A Novel Iterative Threshold for Anti-jamming Algorithm and Its Implementation on FPGA

Rugui YAO, Geng LI, Ling WANG, Yanbo BI, Yun CHEN

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, Shaan Xi, 710072, China

yaorg@nwpu.edu.cn, hebeiligeng@163.com

Abstract— **Frequency-domain anti-jamming (FDAJ) algorithms achieve prominent performance on interference suppression with simple calculations; and hence they are widely used in many applications. The interference suppression threshold plays a key role in the algorithm and affects system performance greatly. In this paper, the initial interference threshold is derived theoretically and then a novel FDAJ algorithm is proposed based on the iterative calculation of the threshold. With the iterative process, the optimal threshold for the interference suppression can be approached with little loss of valid signals; and the anti-jamming performance can be further improved. The simulation results validate well that our proposed iterative algorithm has superior performance for interference suppression. Finally, two FPGA-based implementations are presented: pipeline scheme and iterative scheme. After analysing the different characteristics of each implementation, we discuss the suitable applications for these schemes.**

Keywords— **Frequency-domain, anti-jamming, iterative threshold, simulation, FPGA-based implementation**

I. INTRODUCTION

Due to the benefits of information security and high spreading gain, spread spectrum system is prevalent in military and civilian wireless systems. Spread spectrum system has the capability of narrow-band interference suppression. However, if the interference-to-signal ratio (ISR) is close to or higher than the spreading gains, the system will lose its immunity to interference and the receiver cannot work normally. Therefore, it is necessary to introduce interference suppression technologies to improve the anti-jamming capability.

In general, the interference suppression technologies can be classified into three schemes: time domain scheme, frequency domain scheme [1] and code-aided interference suppression scheme [2]. Wherein, the FDAJ algorithms with good application prospects, transform the complex filtering process in time domain into simple threshold comparison in frequency domain. With the large amount of applications on FPGA chips, FPGA-based implementation for FDAJ algorithms becomes another research hotspot. Currently, abundant research achievements have been presented on FDAJ algorithms. An adaptive threshold FDAJ algorithm and its FPGA implementation were proposed in [3]. With the statistical analysis of interfered signal via MATLAB, the judgment threshold can be determined off line. The simulation results shows this algorithm has better anti-jamming performance in the case of fixed interference signal. However, considering real-time dynamic interference, the selected threshold cannot be adjusted to make it matched to the interference, which will limit the application scenarios. In [4], the distributions of signal envelope and squared envelope in spread spectrum system were analyzed in detail. Based on this distribution, an interference suppression algorithm with segmented thresholds was proposed. This algorithm requires segmented interference suppression according to the amplitude of the frequency-domain sample, which would result in multiple complex comparisons to decide interference. In [5], Zou proposed an overlapped windowing FDAJ algorithm for narrowband interference, which iteratively computed the adaptive threshold. However, this algorithm simply reset the amplitude of interfered sample to zero when this sample is determined to be interfered. This operation would be expected to cause losses to the valid signal in the case of high ISR. In this paper, the proposed FDAJ algorithm utilizes an adaptive iterative threshold which considers the above defects and can effectively suppress both single frequency and narrowband interference.

The remainder of this paper is organized as follows: Section II generally introduces the FDAJ algorithm, and intensively proposes the iterative calculation of threshold and theoretically derives the initial threshold. Some MATLAB simulation results and the analysis on interference suppression performance are given in Section III to validate our algorithm. In Section IV, two different FPGA-based implementation schemes and the corresponding applications are presented. Finally, Section V concludes both the algorithm and the implementation schemes.

II. ITERATIVE THRESHOLD FDAJ ALGORITHM

In this section, after briefly introducing the FDAJ algorithm, we derive the initial threshold and then propose the iterative calculation of the threshold. Finally, a simplified calculation of iterative threshold is also presented regarding implementation.

A. FDAJ Algorithm

The flow chart of FDAJ algorithm is shown in Figure 1.

