Fast Chirplet Transform With FPGA-Based Implementation

Yufeng Lu, *Member, IEEE*, Erdal Oruklu, *Member, IEEE*, and Jafar Saniie, *Senior Member, IEEE*

***Abstract*—This letter presents a fast chirplet transform (FCT) algorithm, a computationally efficient method, for decomposing highly convoluted signals into a linear expansion of chirplets. The FCT algorithm successively estimates the chirplet parameters in order to represent a broad range of chirplet shapes, including the broadband, narrowband, symmetric, skewed, nondispersive, or dispersive. These parameters have significant physical interpretations for radar, sonar, seismic, and ultrasonic applications. For the real-time application and embedded implementation of the FCT algorithm, an FPGA-based hardware/software co-design is developed on Xilinx Virtex-II Pro FPGA development platform. Based on the balance among the system constraints, cost, and the efficiency of estimations, the performance of different algorithm implementation schemes have been explored. The developed system-on-chip successfully exhibits robustness in the chirplet transform of experimental signals. The FCT algorithm addresses a broad range of applications including velocity measurement, target detection, deconvolution, object classification, data compression, and pattern recognition.**

***Index Terms*—Detection, estimation, fast chirplet transform, field programmable gate arrays, hardware/software co-design.**

I. Introduction

In RADAR, sonar, speech processing, seismic exploration, and ultrasound applications, the detected echoes are often nonstationary and corrupted by the measurement noise and/or undesired scattering echoes (clutter). Consequently, isolating these echoes becomes a challenging problem, and conventional signal analysis techniques fails to unravel the desired signal information necessary for target detection, speech analysis, seismic reflections, and ultrasonic imaging. The chirplet is a type of signal often encountered in radar applications [1]. Chirplet transform was introduced in [2] as an expansion of arbitrary function onto a basis of multiscale chirps, and it was used for detection of floating objects in marine radar applications. In [3], a chirplet-based adaptive signal representation algorithm was applied to extract features from ISAR data of a target with a rigid main body and a rotating part. Similarly, the chirplet transform was used for seismic event recognition and extracting time-varying information from seismic records [4]. Parameter estimation of chirp signals and derivation of several estimators were studied in [5].

Signal modeling and parameter estimation, although algorithmically complex and computationally heavy, can be an effective procedure to decompose and estimate echoes with important diagnostic information. The parameters of a chirplet, i.e., time-of-arrival, center frequency, amplitude, bandwidth factor, chirp rate, and phase, are capable of representing a broad range of echo shapes, including the narrowband, broadband, symmetric, skewed, nondispersive, or dispersive [6]. The chirplet parameter estimator is an unbiased and minimum variance estimator. Moreover, the estimated parameters closely represent the physical properties of the system, such as position and velocity of a target in radar or sonar, target size, and orientation in ultrasonic imaging. In this letter, we present a highly efficient realization of chirplet transform and also an embedded system running on an FPGA.

In the following section, the fast chirplet transform (FCT) algorithm and the procedure for successive parameter estimation is presented. Section III discusses hardware/software (HW/SW) co-design of an FPGA-system for the FCT algorithm. Moreover, a case study of the FPGA system using an ultrasonic experimental signal and a bat chirp signal is presented.

II. Chirplet Transform

In most detection applications, a single echo can be modeled as a chirplet [6], [7] as follows:

$$f_\Theta(t) = \beta \exp(-\alpha_1(t-\tau)^2 + i2\pi f_c(t-\tau) + i\phi + i\alpha_2(t-\tau)^2) \quad (1)$$

where $\Theta = [\tau, f_c, \beta, \alpha_2\phi, \alpha_1]$ denotes the parameter vector. The term τ is the time-of-arrival, f_c is the center frequency, β is the amplitude, α_2 is the chirp rate, ϕ is the phase, and α_1 is the bandwidth factor of the echo. The chirplet transform of the echo is defined as [2], [6] follows:

$$\mathrm{CT}(\hat{\Theta}) = \int_{-\infty}^{+\infty} f_\Theta(t)\Psi^*_{\hat{\Theta}}(t)\,dt \quad (2)$$

