IMPROVED FAST MODIFIED DOUBLE-BLOCK ZERO-PADDING (FMDBZP) ALGORITHM FOR WEAK GPS SIGNAL ACQUISITION

Wen Zhang[1,2] *and Mounir Ghogho*[2,3]

[1]College of Opto-Electronic Science and Engineering,National University of Defense Technology,China
[2]School of Electronic and Electrical Engineering, The University of Leeds, UK
[3]International University of Rabat, Morocco
Email: m.ghogho@ieee.org

ABSTRACT

Acquisition of weak GPS signals requires long Predetection, Integration Time (PIT). The double-block zero-padding (DBZP) algorithm can use long PIT without any assisting information to acquire weak GPS signals. Modified double-block zero-padding (MDBZP) algorithm accounts for the Doppler effect on the code duration. It also develops the Doppler frequency bin elimination method during the noncoherent integration steps. The fast modified double-block zero-padding (FMDBZP) algorithm based on MDBZP algorithm eliminates the redundant FFT computations by considering possible data bit combinations and bit edge positions. Although the computation burden of FMDBZP is lower than that of MDBZP, it may still be very heavy. This paper aims to reduce the computation complexity and save memory space further. Three methods are introduced to improve the FMDBZP algorithm. One consists of unlikely data bit combination path elimination during coherent integration steps. The second one perform unlikely code phase elimination during noncoherent integration steps. The third one perform unlikely bit edge elimination also during noncoherent integration steps. The good performance of the proposed improved FMDBZP algorithm is illustrated using both simulation and real data.

1. INTRODUCTION

GPS is a Code Division Multiple Access (CDMA) system in which each satellite transmits a direct sequence spread spectrum signal with a particular pseudorandom code. It is widely used nowadays but still faces limitations in indoor and dense-urban environments because of the received weak signals. Acquisition is the first step of GPS signal processing. The goal of the acquisition is to find the visible satellites, the start of each C/A code and the Doppler shift. Acquisition of weak GPS signals requires long integration time to boost the post-correlation signal-to-noise ratio (SNR). When the coherent integration time is a multiple of one data bit duration, a search over all data bit combinations is required. Since the data bit duration is 20ms and the data bit edge is synchronized with the 1ms PRN code, there are twenty possible bit edge positions. Thus, for weak signals, we have to search over both possible data bit combinations and possible bit edge positions, besides the search of visible satellites, Doppler shifts and code phases.

There have been many acquisition approaches that aim to reduce the processing time and increase the coherent integration. We can sort all these methods into 4 categories. The first one is the conventional hardware approach which sequentially searches for visible satellites at each possible code delay and Doppler frequency shift, [1] and [2]. The second one aims to reduce processing complexity by performing circular correlation using Fast Fourier Transform (FFT) methods, [3] and [4]. Complexity is reduced because the correlation at all code delays is computed at once, for each Doppler bin. The third one consists of using the delay and multiply method to eliminate all Doppler bins, [3] and [5], but it has low sensitivity. The fourth one is using an approach called Double-block zero padding (DBZP), which uses a frequency domain method to search over a delay-Doppler space and acquire the GPS signal, using coherent integration time that is a multiple of the 20ms data bit period, [3], [6], and [7]. DBZP is one of the most advanced algorithms that is especially suitable for software-defined GPS receivers. Different improvements have been proposed to extend the original DBZP algorithm for long integration periods and reduce the computational complexity. [8] introduces the MDBZP algorithm for weak signal acquisition. This algorithm uses a coherent integration time that is a multiple of one data bit, without assuming the availability of any bit edge assisting information. Furthermore, this method circumvents DBZP's limitation resulting from the unaccounted effect of Doppler shift on the code duration. It also introduces a frequency bin elimination method after one or several coherent integration steps. [9] introduces the FMDBZP algorithm based on MDBZP algorithm by eliminating the redundant FFT computations. The number of FFT operations in FMDBZP was found to scale linearly with the coherent integration time, in contrast to the geometric growth experienced in MDBZP. Although the computation complexity of FMDBZP is less than that of MDBZP, it is still very heavy. As numerical examples, given in [9], the computation costs under minimal data bit length for target C/N_0 25dB-Hz, 20dB-Hz, 15dB-Hz and 10dB-Hz need 8×10^8, 3.6×10^9, 2.02×10^{10}, and 9.505×10^{11} total operations (additions and multiplications), respectively.

