

# A Single-Chip Narrow-Band Frequency-Domain Excisor for a Global Positioning System (GPS) Receiver

Paul T. Capozza, *Member, IEEE*, Brian J. Holland, Thomas M. Hopkinson, *Member, IEEE*, and Roberto L. Landrau, *Member, IEEE*

**Abstract**—In recent years, we have witnessed the rapid adoption of the *Department of Defense's Global Positioning System (GPS)* for navigation in a number of military and civilian applications. Unfortunately, the low-power GPS signal is susceptible to interference. This paper presents a novel VLSI architecture that removes narrow-band signals from the wide-bandwidth GPS spectrum. The interference suppression technique employed is frequency-domain excision. The single-chip frequency-domain excisor transforms the received signal (GPS signal + noise + interference) to the frequency domain, computes signal statistics to determine an excision threshold, removes all spectral energy exceeding that threshold, and restores the remaining signal (GPS signal + noise) to the temporal domain. The heart of this VLSI implementation is an on-chip 256-point fast Fourier transform processor that operates at 40 million complex samples per second. It processes 12-bit (for each  $I$  and  $Q$ ) sampled complex data. The 1.5-million-transistor chip was fabricated in 0.5- $\mu\text{m}$  CMOS triple metal technology and is fully functional.

**Index Terms**—Adaptive narrow-band filter, frequency-domain excision, frequency-domain interference suppression, GPS.

## I. INTRODUCTION

A GLOBAL Positioning System (GPS) receiver computes its position, velocity, and time solution by processing navigation signals from a constellation of GPS satellites. These signals arrive at the receiver at a very low-power level, typically 20–30 dB below the receiver's thermal noise level. By integrating over time a receiver can exploit the inherent spread-spectrum processing gain of the navigation signals to provide positive postcorrelation signal-to-noise ratio (SNR). Current generation receivers typically employ a radio-frequency (RF) front-end and digitization implementation that provides very low dynamic range. Their adaptive analog-to-digital (A/D) conversion schemes provide a modest level of protection against low-power, in-band interference energy. Unfortunately, this falls far short of the level of protection required in any system that depends on reliable GPS service and must operate in even a modestly challenging electronic interference environment. This class of user includes most military and many civil applications. The realization of this vulnerability leads to the requirement for a new generation of GPS user equipment with

significantly greater interference protection. In particular, an “all digital” approach offers promising performance.

One class of interfering signal that is particularly troublesome is narrow (or partial) band interference. A  $-100$  dBm single-tone (CW) signal presented to the antenna of a typical commercial receiver is sufficient to prevent GPS acquisition. Frequency-domain implementations of adaptive narrow-band filters (ANBF's) have been shown to provide excellent protection against narrow-band interference [1]. In this paper, we describe a single-chip custom VLSI implementation of an ANBF in the form of a frequency-domain interference suppressor (FDIS). The FDIS IC accepts 12-bit inputs for each  $I$  and  $Q$  sample and maintains an internal computational precision of 20 bits. It operates at a continuous data throughput rate of up to 10 million complex samples per second (MCSPS) and is suitable for high-performance C/A code GPS receiver applications. When included in a properly designed GPS receiver, FDIS provides protection against a single tone narrow-band interfering signal with an interference-to-signal ratio (ISR) of up to 97 dB. Furthermore, this technology provides protection against a broad class of interference environments including multiple tones, swept CW, and pulsed CW. An important consideration in any time- or frequency-based ANBF implementation is the resultant degradation of the desired signal. A key feature of FDIS is its inherent ability to accurately identify narrow-band interference and eliminate it with minimal adverse impact on the GPS signal. When no such interference is present, FDIS introduces negligible insertion loss. When narrow-band interference exists in concert with broad-band interference, FDIS removes the narrow-band interference only. It is ideally suited to combined anti-jam (AJ) implementations where FDIS is integrated with a spatial processor aimed at removing broad-band interference [2].

The following sections will describe FDIS processing for the GPS application and provide a detailed description of the excision algorithm implemented. The final section will motivate and describe the VLSI architecture along with practical design considerations.

## II. FREQUENCY-DOMAIN INTERFERENCE SUPPRESSION PROCESSING

Frequency-domain interference suppression processing in its simplest form is shown in Fig. 1. The fast Fourier transform

Manuscript received July 30, 1999; revised October 11, 1999. This work was supported by the GPS Joint Program Office.

The authors are with The MITRE Corporation, Bedford, MA 01730 USA.

Publisher Item Identifier S 0018-9200(00)00535-7.



Fig. 1. Simple FDIS processing.

(FFT) translates the input signal into the frequency domain, an excisor block modifies the spectrum to eliminate narrow-band interference, and an inverse FFT transforms the modified spectrum back to the time domain. The desired output signal is the input signal minus the narrow-band interference with a minimal degradation to the signal of interest.

The input signal,  $x(n) = s(n) + j(n)$ ,  $n = 0, 1, \dots, N_p - 1$ , is composed of two components: the desired composite signal  $s(n)$  (GPS signal + AWGN) and the undesired interference  $j(n)$ . The  $k$ th discrete Fourier transform (DFT) output sample for each block of  $N_p$  samples,  $X(k)$ , is given by

$$X(k) = \sum_{n=0}^{N_p-1} x(n)e^{-j2\pi kn/N_p}, \quad k = 0, 1, \dots, N_p - 1. \quad (1)$$

The DFT can distinguish exactly  $N_p$  distinct frequencies  $f_k = 2\pi k/N_p$ ,  $k = 0, 1, \dots, N_p - 1$ . The DFT of a signal that contains frequency components that are not exactly these frequencies exhibits the phenomenon of spectral leakage. In other words, the DFT assumes a periodic extension of a finite sequence that is length  $N_p$  samples. If the  $N_p$  samples of the input sequence are not periodic in the DFT's window of observation, a discontinuity occurs at the DFT's block boundary. This results in spectral leakage, as further elaborated in [3]. For the case of a nonwindowed DFT, the discontinuity is abrupt and the spectral leakage can be significant. To illustrate this effect and to demonstrate the performance of the simple FDIS processing case, Fig. 2 depicts the DFT output for an input signal that represents an additive white Gaussian noise signal in the presence of a single CW interferer

$$x(n) = s(n) + \exp\left[j\left(\frac{2\pi f_j}{N} + \phi\right)\right]. \quad (2)$$

In this case, the interference frequency is set to  $f_j = (f_k + f_{k+1})/2$ . This places the CW tone in the middle of two of the DFT's distinct frequency bins. As can be seen in Fig. 2, the spectral energy of the tone is spread across the entire spectrum. The dashed line indicates a calculated excision threshold. Any frequency location that is above the calculated threshold would be set to zero. Through this operation, the interference energy is removed from the received spectrum. It is obvious that no threshold can be selected that will allow the removal of the interference energy without also removing excessive signal spectrum.