where

$$\Psi^*_{\hat{\Theta}}(t) = \left(\frac{2\gamma_1}{\pi}\right)^{\frac{1}{4}} \exp\left(-\gamma_1(t-b)^2 - i\omega_0\left(\frac{t-b}{a}\right) - i\theta - i\gamma_2(t-b)^2\right) \quad (3)$$

denotes the chirplet kernel, and

$$\hat{\Theta} = [b, (\omega_c)/(2\pi a), \eta, \gamma_2, \theta, \gamma_1]$$

Manuscript received December 26, 2007; revised March 15, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Athanassios Skodras.

Y. Lu is with the Electrical and Computer Engineering Department, Bradley University, Peoria, IL 61625 USA (e-mail: ylu2@bradley.edu).

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

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

Digital Object Identifier 10.1109/LSP.2008.2001816

denotes the parameter vector of the chirplet used for transformation. The parameter can be successively estimated [6] as shown in the following:

$$\left.\begin{aligned}\frac{\partial|\mathrm{CT}(\hat{\Theta})|}{\partial a}&=0\\ \frac{\partial|\mathrm{CT}(\hat{\Theta})|}{\partial b}&=0\end{aligned}\right\}\quad\Rightarrow\quad b=\tau \text{ and } \frac{\omega_0}{a}=\omega_c \tag{4}$$

$$\left.\frac{\partial|\mathrm{CT}(\hat{\Theta})|}{\partial\gamma_2}\right|_{b=\tau,\frac{\omega_0}{a}=\omega_c}=0\quad\Rightarrow\quad\gamma_2=\alpha_2 \tag{5}$$

$$\left.\frac{\partial|\mathrm{CT}(\hat{\Theta})|}{\partial\gamma_1}\right|_{\substack{b=\tau,\\ \frac{\omega_0}{a}=\omega_c,\\ \gamma_2=\alpha_2}}=0\quad\Rightarrow\quad\gamma_1=\alpha_1 \tag{6}$$

$$\left.\frac{\partial\mathrm{Re}(\mathrm{CT}(\hat{\Theta}))}{\partial\theta}\right|_{b=\tau,atop\frac{\omega_0}{a}=\omega_c,\atop\gamma_2=\alpha_2}=0$$

$$\Rightarrow\theta=\phi\pm 2k\pi,\quad k=1,2,3,\ldots \tag{7}$$

The objective of the FCT algorithm is to decompose the signal, $s(t)$, into a linear expansion of chirplets and efficiently estimate the parameter vectors defining these chirplets (this is also known as adaptive chirplet transform ACT [8])

$$s(t)=\sum_{j=0}^{N-1} f_{\Theta_j}(t). \tag{8}$$

The parameter vector Θ_j can be estimated based on the chirplet transform of the signal $s(t)$. By localizing the dominant echo in a time-frequency representation of the signal (i.e., chirplet transform), we can estimate the time-of-arrival, center frequency, and amplitude of the dominant echo and then successively estimate the remaining parameters [6]. In an iterative manner, the residual signal is obtained by subtracting the estimated single dominant echo from the signal. The decomposition process is repeated until the energy of residual signal reaches below a predefined reconstruction condition.

The chirplet transform algorithm is capable of achieving a high-resolution time-frequency representation and accurate estimation of parameters. Nevertheless, the time-frequency representation used for decomposition in the algorithm is computationally heavy due to chirplet transform. In each iteration stage, the entire chirplet transform matrix is generated for the echo isolation process, which hinders the algorithm from real-time signal processing applications. This problem can be overcome by utilizing a fast implementation scheme of the algorithm. Instead of two-dimensional transform, we use one-dimensional transform and iteratively estimate the time-of-arrival, center frequency, and amplitude of the dominant echo. Fig. 1 shows the flow chart of the FCT algorithm. The steps involved in the iterative estimation of an experimental signal are outlined as follows.

