FPGA-Based Configurable Frequency-Diverse Ultrasonic Target-Detection System

Joshua Weber, *Student Member, IEEE*, Erdal Oruklu, *Senior Member, IEEE*, and Jafar Saniie, *Fellow, IEEE*

***Abstract*—In this paper, we present field-programmable gate-array (FPGA)-based configurable architectures that are able to perform frequency-diverse target detection for real-time ultrasonic imaging. Three design methodologies are explored including the execution of the detection algorithm on an embedded microprocessor, the creation of a dedicated hardware solution, and the use of hardware/software codesign principles. In addition to the design flow, this paper presents the impact of parameter changes on the detection-algorithm performance and FPGA implementation results. Experimental studies show that the proposed configurable systems are able to meet real-time operation requirements, and the algorithm performs robustly.**

***Index Terms*—Acoustic applications, field-programmable gate arrays (FPGA), reconfigurable architectures, signal processing, system-on-a-chip (SoC).**

I. Introduction

ULTRASONIC target (flaw) detection has applications in many industries ranging from manufacturing, medical imaging, maintenance, and nondestructive testing [1]. To achieve target detection, high scattering-noise (clutter) echoes must be suppressed by signal-processing methods. In addition, real-time requirements place tight constraints on computation time and increase communication bandwidth needs for the capture, processing, analysis, and presentation of data.

The system architecture presented in this paper takes advantage of well-established flaw/target-detection algorithms, specifically, split-spectrum processing (SSP) [2]. The SSP algorithm is based on the subband-decomposition technique and contains many parameters (such as number of bandpass filters, filter bandwidths, and locations) that strongly impact its performance with different data sets. Through the configurable implementation of the real-time SSP processing unit, the flexibility of the design to the changes in SSP parameters can be maximized, and all the requirements can be met. Challenges to overcome include balancing its high computational complexity with the need for a high-performing real-time system in a compact cost-effective unit.

Embedded-processing techniques are rapidly developing and making a large impact in many fields, including industrial electronics. Specifically, field-programmable gate-array (FPGA) devices facilitate fast development time and adaptable architectures for signal-processing applications in many domains [3], including ultrasonic testing and measurements [4]. FPGA devices have been used in chirplet-based signal processing [5], for autonomous robot navigation [6], neural networks [7], industrial control systems [8], [9], microelectromechanical-system prototyping [10], and for fault detection and classification [11]. Within the field of ultrasonics, FPGA use is becoming more prevalent, with studies demonstrating their use in multimode techniques for reducing scanning time [12] and in pulsed-Doppler-based flow measurements [4]. In [13], a hardware platform was presented to demonstrate the feasibility of FPGA-based SSP applications. In this paper, configurable architectures are designed using multiple embedded-processing methodologies. All architectures follow the principles of a system-on-a-chip (SoC) development. SoC provides a high-performance design through integration of all processing components onto a single chip. This allows high computational densities and improves system performance by reducing the communication overhead between processing units. Furthermore, a hardware/software (HW/SW) codesign methodology is employed. This methodology combines the design of both the hardware and software subsystems into a unified synergistic design approach [14] and emphasizes configurability. The system flexibility is presented in two forms. The first is through a highly modular design. With the emphasis on decomposition of the algorithm to independent processing blocks, a flexible system can be created. By increasing or decreasing the total number of processing units or by replacing individual units, the impact of each change on the overall system can be evaluated in a quick and robust manner. The second core flexibility is the creation of a control channel to the host computer. Through this communication channel, the host computer can configure many parameters of the design on the fly (during runtime). This allows the system operator to tailor the parameters to the current conditions. These two forms of flexibility allow the user to evaluate the parameter impact on the results, both in performance of the algorithm and in resource utilization. Therefore, the design space can be optimized through these interactions to meet all requirements while, at the same time, minimizing resource usage.

Manuscript received May 30, 2008; revised March 30, 2009; accepted July 2, 2009. Date of publication September 1, 2009; date of current version February 11, 2011.

The authors are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: jweber8@iit.edu; erdal@ece.iit.edu; sansonic@ece.iit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2009.2030214

II. SSP

Ultrasonic target detection is made difficult by the presence of high scattering microstructure noise. The use of frequency-diverse ultrasonic testing provides signal-echo data containing high amounts of statistical variation in scattering noise. This scattering noise is the result of a large number of small

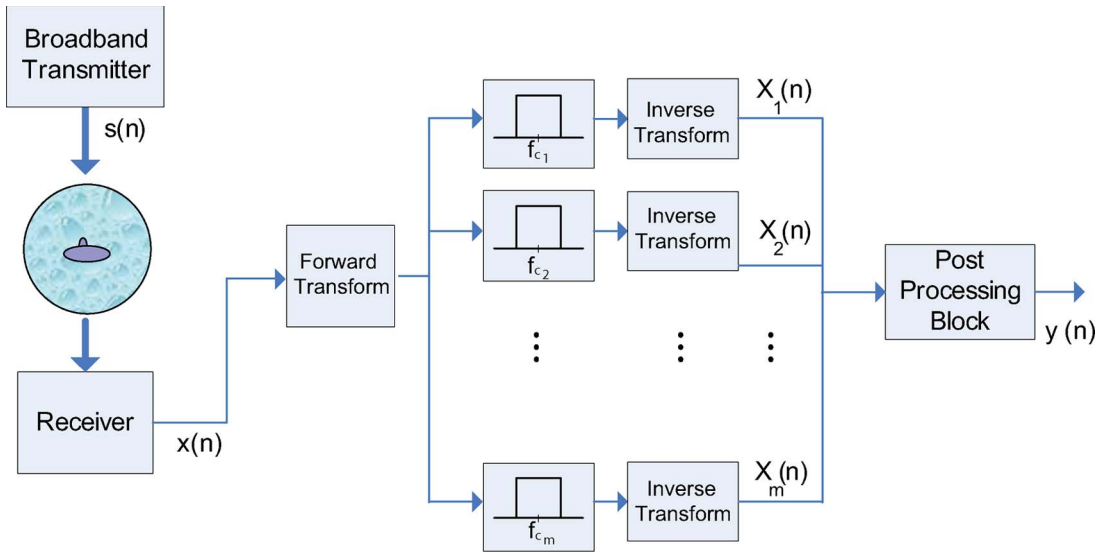

Fig. 1. Split-spectrum processing system overview.

randomly distributed scatters arising from the microstructure of the material.

As this is a common problem for target detection in many fields, many techniques have been researched to provide suppression of this scattering noise while retaining target-echo information [15]–[17]. Signal averaging has been proposed but was demonstrated to perform weakly when the scatters are stationary. Additionally, since scattering noise and target echo contains very closely related frequency responses, techniques to apply bandpass filtering have also been ineffective.

Even though standard bandpass filtering is ineffective, it is possible to achieve decorrelation of scattering noise from target echoes through the use of a shifting-frequency transmitter/receiver. Research was performed into this area of frequency-diverse target detection and further refined by research into techniques of postprocessing utilizing Bayesian and order statistics postdetection processors.

When the transmitted wavelength of the ultrasonic signal is larger than the microstructure of the material under test, the echoes exhibit Rayleigh scattering. These clutter echoes exhibit a large degree of randomness in amplitude and sensitivity to frequency shifts. On the other hand, targets (flaws) are most often much larger than the transmitted wavelength. Correspondingly, the echoes are less susceptible to frequency shifts. The SSP algorithm uses this fact to achieve decorrelation between the target echo and clutter noise. The algorithm, as shown in Fig. 1, works by decomposing the wideband input signal into a series of overlapping narrow subbands. These subbands contain information about the target echoes and a subset of the scattering noise. Due to the fact that clutter noise is highly frequency sensitive and target-echo information is insensitive, the subbands clutter noise will exhibit high degrees of randomness, while the target information will remain relatively constant across subbands. These subbands are then combined back in a postprocessing unit utilizing Bayesian [17] or order statistics [2], [18], [19]. Within order statistics, it is shown that an absolute minimizer is able to achieve substantial performance improvements.

The choice of the narrow subbands in the SSP algorithm is very important for the performance of the algorithm. It is essential to have subbands that are wide enough to contain a full set of target-echo information. However, the overlap between subbands should be kept minimum for high decorrelation. The choice of bands has to be balanced with respect to the minimizer postprocessor. If any subband contains null target information (where the band placement results in a no-target information), then the output performance will be significantly degraded.

The subbands are placed as a series of overlapping bandpass filters. Their placement is controlled by three primary parameters, starting-frequency offset, subbandwidth, and step. N filters of width W are placed, each starting at

$$\text{Offset} + (n-1) \times \text{Step} \quad \text{for } n = 1, \ldots, N.$$

The performance of the algorithm is also dependent on the number of subbands used, with a trend of higher flaw-to-clutter (FCR) ratios for more subbands. However, more subbands also indicate significantly more computation time.

For SSP, the most important performance metric is the FCR ratio, and it is used to judge the overall performance of the algorithm. An example SSP implementation result of ultrasonic experimental data is shown in Fig. 2. The results (typically $>$ 10-dB improvement) demonstrate the ability of the SSP algorithm to perform flaw detection robustly even when the input FCR is very poor.

III. System Implementation

A. System Overview

A test system was designed and implemented to process incoming data utilizing the SSP algorithm. An overview of the test system is shown in Fig. 3. The system is composed of three major components. The first is a host computer, providing control of the system to the user and presentation of all the data and results. Running on this PC system is an application software developed to enable communication with the hardware

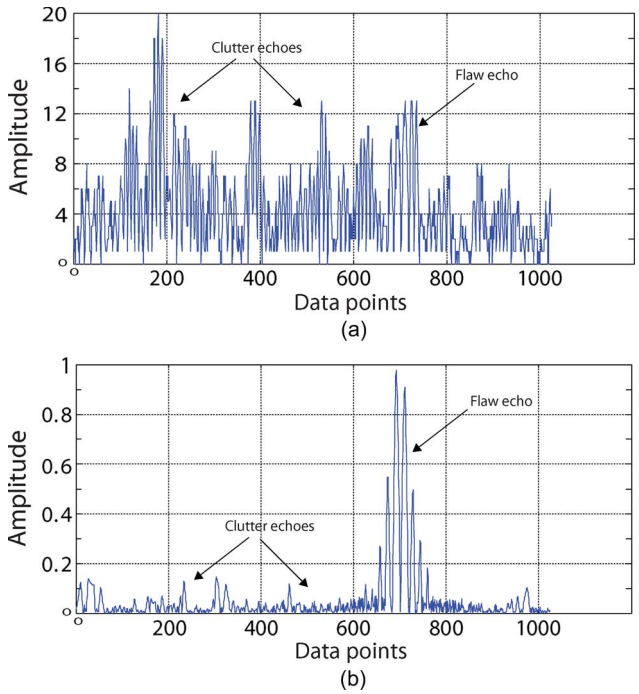

Fig. 2. FCR improvement using SSP algorithm. (a) Ultrasonic experimental data from a steel block. (b) FCR improvement using SSP algorithm.

components. The next component is an XtremeDSP hardware development board produced by Nallatech [20]. This board combines a Xilinx Virtex 4 FPGA with dedicated analog-to-digital converter (ADC) and digital-to-analog converter (DAC) chips into a convenient package. The final component is the transducer and pulse generator. The pulse generator takes in a low-voltage TTL level signal and generates the high-voltage pulse to fire the transducer. In addition, it provides preprocessing of the reflected echo signal.

B. Hardware Realization

1) Hardware Overview: The hardware design facilitates increased modularity. The modularity gives the ability to quickly modify the design to support the many parameter changes necessitated by the SSP algorithm. As the first step for achieving this goal, the system tasks are broken into three tasks: communications, signal processing, and signal capture.

The modules for the system have been designed in such a manner that they act independently of each other. They are all clocked from separate clock domains, allowing very fine grained control of clocking to each part of the design, utilizing asynchronous communication techniques such as first in–first out structures and dual-port RAMs. In addition, they do not make assumptions about the other modules within the system. This allows the modules to be highly reusable. Any module can be redesigned or changed in the system without having to make structure changes to any other part of the design. This, furthermore, isolates all timing constraints to each individual module. All modules can be built and designed independent of other module clock constraints. This technique allows for quick reconfigurations of the system to test individual parameter impacts, allowing isolated module changes to be examined.

All three modules are packaged in a Xilinx Virtex 4 FPGA [21]. This FPGA is provided on the Nallatech ExtremeDSP development kit as the main user FPGA. As shown in Fig. 4, this board also provides other components used in the system. They include dedicated ADC and DAC chips, fixed and programmable oscillators, a dedicated Virtex-II FPGA for clock management, and a dedicated Spartan-II FPGA for Peripheral Component Interconnect/Universal Serial Bus (PCI/USB) communications control.

2) Data-Acquisition Unit: At the front end of the system is the data-acquisition unit. This is composed of ADC chips which capture the incoming data from the ultrasonic transducer. The ADC allows continuous data acquisition at 14 b of precision and 105-MHz sample rate. Furthermore, it has a dynamic range of ± 1 V, giving the ability to resolve very small signal changes. The incoming signal data is captured using a single ADC chip. The sampled data is the echo response from the pulsed firing of a transducer with a center frequency of 5 MHz with a 2.5-MHz 3-dB bandwidth.

3) Signal-Capture Module: The signal-capture module controls the firing of the transducer and capturing of all echo data coming in from the dedicated ADC chips. It also provides a level of preprocessing, by adding a configurable amplifier to the incoming data. It is important to note that the clock domain in this module is separate from the data-processing element. The result of this is that the sampling rate of the device is independent of the clock rate of the processing unit. This enables the optimization of the clock rate and the sample rate independently for maximum performance.

4) Communication Module: The communication module provides an interface to the host PC and oversees all communications. It provides two primary services. The creation of a register file and the ability to access those registers through a memory-mapped interface and a direct-memory-access (DMA) interface that can be easily connected to internal block RAMs (BRAMs). It accomplishes this through the aid of a separate dedicated FPGA component. This FPGA provides all the PCI or USB interfacing requirements, simplifying the design to a more manageable interface. The design of the register file is independent of the overall system. Registers can be added or removed through small code changes. In addition, registers can be made read or write only with minimal code impact.

In addition to direct memory-mapped access is support for DMA requests. The system provides 16 separate channels for DMA access. These 16 channels can be connected to any internal BRAM within the FPGA. This is very beneficial in debugging and verification of the system.

5) Signal-Processing Module: This module implements the SSP algorithm. It performs all the transformations and computations to produce the final output data. It gathers its input data from the signal-capture module and obtains parameter values from the communication module. As data flow into the module, it is first transformed to the frequency domain through a fast-Fourier-transform (FFT) module. The FFT module used is a radix-2-based intellectual property (IP) core provided by Xilinx [22]. This module is highly optimized by Xilinx to take full advantage of the dedicated DSP blocks within the Virtex 4 device. As a design-configuration parameter, FFT operation is

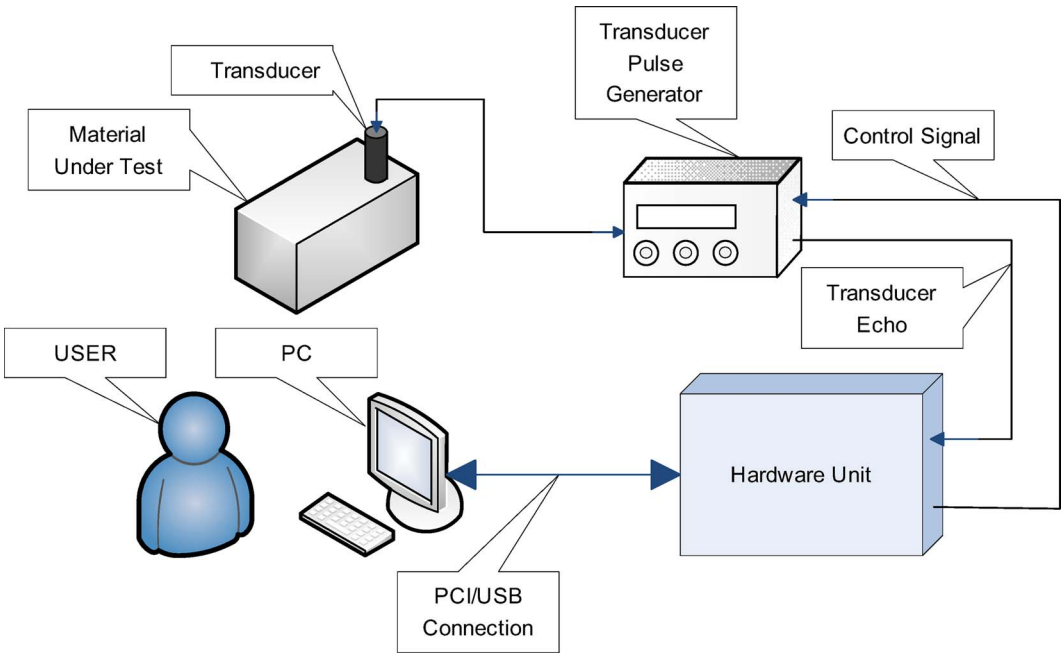

Fig. 3. Configurable frequency-diverse ultrasonic target-detection system.

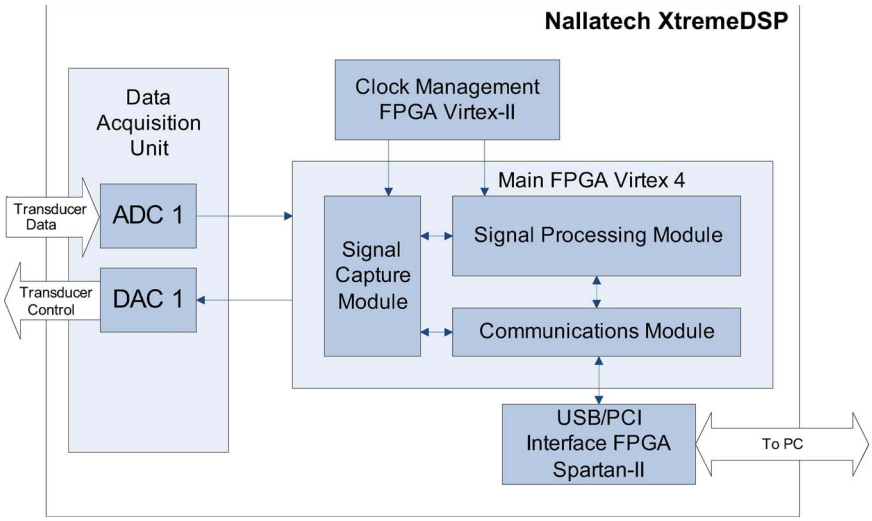

Fig. 4. Configurable frequency-diverse ultrasonic target-detection hardware.

implemented with an 8-b input precision. The data is allowed to dynamically grow to a 19-b output, which is then truncated to 16 b to be passed on to the rest of the design. The output of the FFT module is then decomposed by narrow-band filters. The filters are dynamic in their set points. Both the center frequency and bandwidth of the filters are controlled by user-writable registers in the communication module. This allows the user to change the narrow-band-filter parameters without reconfiguration of the FPGA device. These filter parameters are chosen to optimize the subbands. They are selected to cover the entire bandwidth of the echo response, with suitable offsets to decorrelate noise information. For the four-channel implementation, we utilize filter windows each covering approximately 3.5 MHz with a frequency step of 0.4 MHz for separation. The subband channels are then transformed through the use of inverse FFT (IFFT) modules. Since all subbands are independent, parallel IFFT modules are used to increase performance. The IFFT modules, like the FFT module, are highly optimized Xilinx IP cores. They take the 16-b input and allow it to grow to 27 b. The resulting data are then processed without further truncation. By eliminating truncation, a very high dynamic range is achieved. This helps the hardware design to perform very closely to the ideal software representation.

As a final step, the subbands are combined into the final output. The system currently uses an absolute minimum post-processor. This processor takes as input the time-domain sub-band signals and chooses the absolute minimum for each discrete time value. It has been shown in prior work that this technique is simple to implement and allows for robust FCR performance [18].

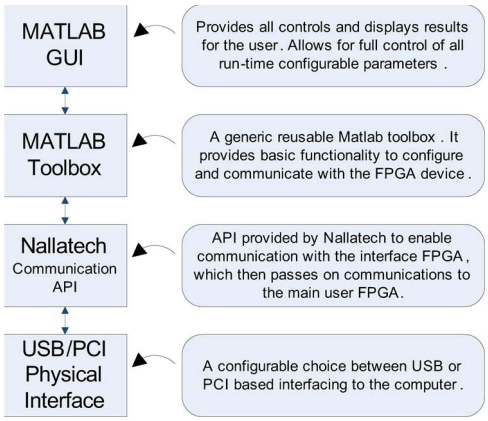

Fig. 5. User-interface software stack.

6) Application Software: The application software, as shown in Fig. 5, has been developed to work in conjunction with the hardware. It primarily acts as a user-friendly graphical user interface. This allows the configuration and programming of the FPGA, control of all system functions, support for modification of all runtime configurable algorithm parameters, and support for display and analysis of results.

In order to maintain the modularity and reusability, the software package has been designed in a layered manner. At the most basic level is the PCI/USB communication drivers, controlled by the PC operating system. On top of that is the communication application programming interface (API) developed by Nallatech. This provides support for basic communications of two types between the host PC and the hardware components. It provides memory-map register access and 16 DMA channel access to the internal memory blocks.

To enable a simple interface to the hardware, a new MATLAB toolbox has been developed. This toolbox acts as a wrapper around the Nallatech-developed API. This provides a more convenient design for the most common communications between the host PC and the hardware. Furthermore, the toolbox can be easily updated for accelerating the development of new applications.

In the following sections, a full hardware implementation is compared with various HW/SW-partitioning schemes.

C. Software Realization

1) Software Overview: This approach takes advantage of the rapidly increasing computational power of embedded processors, in particular, the soft-core processors implemented on FPGA logic. The architecture continues to focus on modularity and design flexibility. It is able to achieve these goals by implementing the entire algorithm in C code which is then executed on an embedded Microblaze processor. By implementing the algorithm in the software, developers can utilize many existing software-engineering optimization techniques to achieve performance gains. An overview of the architecture used is shown in Fig. 6.

The architecture was implemented using the Xilinx embedded development kit (EDK). This allows an integrated tool chain for developing embedded systems on FPGA devices. The core of the design is the embedded softcore Microblaze processor. Connected to the microprocessor through individual local memory bus are BRAMs and BRAM controllers acting as both data and instruction memory. While locating the memory on chip does place sharp constraints on memory use, it allows high-performance access to the memory with no need for complex offchip memory controllers or cache structures. The processor has access to a number of peripheral devices connected via the on-chip peripheral bus (OPB). Connected are a debug port, providing serial-based user interaction via JTAG, and a timer peripheral providing timer counters for profiling and performance analysis.

2) Embedded Microblaze Processor: The Microblaze processor in Virtex-4 FPGA is a highly configurable 32-b soft-core processor, programmed to operate at 100 MHz. In addition to the core processor functionality, a set of processor peripherals are instantiated to support the design. These include both internal BRAMs for instruction and data caches, a JTAG debug module to support standard I/O between the user-controlled PC, and a timer module to provide support for code-execution profiling.

As a demonstration of the configurability of the processor and flexibility of the architecture, three implementations were created. The first uses a very basic minimum-resource implementation, the second adds on a floating-point unit (FPU), and the third implementation uses a hardware FFT accelerator. The rapid creation of these variations demonstrates the ability to make architectural changes to a subsystem component while evaluating the whole system performance.

3) Software Implementation: An important challenge for embedded implementations is the need for careful memory management. Since an embedded processor has very limited memory resources, providing only 256 kB of on-board BRAM memory, careful memory allocation is the key to achieve top performance. To reduce memory requirements, all calculations on the data set are performed in place, and a straightforward implementation of the Cooley–Tukey decimation-in-time FFT has been used [23].

This paper also explored the impact of data precision on the algorithm. Two designs were produced, one using double-point precision and another using only a single-point precision. Through testing, it has been found that both designs perform equally. The increase in precision from single to double fails to show up in the final system results. Inspection of the results show that losses from lower single-precision calculations are much smaller than the noise floor of the signal itself and, hence, have no noticeable impact on overall performance.

D. HW/SW Codesign

1) HW/SW Codesign Methodology: In order to combine the advantages of both the hardware- and software-based architectures, an HW/SW codesign methodology has been utilized. This codesign approach integrates the design flow to achieve a more robust design which can comfortably meet the system requirements. The hardware system and the software application are developed jointly. The focus is to design the system from a top–down approach, and the partitioning of the system

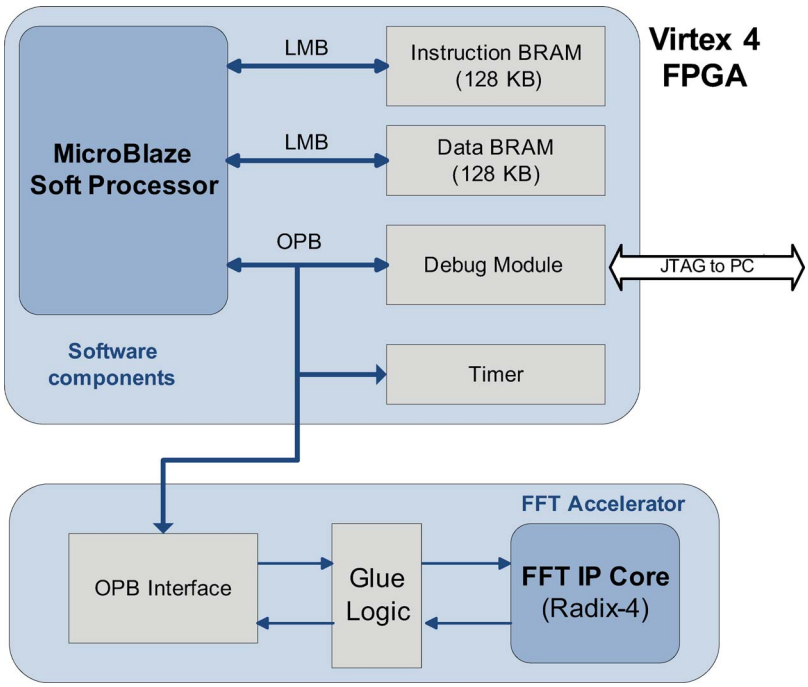

Fig. 6. Software components and HW/SW interface.

TABLE I
FPGA PROCESSING TIME OF THE SSP ALGORITHM

Algorithm Stage	Software Without FPU (Cycles)	Software With FPU (Cycles)	HW/SW Codesign (With FFT accelerator) (Cycles)	Hardware (Radix-2 FFT) (Cycles)	Hardware (Radix-4 FFT) (Cycles)
FFT	28,764,300	599,026	3,456	5,190	1,322
Window Filtering	98,884	99,153	91,456	1,024	1,024
IFFT	109,978,539	2,396,039	13,824	5,190	1,322
Post Processing	1,086,792	66,169	196,661	1,024	1,024
Total cycles	140,540,767	3,772,536	305,397	12,428	4,692
Total time	1,405.5ms	37.7ms	3.05ms	$108\mu s$	$40.8\mu s$

is performed late in the design process. The starting point of the HW/SW codesign approach is the architecture presented in the software realization section. The HW/SW design has been developed with the Xilinx EDK and targeted at an FPGA device.

This architecture is very similar to that of the software architecture shown in Fig. 6. The only modification is the addition of a hardware accelerator block. The HW/SW codesign methodology calls for profiling of the software system to identify areas that can be accelerated by repartitioning to the hardware domain. The iterative repartitioning of the software architecture to produce an HW/SW codesigned-based design is discussed next.

2) Profiling Results: The first step in the iterative process is to profile the current system. Both the GNU profiling tool *gprof* and the internal timing code were used to provide profiling data on the executing software. The results of this can be seen in Table I. Upon inspection of the results and the *gprof* analysis, it can be seen that the slow execution of floating-point operations causes a large increase in execution time. From the review code, it is clear that the major contributor to the floating-point instructions come from the processing of the FFT transforms. As a first iterative step to improve performance using HW/SW codesign, an FPU was added to the microprocessor architecture. The addition of this dedicated unit to the microprocessor accelerates the floating-point intensive FFT transforms.

The results of profiling after the FPU addition are also shown in Table I. It is easily seen that the addition of the FPU has provided a noticeable performance improvement. However, it still fails to meet all system requirements. Inspection of the results shows that the FFT execution time continues to dominate the total execution time. To further improve on performance, the design is once again repartitioned, this time placing the full execution of the FFT transform in the hardware domain through the addition of an FFT accelerator coprocessor.

3) FFT Accelerator: The FFT accelerator coprocessor is implemented as a custom very high speed integrated circuit (VHSIC) hardware description language-based design. This accelerator is interfaced to the Microblaze processor through the OPB. By keeping the accelerator, processor, and bus structure all within the same FPGA device, it allows a very high performance. Beyond the interfacing, the FFT calculation is implemented utilizing the same Xilinx IP core that is used in the pure-hardware architecture. A software driver is created, and the software design is updated to pass data to the accelerator and receive back the FFT results. This accelerator demonstrates the integration of hardware-design techniques with software design.

TABLE II
COMPARISON OF FCR IMPROVEMENT USING DIFFERENT HW/SW CODESIGN TECHNIQUES

Data Set	Input FCR (dB)	Software Implementation 4-channel (dB)	Software Implementation 8-channel (dB)	Hardware/Software Co-Design 4-channel (dB)	Hardware Radix-2 4-channel (dB)	Hardware Radix-2 8-channel (dB)
1	3.69	6.50	8.91	8.96	3.02	11.90
2	-2.33	15.80	17.12	15.46	12.31	13.20
3	2.24	12.67	11.96	9.67	8.73	8.90
4	-3.74	16.36	17.44	14.00	10.11	13.30
5	0.00	10.39	12.18	10.53	7.31	9.95
6	1.54	10.22	9.89	7.24	7.43	8.78
7	2.69	11.10	13.77	9.08	8.49	10.29
8	4.14	7.21	12.31	9.99	5.50	10.78
9	4.25	8.38	7.35	4.44	0.87	7.44
10	-1.70	10.27	10.42	11.43	10.48	12.39
Average	1.07	10.89	12.14	10.08	7.43	10.69

IV. PERFORMANCE AND RESOURCE UTILIZATION

A. FCR Performance

The most important performance metric for evaluating design performance is that of the FCR ratio. The equation to calculate the FCR is shown next. It relates the strength of the target echo to the surrounding clutter noise. This ratio is a direct representation of how a target echo can be detected in the surrounding noise

$$\text{FCR} = 20 \times \log_{10}\left(\frac{\text{Maximum Amplitude of Target Echo}}{\text{Maximum Amplitude of Clutter Echoes}}\right).$$

The equation for FCR improvement is shown next. As FCR is an indicator of the ease of target detection, the improvement value provides an indication of how much the SSP algorithm aided in target detection.

$$\text{Improvement} = \text{FCR after SSP Processing} - \text{Input FCR}.$$

As a reference point for all the designs, a MATLAB representation of the algorithm has been created. This representation is then applied to all the data sets to obtain the data results shown in Table II. This table can be used as a theoretical performance benchmark of the SSP algorithm in both the four- and eight-channel configurations. The data values exhibit fluctuation in performance due to the statistical randomness in the noise. This also accounts for the variations between different implementations. The critical measure for the performance of a design is not the overall performance on one data set but its average performance over all sets.

The detection performance of the software architectures performs identically to the theoretical MATLAB implementation. This performance result is expected. The software implementations use a floating-point representation of the data, the same as the internal representation within MATLAB. For this reason, there is no change in the performance for the software implementations.

On the other hand, both the hardware architectures and the HW/SW codesign with the FFT accelerator use fixed-point hardware-based FFT processing units. Because of the use of a fixed-point representation, of limited bit width, there is some inherent data imprecision when calculating the FFTs. The final impact of this result is a decrease in overall system performance. The performance for the hardware architectures can be seen in Table II. Since all hardware designs, both the pure hardware and HW/SW codesign, utilize similar IP-core-based FFTs, they exhibit a similar impact on performance. On average, the use of the fixed-point hardware FFT incurs a penalty of 1–3 dB in performance, while the HW/SW codesign implementation produces a much smaller 0.5–1 dB decrease. The reason for the better performance of the coprocessor implementation is due to bit precision. Since the coprocessor uses just a single FFT core, instead of the five FFTs in the pure hardware, it is more efficient to allocate more resources to a single more precise FFT implementation, using 16-b precision instead of the 8 b used in the pure hardware. The comparison of these two implementations demonstrates the impact bit precision has on the overall system performance. For both implementation methods, the small performance penalty is well worth the gains in execution time and resource usage.

It is also important to notice that the eight-channel implementation is able to achieve results much closer to the theoretical ideal. In addition to the overall performance, the eight-channel implementation exhibits a standard deviation of 1.98 compared with a standard deviation of 3.48 for the four-channel approach. This shows that the eight channels are able to produce more consistent performance increases. The more consistent performance in combination with results closer to theoretical maximum demonstrates that the eight-channel implementation provides a much more robust performance.

The reason for this more robust performance is due to the placement of the window filters. The eight-channel result is able to use smaller filters with a similar overlap and still be able to cover a larger portion of the frequency spectrum. Smaller filters with a similar overlap provide for more decorrelation between subband-clutter information. In addition, the wider frequency range provides a more robust performance when there are variations in input signal, since all defects do not reflect in a perfectly similar manner. This allows the eight-channel design to perform at both a higher performance metric and in a more consistent manner.

B. Execution Time

All of the architectures presented have a system goal of real-time processing. For ultrasonic imaging, a real-time rate is a processing rate exceeding 1 kHz. This gives only a short 1-ms-time window to perform all capture and processing of

data. The execution-time results for all the architectures are shown in Table I. This table goes into further depth by breaking down the execution time for each algorithmic processing step.

It can be shown that the pure-software approaches fail to reach the real-time requirement. The basic software implementation with only a simple processor without an FPU struggles to perform all the computations. With an execution time of 1.4 s, it comes nowhere near the needed execution rate. With the move to a more complicated microprocessor, incorporating an FPU unit, the design is able to improve to a time of 37.7 ms, although still well below the required rate. Both of these designs demonstrate that a software-only solution is not reasonable for this application. In order to meet the goals needed, the processor design would need to be greatly increased in complexity and clock rate, at a cost that can be better spent on alternate architectural techniques.

Table I also shows the execution time for both radix-2 and radix-4 pure hardware architectures. These implementations aim for high performance using fast hardware-processing elements and a highly parallel implementation, with multiple FFT cores for faster processing. Correspondingly, the designs easily exceed the processing rates required and provide orders of magnitude improvement in execution time compared with the software implementations, being 370 times faster.

As expected, the HW/SW codesign architecture is able to achieve a middle-ground performance between the two pure designs. It is important to highlight the progression of performance improvements when iterating through each step of the HW/SW codesign methodology. Starting with the pure software with basic microprocessor and continuing to the addition of an accelerator coprocessor, the design is able to achieve orders of magnitude improvement at each step. The HW/SW codesign process allows a targeted approach, applying engineering redesign to the algorithmic elements that will provide the most improvement.

The HW/SW codesign is able to achieve a performance rate of 3.05 ms. While the design currently does not meet the real-time requirements, it is close enough to be optimized to achieve the needed performance. Through the use of final optimization of clock rate and code optimization, it would be reasonable to expect performance gains to meet the requirements.

C. Resource Usage

For an FPGA device, there are many types of resources provided [21]. This research is concerned with the three major resources provided by the Xilinx Virtex 4 FPGA device, logic slices, BRAMs, and DSP48s. Logical slices provide the fundamental configurable logical resources in an FPGA device. They provide the ability to perform all logic, arithmetic, and ROM storage and have additional elements to enable distributed RAM storage. The device used provides support for up to 30 720 slices. For larger RAM storage needs, the device also provides 192 distributed dual-port 18-Kb RAM blocks. The final important resource provided are 192 DSP48 elements [24].

The evaluation of these three resources is a good representation of the cost of the implementation. The resource consumption for all the designs is shown in Table III. The resource usage

TABLE III
FPGA RESOURCE USAGE

	Software Architecture without FPU	Software Architecture with FPU	Hardware/ Software co-design	Hardware Architecture 4-channel	Hardware Architecture 8-channel
Slices	1,307	1,886	5,836	8,378	14,505
DSP48	3	7	25	30	54
RAM16	35	35	39	28	48

tends to follow the trend in execution-time performance. As the designs become more complex, they are able to achieve higher performance rates.

It is interesting to see the use of BRAM resources. For the most part, the BRAM tends not to scale evenly as the other resources do. The BRAM usage stays rather high for the software implementations, due to the need to store not only data flow but also the instruction code on the FPGA device. For this reason, the software designs have a high initial BRAM cost. This cost does not increase as more complex designs are created. In addition, the BRAM usage for the pure-hardware designs increases slowly as storage of the data grows slowly compared with processing elements.

V. DESIGN SUMMARY AND CONCLUSION

The frequency-diverse ultrasonic target-detection-system architecture has been created with modularity and reconfigurability in mind. This configurability is provided at two levels.

1) Design Configurability

   Design configuration takes advantage of the modularity of the system. It enables subsystem modifications within the design to facilitate large changes to the algorithm, such as changes in number of channels or changes between types of modules themselves.
   a) number of subband channels;
   b) bit lengths of processing;
   c) choice of frequency-transformation implementations, radix-2/radix-4 for FFT;
   d) choice of postprocessing techniques, e.g., minimization, maximization, averaging, neural networks.
2) Operational Configurability

   Operational configuration provides runtime flexible parameters, algorithmic changes that can be controlled through the user interface, without requiring a reprogramming of the device.
   a) subband window parameters (width, overlap, and offset);
   b) data sampling rate and acquisition-window location;
   c) signal preprocessing and gain.

In this paper, we have presented multiple configurable FPGA-based architectures for the SSP algorithm and their application to achieve real-time ultrasonic signal processing. The SSP-algorithm parameters have a large impact on its overall performance. With this in mind, all the architectures have been designed with a modular approach. All the designs are able to provide a high degree of flexibility in configuration.

This highly configurable modular approach is valuable for many reasons. It provides a valuable research tool which can be used to fully explore the design space of the SSP algorithm,

to evaluate the implementation impact caused by parameter changes. Furthermore, the architectures presented here are applicable to many embedded applications in other fields.

REFERENCES

[1] A. J. Al-Khalili, A. Brown, and D. M. Al-Khalili, "A microprocessor-based flaw detection system for offshore steel structures," *IEEE Trans. Ind. Electron.*, vol. IE-34, no. 2, pp. 138–147, May 1987.

[2] J. Saniie, D. Nagle, and K. Donohue, "Analysis of order statistic filters applied to ultrasonic flaw detection using split spectrum processing," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 38, no. 2, pp. 133–140, Mar. 1991.

[3] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domains of FPGAs," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1810–1823, Aug. 2007.

[4] H. Chang-Hong, Q. Zhou, and K. K. Shung, "Design and implementation of high frequency ultrasound pulsed-wave Doppler using FPGA," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 55, no. 9, pp. 2109–2111, Sep. 2008.

[5] Y. Lu, E. Oruklu, and J. Saniie, "Fast chirplet transform with FPGA-based implementation," *IEEE Signal Process. Lett.*, vol. 15, no. 1, pp. 577–580, 2008.

[6] L. Vachhani and K. Sridharan, "Hardware-efficient prediction-correction-based generalized-Voronoi-diagram construction and FPGA implementation," *IEEE Trans. Ind. Electron.*, vol. 55, no. 4, pp. 1558–1569, Apr. 2008.

[7] S. Jung and S. S. Kim, "Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 265–271, Feb. 2007.

[8] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems—A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007.

[9] Y. Chan, M. Moallem, and W. Wang, "Design and implementation of modular FPGA-based PID controllers," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1898–1906, Aug. 2007.

[10] Y.-A. Chapuis, Z. Lingfei, Y. Fukuta, Y. Mita, and H. Fujita, "FPGA-based decentralized control of arrayed MEMS for microrobotic application," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1926–1936, Aug. 2007.

[11] S. Valsan and K. Swarup, "High-speed fault classification in power lines: Theory and FPGA-based implementation," *IEEE Trans. Ind. Electron.*, vol. 56, no. 5, pp. 1793–1800, May 2009.

[12] Á. Hernández, J. Ureña, D. Hernanz, J. J. García, M. Mazo, J.-P. Dérutin, J. Sérot, and S. E. Palazuelos, "Real-time implementation of an efficient golay correlator (EGC) applied to ultrasonic sensorial systems," *Microprocess. Microsyst.*, vol. 27, no. 8, pp. 397–406, Sep. 2003.

[13] J. Weber, E. Oruklu, and J. Saniie, "Configurable hardware design for frequency-diverse target detection," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2008, pp. 890–893.

[14] G. D. Micheli and R. K. Gupta, "Hardware/software co-design," *Proc. IEEE*, vol. 85, no. 3, pp. 349–365, Mar. 1997.

[15] V. Newhouse, N. Bilgutay, J. Saniie, and E. Furgason, "Flaw-to-grain echo enhancement by split-spectrum processing," *Ultrasonics*, vol. 20, no. 2, pp. 59–68, Mar. 1982.

[16] J. Saniie, T. Wang, and N. Bilgutay, "Statistical evaluation of backscattered ultrasonic grain signals," *J. Acoust. Soc. Amer.*, vol. 84, no. 1, pp. 400–408, Jul. 1988.

[17] J. Saniie and D. Nagle, "Analysis of order statistic CFAR threshold estimators for improved ultrasonic flaw detection," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 35, no. 5, pp. 618–630, Sep. 1992.

[18] J. Saniie, K. Donohue, and N. Bilgutay, "Order statistic filters as postdetection processors," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 10, pp. 1722–1732, Oct. 1990.

[19] D. Nagle and J. Saniie, "Performance analysis of linearly combined order statistic CFAR detectors," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 31, no. 2, pp. 522–533, Apr. 1995.

[20] *XtremeDSP Development Kit-IV User Guide*, Xilinx, Inc., San Jose, CA, Sep. 2005. [Online]. Available: http://www.xilinx.com/support/documentation/boards_and_kits/ug_xtremedsp_devkitIV.pdf

[21] Xilinx, Inc., Virtex-4 User Guide, San Jose, CA, Apr. 2008. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug070.pdf

[22] *LogiCORE IP Fast Fourier Transform v7.1*, Xilinx, Inc., San Jose, CA, Apr. 2010. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf

[23] H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 6, pp. 849–863, Jun. 1987.

[24] *ExtremeDSP for Virtex-4 FPGAs User Guide*, Xilinx, Inc., San Jose, CA, Oct. 2007. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug073.pdf

Joshua Weber (S'07) received the B.S.E. degree in computer systems engineering from Arizona State University, Tempe, in 2005 and the M.S. degree in computer engineering from Illinois Institute of Technology, Chicago, in 2008. He is currently working toward the Ph.D. degree at Illinois Institute of Technology.

His research interests are in embedded systems, reconfigurable computing, hardware/software codesign and digital-logic design with field-programmable gate arrays. He is also a member of the VLSI and SoC Research Laboratory.

Erdal Oruklu (S'01–M'05–SM'09) received the B.S. degree in electronics and communication engineering from Technical University of Istanbul, Istanbul, Turkey, in 1995, the M.S. degree in electrical engineering from Bogazici University, Istanbul, in 1999, and the Ph.D. degree in computer engineering from Illinois Institute of Technology, Chicago, in 2005.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, where he is also the Director of VLSI and SoC Research Laboratory. His research interests are reconfigurable computing, advanced computer architectures, hardware/software codesign, and embedded systems.

Jafar Saniie (S'80–M'81–SM'91–F'10) received the B.S. degree in electrical engineering from the University of Maryland, College Park, in 1974, the M.S. degree in biomedical engineering from Case Western Reserve University, Cleveland, OH, in 1977, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1981.

Since 1983, he has been with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, where he is a Filmer Professor, an Associate Chair and a Director of the Embedded Computing and Signal Processing Research Laboratory. His research interests and activities are in embedded digital systems, digital-signal processing with field-programmable gate arrays, and ultrasonic signal and image processing.

Dr. Saniie is an IEEE Fellow for his contributions to ultrasonic signal processing for detection and estimation.