In this paper, we propose three methods based on FMDBZP, aiming to reduce the computational complexity and memory space requirements. These are i) unlikely data bit combination path elimination during coherent integration steps, ii) unlikely code phase elimination during the noncoherent integration steps. iii) unlikely bit edge elimination dur-

ISSN 2076-1465

ing the noncoherent integration steps. Examples using simulation and real data are given.

The rest of the paper is organized as follows. Firstly, the implementation of FMDBZP is illustrated using matrices and figures. Secondly, the unlikely data bit combination path elimination inside coherent integrations, as well as unlikely code phase and bit edge elimination are introduced. Thirdly, examples using simulation and real data will be presented. Last, conclusions will be drawn.

2. IMPLEMENTATION OF FMDBZP

The MDBZP [10] algorithm aims to make the DBZP algorithm suitable for large PIT T_I. Let the Doppler frequency coverage be from $-f_{d_{max}}$ to $f_{d_{max}}$, the sampling frequency be f_s. For PIT T_I, the number of Doppler frequency bins N_{fd} is:

$$N_{fd} = 2 f_{d_{max}} T_I \quad (1)$$

The number of blocks N_b should be chosen to be equal to the number of Doppler frequency bins N_{fd} in MDBZP algorithm [3]:

$$N_b = N_{fd} \quad (2)$$

The size of each block S_b is:

$$S_b = \frac{f_s T_I}{N_b} \quad (3)$$

The number of steps N_{step} needed to calculate the correlation for all delays of the 1ms PRN code is equal to the number of blocks in 1ms:

$$N_{step} = \frac{N_b}{10^3 T_I} = \frac{10^{-3} f_s}{S_b} \quad (4)$$

The duration of each navigation data bit is 20ms, and the C/A code has a code period of 1ms. So there are 20 C/A code periods for each data bit. Letting the number of navigation data bits in T_I be N_{db}, the number of blocks N_b can also be expressed as:

$$N_b = 20 N_{db} N_{step} \quad (5)$$

The total number of samples over T_I is thus:

$$N_T = T_I f_s = S_b (20 N_{db} N_{step}) \quad (6)$$

The number of samples N_{ms} over 1ms is:

$$N_{ms} = N_{step} S_b \quad (7)$$

The separation between the Doppler bins is $f_{res} = 1/T_I$, also known as frequency resolution. So, the Doppler bin values f_{d_m}, can be calculated as:

$$f_{d_m} = f_{res}(m - N_b/2 - 1), \text{where } m = 1, 2, \ldots, N_{fd} \quad (8)$$

To enlarge the PIT T_I, the replica C/A code has to account for the effect of the Doppler shift on the code duration. This is handled by dividing the frequency range into a small number of ranges N_r with f_r being the frequencies that lie in the middle of each range, where $r = 1, 2, \ldots, N_r$. Then each of the N_r replica codes is generated by compensating for the effect of one of the frequencies f_r. The N_r Doppler compensated replica codes are:

$$C_{L_{kr}} = C_L\left(t_k\left(1 + \frac{f_r}{f_{L1}}\right)\right), \text{where } r = 1, 2, \ldots, N_r \quad (9)$$

where t_k is the sample time with index k, f_{L1} is the L1 carrier frequency which is equal to 1575.42MHz, C_L is the locally generated code. The DBZP is calculated once for each of the replica codes. Then in forming the result, the cells corresponding to the correct range from each calculation are preserved, and the other cells are discarded. This is coherent integration.

