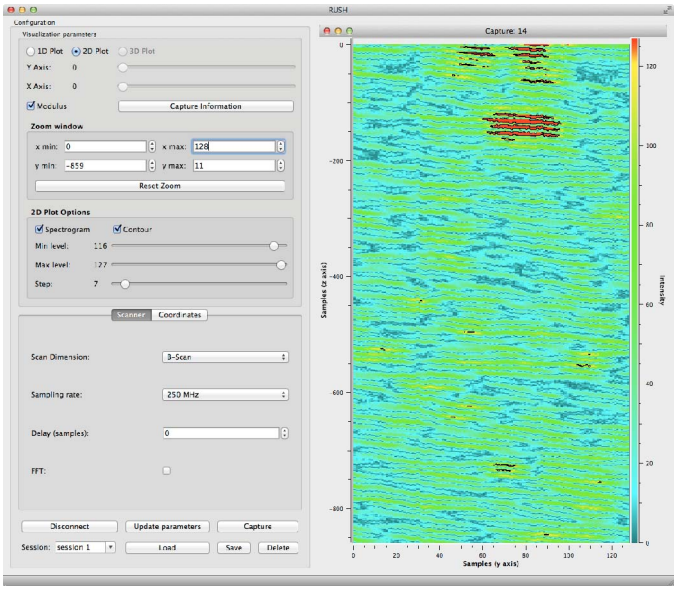

Fig. 4. RUSH GUI displaying a B-Scan

When a B-Scan is captured, a two-dimensional plot is created and displayed using QWT libraries, as shown in Fig. 4. The amplitude of the ultrasonic capture echoes are represented with colors, from blue (minimum value) to red (maximum value). The previsouly introduced features Zoom and Modulus, are also implemented for B-Scans. In addition, 2D Countour options are presented to the user. They allow plotting contour curves so that every curve corresponds to a determined value. The minimum and maximum threshold sliders allow setting the minimum and maximum value to be considered, respectively. Furthermore, the user can choose to only plot the curves

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on October 02,2020 at 06:23:35 UTC from IEEE Xplore. Restrictions apply.

without the spectrogram representation, just by disabling the Spectrogram checkbox.

D. C-Scan

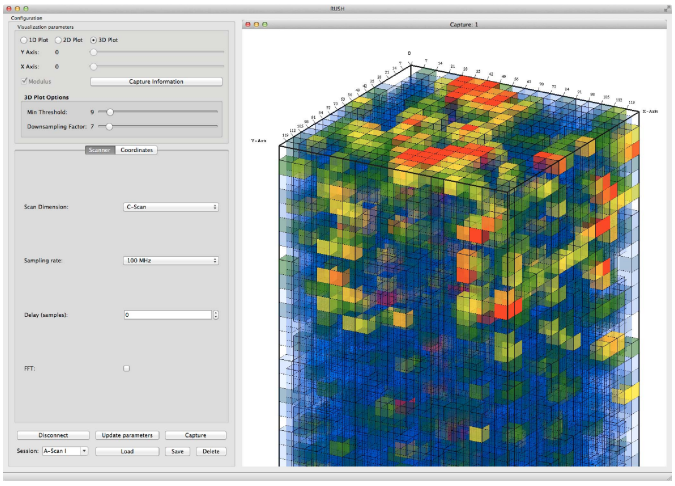

Fig. 5. RUSH GUI displaying a C-Scan

When a C-Scan is displayed, a three-dimensional spectrogram with color as the amplitude is displayed using OpenGL libraries, as shown in Fig. 5. In order to extend the concept of B-Scans into 3D, we have taken the following approach:

- Given that a 2D pixel can be seen as a square, cubes have been used to extend pixels for 3D rendering.
- In order to be able to see the inside of the scanned object and to detect high amplitudes (discontinuity of density during the progression of the wave, meaning new material or flaws), a special OpenGL feature has been used: the density. For this application, both the color and density of the pixels are set according to the amplitude: representation of values go from full transparent and blue for the lowest amplitudes, to opaque and red for the highest amplitudes.