In this paper, we assume an overlapped window FDAJ algorithm is used to reduce the spectrum leakage caused by DFT. The interfered samples in time domain are windowed by

generalized hamming window and then fed into FFT module to obtain frequency domain samples. The interference judgment and suppression (IJAS) module will deal with these frequency domain samples to filter out the interference. Then the anti-jamming frequency domain samples are recovered to time domain samples by IFFT. At the end of the process, an overlap-add synthesis output is done to reduce the loss of signal-to-noise ratio (SNR) [6].

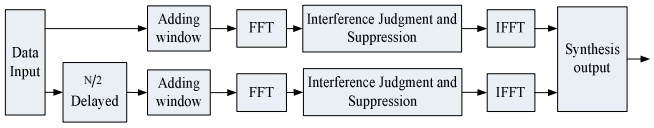


Figure 1. FDAJ algorithm flowchart

From Figure 1, the IJAS module is critical to the FDAJ algorithm, which consists two steps: interference judgment and interference suppression. Interference judgment is essentially to compare the frequency domain sample with a given threshold. If the sample is larger than the threshold, then it is determined to be interfered and required interference suppression; otherwise, the sample is considered valid signal and just kept intact. Interference suppression is mainly classified into two methods: zero clamping and threshold clamping. Because zero clamping will lose some amplitude information of valid signal and make iterative calculation difficult, we tend to threshold clamping in our algorithm. After the operation of interference judgment and suppression, the interference among output samples is reduced to an acceptable level.

B. Derivation of Initial Threshold for Interference Judgment

As mentioned above, all the frequency domain samples should be compared with the threshold in the IJAS process; therefore, how to set up the threshold plays a decisive role in anti-jamming performance. Here, due to the simplicity, the first-order statistic is applied for interference threshold calculation and the corresponding formula can be formatted as below:

$$TH = K \times u \tag{1}$$

where K is the threshold optimization coefficient; u is the mean of the amplitude of frequency-domain samples. What we will discuss below is how to optimize the coefficient K theoretically with the help of Narrow-band Gaussian (NBG) model.

The spread spectrum signal can be expressed as $S(k)+N(k)$ without interference, where $S(k)$ is the valid signal and $N(k)$ is the additive noise. After N-point DFT, the signal is transformed into $S(n)+N(n)$, which can be approximated to be NBG distributed [4]. According to the characteristics of the NBG model, the envelope $|S(n)+N(n)|$ is Rayleigh distributed while the squared envelope $|S(n)+N(n)|^2$ is exponentially distributed. The interference threshold can be derived with the probability density function (pdf) of exponential distribution in [4][5][7]. However, it is found that the value of $|S(n)+N(n)|^2$ tends to be very large which will consume a lot of storage resources when hardware implementation is considered. For this purpose, we will deduce a reasonable value of K according to the Rayleigh distributed envelope.

The mean and pdf of signal $|S(n)+N(n)|$ which is Rayleigh distributed can be formulated as follows:

$$E = \sigma\sqrt{\frac{\pi}{2}} \tag{2}$$

$$f(x) = \frac{x}{\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3}$$

When first-order statistic, that is, mean is used, the threshold can be computed as

$$TH = K \times E = K \times \sigma\sqrt{\frac{\pi}{2}} \tag{4}$$

Here we just regard $K = 1,2,3,4,5$. We should minimize the loss of valid signal and the formula in probability form will be

$$\begin{aligned} P\left(|S(n)+N(n)| < TH\right) &= \int_0^{TH} \frac{x}{\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right)dx \\ &= 1-\exp\left(-\frac{x^2}{2\sigma^2}\right)\Big|_{x=TH} \end{aligned} \tag{5}$$

By (5), the probability corresponding to different K can be computed, as is shown in Table 1:

TABLE 1. PROBABILITIES FOR DIFFERENT VALUES OF K

$K=1$	$K=2$	$K=3$	$K=4$	$K=5$
0.5441	0.9568	0.9991	1.0000	1.0000

Obviously, when $K=4$, all valid signals would not be lost. What's more, the optimized K is easy for hardware implementation by simple bit-shift operation.