After T_I seconds, the C/A code will be shifted by a number of samples N_s equal to:

$$N_s = T_I f_s \frac{f_{d_{L1}}}{f_{L1}} = T_I f_s \frac{f_{d_{ca}}}{f_{ca}} \quad (10)$$

where $f_{d_{L1}}$ is the Doppler shift on L1 frequency and $f_{d_{ca}}$ is the Doppler shift on the C/A code, and f_{ca} is the C/A code frequency. To add the coherent integration result incoherently to the previous one, first it has to be circularly rotated, at each Doppler frequency bin, by N_{s_m} samples [10]:

$$N_{s_m} = \text{sign}(f_{d_m} - f_r)\text{round}\left((n-1) T_I f_s \frac{|f_{d_m} - f_r|}{f_{L1}}\right) \quad (11)$$

Where n is the noncoherent integration step.

Let the number of possible data bit edge over T_I be N_{be}. As a consequence of the 50Hz data message, there are 20 possible bit edge positions in 20ms. Each one is aligned with the start of a 1ms PRN code period. If the start of the coherent integration is misaligned with respect to the start of the data bits, then there will be a loss in the integration due to the bit edge transitions. So the 20 possible bit edge positions have to be searched.

The received GPS signal first has to be down-converted to baseband and then divided into N_b blocks. 1ms of C/A code consists of N_{step} blocks, so for the computation of the correlation when performed over T_I, this sequence of N_{step} blocks has to be repeated $(20N_{db} - 1)$ times to match the length of the signal as shown in Fig.1. Then, double-blocking the baseband signal and zero-padding the local C/A code are performed. Each two blocks i and $(i+1)$ are combined into one section to produce N_b sections of size $2S_b$; the locally generated 1ms C/A code is also divided into N_{step} blocks, each block is padded at end with S_b zeros to produce sections of size $2S_b$ each. We will use *section* instead of *block* after DBZP operation. Double-blocking and zero-padding operation is adopted in order to use the circular correlation operation. Circular correlation operation is then performed between each corresponding sections of signal and code; we only save the first

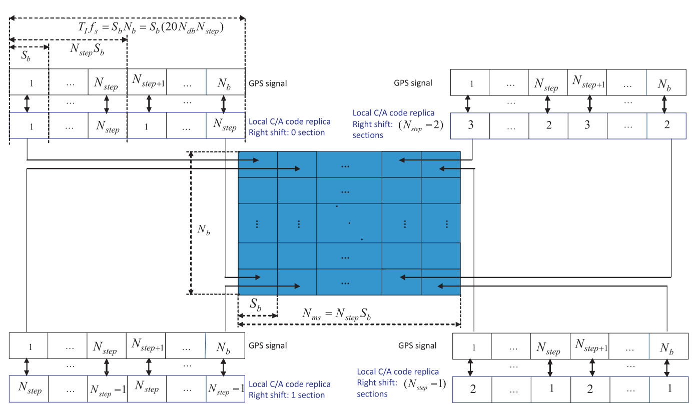

Fig. 1. The DBZP matrix M_{cc}

S_b samples of each circular correlation. We form a matrix called M_{cc} of size $N_b \times N_{ms}$, which is illustrated in Fig.1. The matrix M_{cc} is formed in the following way. When the local code is not shifted, save the first S_b samples of the circular correlation of section 1 of GPS signal and section 1 of local code into $1 \times S_b$ (row×column) of M_{cc}, section 2 of GPS signal and section 2 of local code into $2 \times S_b$ of M_{cc},$\cdots$, section $(N_b - 1)$ of GPS signal and section $(N_b - 1)$ of local code into $(N_b - 1) \times S_b$ of M_{cc}, section N_b of GPS signal and section N_b of local code into $N_b \times S_b$ of M_{cc}. Then right shift the code by one section, save the first S_b samples of the circular correlation of section 1 of GPS signal and section N_{step} of local code into $1 \times (2S_b)$ (row×column) of M_{cc}, section 2 of GPS signal and section 1 of local code into $2 \times (2S_b)$ of M_{cc},$\cdots$, section $(N_b - 1)$ of GPS signal and section $(N_{step} - 2)$ of local code into $(N_b - 1) \times (2S_b)$ of M_{cc}, section N_b of GPS signal and section $(N_{step} - 1)$ of local code into $N_b \times (2S_b)$ of M_{cc}. Then going on right shifting the code section by section and save the results in the corresponding position of M_{cc}. The last two steps of code shift are also shown in Fig.1.