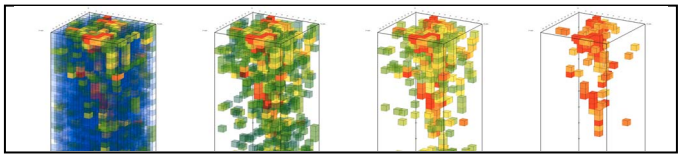

Fig. 6. The C-Scan capture shown in Fig. 5, with threshold value set to 0, 53, 91 and 99, respectively.

The C-Scan visualitazion features are completely different from A-Scans and B-Scans. One of the main features of 3D plots is that their values are always displayed in absolute value. The GUI provides the option of cut off for low amplitudes to display only the high amplitudes (flaws), by setting a threshold, as shown in the screenshots in Fig. 6.

E. Sub-Dimension Plots

In general, an N-dimensional representation can be displayed in a lower dimension. We have used this feature for the RUSH interface. When capturing a C-scan, besides displaying 3D captures, the GUI provides the option of displaying in 2D plots (collection of B-scans) or 1D plots (collection of A-scans). Similarly, the user can also display a B-scan in 1D (as a collection of A-scans).

By default, a capture is plotted in its full dimension. As soon as the user selects to plot it in a lower dimension, the axis sliders are enabled so it can move through the higher dimension.

F. File I/O and Print features

1) Open and Save

The RUSH GUI allows the user to open and save captures, from the file menu. When saving a capture, its raw data and its capture information are saved. In addition, the visualization configuration is also appended to the file. As a result, when opening a file, the user retrieves the capture exactly as it was shown when he or she saved the file. This allows the user to adjust the visualization parameters for his or her convenience, save the file, and then come at another time and open the file exactly as it used to be before saving it.

The plots themselves are not saved into the file, for efficiency. Only the raw data and desired visualization parameters are saved using the RUSH serialization. When the user sets a file name, the software makes a block of the desired variables to save and serializes them into the file. Then, when opening a file, the system retrieves the variables. In other words, it is like recovering the context of a plot at the time it was saved. Then, the software immediately generates the plot based on the retrieved variables. Thus the regenerated plot displays the same features as when it was saved.

2) Print

When the user requests to print, a standard OS printing dialog is shown which presents the current plot maximized to the paper size.

3) Export Formats

The user can export graphs in additional standard formats, such as PDF, SVG, PS and BMP. The first 3 formats benefit from the advantage of storing plots in vector format, so the resulting file size is considerably smaller than a standard image, and presents smooth edges and contours, that can adapt to any resolution.

IV. Conclusion

In this paper, the objectives of designing a Graphical User Interface to visualize ultrasonic captures of the RUSH platform along with providing the user with a convenient and intuitive way to set and manage FPGA parameter values, have been achieved. Moreover, a communication protocol has been specifically designed for this platform. The RUSH high performance hardware/software co-designed ultrasonic system has been extended by creating a fully customized GUI and its server application by using object-oriented programming approach, while incorporating scientific plot libraries and implementing full 3D rendering.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on October 02,2020 at 06:23:35 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] S. Gilliland, J. Saniie, and S. Aslan, "Linux based reconfigurable platform for high speed ultrasonic imaging", *IEEE International Midwest Symposium On Circuits and Systems (MWSCAS)*, pages 486–489, 2012.

[2] P. Govindan, S. Gilliland, A. Kasaeifard, T. Gonnot and J. Saniie," Performance analysis of reconfigurable ultrasonic system-on-chip hardware platform", *IEEE International Conference On Instrumentation and Measurement Technology*, pages 1550–1553, 2013.

[3] P. Govindan, S. Gilliland, T. Gonnot and J. Saniie, "Reconfigurable ultrasonic system-on-chip hardware (RUSH) platform for real-time ultrasonic imaging applications", *IEEE International Ultrasonics Symposium (IUS)*, pages 463–466, 2012.

[4] J. Blanchette and M. Summerfield, "*C++ GUI Programming with QT4"*, 2nd Edition, Prentice Hall Press, Upper Saddle River, NJ, USA, 2008.

[5] Uwe Rathmann and Josef Wilgen. "*QWT User's Guide*", 2013.

[6] Khronos Group, http://www.opengl.org/ , 1997-2014.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on October 02,2020 at 06:23:35 UTC from IEEE Xplore. Restrictions apply.