For the static statistical characteristics of NGB model, the expectation, $E(\cdot)$ can be approximated by the statistical average $\frac{1}{N}\sum_{n=1}^{N}|S(n)+N(n)|$, when N is large enough (i.e. $N > 256$). Till now, we have deduced the threshold optimization coefficient and also the initial threshold for interference suppression.

C. Iterative Calculation of Interference Threshold

Because the input signal contains strong interference which will result in a higher interference threshold according to the initial threshold, interference often cannot be suppressed completely and remain a lot after only one suppression. With this consideration, we present an iterative interference suppression algorithm. By several judgments and suppressions, the calculated threshold can gradually approximate the optimal one for interference suppression, which can keep the

valid signal and suppress the interference as much as possible. The detail iterative algorithm is designed as follows.

1) For the first interference suppression, the mean of frequency-domain samples and the initial threshold could be calculated as

$$M_1 = \frac{1}{N}\sum_{n=0}^{N-1}\left|R(n)\right| \tag{6}$$

$$TH_1 = K \times M_1 = 4M_1 \tag{7}$$

where $\left|R(n)\right|$ is the amplitude of frequency domain samples.

2) Compare $\left|R(n)\right|$ with TH_1 and do the interference judgment and suppression as

$$R_1(n) = \begin{cases} TH_1, & \left|R(n)\right| > TH_1 \\ R(n), & else \end{cases} \quad (n = 0,1,2\cdots N-1) \tag{8}$$

Here, $R_1(n)$ is the input of the second interference suppression process.

3) The mean and interference threshold for the second suppression can be recomputed as (6) and (7), that is,

$$M_2 = \frac{1}{N}\sum_{n=0}^{N-1}\left|R_1(n)\right| \tag{9}$$

$$TH_2 = K \times M_2 = 4M_2 . \tag{10}$$

4) Use TH_2 to do the second interference judgment and suppression and obtain the new output $R_2(n)$.

$$R_2(n) = \begin{cases} TH_2, & \left|R_1(n)\right| > TH_2 \\ R_1(n), & else \end{cases} \quad (n = 0,1,2\cdots N-1) \tag{11}$$

5) The subsequent threshold calculation and interference judgment and suppression are similar to step 3) and 4), which always use the last output as the input for the new iteration. Here we do not repeat it for simplicity.

D. Simplified Calculation of Iterative Threshold

The re-computation of the mean is complicated when the number of accumulated samples is large in IJAS process. To simplify the calculation, we propose another algorithm for the mean as

$$SUM_i = SUM_{i-1} - \sum_{j=0}^{L}\left(\left|R_i(j)\right| - TH_{i-1}\right) \tag{12}$$

Where $R_i(j)$ is the amplitude of spectrum sample which is bigger than TH_{i-1} and L is the total number of interfered samples in the *i*-th IJAS process. The simplified calculation is favorable in engineering applications when the number of iterations and the points of FFT are large.

This iterative anti-jamming algorithm can track well with the interference by real-time updating the threshold, and be effect even in the case of no interference. When there is no interference, based on the threshold coefficient optimization strategy, all the valid signals are retained for all the frequency domain samples will be less than the interference threshold; when interference exists or is stronger, we can get the optimal threshold for interference suppression by the iterative means. In practical applications, the number of iterations can be determined adaptively according to the number of samples with clamping operation for an iteration. We will study on the adaptive number of iterations or early-stopping strategies in the future work. In this paper, three iterations are enough for effective interference suppression in Matlab simulation and FPGA implementation.