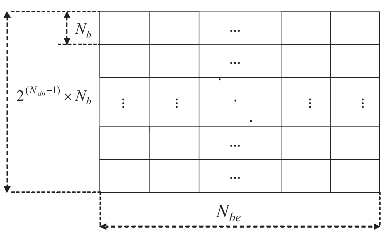

Fig. 2. One of N_{ms} matrix, M_{ms}, formed by both considering $2^{N_{db}-1}$ possible data bit combination and N_{be} possible data bit edge positions

The FMDBZP method [9] breaks each column of the matrix M_{cc} into $N_{db}N_{be}$ parts and then zero-pads the corresponding places to form columns of the same length $20N_{db}N_{step}$ before dividing. Considering the N_{be} possible bit edges, extra $(N_{be} - 1)$ parts are also calculated in the above DBZP and then are divided and zero-padded to form columns of length $20N_{db}N_{step}$, too. The $20N_{db}N_{step}$-point FFT is then performed on the $((N_{db} + 1)N_{be} - 1)$ columns, and the results in frequency domain are expressed as X_1, X_2, $\cdots$, $X_{(N_{db}N_{be}+1)}$, $X_{(N_{db}N_{be}+2)}$, $\cdots$, $X_{(N_{db}N_{be}+1)}$, $X_{((N_{db}+1)N_{be}-1)}$. In order to construct the correlation result by considering N_{be} possible data bit edge positions and $2^{(N_{db}-1)}$ possible data bit combinations, one of the N_{ms} matrices, called M_{ms}, is formed as illustrated in Fig.2. In matrix M_{ms}, the N_{be} columns are formed by taking the data from X_1 to $X_{N_{db}N_{be}}$, X_2 to $X_{(N_{db}N_{be}+1)}$, $\ldots$, $X_{N_{be}}$ to $X_{((N_{db}+1)N_{be}-1)}$. The multiple $2^{(N_{db}-1)}$ in row is formed by adding the $N_{db}N_{be}$ $20N_{db}N_{step}$-point FFT results by taking into account the $2^{(N_{db}-1)}$ possible data bit combinations at each of the N_{be} possible bit edge positions.

After processing all the columns in Fig.1, we get the full matrix, called M_{full}, of size $(2^{(N_{db}-1)} \times 20N_{db}N_{step}) \times N_{be}N_{ms}$. Matrix M_{full} is the result of the coherent integration.

To add this coherent integration matrix M_{full} incoherently to the noncoherent integration results matrix, first it has to be circularly rotated at each Doppler frequency bins using Equation (11). Then among all the $2^{(N_{db}-1)}$ possible data bit sign combinations of any tentative Doppler frequency bin, any tentative bit edge position and any tentative code phase, [10] preserve the most likely data bit sign combination and discard the others during the noncoherent integration steps; the resulted new noncoherent integration matrix is of size $(20N_{db}N_{step}) \times N_{be}N_{ms}$.

3. IMPROVED FMDBZP

Although the computation complexity of FMDBZP is less than MDBZP , it's still very heavy and needs large memory size. So it is needed to research on further computation reduction and less memory occupation methods. Here we propose three methods to improve FMDBZP to achieve the goal. The Improved FMDBZP has three main considerations. One is data bit combination path elimination during coherent Integration steps. The second one is code phase elimination during nonoherent integration steps. The third one is bit edge elimination during nonoherent integration steps. Because the last two methods are both processed on noncoherent integration steps, we describe these two together.