To mitigate the effect of spectral leakage, DFT-based ANBF techniques typically use windowing. Windowing applies a weighting factor to the input signal prior to computing the DFT (Fig. 3). Windowing smooths the discontinuities at the block boundary and therefore lessens the effect of spectral leakage.

The  $k$ th DFT output sample for each block of  $N_p$  samples,  $X(k)$ , for this case is given by

$$X(k) = \sum_{n=0}^{N_p-1} w(n)x(n)e^{-\frac{j2\pi kn}{N_p}}, \quad k = 0, 1, \dots, N_p - 1. \quad (3)$$

The selection of the window coefficients,  $w(n)$ ,  $n = 0, 1, \dots, N_p - 1$ , determines the amount of spectral leakage in the DFT output. To illustrate this, consider the nonwindowed processing, which is equivalent to using a rectangular window. The spectral leakage problem is most easily visualized in the frequency domain. The Fourier transform of the rectangular window is a sinc function with the first sidelobe 13 dB down relative to the main lobe and with subsequent sidelobes that fall off at 6 dB/octave. When the frequency of a signal is not exactly one of the DFT frequencies, the signal energy will be spread across the spectrum proportional to the width of the main lobe and the height of the sidelobes of the window. Selecting a window with lower sidelobes will reduce the amount of spectral leakage. However, a lower sidelobe usually results in a wider main lobe (i.e., reduced spectral resolution).

There are a number of window functions described in [3]. For the GPS application, the objective is to minimize the frequency spreading of each CW tone in order to minimize the number of frequency bins that will be excised. At the same time, one would also want to minimize the degradation of the GPS signal when an interferer is not present. The window selection requires a tradeoff between the reduction in SNR due to the signal attenuation incurred by multiplying the data sequence by a window and the effectiveness of the spectral containment for a CW tone. The degradation of the SNR can be expressed as

$$\frac{\left(\sum_{n=0}^{N-1} w(n)\right)^2}{N \sum_{n=0}^{N-1} w^2(n)}. \quad (4)$$

The SNR loss can be as high as 3 dB for some window functions with extremely low sidelobes. Similarly, frequency containment is proportional to a window's lowest sidelobe level. For example, a minimum four-sample Blackman–Harris window has a  $-92$  dB sidelobe that provides excellent frequency containment for each CW interferer. The low sidelobe levels essentially restrict the spectral leakage to the width (approximately seven frequency bins) of the window's main lobe. This window, however, results in a 3 dB SNR loss due to high signal attenuation at the DFT block transitions. Conversely, a Hamming window which has a  $-40$  dB sidelobe level, provides frequency containment equivalent to approximately three frequency bins for each CW interferer provided the interference level is significantly below the  $-40$  dB sidelobe level. The SNR degradation resulting from a Hamming window is less than 1.36 dB. In Fig. 4, we see the reduction in the number of frequency bins occupied by a CW interferer with a 30 dB interference-to-noise ratio (INR). Since the interference is well above the  $-13$  dB sidelobes of the rectangular window, the spectrum for this case exhibits significant spectral leakage similar to that shown in Fig. 2. The Hamming window and the four-sample Blackman–Harris window have sidelobe levels that are below the interference level. Windowing the input

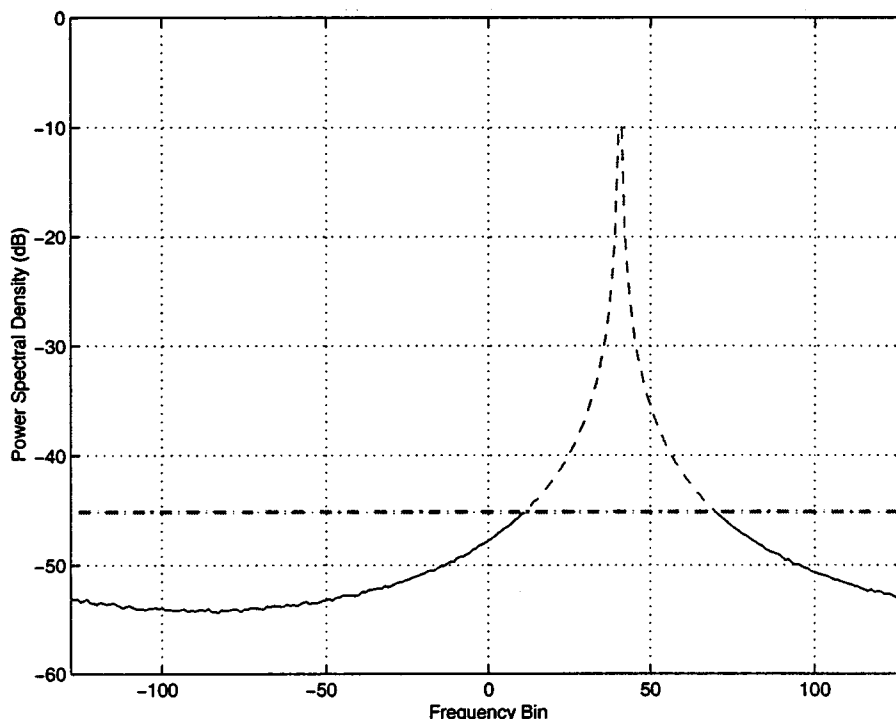


Fig. 2. DFT output for a single interferer without windowing.

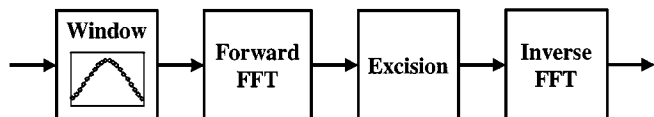


Fig. 3. Windowed FDIS processing.

signal with Hamming or Blackman–Harris reduces the amount of spectrum that must be excised, thus preserving more of the desired signal spectrum. For the intended GPS application, we expect an INR on the order of 65 dB; clearly the Hamming window is not sufficient. A window with similar characteristics to the four-sample Blackman–Harris window is required.

To minimize the degradation of the SNR due to the window while maintaining CW frequency containment, the FDIS processing uses a well-known technique that employs a 50% overlap [3]. The overlap function unfortunately doubles the processing since it requires a second processing path that includes a window, forward FFT, excisor, and inverse FFT, as shown in Fig. 5. The advantage of the overlap processing is that it reduces the effect of the signal attenuation from the window on the output SNR. Each path in the processing chain produces one-half of the usable output sequence as shown in Fig. 6. This figure shows the contribution of each of the two data paths to the overall result. It takes into account the windowing but for this simple example does not include excision. The input signal is shown in Fig. 6(a). The input signal is broken into two paths offset by  $N_p/2$  samples with respect to each other [Fig. 6(b) and (c)]. The  $N_p/4$  samples at the beginning and end of each inverse FFT output block are discarded, leaving the middle  $N_p/2$  samples indicated by the solid line in Fig. 6. The  $N_p$  samples from each path are then combined to form the final  $N_p$  samples. The resultant output waveform is shown in Fig. 6(d).