1) Find the maximum location of $s(t)$ in time domain and use it as the initial guess of time-of-arrival and the starting point of iteration.
2) Estimate the center frequency which maximizes the chirplet transform, given initial guess of time-of-arrival.

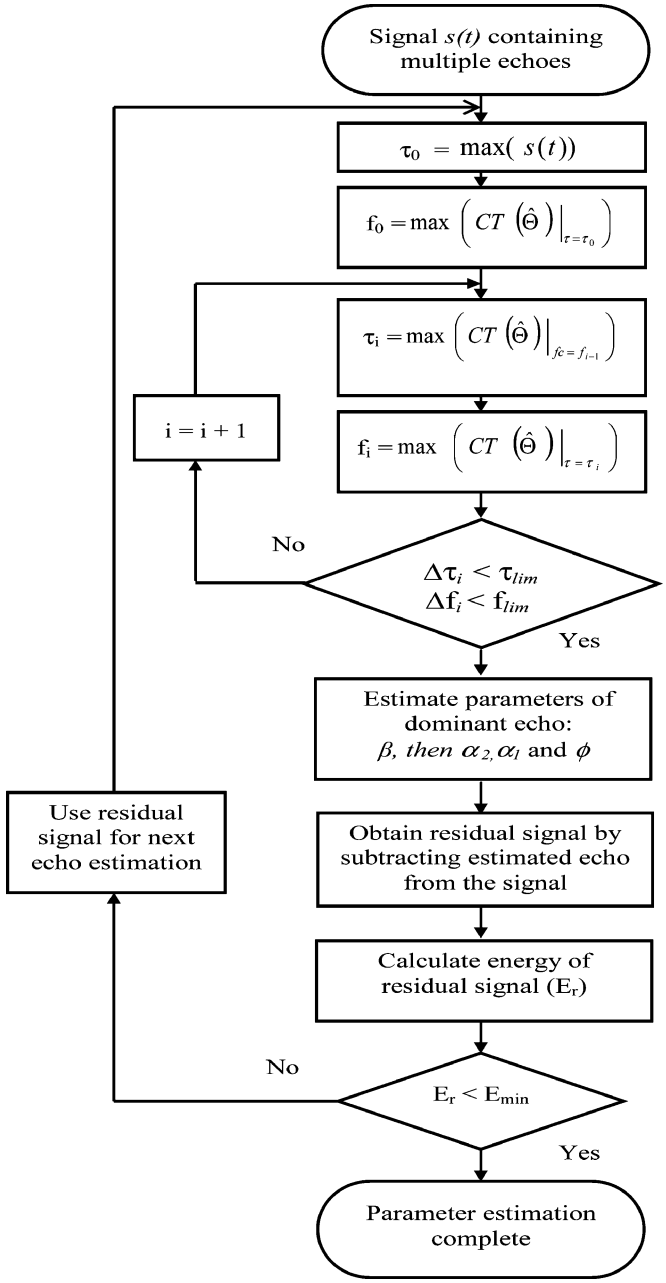

Fig. 1. Flow chart of the FCT algorithm.

3) Estimate the time-of-arrival which maximizes the chirplet transform, given the estimated center frequency from the previous step.
4) Estimate the center frequency which maximizes the chirplet transform, given the new estimated time-of-arrival from Step 3.
5) Check convergence: If $\Delta\tau<\tau_{\mathrm{lim}}$ and $\Delta f<f_{\mathrm{lim}}$ (here, τ_{lim} and f_{lim} are predefined convergence conditions), then go to Step 6; otherwise, go to Step 3.
6) Estimate the amplitude β and the remaining parameters α_2,ϕ, and α_1 successively.
7) Obtain the residual signal by subtracting the estimated echo from the signal.

8) Calculate energy of the residual signal (E_r) and check convergence ($E_{\min}$ is predefined convergence condition): If $E_r < E_{\min}$, STOP; otherwise, go to Step 1.

From the above steps, it can be seen that the FCT algorithm significantly relaxes the computation load of the algorithm.