3.1. Data Bit Combination Path Elimination during Coherent Integration Steps

The coherent integration matrix size will be $(2^{(N_{db}-1)}20N_{db}$ $N_{step})(N_{be}N_{ms})$ without considering data bit combination path elimination and other processing. We aim at reducing 75%

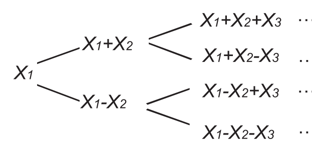

Fig. 3. Data bit combination path inside of coherent integration

of the possible data bit path. The data bit combination path elimination during coherent integration steps is implemented as follows. This method can be justified based on the fact that the correct coherent integration cell will always contain information due to the correlation between the received signal and the local code replica. If the correct cell does not produce the maximum power value, its value will be within a certain range of the high values obtained from all the cells. Taking the FFT results X_1 to $X_{N_{db}N_{be}}$ for example, considering the $2^{(N_{db}-1)}$ possible data bit path to get the possible results, the coherent integrations are added in N_{db} steps as illustrated in Fig.3. We add and save the different paths until step $(N_{db} + 1)/2$ (when N_{db} is odd) or $(N_{db} + 2)/2$ (when N_{db} is even), that is equal to the middle of the addition steps or one step more. At the next step the addition results will be doubled by considering possible data bit ± 1. The important strategy to eliminate possible paths is lie here. We compare all the results at this new step and only save the maximum half of the results, which means save half of the paths for later use and discard the others. Then at the remain steps, we apply the same method and only keep half of the results. The last coherent integration matrix size will be $(2^{(N_{db}-3)}20N_{db}$ $N_{step})(N_{be}N_{ms})$, which is only 25% of the full matrix M_{full}.

To add this coherent integration matrix incoherently to the previous one, first it has to be circularly rotated at each

Doppler frequency bin using Equation (11). Then among all the $2^{(N_{db}-3)}$ possible data bit sign combinations of any tentative Doppler frequency bin, any tentative bit edge position and any tentative code phase, we preserve the most likely data bit sign combination and discard the others during the noncoherent integration steps; the resulting noncoherent integration matrix is of size $(20N_{db}N_{step}) \times N_{be}N_{ms}$.

The unlikely data bit path elimination will render the coherent integration operation a little bit more complicated than the conventional method, but it will save 75% of memory space. It will also slightly increase the probability of false alarm, but there is no loss at the incorrect data bit combination in the incorrect paths.

3.2. Code Phase and Bit Edge Elimination Noncoherently

The most likely bit edge positions and code phases obtained in a PIT can be saved and used in the next coherent integration step to reduce the bit edge and code phase search range. This method can be justified by the fact that the correct noncoherent integration cell will always contain useful information due to the correlation between the received signal and the local code replica. If the correct cell does not produce the maximum power value, its value will be within a certain range of the high values obtained from all the cells. Thus, it is not necessary to keep track of all the code phases and bit edge positions throughout all of the processing steps. Code

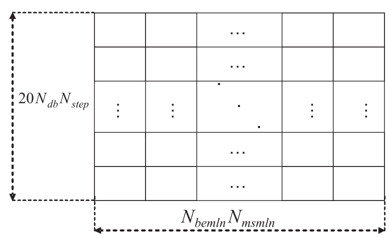

Fig. 4. Matrix formed by choosing the most likely N_{msmln} of N_{msn} possible code phase and N_{bemln} of N_{ben} possible bit edges at noncoherent integration step n

phase and bit edge elimination after one or several noncoherent integrations is introduced here to reduce the matrix size and computation. N_{msmln}, respectively N_{bemln}, is set to be a certain percentage of N_{msn}, respectively N_{ben}. The subscript n refers to the noncoherent integration step. After each coherent integration, the most likely data bit combination is determined. We sort the N_{msn} integration results among every $20N_{db}N_{step} \times N_{ben}$ units, then we only save N_{msmln} maximum out of $20N_{db}N_{step} \times N_{ben}$ units and discard the others. This step aims to save the most likely code phases. Then, we save the largest N_{bemln} of $20N_{db}N_{step} \times N_{msmln}$ units and discard the others. This step aims to save the most likely data edges. The resulting matrix, of size $20N_{db}N_{step} \times N_{bemln}N_{msmln}$, is shown in Fig.4. We save the most likely code phases and bit edges at this step and pass them to the next coherent integration. The next coherent integration only has to consider and search over these saved most likely code phases and bit edges. Computing the next coherent integration to form a matrix, first circularly rotate this matrix at each Doppler frequency bins using Equation (11) and then only add the overlapped most likely possible code phases and bit edges to the first one incoherently, and discard the other units. The same process on the rest of coherent and noncohernet integration processing will be used. This method can eliminate the possible bit edge detection, reduce the computation and save the memory space. The computation reduction at the next coherent integration step is $1-(N_{msmln}N_{bemln})/(N_{ms}N_{be})$.