III. Simulation Results and Analysis

In this section, all the simulations are based on these parameters: COMPASS C/A code with bandwidth 20.46MHz, sampling frequency 62MHz and digital IF 15.48MHz; colored Gaussian noise with bandwidth 20.46MHz and SNR -20dB; and narrow-band interference with bandwidth 2MHz, center frequency 15.48MHz and SIR -30dB.

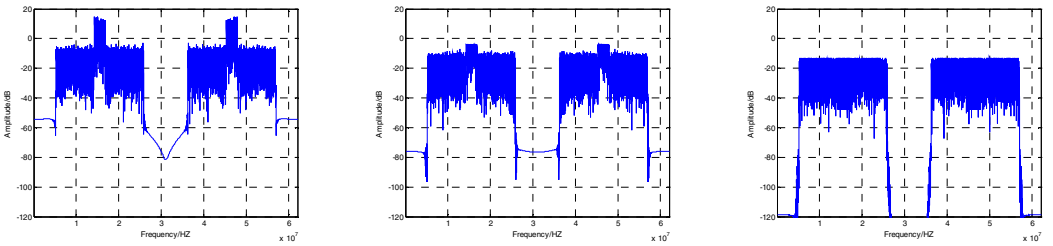
According to the characteristics of spread spectrum signal, the correlation between the received signal and local C/A code can be used to measure the interference suppressing performance.

The spectrograms and corresponding correlation for the original input, after one iterative suppression and after three iterative suppression are illustrated in Figure 2 and Figure 3.

From the above comparison, we can easily conclude that, 1) there is no distinct correlation peak without interference suppression and the receiver cannot surely work properly; 2) after one iterative anti-jamming, the interference is suppressed to some extent, however still left over and therefore the correlation peak is not obvious, which results in poor performance of position; 3) after three iterative anti-jamming, the interference is almost completely removed and the correlation peak is highlighted, which brings out superior performance.

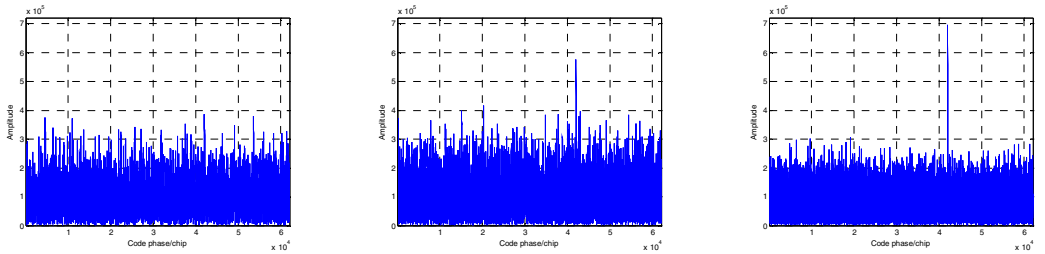
IV. FPGA-based Implementation of the Proposed Iterative Threshold Anti-jamming Algorithm

As is shown in Figure 1, the structure of the FPGA implementation for the iterative threshold anti-jamming algorithm we propose is divided into five major parts: add window module, FFT module, IJAS module, IFFT module and the synthesis output module.



(a) original input (b) after one iterative suppression (c) after three iterative suppression

Figure 2. The comparison of spectrogram for original input and with one or three iterative suppression



(a) original input (b) after one iterative suppression (c) after three iterative suppression

Figure 3. The comparison of correlations for original input and with one or three iterative suppression

The development platform for FPGA is Xilinx ISE10.1 and the selected FPGA chip is XC5VLX155 which has large logical resources in Vertex-5 series.

As for parameter settings in implementation, the window length is set to be 512-point as well as the depth of all used RAM. In addition, the FFT and IFFT operation are implemented with IP core in scaled mode to reduce the subsequent storage utilizations.

With the idea of exchanging storage resources and processing rate in FPGA implementation, this paper proposes two implementations: pipeline scheme and iterative scheme, to realize the function of IJAS module. Because all the modules except IJAS module are identical for the two different schemes, we just focus on the implementations of IJAS module in the following part.