The beneficial effect of overlap processing in reducing the signal distortion can be seen from Fig. 6 when the output signal is compared to the signal from one of the individual paths. For the four-sample Blackman–Harris window, the SNR degradation after a 50% overlap is reduced to 0.6 dB from 3 dB. The next section will describe the FDIS IC's excision algorithm in detail.

### III. $N$ -SIGMA EXCISION ALGORITHM

Our objective is to remove narrow-band interferers from the wide-bandwidth signal of interest. By definition, these interferers occupy relatively few frequency bins, and their amplitudes are above the noise floor of the FFT. There are a number of potential narrow-band excision algorithms [4]. The following section describes the  $N$ -sigma algorithm implemented on the FDIS IC.

The  $N$ -sigma algorithm is an adaptive algorithm that excises a variable number of the frequency bins from each FFT data block. The percentage of the spectrum that is excised is dependent on the number and the magnitude of the interferers that are present. The adaptive nature of the  $N$ -sigma algorithm allows it to eliminate the narrow-band interferers without excising other frequency components of the signal unnecessarily. The  $N$ -sigma algorithm takes advantage of the fact that the distribution of the (GPS signal + noise) is approximately Gaussian in the absence of narrow-band interference. As interferers are added to the signal, the distribution of the spectrum becomes distorted—the portion of the signal unaffected by the interference retains a Gaussian shape, and the frequency components affected by the interference are distributed above the desired signal components. The effect of this distortion on the distribution is illustrated in Fig. 7. The

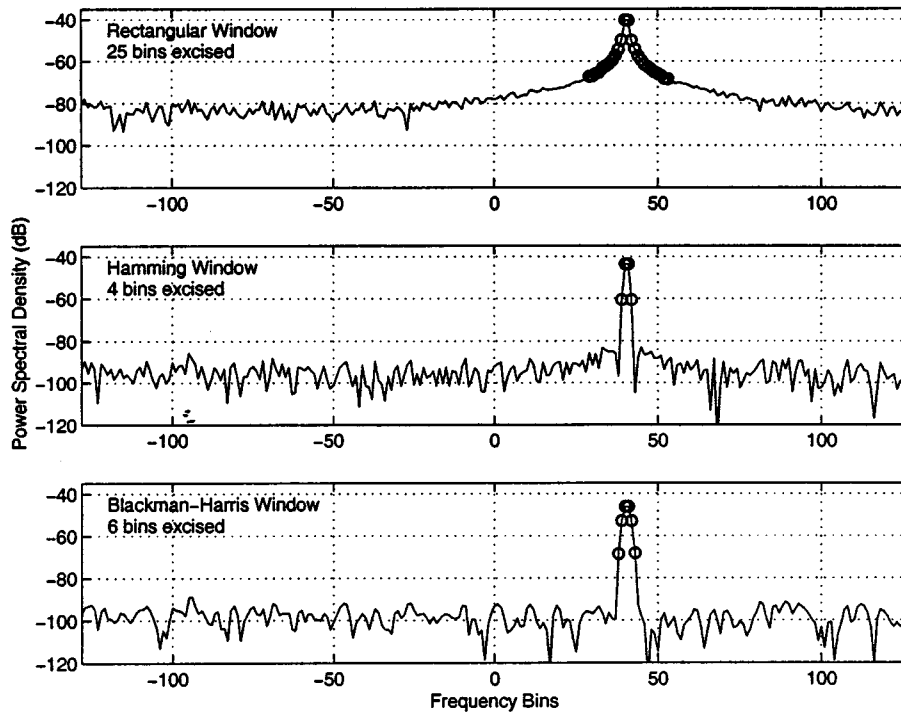


Fig. 4. Window comparisons: (top) rectangular, (middle) Hamming, and (bottom) four-sample Blackman-Harris.

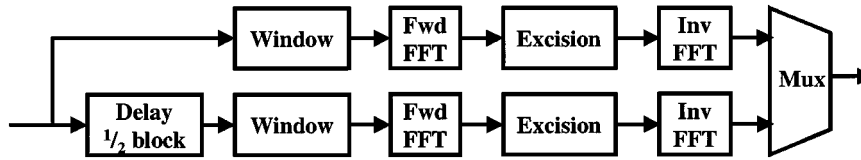


Fig. 5. 50% overlap and select FDIS processing.

histogram in Fig. 7(a) corresponds to the magnitudes of the FFT output,  $10 \cdot \log(|X(k)|)$ , without interference present, and the histogram in Fig. 7(b) corresponds to  $10 \cdot \log(|X(k)|)$  with five interferers present. As illustrated in the figure, in the absence of interference, most of the signal is less than two standard deviations above the mean. As the number of interferers and/or the total interference power is increased, the mean is shifted with respect to the center of the Gaussian portion of the signal, and the standard deviation is increased by the dominating interfering frequency components. This distortion requires that the  $N$ -sigma algorithm select a threshold that is closer to the mean in order to excise the frequency components associated with the interference.

A conceptual block diagram of the  $N$ -sigma algorithm is illustrated in Fig. 8. For each block of 256 complex data points, the  $N$ -sigma algorithm computes the magnitude in decibels for each of the frequency bins and then calculates the standard deviation ( $\sigma_{\text{calc}}$ ) and mean ( $\mu_{\text{calc}}$ ) of the magnitudes for that block. The excision threshold is computed by first comparing  $\sigma_{\text{calc}}$  to four levels,  $\sigma_{l0}$  to  $\sigma_{l3}$ . Based on the comparisons, one of five possible scale factors is used as the value of  $N$  for the threshold calculation

$$\text{Excision}_{\text{threshold}} = (\mu_{\text{calc}} + N \cdot \sigma_{\text{calc}}). \quad (5)$$

A larger  $\sigma_{\text{calc}}$  indicates the presence of interferers and requires the selection of a smaller scale factor  $N$  to maintain the threshold at the top of the noise floor. The five choices for  $N$ , as well as  $\sigma_{l0}$ – $\sigma_{l3}$ , are user programmable. This allows for the adaptation of the FDIS IC to other spread-spectrum communication applications. Each frequency bin is then compared to the excision threshold, and if the magnitude of the frequency bin is greater than the threshold, its value is set to zero.

The adaptive nature of the  $N$ -sigma algorithm is illustrated in Fig. 9. For a single interferer,  $N$ -sigma selects  $2\sigma$  as the threshold—resulting in the nulling of four frequency bins. For the five-interferer case, the threshold is lowered to  $\sigma/4$ —resulting in the nulling of 35 frequency bins. Note that the  $N$ -sigma algorithm adjusted the excision percentage from 1.5% to 13.7% of the spectrum as the spectral occupancy of the interference increased. As a result, it provided excellent interference rejection performance while preserving as much of the desired signal as possible.

## IV. VLSI ARCHITECTURE

### A. Overview

As described in Section II, the frequency-domain interference processing algorithm requires transforming the signal of interest

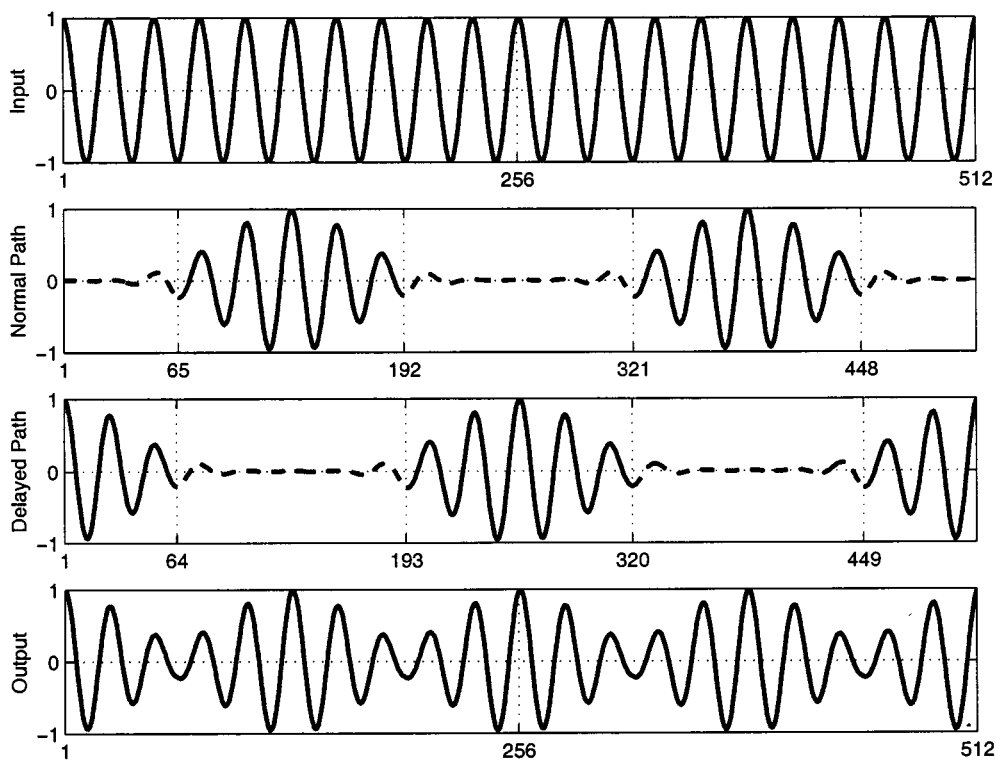


Fig. 6. (a) Input sequence, (b) *normal path*—windowed input sequence, (c) *delayed path*—input sequence delayed  $N_p/2$  samples and windowed, and (d) combined output sequence.

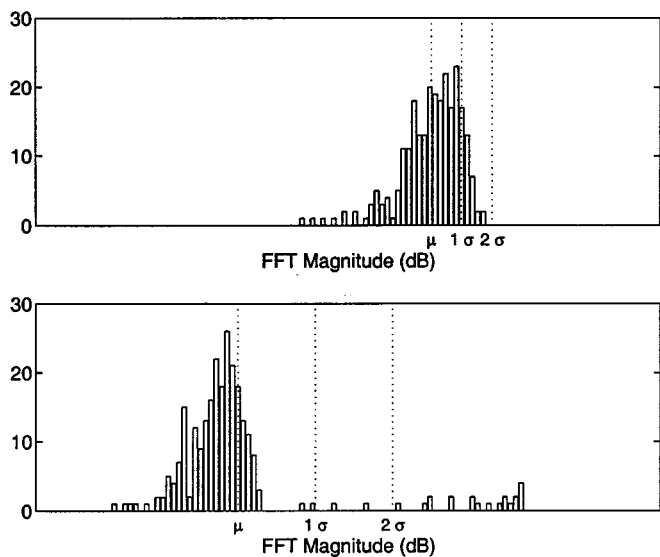


Fig. 7. Histogram of FFT block output: (a) no interferers and (b) five CW interferers.

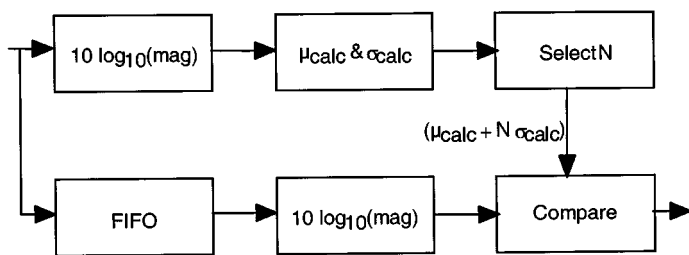


Fig. 8.  $N$ -sigma block diagram.

from the time domain to the frequency domain, applying the excision algorithm, and transforming the result back to the time domain.

An FFT implementation of the transform requires on the order of  $N_p \log(N_p)$  operations for each block ( $N_p$  samples) of data. The most reasonable tradeoff between area and speed was to multiplex the four FFT operations shown in the conceptual processing diagram of Fig. 5, using a pipelined FFT architecture. An easily extendable  $N_p = 256$  point FFT core has been designed that supports a continuous throughput of 40 MCSPS. This results in an overall FDIS processor that performs more than 2 billion operations per second. The FFT core will be described in the following sections along with the hardware implementation of the  $N$ -sigma excisor.

### B. Pipelined FFT Architecture

The FFT contains a significant portion of the computational resources of the FDIS IC and accounts for approximately one-half of the total area. Our goal was to design an FFT architecture that could meet the current processing requirements of the 2-MHz bandwidth C/A code—the standard positioning system (SPS) GPS signal [5]—and could also be easily extended in the future for a higher bandwidth P-code receiver (20-MHz bandwidth).

We selected a pipelined radix-2 FFT architecture because it offers efficient memory usage, low latency, and a high continuous throughput rate, and is easily scalable for future applications [6]. The architecture contains eight ( $\log_2(N_p)$ ) cascaded processors referred to as *stages*, as shown in Fig. 10. All stages operate simultaneously, with every stage performing 128

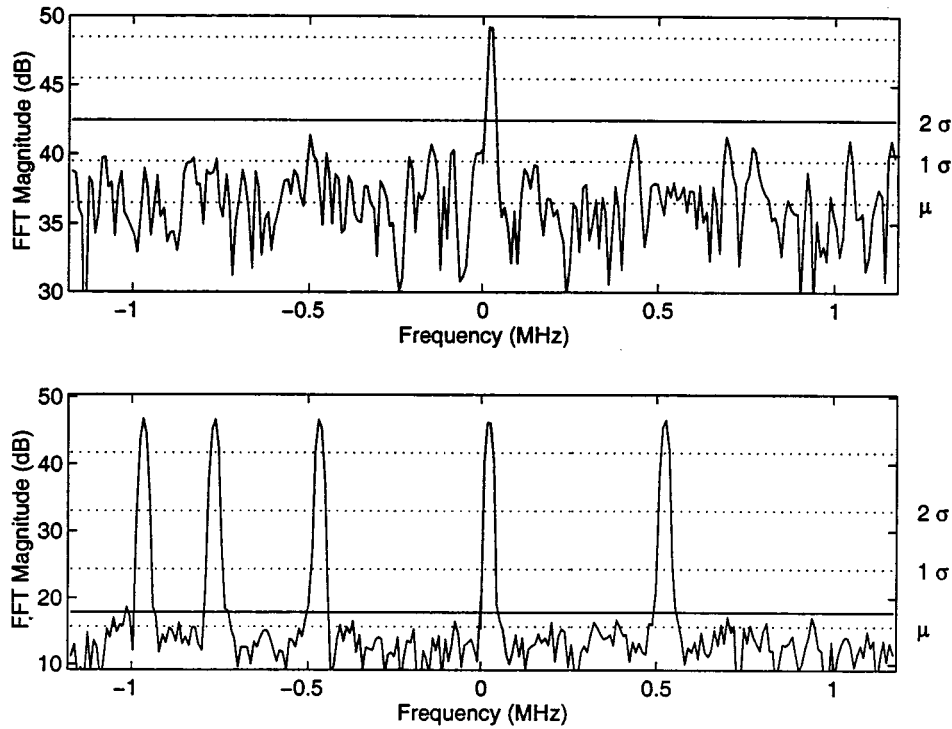


Fig. 9. (top)  $N$ -sigma exciting a single interferer and (bottom)  $N$ -sigma exciting five interferers.

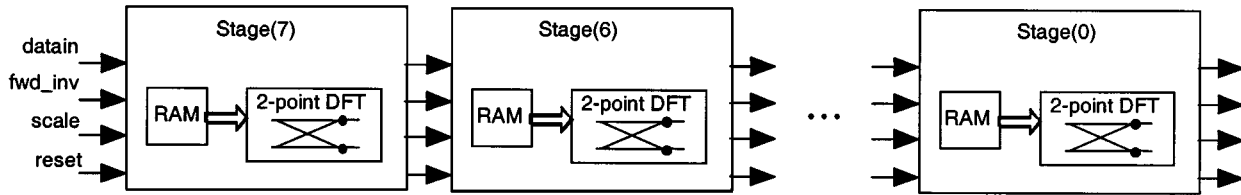


Fig. 10. Pipelined FFT architecture.

$(N_p/2)$  radix-2 FFT butterfly operations per block of data. Unlike other common implementations, this implementation has hardware dedicated to performing the radix-2 butterfly computation and the data shuffling for each stage.

The input to the FFT core is a stream of complex numbers, represented in fixed-point notation using 12 bits. The FFT core uses an oversized internal word length of 20 bits (input word size +  $\log_2(N_p)$ ). This provides sufficient dynamic range to allow for the growth of coherent signals (interference sources) through the FFT at a rate of one bit per stage without sacrificing numerical precision. We considered a traditional floating-point and a convergent block floating-point representation, as described in [7]. Both of these representations, for a given number of bits of numerical representation, trade off precision for increased dynamic range. Neither of these floating-point representations was appropriate for GPS applications, since a loss of precision corresponds to an unacceptable degradation of the SNR. The improved precision accuracy of an oversized internal word length is necessary and prevents us from taking advantage of the potential hardware savings obtained by using a floating-point representation with a smaller number of bits.

In order to multiplex the four FFT operations, we have designed an FFT core that can continuously process forward or

inverse FFT's concurrently. The forward and inverse DFT equations for a block of  $N_p$  samples

$$X[k] = \sum_{n=0}^{N_p-1} x[n] e^{-j2\pi nk/N_p} \quad \text{Forward DFT} \quad (6)$$

$$x[n] = \left(\frac{1}{N_p}\right) \sum_{k=0}^{N_p-1} X[k] e^{+j2\pi nk/N_p} \quad \text{Inverse DFT} \quad (7)$$

differ only by a scale factor and a change of sign in the exponent of the complex coefficients—also called *weights*. The scaling is easy to implement by shifting the output of the inverse FFT to the right and rounding accordingly. Since the weights for the forward and inverse FFT's are computed *a priori* and stored in on-chip ROM's, the change of exponent in the weights can be accommodated easily by accessing a different section of the ROM's. To process data continuously, independently of the FFT direction, and without the need to reset or flush the stage pipeline, we propagate control signals from one stage to the next along with the data. This allows a stage to process a transform in one direction on a data block while an adjacent stage can perform a transform in the other direction.

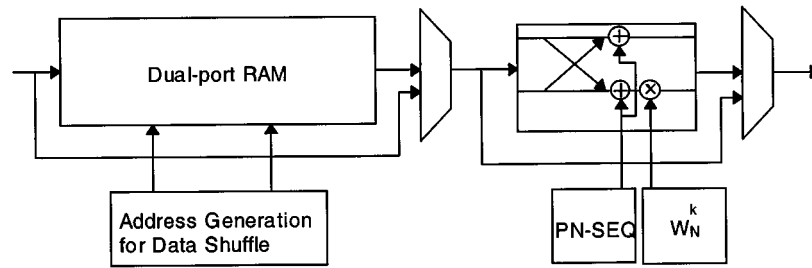


Fig. 11. Block diagram for an FFT stage.

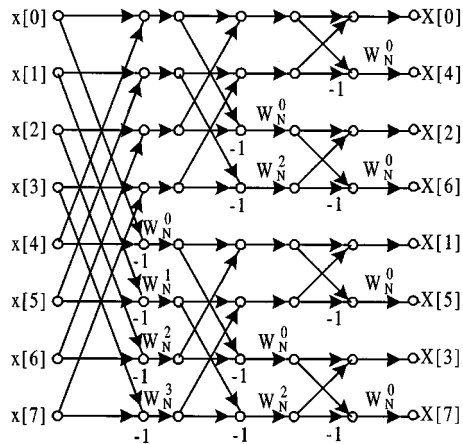


Fig. 12. Eight-point FFT flowgraph.

### C. FFT Components

As described above, the FFT core consists of  $\log_2(N_p)$  stages operating simultaneously. Each stage includes a random-access memory (RAM), an address generator, a read-only memory (ROM), a computational engine (butterfly processor), a random-number generator (provides for unbiased rounding), and multiplexers to facilitate testing as shown in Fig. 11.

The FFT architecture uses a dual-port RAM, a controller, and an address generator to shuffle data from stage to stage using a read-modify-write sequence. To reduce the memory requirements, we took advantage of the fact that in an FFT, each value that enters a particular stage is used only once by its butterfly processor. When a value is read from the memory at a particular stage, a memory location becomes available to store the result from the previous stage. On the first half of every clock cycle, the stage controller reads two complex values from the memory, leaving two locations free. The butterfly performs a two-point DFT on those two values and passes the results to the next stage for further buffering, reordering, and processing. On the second half of the clock cycle, a new pair of complex values—the results of the two-point DFT from the preceding stage—are written into the two locations that were freed during the read cycle.

### D. Address Generation

For simplicity, we will use the case of an eight-point FFT to illustrate the address generation. Refer to Fig. 12 for a flowgraph of the decimation-in-frequency decomposition of an eight-point FFT.

Let us assume that the data entering the FFT are in natural (time-sequential) order

$$x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8], x[9], \dots$$

where the samples  $x[0]$ – $x[7]$  constitute the first block of data. We will use  $x'[n]$  to denote the output of the first stage,  $x''[n]$  for the output of the second stage, and  $X[n]$  for the output of the third stage (the final output of the FFT). The first block of data will be written to the memory in the first stage in natural order (e.g.,  $x[0]$  will be written to memory location 0,  $x[1]$  to memory location 1,  $\dots$ ,  $x[7]$  to location 7). The flowgraph shows that the butterfly processor in the first stage should compute

$$\begin{aligned} x'[0] &= (x[0] + x[4]) \\ x'[4] &= (x[0] - x[4])W_8^0 \\ x'[1] &= (x[1] + x[5]) \\ x'[5] &= (x[1] - x[5])W_8^1 \\ x'[2] &= (x[2] + x[6]) \\ x'[6] &= (x[2] - x[6])W_8^2 \\ x'[3] &= (x[3] + x[7]) \\ x'[7] &= (x[3] - x[7])W_8^3 \end{aligned}$$

where  $W_N = e^{-j(2\pi/N)}$ . Note that the input values  $x[n]$  needed to compute the results of the first stage  $x'[n]$  are spaced  $N_p/2$  samples apart (e.g., points 0 and  $N_p/2$ , 1 and  $N_p/2 + 1$ ,  $\dots$ ). Therefore, the stage controller must generate a new sequence of addresses when the first stage is processing the first block of data ( $x[0]$ – $x[7]$ ). The situation complicates even further due to the fact that, while the first stage processes the first block, the second block of data is entering the FFT. The controller must store the incoming samples in the locations that are free. As a result, the second block of data will not be stored in the same order as the first block of data. After the last pair of values of the first block of data is processed, the memory will contain all the values for the second block of data ( $x[8]$  to  $x[15]$  for the case of an eight-point FFT) stored as follows:

$$0, \frac{N_p}{2}, 1, \frac{N_p}{2} + 1, 2, \frac{N_p}{2} + 2, \dots, i, \frac{N_p}{2} + i, \dots, \frac{N_p}{2} - 1, N_p - 1$$

for  $0 \leq i \leq \frac{N_p}{2}$ . An example using the eight-point FFT is listed in Table I. When the last point of the second block is stored, the controller must start generating another sequence of addresses

TABLE I  
STORAGE OF DATA POINTS IN THE FIRST STAGE FOR THREE CONSECUTIVE  
BLOCKS OF AN EIGHT-POINT FFT

Memory Location	First Block	Second Block	Third Block
0	x[0]	x[8]	x[16]
1	x[1]	x[10]	x[20]
2	x[2]	x[12]	x[17]
3	x[3]	x[14]	x[21]
4	x[4]	x[9]	x[18]
5	x[5]	x[11]	x[22]
6	x[6]	x[13]	x[19]
7	x[7]	x[15]	x[23]

to start processing the second block and storing the values for the third block using the pattern

$$0, \frac{N_p}{4}, \frac{N_p}{2}, \frac{N_p}{2} + \frac{N_p}{4}, 1, 1 + \frac{N_p}{4}, 2, 2 + \frac{N_p}{4}, 3 + \frac{N_p}{4}, \dots, \frac{3N_p}{4} - 1, N_p - 1.$$

This pattern is similar to the one used previously, with the address bits rotated to the right by one. As expected, the third block of data will also be stored out of order, using a different sequence. To retrieve the third block, the address generator needs to provide a sequence similar to the previous one, with the address bits rotated to the right by one more position. In general, the stage controller will continue to rotate the base address by one additional position every time it sees a block boundary. The pattern continues until the bits have been rotated  $\log_2(N_p) - 1$  times, since rotating by  $\log_2(N_p)$  is equivalent to no rotation at all. Refer to Table II for a list of the sequences needed to store, shuffle, and retrieve the data in the correct order for an eight-point FFT.

The address permutations can be generated with a circuit that counts from zero to  $N_p - 1$ , followed by a circuit that performs a circular rotation of the counter output bits as a function of the block number. For the first block, no rotation is needed. For the second block, the bits should be rotated to the right by one bit. As the block number increases, the output of the counter should be rotated by an additional bit, until the number of bits to rotate equals the total number of bits in the address (which is equivalent to no rotation). Refer to Fig. 13 for a block diagram.

The address generation for the subsequent stages is different. When the first block of data enters the second stage, it is no longer in natural order, due to the shuffling that occurs in the first stage. Instead, the data arrive according to the sequence generated by the address generator in the first stage

$$0, \frac{N_p}{2}, 1, \frac{N_p}{2} + 1, 2, \frac{N_p}{2} + 2, \dots, i, \frac{N_p}{2} + i, \dots, \frac{N_p}{2} - 1, N_p - 1$$

for  $0 \leq i \leq N_p/2$ . For an eight-point FFT, the sequence is

$$x'[0], x'[4], x'[1], x'[5], x'[2], x'[6], x'[3], x'[7].$$

The values are stored in memory in the same sequence they enter the stage. Refer to Table III for a list of the values and the locations in which they are stored for the eight-point FFT

TABLE II  
PERMUTATIONS OF ADDRESS GENERATOR SEQUENCES FOR THE FIRST STAGE  
OF AN EIGHT-POINT FFT

First Permutation		Second Permutation		Third Permutation	
DEC	BIN	DEC	BIN	DEC	BIN
0	000	0	000	0	000
1	001	4	100	2	010
2	010	1	001	4	100
3	011	5	101	6	110
4	100	2	010	1	001
5	101	6	110	3	011
6	110	3	011	5	101
7	111	7	111	7	111

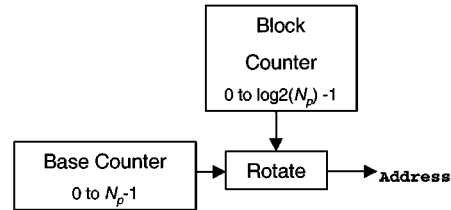


Fig. 13. Address generator circuit for the first stage.

example. To compute  $x''[n]$ , the results of the second stage, the butterfly must calculate

$$\begin{aligned} x''[0] &= (x'[0] + x'[2]) \\ x''[2] &= (x'[0] - x'[2])W_8^0 \\ x''[1] &= (x'[1] + x'[3]) \\ x''[3] &= (x'[1] - x'[3])W_8^2 \\ x''[4] &= (x'[4] + x'[6]) \\ x''[6] &= (x'[4] - x'[6])W_8^0 \\ x''[5] &= (x'[5] + x'[7]) \\ x''[7] &= (x'[5] - x'[7])W_8^2. \end{aligned}$$

Note that the samples of  $x'[n]$  must be supplied as pairs of numbers spaced  $N_p/4$  apart. To retrieve these values from memory in the proper order for the butterfly, the stage controller must generate the sequence of addresses listed in Table IV. For the third block, the controller needs to generate addresses in natural order again, repeating the cycle.