The saved data edge positions in the last step may be deviate from the correct ones by ± 1; so at the current step we may have to take this uncertainty into account by considering all these possible positions.

4. EXAMPLES

Simulation data is used here to verify the validity of unlikely data bit path elimination during the coherent integration steps. The $C/N0$ is set as 24dB-Hz, and the satellite signal strength 1. Searching over all the possible data bit paths, the output matrix at the right bit edge position is shown in Fig.5. The maximum coherent integration value, Doppler frequency bin and code phase are 1.02e8, 647, 505, respectively.

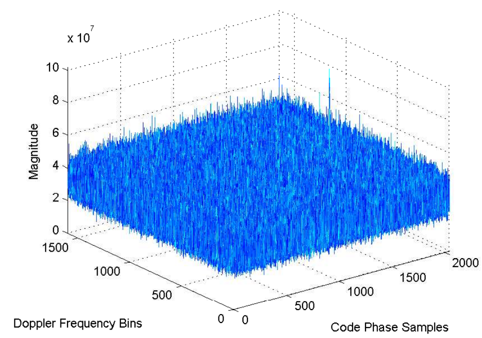

Fig. 5. Output matrix at the right bit edge position, before data bit combination path elimination

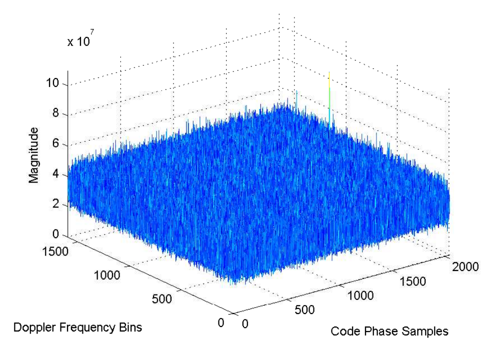

Fig. 6. Output matrix at the right bit edge position, after correct data bit path combination elimination

Applying the 75% unlikely data bit path elimination, the output matrix at the right bit edge position is shown in Fig.6. The maximum coherent integration value, Doppler frequency bin and code phase are 1.10e8, 647, 505, respectively. They are nearly the same as the results obtained using all possible path search method. The simulation was repeated nine more times and and similar observations were made. This supports the claim that the 75% unlikely data bit path elimination method achieves the acquisition goal at the setted simulation environment. We have also tested the saving 1 over 2 possible path method (this mehtod only keep the larger one of the two possille paths on every addition step illustrated in Fig.3) at each step of the N_{db} steps using the same simulation environment. This method can reduce more processing but failed to keep the right path because the increased probability of false alarm is unacceptable.

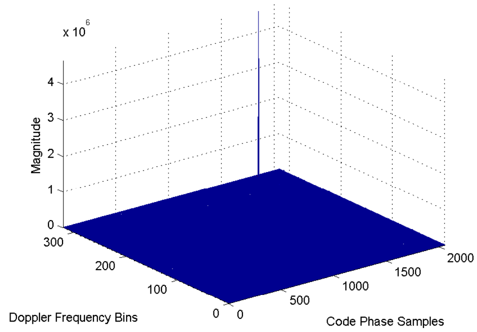

Fig. 7. Output matrix before bit edge and code phase elimination at the right bit edge position