III. FPGA-Based Hardware/Software Co-Design

The FCT algorithm has been implemented as a system-on-chip (SoC) based on a Xilinx Virtex II Pro FPGA to explore its suitability for real-time applications. FPGA has been chosen due to its reconfigurable and reprogrammable ability. From the system point of view, the FPGA-based SoC design consists of two parts: hardware architecture and software code, where hardware architecture is a structured combination of processing elements (including hard/soft processor core, IP cores, and hardware accelerators), and external interfaces to different I/Os (such as UART, Ethernet, USB, and VGA, etc.), and the software part is the executable code running on the on-chip processor.

The objective of the design is to partition the FCT algorithm into hardware and software components based on their computational demands. The profiling results of the FCT algorithm revealed that several computationally intensive functions cost a significant amount of the execution time. These functions and their associated percent execution time are:

Est_T : estimate time-of-arrival given the center frequency (see step 3 of the FCT algorithm). Execution time: 24%.

Est_F : estimate frequency given the time-of-arrival (see step 4 of the FCT algorithm). Execution time: 39%.

Est_Para: estimate the remaining parameters of chirplets (see step 6 of the FCT algorithm). Execution time: 36%.

It can be seen that these functions comprise around 99% of the total execution time, and they are good candidates for hardware accelerators and/or software optimization. Further analysis shows that calculating multiplications, trigonometric, exponential, and FFT functions take most of the execution time in implementing the FCT algorithm. For example, FFT function accounts for 68% of Est_T and 12% of Est_Para execution times. To implement the hardware accelerator, the challenge is to substitute the function in the software implementation with the code to access the accelerator. Xilinx core generator system provides a catalog of user-customizable cores to streamline the design process and improve design quality on Xilinx FPGA devices. For the FFT function, we use the FFT core with run-time programmable transform size and streaming pipeline implementation. Multiplier core is chosen to implement the multiplier functions. Exponential function can be implemented in hardware [9]. However, the performance improvement through hardware accelerator for exponential function becomes limited due to access-time overhead. It is important to point out that the chirplet transform is only applicable to band-limited signals, and this allows an efficient and practical use of software look-up table to implement the exponential function, $e^{-\alpha_1(t-\tau)^2}$ which is a monotonic decreasing function for $\alpha_1 \geq 0$. Similarly, we utilize look-up tables and the existing multiplier cores within the FPGA to calculate the phase (i.e., $2\pi f_c(t-\tau)^2+\alpha_2(t-\tau)^2+\phi$) and to obtain the values of trigonometric functions.

TABLE I
PERFORMANCE COMPARISON OF SCHEME 1–SCHEME 4

Function	Scheme 1 (Clock Cycles)	Scheme 2 (Clock Cycles)	Scheme 3 (Clock Cycles)	Scheme 4 (Clock Cycles)	Performance Improvement Scheme 4 versus Scheme 1
Est_T	2,379,887,529	102,922,974	28,574,231	2,313,609	1028.64 X
Est_Para	3,896172,993	161,861,305	14,267,866	5,840,809	677.06 X
Est_F	3,636,716,247	171,816,738	2,823,983	2,823,983	1287.79 X
FCT Algorithm	9,915,638,571	438,143,059	47,474,555	12,786,876	775.45 X

For the software optimization, the embedded processor IP core (PowerPC 405) can be used with an optional and size-configurable instruction cache and data cache for improved performance. The FCT algorithm has been successfully implemented on the embedded PowerPC405 core of XUP-V2P (Xilinx Virtex-II Pro FPGA development system) [10]. The signal source used in the embedded system is from an application of ultrasonic target detection [6]. The center frequency of the ultrasonic transducer is 5 MHz and the sampling rate is 100 MHz. A 512-point experimental data set is used to test the performance of FCT algorithm on this system. The effort of acceleration and optimization varies in the different stages of hardware/software co-design. Four implementation schemes have been prototyped. These implementation schemes are:

Scheme 1: Software implementation on PowerPC405;

Scheme 2: Scheme 1 with data and instruction caching;

Scheme 3: Scheme 2 plus hardware multiplier accelerator, software look-up tables for trigonometric, and exponential functions;

Scheme 4: Scheme 3 with hardware FFT accelerator.

