Performance Analysis of Reconfigurable Ultrasonic System-on-Chip Hardware Platform

Pramod Govindan, Spenser Gilliland, Alireza Kasaeifard, Thomas Gonnot and Jafar Saniie
Department of Electrical and Computer Engineering
Illinois Institute of Technology,
Chicago, Illinois 60616

***Abstract* - High-speed implementations of computationally-intensive algorithms are essential for real-time ultrasonic signal processing application development. Hardware-software co-design allows flexible system development to carry out such experiments. In this paper, we developed a reconfigurable ultrasonic system-on-chip hardware (RUSH) platform which offers a Linux based development environment by integrating FPGA firmware with specially designed software for high performance ultrasonic signal processing applications. Furthermore, the performance and resource utilization of the RUSH platform is analyzed for various ultrasonic signal processing algorithm implementations.**

I. INTRODUCTION

In this study, a reconfigurable ultrasonic system-on-chip hardware (RUSH) platform is developed to implement several time-critical ultrasonic signal processing algorithms. The major component of the platform is a Xilinx Virtex-5 (XC5VLX110T) FPGA which instantiates an embedded Microblaze processor within the FPGA firmware. This enables hardware-software (HW/SW) co-design for improved system performance. The platform provides real-time signal processing for non-destructive evaluation (NDE) and imaging applications using ultrasonic transducers with operational frequencies ranging from 20 KHz to 20 MHz. Fig.1 shows the RUSH platform setup for NDE and imaging applications. In order to analyze the performance of the RUSH platform, various algorithms are implemented as Linux Applications running on an embedded Microblaze processor configured within the FPGA. Drivers are written for the gigabit Ethernet controller and various processor features. The resource utilization and performance are studied to analyze the efficiency of the algorithms.

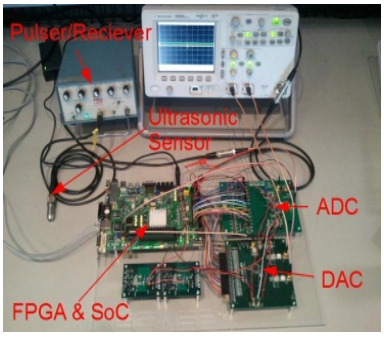

Fig. 1. RUSH system including Virtex-5 FPGA, ADC, DAC, pulser/receiver and ultrasonic transducers

This paper is organized as follows. Section II provides a brief description of the RUSH system. Section III explains the Software details of the RUSH platform. Section IV presents various ultrasonic signal processing applications implemented on RUSH platform to analyze the RUSH system performance. Section V concludes this paper.

II. RUSH SYSTEM FEATURES

The RUSH system includes a Xilinx XUPV5 FPGA development platform (Xilinx 2010), a Maxim MAX1215N ADC evaluation kit, (Maxim1215N 2012), a Maxim MAX5874 DAC evaluation kit (Maxim5874 2012) and two Maxim MAX1536 power supply evaluation kits. The interconnections of these components are shown in Fig. 2. The FPGA system-on-chip (SoC) integrates the following Xilinx IP cores: A multi-port memory controller (MPMC), universal asynchronous receiver transmitter (UART), gigabit Ethernet, system advanced configuration environment (Sys ACE) compact flash controller, timer, clock generators and a Microblaze processor. These cores are interconnected via processor local bus (PLB). In the RUSH design, the high performance signal processing algorithms are implemented on the FPGA. The RUSH system communicates with the user using TCP/IP through gigabit Ethernet.

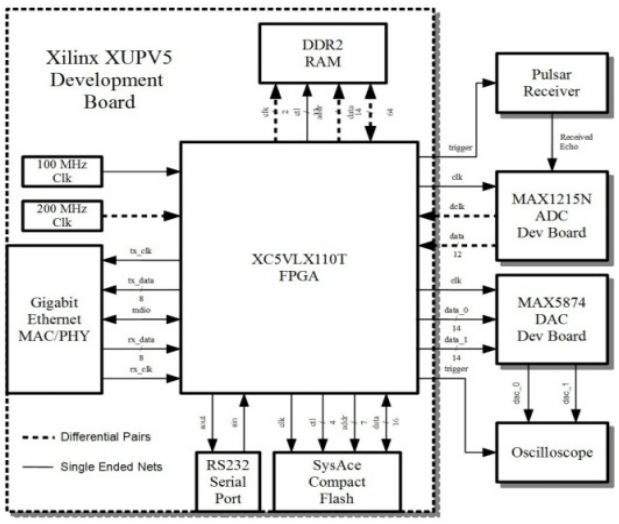

Fig. 2. RUSH block diagram showing hardware components and interconnections.