We implemented the address generator with a simple circuit that consists of a modulo- $N_p$  counter and a selector that swaps the MSB and the LSB of the address when storing odd-numbered blocks and processing even-numbered blocks. A conceptual block diagram of the circuit is shown in Fig. 14. The two permutations (*Normal* and *Reversed*) are listed in Table IV. Tracking the first data blocks through any of the subsequent stages reveals that the address generation for all stages except the first one are identical.

#### E. Memory Requirements and Latency

Based on the data shuffle and address generator circuits described previously, the first butterfly cannot start processing until the sample  $x[N_p - 1]$  is stored. For this reason, the first stage needs to buffer a complete block ( $N_p$  samples). The total memory requirement for the first stage of our 256-point FFT is 256 words

TABLE III  
STORAGE OF DATA POINTS IN THE SECOND STAGE FOR THREE CONSECUTIVE  
BLOCKS OF AN EIGHT-POINT FFT

Memory Location	First Block	Second Block	Third Block
0	x'[0]	x'[8]	x'[16]
1	x'[4]	x'[10]	x'[20]
2	x'[1]	x'[9]	x'[17]
3	x'[5]	x'[11]	x'[21]
4	x'[2]	x'[12]	x'[18]
5	x'[6]	x'[14]	x'[22]
6	x'[3]	x'[13]	x'[19]
7	x'[7]	x'[15]	x'[23]

by 40 bits (20 for the real and 20 for the imaginary part). For the second stage, the butterfly cannot start processing until the sample  $x'[\frac{N_p}{4}]$  is received. Since the data does not arrive in natural order, this sample is not received until  $N_p/2$  samples are stored. Based on the circuit described previously, this stage must wait until a complete block ( $N_p$  samples) is received, making the memory requirements similar to the first stage. The situation simplifies starting at the third stage. The butterfly needs the sample  $x''[\frac{N_p}{8}]$  to start processing. This point arrives as point number  $N_p/4$ , which allows the butterfly to start processing after  $N_p/2$  samples have been received. The memory requirement for the third stage is  $N_p/2$  instead of  $N_p$ . Similarly, the requirements are reduced by one-half at every subsequent stage.

#### F. $N$ -Sigma Excisor

The  $N$ -sigma excisor calculates the statistics, mean ( $\mu_{\text{calc}}$ ) and variance ( $\sigma_{\text{calc}}^2$ ), for each FFT block. The statistics are used to compute the threshold as described in Section III. In order to generate the block statistics, the excisor calculates the magnitude in decibels for each complex FFT output,  $10 \cdot \log(|X(k)|)$ . A logarithmic scale is used to minimize the distortion of the spectrum statistics for the combined GPS signal and thermal noise  $s(n)$  in the presence of interference and to reduce the numerical precision required to represent  $(|X(k)|)$  for the subsequent computation of the block statistics. To simplify the hardware implementation, several approximations are used for the calculation of  $10 \cdot \log(|X(k)|)$  and the statistics. The reduced accuracy yielded by these approximations proved suitable for identifying CW tone interferers from the combined GPS signal and thermal noise.

The magnitude is approximated as follows:

$$|X(k)| \approx \left[ \frac{\max(\text{Re } X(k), \text{Im } X(k))}{4} + \frac{\min(\text{Re } X(k), \text{Im } X(k))}{4} \right]. \quad (8)$$

The above approximation has a 0.6% accuracy on average with a maximum error of 11.6% at  $(\pi/4, 3\pi/4, 5\pi/4, 7\pi/4)$ . The second approximation for logarithmic scale conversion is simplified by noting that the logarithm calculation

$$10 \cdot \log_{10}(|X(k)|) = 10 \cdot \log_{10}(2) \cdot \log_2(|X(k)|) \quad (9)$$

can be reduced to  $\approx 3 \cdot \log_2(|X(k)|)$ . The magnitude calculation is implemented using simple shift and add operators instead of area intensive multipliers and square root operations. The logarithmic approximation has a 0.5% error on average with a maximum error of 2.9%.

TABLE IV  
PERMUTATIONS OF ADDRESS GENERATOR SEQUENCES FOR SUBSEQUENT  
STAGES OF AN EIGHT-POINT FFT

Normal Permutation		Reversed Permutation	
DEC	BIN	DEC	BIN
0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111

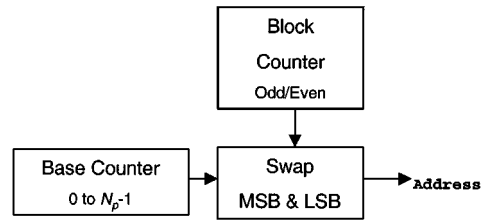


Fig. 14. Address generator circuit for the first stage.

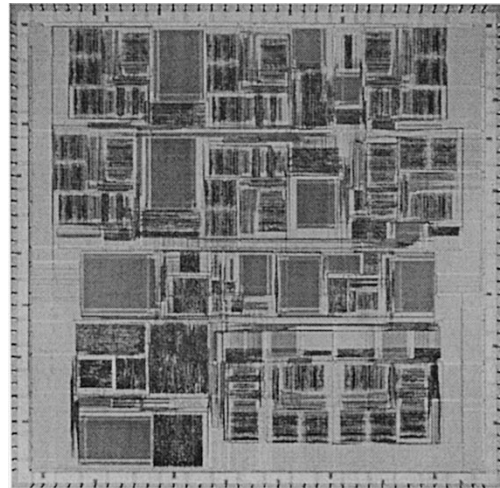


Fig. 15. Die microphotograph of the FDIS IC.

The statistics

$$\mu_{\text{calc}} = \sum_{k=0}^{N_p-1} \frac{(10 \cdot \log(|X(k)|))}{N_p} \quad (10a)$$

$$\sigma_{\text{calc}} = \frac{1}{N_p} \left[ \sum_{k=0}^{N_p-1} (10 \cdot \log(|X(k)|))^2 - \frac{1}{N_p} \left( \sum_{k=0}^{N_p-1} (10 \cdot \log(|X(k)|)) \right)^2 \right] \quad (10b)$$

are computed for each FFT block. Based on the above equations, the hardware accumulates for  $N_p$  samples the  $10 \cdot \log(|X(k)|)$  and the  $(10 \cdot \log(|X(k)|))^2$ . At the end of the block processing, the accumulated values are used to generate the  $\mu_{\text{calc}}$  and  $\sigma_{\text{calc}}^2$ . At this point, the standard deviation,  $\sigma_{\text{calc}} = \sqrt{\sigma_{\text{calc}}^2}$ , is computed to generate the final input for the threshold equation (5).

Since, the square root is computationally and area intensive, an approximation is applied. The calculation for the square root of any nonnegative value  $y$  can be simplified by defining  $y = a \cdot 2^b$  for  $1 \leq a < 2$  and  $b$  an integer. The calculation then becomes

$$\sqrt{y} = \sqrt{a} \cdot 2^{b/2} \cdot \sqrt{2}, \quad \text{for } b \text{ odd} \quad (11)$$

$$\sqrt{y} = \sqrt{a} \cdot 2^{b/2}, \quad \text{for } b \text{ even} \quad (12)$$

where  $\sqrt{a}$  can be calculated using a linear approximation of the square root for  $1 \leq a < 2$ . The maximum resultant error for this approximation is 1.3%.

The calculated standard deviation is compared with four internally stored constants to select the scale factor  $N$  from one of five internally stored values. Both the standard deviation and  $N$  values are user programmable. The selected  $N$  is then used to calculate the excision threshold using (5). Control logic in the excisor block synchronizes the calculation and comparison of the threshold to the stored  $X(k)$  values in memory. Instead of storing  $10 \cdot \log(|X(k)|)$  in addition to the  $X(k)$  values for each complex point, the magnitude is recalculated and compared to the threshold. The tradeoff of memory size versus arithmetic processing in this case proved to be a clear area savings. If excision is turned on (note that the excision operation can be enabled/disabled by the user) and the recalculated magnitude does not exceed the threshold, the  $X(k)$  values are passed to the inverse FFT; otherwise, the  $X(k)$  values are set to zero before being passed to the inverse FFT.

## V. RESULTS AND FUTURE DIRECTIONS

The FDIS IC (Fig. 15) was fully tested in June 1997. The IC was fabricated in a 0.5- $\mu$  CMOS triple-metal process. The 1.5-million-transistor design operates at a complex throughput rate of 10 MCSPS, dissipates 1.5 W, and has a latency of 70  $\mu$ s. We have integrated the FDIS IC into a high-performance GPS receiver testbed [8]. Experiments have been performed using the FDIS design in conjunction with an 8.0-MHz C/A code receiver from Novatel. A complete analysis of the measured receiver data has been presented in [9].

The experiments varied from single to multiple CW interferers at different ISR levels. The results demonstrate that FDIS provides substantial narrow-band interference suppression while modestly affecting the navigation accuracy of a GPS receiver.

## ACKNOWLEDGMENT

The authors would like to thank G. M. Butler and R. T. Carroll for their support, guidance, and insight.

## REFERENCES

- [1] R. C. DiPietro, "An FFT-based technique for suppressing narrow-band interference in PN spread-spectrum communications systems," in *ICASSP'89*, Glasgow, Scotland, U.K., May 23–26, 1989, pp. 1360–1363.
- [2] R. Fante and J. Vaccaro, "Enhanced anti-jam capability for GPS receivers," in *ION 98 Conf.*, Nashville, TN, Sept. 1998, pp. 251–254.

- [3] F. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proc. IEEE*, vol. 66, pp. 51–83, Jan. 1978.
- [4] J. Young, "Analysis of DFT-based frequency excision algorithms for direct-sequence spread-spectrum communications," *IEEE Trans. Commun.*, vol. 46, pp. 1076–1087, Aug. 1998.
- [5] B. W. Parkinson and J. J. Spilker, *Global Positioning System: Theory and Applications Volume I*. Washington, DC: American Institute of Aeronautics and Astronautics, 1996.
- [6] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.
- [7] E. Bidet, D. Castelain, C. Joanblanc, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, vol. 30, pp. 300–305, Mar. 1995.
- [8] P. T. Capozza, B. J. Holland, T. M. Hopkinson, D. Moulin, and M. Solomon, "A test facility for evaluating GPS anti-jam techniques," in *Proc. 1999 National Tech. Meeting and 19th Biennial Guidance Test Symp.*, San Diego, CA, Jan. 25–27, 1999, pp. 605–613.
- [9] P. T. Capozza, B. J. Holland, T. M. Hopkinson, C. Li, D. Moulin, P. Pacheco, and R. Rifkin, "Measured effects of a narrowband interference suppressor on GPS receivers," in *55th Annu. Meeting*, Cambridge, MA, June 25–27, 1999, pp. 645–651.



**Paul T. Capozza** (S'87–M'88) received the B.S. degree in computer electronics engineering and the M.S. degree in electrical engineering from the University of Rhode Island, Kingston, in 1986 and 1988, respectively.

In 1989, he joined The MITRE Corp., Bedford, MA, where he currently is a Principal Engineer in the Communications and Networking division. He has been involved in the design of integrated circuits for radar, navigation, and communications systems. His research interests include circuit design and design automation. He currently represents MITRE on the SRC's Integrated Circuit & Systems Sciences technical advisory board. He is also a contributor to the International Technology Roadmap for Semiconductors.



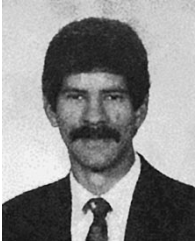
**Brian J. Holland** received the B.S.E.E. degree from Tufts University, Medford, MA, in 1986.

In 1987, he joined The MITRE Corp., Bedford, MA, as a Member of Technical Staff in the VLSI/VHSIC Center. He is currently a Lead Engineer in the Communications and Networking Division. Throughout his career, he has worked on embedded systems design for networking and communication applications. Most recently, he has been a contributor to MITRE's GPS initiative where he has been involved in the system simulation, ASIC development, and digital hardware design of a GPS anti-jam hardware testbed.



**Thomas M. Hopkinson** (S'84–M'87) received the B.S.E.E. and M.S.E.E. degrees with a concentration in communications and signal processing from Northeastern University, Boston, MA, in 1985 and 1988, respectively.

He is with The MITRE Corp., Bedford, MA, where he has contributed to and led the design and development of a variety of communication and navigation systems. He is currently an Associate Section Leader in the Communications and Networking Division, where he manages a group that specializes in architecture development and design of high-performance digital VLSI and hardware. In his current assignment, he is the Program Manager for MITRE's GPS anti-jam user equipment work, concentrating on digital receiver design, antenna research, and advanced signal processing.



**Roberto L. Landrau** (S'94–M'95) received the S.B.E.E. degree from the Massachusetts Institute of Technology, Cambridge, in 1985 and the M.S.E.E. degree from Northeastern University, Boston, MA, in 1995.

In 1985, he joined the Charles Stark Draper Laboratory as a Member of Technical Staff. He was involved in the analysis and design of high-precision control systems, including temperature controllers, low-drift voltage and current sources, and data-acquisition systems. In 1989, he joined the VLSI/VHSIC Technical Center at The MITRE Corp., Bedford, MA, where he is currently a Lead Integrated Circuits Engineer in the Communications and Networking division. He has been involved in the design and implementation of digital and analog integrated circuits in CMOS and GaAs. He is also a key contributor to the electronic design automation capabilities of his division. He currently represents MITRE on the SRC's Computer Aided Design and Test Sciences technical advisory board.