The performance for these four schemes is presented in Table I. Caching a small size of local memory in the processor core can boost the performance of CPU and save the time of accessing data and instruction by buffering recent visited data or instructions. The caching effect can be examined from the system performance of Schemes 1 and 2. It can be seen that the overall performance of FCT algorithm is increased 22.6 times by enabling the data and instruction caching of the embedded PowerPC405 processor.

As discussed in the profiling results of the algorithm, the Est_F function is more dependent on the efficiency of multiplier, trigonometric, and exponential functions; the Est_T function heavily depends on the computation of FFT; the computation of Est_Para is more evenly distributed on FFT and trigonometric functions.

In Scheme 3, the hardware multipliers and software look-up tables are used to accelerate the algorithm. Compared with Scheme 2, the execution time of Est_F is significantly improved by 60.8 times in Scheme 3; whereas, the Est_T only gains performance improvement by 3.6 times in Scheme 3. This is confirmed by the inspection of the algorithm: the Est_F is more dependent on the efficiency of the multiplier, trigonometric, and exponential functions.

As a comparison, in Scheme 4, a hardware FFT core is used to accelerate the system, especially the Est_T and Est_Para functions. As expected, the FFT accelerator has no effect on the performance of the Est_F function since there is no FFT operation in Est_F function. The execution time of Est_T function is improved by 12.3 times, compared with the one

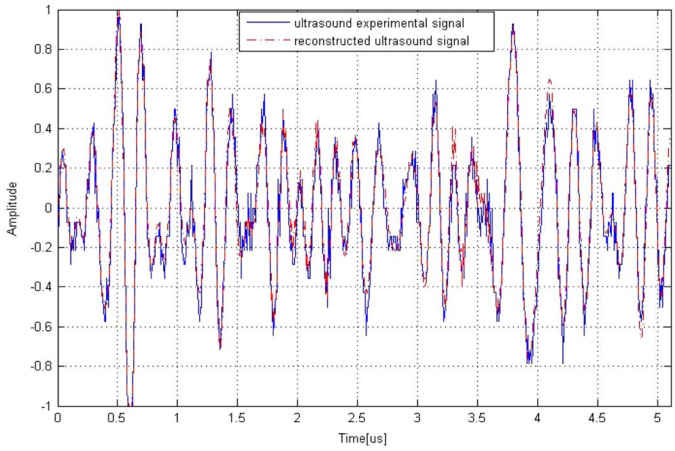

Fig. 2. Ultrasonic backscattered signal superimposed with the reconstruction echoes.

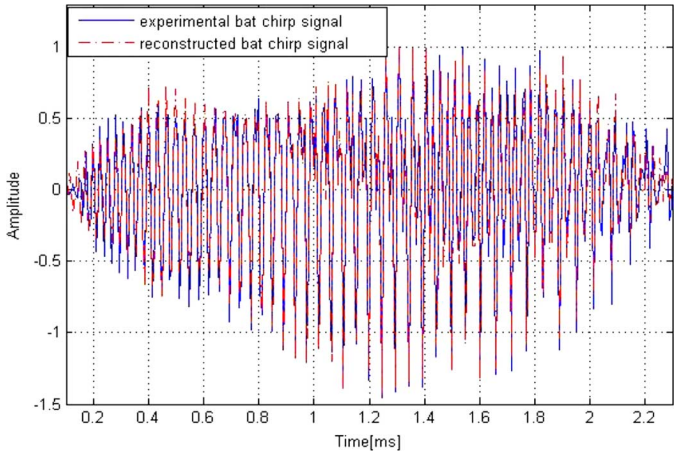

Fig. 3. Bat chirp signal superimposed with the reconstructed echoes.

in Scheme 3. The Est_Para gets a moderate performance gain from the FFT accelerator. It can be seen that the overall performance of the FCT algorithm has been improved by 775.45 times through the FPGA-based hardware/software co-design. The performance improvement from hardware accelerator comes at the expense of logic usage in FPGA. In the embedded implementations of the FCT algorithm, the trade-off between performance and logic usage for Schemes 1 and 3 is: 5.1% more number of logic slices, 3.2% more number of slice flip flops, and 4.3% more number of four-input LUTs due to the multiplier hardware accelerators. The trade-off for Schemes 3 and 4 is: 11% more number of logic slices, 5.8% more number of slice flip flops, and 9.7% more number of four-input LUTs, and 1.5% more BRAMs due to the FFT hardware accelerators. From the analysis of different design schemes, it can be seen that HW/SW co-design on FPGA can switch some functions of the FCT algorithm between hardware and software in a reconfigurable way. In practice, choosing a hardware accelerator is based on the trade-off of the system constraints (cost, performance, etc.).

For a case study, the FPGA-based system of the FCT algorithm is applied to the experimental ultrasonic backscattered signal [6]. The estimated parameter vectors consisting of 20 chirplets are used to reconstruct the original experimental signal. The experimental ultrasonic signal superimposed with the reconstructed chirplets is shown in Fig. 2. The reconstructed chirplets match closely to the original experimental signal. The estimation process of 20 chirplets takes about 1.0 s. Furthermore, the performance of the FCT algorithm has been tested using a bat chirp signal emitted by a large brown bat, which is digitized within 2.2 ms duration with a 7 μs sampling period [11]. The comparison of the decomposition and the original signal is shown in Fig. 3. These results confirm the feasibility of FCT algorithm for signal estimation and representation.

IV. Conclusion

In this letter, an FPGA-based hardware/software co-design for the fast chirplet transform algorithm is presented. The prototype design shows that hardware accelerators and software optimization have improved the system performance significantly. The developed embedded system successfully decomposed the ultrasonic experimental data and bat chirp signal into chirplets and estimated all the parameters. This type of study addresses a broad range of applications in target detection, deconvolution, object classification, data compression, and pattern recognition.

References

[1] M. I. Skolnik, *Radar Handbook*, 2nd ed. New York: McGraw-Hill, Jan. 1990.
[2] S. Mann and S. Haykin, "The chirplet transform: A generalization of Gabor's logon transform," in *Proc. Vision Interface'91*, Jun. 1991, pp. 205–212.
[3] J. Li and H. Ling, "Application of adaptive chirplet representation for ISAR feature extraction from targets with rotating parts," *Proc. Inst. Elect. Eng., Radar, Sonar, Navig.*, vol. 150, no. 4, pp. 284–291, Aug. 2003.
[4] W. Fan, H. Zou, Y. Sun, Z. Li, and R. Shi, "Decomposition of seismic signal via chirplet transform," in *Proc. IEEE 6th Int. Conf. Signal Processing*, Aug. 2002, vol. 2, pp. 1778–1782.
[5] P. Djuric and S. Kay, "Parameter estimation of chirp signals," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-38, no. 12, pp. 2118–2126, Dec. 1990.
[6] Y. Lu, R. Demirli, G. Cardoso, and J. Saniie, "A successive parameter estimation algorithm for chirplet signal decomposition," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 53, no. 11, pp. 2121–2131, Nov. 2006.
[7] S. Mann and S. Haykin, "The chirplet transform: Physical consideration," *IEEE Trans. Signal Process.*, vol. 43, no. 11, pp. 2745–2761, Nov. 1995.
[8] S. Mann and S. Haykin, "The adaptive chirplet: An adaptive wavelet like transform," in *Proc. SPIE, 36th Annu. Int. Symp. Optical and Optoelectronic Applied Science and Engineering*, Jul. 1991.
[9] J. Detrey and F. Dinechin, "A parameterized floating-point exponential function for FPGAs," in *Proc. IEEE Int. Conf. Field-Programmable Technology*, Dec. 2005, pp. 27–34.
[10] Xilinx XUP-V2P Development System. [Online]. Available: http://www.xilinx.com/univ/xupv2p.html.
[11] Bat Chirp Signal. [Online]. Available: http://www.dsp.rice.edu/software/bat.shtml.