A. FPGA-based Implementation in Pipeline Scheme

The flow chart of IJAS module in pipeline scheme is shown in Figure 4, where ICU, AU, TCU and PU are the abbreviations of **Interference Clamping Unit**, **Accumulation Unit**, **Threshold Calculation Unit** and **Processing Unit** respectively. Another notation is we do not consider the simplicity of threshold calculation as (12) because of little consumption to implement AU.

In Figure 4, we only show the iterative process for one windowed signal and the process is designed as follows:

1) Store the 512 samples (include real part $R_r(n)$, imaginary part $R_i(n)$ and their amplitude values $|R(n)|(n=0,1,2\cdots 511)$) into RAM1 after FFT.

2) While RAM1 is storing, AU1 accumulates the amplitude $|R(n)|$ and send the accumulation results *SUM* to TCU1 when RAM1 is overflow.

3) TCU1 calculates the initial threshold TH_1 as (4) and sets it to ICU1. Because K is $4=2^2$ and the number of accumulation is $512=2^9$, TH_1 can be obtained by a direct 7-bit shift to *SUM*.

4) ICU1 compares TH_1 with the amplitude $|R(n)|$ read from RAM1. If $|R(n)|\geq TH_1$, the sample is decided to be interfered, and set $R_r(n)=TH_1$ and $R_i(n)=0$. Otherwise, keep the original stored sample for the sake of no interference.

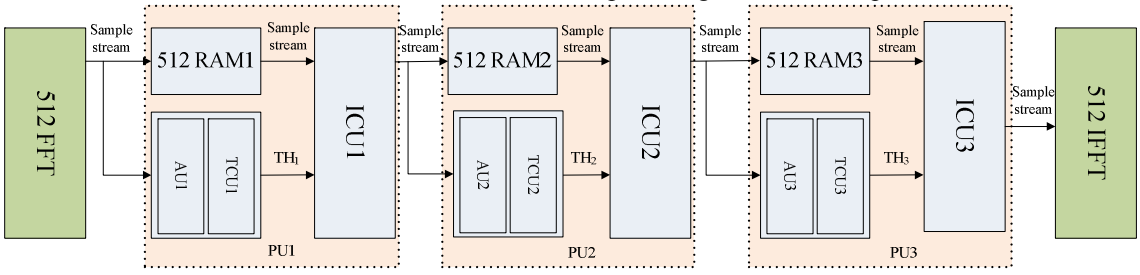


Figure 4. Flow chart of IJAS module in pipeline scheme

5) RAM2 and RAM3 receive the outputs of ICU1 and ICU2 respectively. The operations of sample storage, threshold calculation and interference judgment and suppression in the 2nd and 3rd is similar to 1), 2), 3) and 4). When PU3 completes all the operations, output the anti-jamming samples to the IFFT module.

B. FPGA-based Implementation in Iterative Scheme

In order to save up hardware resources, we propose another FPGA-based implementation in iterative scheme. The flow chart of IJAS module in iterative scheme is shown in Fig. 5, where DISU, AU, DPSU, CU, SMU and FR are the abbreviations of **Data Input Selection Unit**, **Accumulation Unit**, **Data Processing Selection Unit**, **Comparison Unit**, **Sum Modification unit** and **Flag Register** respectively.

In this scheme，the techniques of Ping-Pong operation, high-speed clock RAM reading and the simplicity of threshold calculation are mainly involved to realize three iterative process. Assume that the working frequency of 512-point FFT is f_{in} and that of the IJAS module is f_s which is determined by the number of iterations. If we set up 3 iterative process, then $f_s = 3 \times f_{in}$ at least without conflict for RAM1 and RAM2. The process shown in Fig.5 is designed as follows:

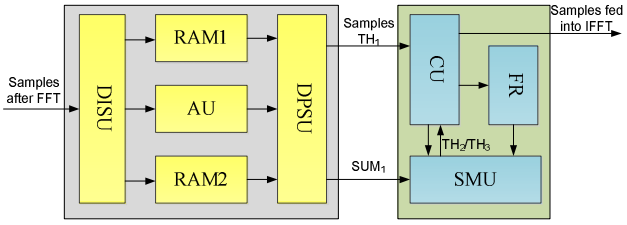


Figure 5. Flow chart of IJAS module in iterative scheme

1) DISU activates RAM1 first to receive and store the 512-point samples (with the same storage operation in pipeline scheme) with frequency f_{in}. In the meanwhile, AU calculates the sum of the amplitude values $|R(n)|$ and then finally obtains the accumulated value SUM_1 as well as initial threshold TH_1 according to (7) at the moment that RAM1 is overflow.

2) When RAM1 is overflow, DISU alternates RAM2 to receive and store the samples from FFT module. Do the same operations of computing SUM_1 and TH_1 as 1) for the next group of samples.

3) While RAM2 is storing, read samples from RAM1 and make the process of them for 3 iterations with the frequency f_s. Here we introduce a 512 bit-width Flag Register (FR) with the aim to simplified computation as (12) and avoidance of write-back of RAM1 and RAM2. The 3 iterative process can be further divided into 3 steps in detail.

i. **1st iteration**. Set two registers *RSUM* and *FR* and reset them. Compare the amplitude values $|R(n)|$ read from RAM1 with TH_1. If $|R(n)| \geq TH_1$, then set $FR[n] = 1$ and make the modification of $RSUM = RSUM + (|R(n)| - TH_1)$ in SMU; otherwise, set $FR[n] = 0$ and keep *RSUM* unchanged. At the end of this iteration, calculate SUM_2 as $SUM_2 = SUM_1 - RSUM$ and $TH_2 = K \times \frac{SUM_2}{N}$.

ii. **2nd iteration**. Reset *RSUM* in SMU. Successively compare $|R(n)|$ with TH_2. In case of $|R(n)| \geq TH_2$, then check whether the corresponding $FR[n]$ is 1 or not. If $FR[n] = 1$, which means $|R(n)| \geq TH_1$, then keep $FR[n] = 1$ and modify *RSUM* as $RSUM = RSUM + (TH_1 - TH_2)$; otherwise, set $FR[n] = 1$ and modify *SUM* as $RSUM = RSUM + (|R(n)| - TH_2)$. In case of $|R(n)| < TH_2$, reset $FR[n] = 0$ and keep *RSUM* unchanged. Similarly, at the end of this iteration, we should compute $SUM_3 = SUM_2 - RSUM$ and $TH_3 = K \times \frac{SUM_3}{N}$.

iii. **3rd iteration**. Start the third reading and comparing process. If $|R(n)| \geq TH_3$, then modify the real and imaginary parts of the sample with $R_r(n) = TH_3$ and $R_i(n) = 0$, and feed this modified version of sample into IFFT module; otherwise, directly output the original sample into the IFFT module.

4) When the process for the samples in RAM1 is finished, the storage for RAM2 is also completed. Repeat the step 3) for the samples stored in RAM2 and start to store new data in RAM1.

C. Comparison and Summary of the Two Schemes

The performance of these two schemes for interference suppression is the same due to completely identical substantial. The pipeline implementation is relatively simpler; however it needs to store the data after each judgment and requires greater processing delay. By contrast, the iterative scheme can avoid the storage of intermediate data in iterative process by using high-speed clock to read RAM and the processing delay is rather smaller. In this paper, from the time when the first valid sample is fed into the IJAS module to the time when the first anti-jamming sample outputs, the pipeline scheme consumes a total of clocks, while the iterative scheme only does 512 clocks. Due to the limitation of clock-rate for hardware, the number of iterations in iterative scheme cannot be very large.

The resource utilizations for the two schemes are shown in Table 2.

TABLE 2. THE RESOURCE UTILIZATIONS FOR THE TWO SCHEMES

Resource Type	Pipeline Scheme	Iterative Scheme
Logic Utilization	14%	12%
Register	13%	15%
LUTs	54%	44%
LUT-FF	6%	6%
RAM/FIFO	37%	37%
DSP48s	14%	12%

From Table 2, we can discover that the resource utilizations for pipeline scheme are slightly more than those of iterative scheme resources except the registers for temporary storage. And this resource gap will be enlarged with the increase of the number of iterations. However the iterative process would require a very high clock, which is difficult or even impossible to achieve for hardware when the number of iterations is larger. From the above comparison, the pipeline scheme can be applied to the scenario requiring multiple iterations and should be provided with sufficient storage resource; on the contrary, the iterative scheme is suitable to the applications with a smaller number of iterations and resource constraint because this scheme requires less resources but needs higher clock rate.

V. CONCLUSIONS

Due to the importance of the threshold in frequency domain interference suppression, we proposed a novel iterative algorithm of threshold calculations and then the interference suppression algorithm based on the computed threshold. In order to accelerate the convergence of the threshold and shrink the loss of valid signals when interference is not strong or does not exist, we also derive the optimal initial threshold based on the Rayleigh-distributed characteristics of the received signal amplitude. For COMPASS C/A code, the MATLAB simulation results validate the correctness and effectiveness of our proposed algorithm. Considering the applications, two FPGA-based implementations are proposed: pipeline scheme and iterative scheme. With comparison of these two schemes, their respective suitable applications are put forward. Note that the idea of iterative threshold can be used easily in some other occasions where the thresholds are difficult to set.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (No. 61271416), the NPU Foundation for Fundamental Research (No. JCY20130132), and the Aerospace Support Funds (No. 2013-HT-XGD).

REFERENCES

[1] Huang K W, Tao R, Wu K, et al. "Study on interference suppression based on joint fractional Fourier domain and time domain," *Science China Technological Sciences*, vol. 54, pp. 2674-2686, 2011.

[2] Buzzi S, Lops M, Poor H V. "Code-aided interference suppression for DS/CDMA overlay systems," *Proceedings of the IEEE*, vol. 90, pp. 394-435, 2002.

[3] ZHU L, WANG Y Q, DU Y, HAN F J. "Transform Domain Method for Suppressing Interference Based on Adaptive Threshold and Its FPGA Implementation," *Electronic Engineer*, vol. 32, pp. 37-39, 2006.

[4] ZHANG A., HU Y., HAN F. "Research on Narrowband Interference Suppression Based on Adaptive Threshold Under Low SNR," *Electronic Information Warfare Technology*, vol. 24, pp. 51-54, 2009.

[5] ZOU NING, XU S., LIU M., LIU K. "An Overlap Window-Based Frequency Domain Narrowband Interference Suppression Algorithm and Its Analysis," *Modern Defence Technology*, vol.38, pp. 121-125, 2010.

[6] YAO J., ZHENG L. "Analysis of Filtering Threshold for Frequency Domain Suppressor," *Modern Electronics Technique*, pp. 1-2, 2007.

[7] MU L., HUO Z. "Research on Threshold Detection and Interference Processing in Frequency Domain Anti-jamming," *China Computer \& Network*, vol.36, pp. 46-48, 2010.

[8] ZHU Q., WANG L., YAO R. "Application of linear prediction filter in multi-NBI suppression," Modern Electronics Technique, vol.35, pp. 1-3, 2012.

[9] SHYNK J. J. "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Magazine*, vol.9, pp. 14-37, 1992.

Rugui YAO (M'12) was born in 1980. He received his B.Sc., M.Sc. and Ph.D. degrees in telecommunications and information system from Northwestern Polytechnical University, Shaan Xi, China, in 2002, 2005 and 2006 respectively.

From 2007, he joined Northwestern Polytechnical University and currently he is an associated professor.

Prof. Yao's research interests include wireless communications, receiver design of navigation system and anti-jamming techniques.