The ADC operates across a wide range of ultrasound frequencies (20 KHz - 20 MHz). At a clock rate of 250 MHz, the total throughput of the ADC is 375 MB/s. To support these high data rates, the ADC uses LVDS (low voltage differential signaling) when interfacing to the FPGA. LVDS is required because the nets between the ADC and the FPGA must be treated as a transmission line. With a measured delay of 170 ps/inch, this line creates a total skew of around 6.23 ns and limits the maximum sampling rate of the connection to 160.50 MHz. To reach the maximum operating frequency of 250 MHz, a recovered clock is provided by the ADC. This recovered clock is aligned to the data which decreases the total skew of the interface to zero. However, the clock and recovered clock are no longer in the same clock domain; thus, a clock crossing scheme is used in the FPGA when sending the data to the processor.

III. Software Design Issues

Buildroot is used to build a linux-based operating system (OS) for the RUSH system. Buildroot provides a set of makefiles and patches that make it simple to generate a complete embedded linux system [15]. Base support for the Microblaze architecture, toolchain configuration and device tree support are implemented with the help of the buildroot developers. A device tree file is a software consumable description of the hardware devices in a SoC. Xilinx provides an automated method for producing device tree files through the device tree generator available from the Xilinx git repository [14]. This method is used to provide hardware addresses and capabilities to the drivers which implement the ADC and DAC features as well as other custom IP cores. This allows the hardware and software to be more decoupled and increase flexibility in the design. For example, the system engineer can increase the maximum number of samples to be obtained by the ADC without changing any software. All software applications are built using the open source Xilinx GNU toolchain.

A. Kernel Development

A kernel driver provides the ability to subscribe to interfaces between the hardware and user land. The only interface initially provided to a driver is the init/exit interface. This is represented by two functions mod_init and mod_exit. The mod_init function is run as soon as the kernel starts loading drivers and provides the opportunity for the driver developer to subscribe to additional interfaces. The mod_exit function is run when the module is being unloaded (i.e. reboot, shutdown). In the implementation of the mod_init function for the ADC driver, a subscription to the platform device is created. When the kernel finds the ADC or DAC device, it runs the probe function where additional interfaces may be subscribed to by the driver. In the ADC driver, the probe function is used to subscribe a character interface to the device and gather information about the hardware from the device tree and hardware registers. The drivers character interface provides open, close, read, write and ioctl (input/output control) methods. The open method is used to create data buffers and set the default configuration of the device. The close method resets the device and deallocates the buffers created in the open method. The read method copies the data from the device to user space and the write method copies the data from user space to the device. The ioctl methods are unused in the current implementation. The benefit of creating the device as explained above is that multiple ADCs can be instantiated in the firmware and multiple users can use an ADC concurrently.

B. User Space Application Interface

The user space of the operating system is provided by BusyBox suite of tools. The system includes the SSH client/server Dropbear and several debugging tools. The system's user space is provided as an initramfs to the kernel and included in the downloaded kernel image. Initramfs is a file system which solely exists in the memory of the system and does not persist across reboots. To make a persistent system, the initramfs could be extracted to a flash card and mounted using the Xilinx compact flash driver. The creation of the initramfs has been automated by buildroot. Buildroot applies the needed compilation flags, configures, builds and installs the cross compiled binaries in the correct file system location and then compresses the initramfs for inclusion in the kernel.

The user applications read data from the device file which interfaces to the character interface of the ADC driver and provides the ADC output in different forms based on the application. These applications then use netcat or ssh to send data over the network to a receiving station. Secure shell (SSH) is a protocol for remote access to a system. It provides access to a shell as well as copy support through the secure copy protocol (SCP). Netcat is a UNIX utility used to transfer data using TCP from the hardware platform to an external monitoring PC. The netcat program takes data on stdin and delivers it to stdout of the receiver. This makes it ideal for simple data transfers between the RUSH system and the monitoring PC. Netcat is part of the BusyBox.

IV. Applications

The performance of the RUSH platform is analyzed by implementing three ultrasonic signal processing applications: real-time ultrasonic flaw detection using split-spectrum processing, real-time embedded chirplet signal decomposition and 3D ultrasonic data compression using the discrete wavelet transform.

A. Split Spectrum Processing

Split spectrum processing (SSP) is a method of discriminating between target echo and noise resulting from microstructure scattering. The goal is to remove the Rayleigh scattering in a system thereby increasing the target-to-clutter ratio. The target echo is hidden by a static noise produced by the stationary grain (microstructure) scatterer echoes. Therefore, to accurately detect the target,

these scatterings must be filtered from the signal. When a measurement is taken using a number of different narrowband pulses with different frequencies, the clutter echoes become randomly distributed across the pulse frequencies and can be filtered using statistical processing. In the SSP realization, a fast Fourier transform (FFT) is performed on the received broadband echo to obtain its frequency spectrum. The signal is then split by spectrum into different frequency bands by using band-pass filters. The inverse FFT is performed on the frequency bands to obtain the time domain signal for each individual frequency band. These signals now represent multiple narrowband signals split from the initial broadband signal. The narrowband signals are then post processed to reduce the clutter by employing different techniques such as averaging, minimization, order statistic filters (e.g. minimization), Bayesian classifiers or neural networks [2]-[6].

The SSP implementation on the FPGA uses a basic one-zero windowing algorithm to split the spectrum and pass-through absolute minimum post processing to improve the visibility of target echo in presence of clutter. Xilinx radix 2-Lite FFT IP Cores (FFT 2012) are used to allow a highly area efficient FFT/iFFT implementation. At a transform size of 4096 points, this IP Core has a latency of 57,539 cycles. It uses only 2 DSP48 components, 7 BRAMs, and approximately 250 slices. This low resource usage allows one FFT core and twelve iFFT cores to be implemented in tandem with the other components of the RUSH. A maximum of 12 channels at 4096 samples are possible given the DSP48 resources available on the FPGA. Table I gives the resource usage for a varying number of channels. The total execution time for 4, 8 or 12 channels is found to be 16 ms for a dataset of 2048 samples due to the parallel implementation style of the design.

TABLE I. RESOURCE USAGE WITH SSP

Resource	Available	Base	4 Ch.	8 Ch.	12 Ch.
Slices	**17,280**	6,743	8,612	10,279	11,326
BRAMs/FIFOs	**148**	43	66	81	97
DSP48	**64**	6	23	41	59

B. *Chirplet Signal Decomposition*

Chirp pulses are often used for ultrasonic imaging as an effective technique for NDE and flow measurement applications. Chirp echoes characterize dispersive media, and are also very effective for pulse compression and improved echo detection. However, chirplet analysis is a challenging problem in terms of computational complexity and susceptibility to noise. An application of chirplet analysis is characterizing defects in large grain dispersive materials where interference and masking of ultrasonic flaw echoes is often caused by grain scattering. Chirp echo parameter estimation and signal decomposition are generally performed using various time-frequency transformations combined with maximum likelihood estimation (MLE) algorithms [10]-[12]. Consequently, efficient techniques to decompose these complex signals are highly desirable for flaw detection and material characterization. When using chirplet excitation in NDE and flow measurement applications, the reflected echo will also be a transformed chirplet influenced by the property of materials or the velocity of targets within the object under inspection. The implemented algorithm is based on the successive ellipse fitting method (EFM) in the time-frequency (TF) plane using short time Fourier transforms (STFT) [13]. Five parameters, i.e., time of arrival, center frequency, chirp rate, amplitude and bandwidth factor are estimated by template matching the STFT of the signal. The fitted ellipse's parameters are obtained by solving a system of linear equations, which have lower computational complexity than that of conventional parameter estimators such as iterative maximum likelihood estimators. A real-time implementation of the EFM algorithm is embedded in the Virtex-5 FPGA in order to decompose ultrasonic experimental signals consisting of many interfering echoes acquired from a steel block using a 5 MHz transducer at 100 MHz sampling rate. The decomposition has been successfully performed in the presence of measurement noise and interference from microstructure scattering. Three schemes have been implemented as listed below:

Scheme 1: Software implementation on Microblaze
Scheme 2: Software implementation on Microblaze with floating point unit
Scheme 3: Hardware/ software co-design including FFT and look-up table IP-cores

The execution times for the scheme 1, 2 and 3 are 1660 ms, 550 ms and 82 ms respectively. From Table II, it is clear that the device resource usage for scheme-3 is almost comparable with other schemes, but scheme-3 architecture is able to achieve an enhanced performance.

TABLE II. RESOURCE USAGE WITH CSD

Resource	Available	Used (Scheme 1)	Used (Scheme 2)	Used (Scheme 3)
Slices	**17,280**	1,930	2,622	2,344
BRAMs/ FIFOs	**148**	80	80	90
DSP48	**64**	-	-	12

C. *Discrete Wavelet Transform based 3D Compression*

Discrete wavelet transform (DWT) is one of the most efficient methods for compressing ultrasonic data. A volumetric image of 128*128*2048 ultrasonic data samples is used for the compression analysis. This resembles a 3D block of data as shown in Fig. 3. This experimental data is generated using a steel block test. A 2 inch by 2 inch surface of a steel block is used for this test. Data are sampled at a rate of 100 MHz by using a 5 MHz transducer.

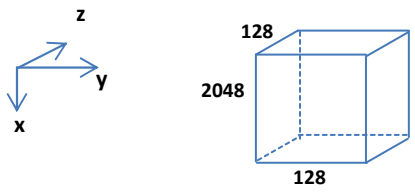

Fig. 3. 3D block of ultrasonic data.

This oversampling is done to retain very fine features of the ultrasonic signal. 128 A-scans (each A-scan is a sample of 2048 points) are taken per line (y-direction). A-scans are taken on 128 such lines (z-direction).

3D ultrasonic data compression is carried out by applying 1D-DWT for x, y and z coordinates. A-scans will have a significant amount of redundant information due to the oversampling. To improve the compaction of the A-scans, a high-order wavelet Daubechies-10 (db10) is used to compress the signal in the x-direction. In the x-direction, the decomposition is done for the low pass (L) components and selected high pass (H) components that have higher energy distribution as shown in Fig.4. By analyzing the high correlation between neighboring A-scans in y and z directions, it is clear that there are some redundancies between nearby A-scans, which can be removed. To reduce the computation time and hardware requirements, a simple Haar kernel is used for DWT in y & z directions.

The DWT coefficients {LLLL, LLLHL, LLLHH, LLHL, LLHH, LH, H} for one particular A-scan is plotted in Fig. 5. It can be seen that H, LH & LLHL have very little energy and thus can be discarded so as to achieve 79% compression in x-direction. Applying Haar wavelet in y & z direction adds an additional 75% reduction of the compressed results from x-direction because of the low energy content of H & LH coefficients. Thus a total of 98.7% compression is obtained for the 3D ultrasonic data. For achieving this level of compression, the 3D compression algorithm implemented in the Virtex-5 FPGA system takes 210 ms to compress ~33 Mbytes of data into ~1.0 Mbytes [8]. The FPGA utilization for implementing 3D compression system is only about 30% [8].

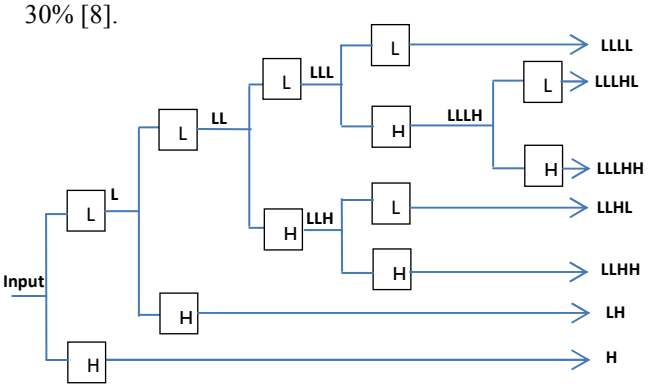

Fig. 4. Wavelet decomposition in x-direction

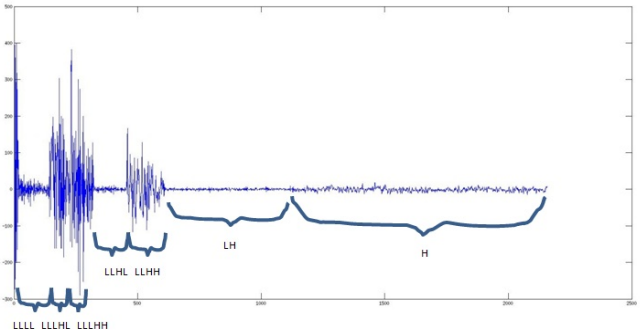

Fig. 5. DWT coefficients for one A-scan

V. Conclusion

Executing ultrasonic signal processing algorithms on the RUSH platform helps to enhance the computational performance. Furthermore, the RUSH platform allows parallel and pipelined implementations using FPGA resources. Finally, the HW/SW co-design increases the efficiency of the system by partitioning the computational load between the FPGA hardware and the embedded processor.

References

[1] J. Saniie and E. Oruklu, "Introduction to Special Issue on Novel Embedded Systems for Ultrasonic Imaging and Signal Processing", *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 59, no. 7, pp. 1329-1332, July 2012 (invited paper).

[2] J. Saniie, E. Oruklu and S. Yoon, "System-on-Chip Design for Ultrasonic Target Detection Using Split-Spectrum Processing and Neural Networks", *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 59, no. 7, pp. 1354-1369, July 2012.

[3] J. Saniie and D. T. Nagle, "Analysis of Order-Statistic CFAR Threshold Estimators for Improved Ultrasonic Flaw Detection", *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 39(5), pp. 618-630, 1992.

[4] J. Saniie, K. D. Donohue, D. T. Nagle and N. M. Bilgutay, "Frequency Diversity Ultrasonic Flaw Detection using Order Statistic Filters", *Proceedings of IEEE Ultrasonics Symposium*, pp. 879-884 vol.2, 1988.

[5] J. Saniie, D. T. Nagle and K. D. Donohue, "Analysis of Order Statistic Filters Applied to Ultrasonic Flaw Detection using Split-Spectrum Processing", *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 38(2), pp. 133-140, 1991.

[6] J. Saniie, T. Wang and X. Jin, "Performance Evaluation of Frequency Diverse Bayesian Ultrasonic Flaw Detection", *The Journal of the Acoustical Society of America*, 91, pp. 2034-2041, 1992.

[7] Y. Lu, E. Oruklu and J. Saniie, "Fast Chirplet Transform With FPGA-Based Implementation", *IEEE Signal Processing Letters*, 15, pp. 577-580, 2008.

[8] C. Desmouliers, E. Oruklu and J. Saniie, "Adaptive 3D Ultrasonic Data Compression using Distributed Processing Engines", *Proceedings of IEEE Ultrasonics Symposium* , pp. 694 – 697, September 2009.

[9] Xilinx, 2012, Platform Specification Format Reference Manual v13.4, http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/psf_rm.pdf

[10] R. Demirli and J. Saniie, "Model-based estimation of ultrasonic echoes, part I: analysis and algorithms ", *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 48, no. 3, pp. 787-802, May 2001.

[11] R. Demirli and J. Saniie, "Model-based estimation of ultrasonic echoes, part II: analysis and algorithms ", *IEEE Trans. on Ultrasonics Ferroelectrics and Frequency Control*, vol. 48, no. 3, pp. 803-811, May 2001.

[12] Y. Li, H. Fu and P. Kam, " Improved, Approximate, Time-Domain ML estimators of chirp signal parameters and their performance analysis ", *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1260-1272, April 2009.

[13] A. Kasaeifard, J. Saniie and E. Oruklu, "Chirplet Parameter Estimator Based on Ellipse Fitting in Time-Frequency Distributions for Ultrasonic NDE Applications", *Proceedings of IEEE Ultrasonics Symposium,* pp. 2024 – 2027, *Oct* 2010.

[14] http://wiki.xilinx.com/using-git

[15] http://buildroot.uclibc.org/