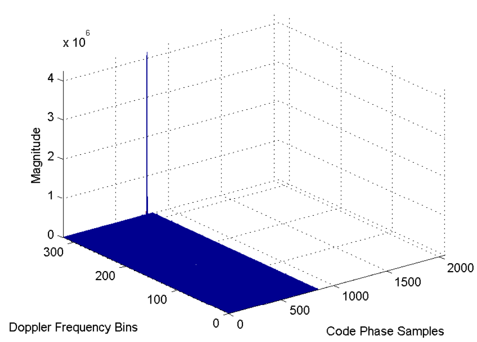

Fig. 8. Output matrix after bit edge and code phase elimination at the right bit edge position

To test the unlikely code phase and bit edge elimination together during noncoherent integration steps, the data used here are collected by a USB RF Front-End in an indoor environment. In order to speed up computation when using FFT, the original data is resampled with frequency $f_s = 2.048$ MHz. For code phase and bit edge elimination after a predefined number of coherent integration steps, we set PIT as $T_I = 20$. We also set $N_{be} = 20$. And the $C/N0$ is about 30dB-Hz.

Taking satellite PRN01 for example, the acquisition results before bit edge and code phase elimination has a size of $320 \times 2048 \times 20 = 13107200$. The results of the integration matrix at the right bit edge position is shown in Fig.7. Now we set the eliminated bit edge numbers as $[10 \quad 2 \quad 2 \quad 2 \quad 2 \quad 1]$, and eliminated code phase numbers as $[248 \quad 200\ 200 \quad 200 \quad 200 \quad 200]$ corresponding to the 6 noncoherent integration steps. The result is shown in Fig.8, where the code phase axis has unit *index* instead of *chips*. Each index corresponds to a value of the code shift chips. The new matrix sizes at the 6 steps are $[5760000 \quad 4096000 \quad 2688000 \quad 1536000\ 640000 \quad 256000]$, which implies that the processing and the memory space are reduced by $[56.1\% \quad 68.8\%\ 79.5\% \quad 88.3\% \quad 95.1\% \quad 98.1\%]$, with an average value of 81.0%.

5. SUMMARY AND CONCLUSION

In this paper, three methods are proposed to improve FMD-BZP to further reduce the computation processing and save memory space. These are unlikely data bit combination path elimination during coherent integration steps, code phase and bit edge elimination during noncoherent integration steps. The examples and results show that the unlikely data bit combination path elimination method can reduce the computation and save the memory space by nearly 75% during the coherent integration, while, on average, the code phase and bit edge elimination methods together reduces these metrics by nearly 81.0% during the noncoherent integration steps. Further study on the impact of the three methods on the probability of false alarm should be considered.

6. REFERENCES

[1] B. Parkinson and J. Spilker, "Global positioning system: Theory and applications," *American Institute of Aeronautics and Astronautics*, 1996.

[2] P. Misra and P. Enge, "Global positioning system: Signals, measurements, and performance," *Ganga-Jamuna Press*, December 2001.

[3] J. B. Y. Tsui, "Fundamentals of global positioning system receivers: A software approach (second edition)," *Publisher: Wiley-Interscience*, November 2004.

[4] D. J. R. Van Nee and A. J. R. M. Coenen, "New fast gps code-acquisition technique using fft," *IEEE Electronics Letters. Vol. 27, No. 2*, January 1991.

[5] D. M. Lin and L. B. Y. Tsui, "Acquisition schemes far software gps receiver," *Proceedings of ION GPS-1998*, pp. 317–325, September 1998.

[6] D. M. Lin and J. B. Y. Tsui, "Comparison of acquisition methods for software gps receiver," *Proceedings of ION GPS-2000*, pp. 2385–2390, September 2000.

[7] D. M. Lin, J. B. Y. Tsui, and D. Howell, "Direct p(y)-code acquisition algorithm for software gps receivers," *Proceedings of ION GPS-99*, September 1999.

[8] N. I. Ziedan and J. L. Garrison, "Unaided acquisition of weak gps signals using circular correlation or double-block zero padding," *Position Location and Navigation Symposium (PLANS) 2004*, pp. 461–470, April 2004.

[9] G. W. Heckler. and J. L. Garrison, "Implementation and testing of an unaided method for the acquisition of weak gps c/a code signals," *Navigation. Vol. 56, No. 4*, pp. 241–259, Winter 2009.

[10] N. I. