Near Real-Time Simultaneous Range and Velocity Processing in a Random Noise Radar

T. Joel Thorson and Geoffrey A. Akers
Graduate School of Engineering and Management
Air Force Institute of Technology
Dayton, OH 45433-7765
Email: timothy.thorson@afit.edu and geoffrey.akers@afit.edu

***Abstract*— The Air Force Institute of Technology (AFIT) has developed an ultrawideband (UWB) random noise radar (RNR) that operates in the UHF band. The large fractional bandwidth and lack of phase coherency within the AFIT RNR necessitate an alternative to classical Doppler processing for velocity estimation. A time-domain approach to simultaneous range and velocity processing has been implemented in the AFIT RNR but results in significant memory requirements and impractical processing times. This paper presents a method to reduce the two-dimensional (2D) processing time by over an order of magnitude by segmenting the signal processing algorithm in order to parallelize on a graphics processing unit (GPU).**

I. Introduction

Multi core processing has found its way into all segments of the computer industry [1]. Two- to eight-cores are no longer used just for the server market but are now commonplace in personal desktop computers. The trend continues to signal processing where graphics processing units (GPUs) with hundreds of processing cores are now widely used for applications other than graphics processing. With the release of the NVIDIA CUDA (Compute Unified Device Architecture) software interface in 2007, the GPU has become more openly exploited in signal processing [2]. No longer solely used for a specific graphical application, the general purpose GPU (GPGPU) is made up of hundreds of individual processing cores that typically share memory resources making it favorable for many parallel systems.

The most recent simultaneous range-velocity processing research at AFIT began the effort of parallelizing the signal processing algorithm [3]. The idea was to reduce the memory required to process the algorithm in order to implement in a GPU. The memory was reduced by replacing an eight-bit analog-to-digital converter (ADC) with a binary ADC. Although the memory was reduced significantly, the signals were too long and still required too much memory to be processed in parallel on a GPU. However, recent strides have been taken to evolve the algorithm for further parallelization.

This paper presents the ongoing research effort conducted at AFIT to reduce the simultaneous range and velocity processing time of the RNR to near real-time. The paper begins with a description of the AFIT RNR and its velocity estimation technique. It continues with the fast Fourier transform (FFT) segmentation method used to parallelize the correlation algorithm used in the AFIT RNR. Finally, it discusses the test, methodology, and results used to prove the technique.

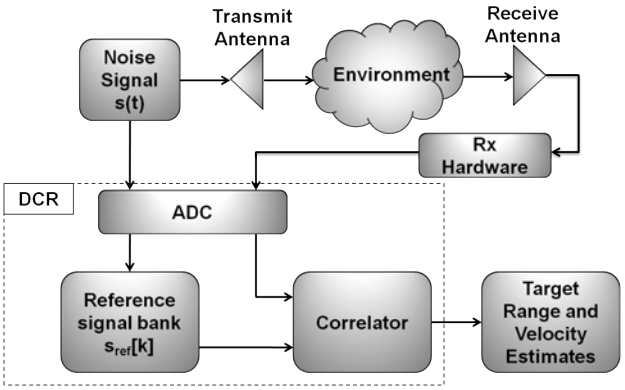

Fig. 1. The functional building blocks of the AFIT RNR.

II. System/Equipment Description

The AFIT RNR was first constructed by Schmitt [4] and was demonstrated in near-monostatic and networked configurations. The bistatic/near-monostatic configuration, broken into its functional blocks, can be seen in Figure 1, and is the focus of this research effort.

A. Transmitter

To generate a random noise transmit signal, the AFIT RNR uses a thermal, white Gaussian noise (WGN) generator that provides a flat frequency response at -82 dBm/Hz up to 1.6 GHz. The source is then filtered using a low-pass filter (LPF) and a high-pass filter (HPF) to generate the band-limited, ultrawideband (UWB), continuous wave signal from 400 to 800 MHz. After filtering, the noise signal is split to the transmit antenna as well as to the direct conversion receiver (DCR), where it is used as a reference signal for correlation processing. The transmit path of the AFIT RNR can be seen in Figure 2.

B. Receiver Front End

After the transmit signal interacts with the environment and is received by the antenna, the received signal is passed through a LPF and HPF combination, identical to the transmit path filters, before being amplified by two, 20-dB low noise amplifiers (LNAs). The LNAs are used to bring the receive

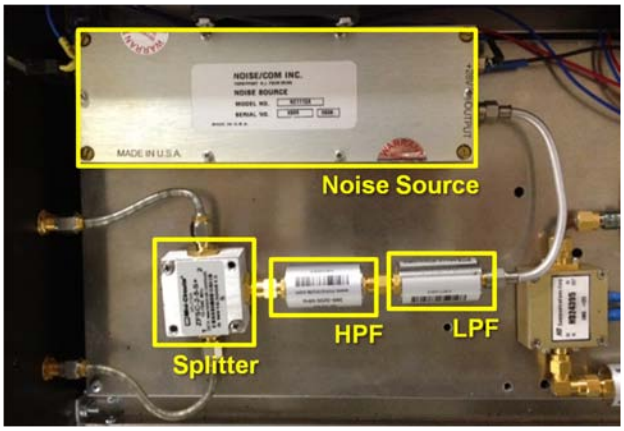

Fig. 2. This figure highlights each component in the AFIT RNR transmit path.

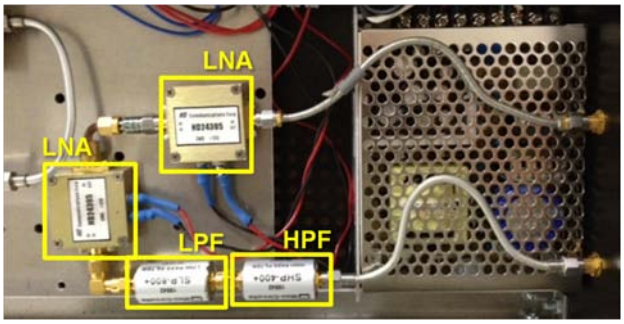

Fig. 3. This figure highlights each component in the AFIT RNR receiver front end (prior to ADC).

signal amplitude within the dynamic range of the ADC. The receive path of the AFIT RNR can be seen in Figure 3, and a detailed description of each hardware component in the AFIT RNR can be found in [4].

C. Direct Conversion Receiver

The direct conversion receiver (DCR) performs the transmit and receive signal analog-to-digital conversion as well as the correlation processing required for range and velocity estimation. The DCR consists of a two-channel ADC, with a maximum of 1 μs acquisition time, connected to a laptop via a Peripheral Component Interconnect Express (PCIe) card. The laptop hosts the MATLAB® routine developed to provide target range and velocity estimates using digital correlation processing.

III. Velocity Estimation

The AFIT RNR was designed to be simple, flexible, and reliable. This was accomplished by designing a DCR, which bypasses the well-known heterodyne receiver architecture [5]. The ADC is placed as close to the receive antenna as possible, providing a digital correlation environment. By implementing the DCR and limiting the RF front-end hardware, noise is minimized and the receiver becomes software modifiable much in the same was a the software defined radio concept. However,

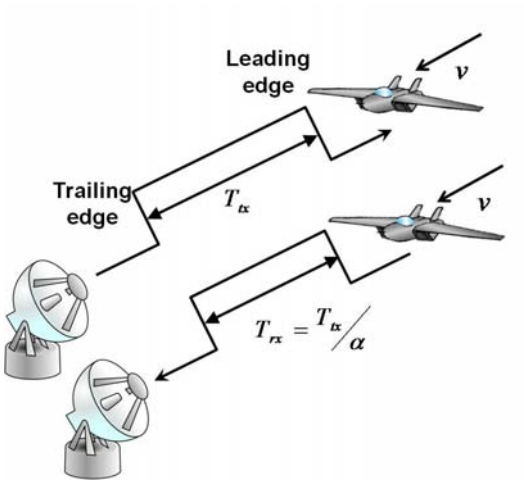

Fig. 4. Illustration of the time domain velocity estimation technique implemented in the AFIT RNR.

no quadrature channel exists in the AFIT RNR, resulting in a phase-incoherent system.

In addition to the AFIT RNR phase-incoherency, the UWB signal implementation consists of a large fractional bandwidth resulting in errors when estimating velocity using the classical narrowband Doppler technique, therefore necessitating an alternative approach to velocity estimation. Lievsay and Akers [6] implemented an alternative approach to simultaneous range-velocity estimation in the AFIT RNR using time domain processing based on Axelsson's RNR processing theory [7] and Rihaczek's general wideband ambiguity function theory [8]. The 2D processing technique implemented by Lievsay and Akers proved to be successful, but required approximately $T_p \approx 42$ minutes of processing time and 32 GB of RAM [6].

A. Time-Domain Signal Dilation

The continuous wave transmit signal must be split into measurement windows for processing. This measurement window, with time extent T, corresponds to the signal processing integration time. A target's relative radial velocity, v, will cause the receive signal to be dilated in time as can be seen in Figure 4. An inbound target will cause a compression to the transmit signal and, conversely, an outbound target will cause the transmit signal to stretch. The measurement window at receive will be scaled by a factor of α, and will take the form

$$T_{rx} = \frac{T_{tx}}{\alpha}, \tag{1}$$

where α is the time scale of the receive signal given by [9]

$$\alpha = \frac{c - v}{c + v}, \tag{2}$$

and c is the speed of light.

$\Delta T = T_{rx} - T_{tx}$ represents the overall dilation of the signal resulting from target motion as illustrated in Figure 5. However, to generate reference signals based on target velocity, it is important to understand how each sample is affected in time

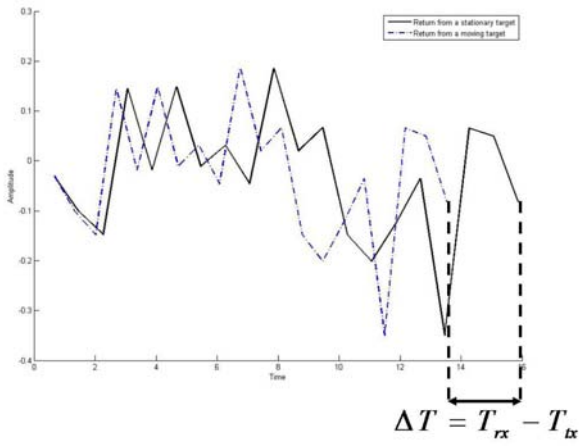

Fig. 5. Ingressing radial velocity shortens the measurement window by ΔT.

due to target motion. This relationship is given by

$$\Delta t = \frac{2v}{(c-v)f_s}, \tag{3}$$

where $f_s = 2f_h$ is the Nyquist sample rate. Not only can the reference signals be generated using (3), but a target's radial velocity can be derived by measuring the time shift at each sample over the length of the measurement window and comparing the result to the reference signal bank.

The significant drawback to the time-domain signal dilation method of velocity estimation is the signal length required for practical velocity resolution. Velocity resolution Δv, defined as [10]

$$\Delta v = \frac{2c}{Tf}, \tag{4}$$

is dependent on the length of integration time (measurement window) T and the transmitted frequency f. In the case of the noise radar, which transmits at a range of frequencies, the best range resolution results from the highest frequency in the range, f_h. By inspection of (4), as T and f_h increase, velocity resolution improves.

Signal length is given by $N = f_s T$ and is dependent on the sample frequency f_s and measurement window T. In order to meet the Nyquist minimum, the sampling frequency must be $2f_h$. So, increasing T and f_h to improve resolution also increases the length of the transmit signal. This relationship highlights the dependence of velocity resolution on signal length, N, given by

$$\Delta v = \frac{4c}{N}, \tag{5}$$

when sampling at Nyquist baseband. For a velocity resolution of $\Delta v = 6$ m/s, the signal length must be $N = 200$ million samples long. Obviously these long signals require significant memory and pose a computational burden for signal processing.

IV. FFT Segmentation

As discussed above and illustrated in Figure 1, the transmit signal, $s(t)$, is split and passed into the DCR, where a bank of reference signals is generated based on the transmit signal and set of pre-defined reference velocities. Based on (3), the selected reference velocity shifts each of the N samples of the transmit signal by Δt to give a reference signal in the form of

$$s_{ref}[k] = s[k - (k-1)\Delta t], \tag{6}$$

where each sample, k, is shifted by Δt which corresponds to the reference velocity.

The receive signal is then correlated with each of the reference signals. The cross correlation function, $r(\tau)$, is defined as [10]

$$r(\tau) = \int_{t=0}^{T} s_R(t) s_{ref}(t-\tau) dt, \tag{7}$$

where $s_R(t)$ is the receive signal and $\tau = 2R/c$ represents the range to the target in terms of the time delay. However, in the DCR, the correlation is performed digitally in the form of

$$r[m] = \sum_{k=0}^{N-1} s_R[k] s_{ref}[k-m], \tag{8}$$

where $m = f_s\tau$ corresponds to the number of samples for delay τ.

Correlation is similar to convolution, and that similarity can be exploited to take advantage of the FFT efficiencies. The receive signal $s_R[k]$ can be multiplied by the conjugated reference signal $s_{ref}[k]$ in the Fourier domain, leading to the equation for correlation

$$r[m] = \frac{1}{N}\mathcal{F}^{-1}\left[\mathcal{F}\{s_R[k]\}\mathcal{F}\{s_{ref}[k-m]\}^*\right], \tag{9}$$

where $\mathcal{F}$ represents the Fourier transform, $\mathcal{F}^{-1}$ represents the inverse Fourier transform, and * represents conjugation.

Although the FFT is efficient, the lengthy signals required for sufficient velocity resolution, and hence the lengthy FFTs, are too long to allow for parallel implementation in a GPU. The signals must be broken into small segments, thus reducing the FFT sizes to allow for parallelization over hundreds of processors.

Meller published a method in [11] to segment lengthy FFTs in a noise radar correlator for range processing. More commonly known as the overlap-save method [12], the FFTs are broken into overlapping segments of length $2M$, where M is equivalent to the number of samples in the time delay corresponding to the range extent R_{max}. The receive signal segments have M samples from $s_R[k]$ and are padded with M zeros to have a segment length of $2M$ samples. The reference signal segments, on the other hand, are a concatenation of M samples from the "previous" segment and M samples from the "current" segment, thus overlapping the FFT segments.

Once the signals have been segmented, the FFTs of the receive signal and reference signal segments are computed. The

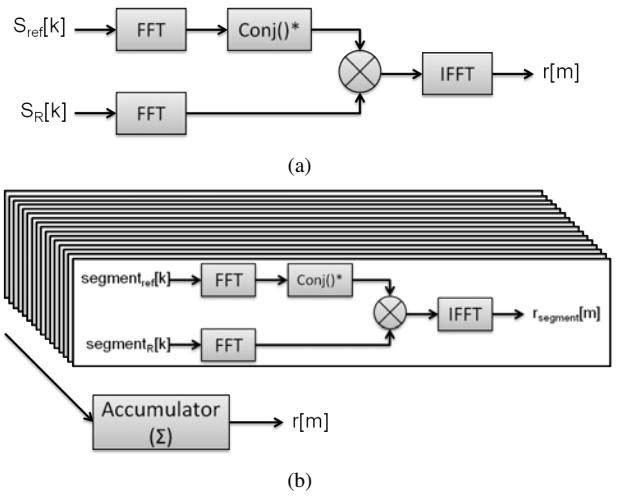

Fig. 6. Comparison of (a) the traditional cross correlation implementation and (b) the segmented cross correlation method proposed by Meller [11].

conjugated reference signal FFT segment is then multiplied with the receive signal FFT segment, and the inverse FFT (IFFT) of the result is computed. The segmented IFFTs are then accumulated (the vectors are added), resulting in the cross correlation of the reference and receive signal. A comparison of the the traditional cross correlation implementation with the segmented method proposed by Meller [11] can be seen in Figure 6.

The benefit of this FFT segmentation method is its potential for parallelization. The cross-correlation of each segment can be computed individually and in parallel before accumulation. Instead of a single cross correlation that has FFTs of length $N_{FFT} = 200$ million, there can be many (thousands) of cross correlation operations that take the place of the single operation. These fast operations can be distributed to hundreds of processing cores operating in parallel, thus significantly reducing the overall processing time.

V. GPU IMPLEMENTATION

Two computers, equipped with NVIDIA® GPUs, were used to process the collected data. The specifications for each of the computers along with the GPUs are presented in Table I.

The data set used for this element of the test effort was generated by Lievsay [6] and used in the most recent effort to replace the eight-bit ADC with a binary ADC [3]. The original data was collected by Lievsay using the AFIT RNR with a Tektronix® Digital Phosphor Oscilloscope (DPO) 7254 as the eight-bit ADC. The data set that is used for the test is of a target at 10 m from the monostatic radar moving directly toward the RNR at 5 m/s. The measurement window was 160 ms and sampled at $f_s = 1.25$ Gsamp/s. In the most recent effort at AFIT, the signal processing time for the same data set was minimized to roughly 5 minutes. Obviously 5 minutes is not a practical processing time. By parallelizing the correlation algorithm on a GPU, the goal is to bring the processing time to a near real-time application.

TABLE I
PROCESSING HARDWARE

	Computer 1	Computer 2
Make	Dell	HP
Model	Precision T7500	Z8000 Workstation
Operating System	Windows 7 Pro	Windows 7 Pro
Processor Make	Intel	Intel
Processor Model	Xeon W5590	Xeon X5667
Number of Processing Cores	8	8
Processor Speed	3.33 GHz	3.07 GHz
Installed Memory	48 GB	48 GB
GPU Make	NVIDIA®	NVIDIA®
GPU Model	Tesla 1060	Tesla C2070
GPU Processing Cores	240	448
GPU Shared Memory	4 GB	6 GB

As discussed in [3], single precision computing is acceptable for AFIT RNR processing. There is no loss of target estimation capability using single precision versus double precision. All computations and results discussed in this paper are based on single precision to take advantage of the memory and processing time savings afforded by single precision processing.

The test procedure for this element of the test effort is as follows:

1) Update the algorithm to include FFT segmentation. Determine processing time on both multi-core PCs without GPU computing.
2) Modify the algorithm for GPU computing using MATLAB®'s GPU interface. Determine processing time on both GPU equipped multi-core PCs.
3) Modify the algorithm for GPU computing using Jacket®'s GPU interface. Determine processing time on both GPU equipped multi-core PCs.

MATLAB® has developed a GPU interface as part of the parallel computing toolbox. A number of GPU specific commands have been created to pass CPU variables to the GPU to perform computations on the GPU and then gather the results back to the CPU.

Another company, AccelerEyes®, has developed a product called Jacket® that claims to be better than the parallel computing toolbox in MATLAB®. Jacket® allows MATLAB® users to interface with the GPU without getting into the low-level programming details. Jacket® supports many MATLAB® functions to make modifying existing algorithms for GPU computing fairly seamless. Both Jacket® and MATLAB®'s parallel computing toolbox will be used to find the best solution to simultaneous range and velocity processing in the AFIT RNR.

VI. RESULTS

After configuring the AFIT RNR 2D processing algorithm for implementation on the GPU, the algorithm was applied to the sample transmit and receive data according to the test procedure. As can be seen in Table I, the two computers used for the 2D processing share similar specifications. The first computer, however, has a faster processor leading to faster

TABLE II

SUMMARY OF COMPUTER 1 PROCESSING TIMES PER REFERENCE VELOCITY

Step	Description	Reference Signal Time (s)	Correlation Time (s)	Total Time (s)
1	No GPU	1.49	8.27	9.76
2	MATLAB®	1.55	3.72	5.27
3	Jacket®	1.92	9.42	11.34

TABLE III

SUMMARY OF COMPUTER 2 PROCESSING TIMES PER REFERENCE VELOCITY

Step	Description	Reference Signal Time (s)	Correlation Time (s)	Total Time (s)
1	No GPU	1.78	10.23	12.01
2	MATLAB®	1.70	2.61	4.31
3	Jacket®	1.98	8.73	10.71

TABLE IV

SUMMARY OF 2D PROCESSING TIMES FOR 25 REFERENCE VELOCITIES

Step	Description	Computer 1 Time (minutes)	Computer 2 Time (minutes)
1	No GPU	4.07	5.00
2	MATLAB®	2.20	1.80
3	Jacket®	4.75	4.46

CPU operations. The second computer is outfitted with a higher-end GPU model and resulted in faster GPU operations. This explains why, when comparing Table II to Table III, the first computer performs the 2D processing faster when only the local CPUs are used. Conversely, the second computer performs the 2D processing faster when the GPU is tasked with the majority of the signal processing.

Another factor affecting the speed of the 2D processing is the GPU interface used. Table IV reveals that the MATLAB® GPU implementation is faster than Jacket® for this 2D processing algorithm. MATLAB® is performs faster because of its inherent speed in matrix math. The FFT segments were placed in matrix format to process many FFTs simultaneously, thus reducing the number of loops required to process all FFT segments for correlation. Jacket®, on the other hand, has a very efficient *gfor* loop not available on the MATLAB® interface, but memory allocation challenges did not allow for efficient matrix FFT calculations using Jacket®.

Tables II and III also reveal the fact that reference signal generation accounts for approximately 25 to 30% of the overall 2D processing time. This signal generation time can be eliminated by using a template playback in place of the thermal noise source [13]. A template playback strategy involves building a digital noise transmit signal and using a digital-to-analog converter (DAC) in place of the thermal noise generator. This template playback approach introduces periodicity and signal randomness, but the desired low probability of intercept attribute can be preserved using strategies as discussed in [14].

Generating the reference signal bank *a priori* would significantly reduce the 2D processing time of the AFIT RNR, but it would require a significant amount of memory to store all the reference signals. The current algorithm steps through a vector of reference velocities. For each reference velocity, it builds a single reference signal and performs the cross-correlation with the receive signal. Once the cross correlation is complete and stored, the algorithm clears the reference signal from memory and generates a new reference signal based on the next reference velocity. This iterative process takes time but minimizes the number of lengthy signals that must be stored in memory.

Although the reference signal generation time is a major factor in the overall processing time, the correlation processing time holds the majority and is still prohibitively long. Improvements to the correlation algorithm need to be made for this algorithm to be applied in a practical radar application.

VII. CONCLUSION

Decreasing the simultaneous range-velocity processing time of the AFIT RNR by more than an order of magnitude from $T_p \approx 42$ minutes to $T_p < 2$ minutes is a significant improvement. Unfortunately, considerable effort remains to bring the processing time to near real-time. Simultaneous range and velocity processing has its advantage in that it can separate two targets traveling at different velocities within the same range bin. This characteristic is highly desired in many RNR niche applications. Future research efforts in the AFIT RNR will continue to set near-real time 2D processing as a primary objective.

ACKNOWLEDGMENT

The authors would like to thank Dr. Meller from the Telecommunications Research Institute in Poland for his contributions and support to our implementation of FFT segmentation.

The views expressed in this article are those of the authors and do not reflect the official policy or position of the U.S. Air Force, Department of Defense, or the U.S. Government.

REFERENCES

[1] A. Gonzalez, J. A. Belloch, F. J. Martinez, P. Alonso, V. M. Garcia, E. S. Quintana-Orti, A. Remon, and A. M. Vidal, "The impact of the multi-core revolution on signal processing," *Waves*, vol. 2, pp. 74–74–85, 2010.

[2] A. Fasih and T. Hartley, "GPU-accelerated synthetic aperture radar backprojection in CUDA," in *Radar Conference, 2010 IEEE*, 2010, pp. 1408–1413.

[3] T. J. Thorson and G. A. Akers, "Investigating the use of a binary ADC for simultaneous range and velocity processing in a random noise radar," 2011.

[4] A. Schmitt, "Radar imaging with a network of digital noise radar systems," Master's Thesis, Air Force Institute of Technology, 2009.

[5] R. M. Narayanan and M. Dawood, "Doppler estimation using a coherent ultrawide-band random noise radar," *Antennas and Propagation, IEEE Transactions on*, vol. 48, no. 6, pp. 868–878, 2000.

[6] J. Lievsay and G. Akers, "Moving target detection via digital time domain correlation of random noise radar signals," in *Radar Conference (RADAR), 2011 IEEE*, May 2011, pp. 784 –788.

[7] S. R. J. Axelsson, "Generalized ambiguity functions for ultra wide band random waveforms," in *Radar Symposium, 2006. IRS 2006. International*, 2006, pp. 1–4.
[8] A. W. Rihaczek, "Delay-Doppler ambiguity function for wideband signals," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. AES-3, no. 4, pp. 705–711, 1967.
[9] J. R. Lievsay, "Simultaneous range/velocity detection with an ultra-wideband random noise radar through fully digital cross-correlation in the time domain," Master's Thesis, Air Force Institute of Technology, 2011.
[10] K. Kulpa, *Continuous Wave Radars, Monostatic, Multistatic and Network*, ser. Advances in Sensing with Security Applications. Springer Netherlands, 2006, vol. 2, pp. 215–242.
[11] M. Meller, "Some aspects of designing real-time digital correlators for noise radars," in *Radar Conference, 2010 IEEE*, 2010, pp. 821–825.
[12] A. Oppenheim and R. Schafer, *Discrete-time Signal Processing*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2009.
[13] P. J. Collins and I. John A. Priestly, "Trading spectral efficiency for system latency in true-random noise radar network through template-replacy diversity," *Waveform Diversity and Design Conference*, 2012.
[14] J. A. Priestly, "AFIT NoNET enhancements: Software model development and optimization of signal processing architecture," Master's Thesis, Air Force Institute of Technology, 2011.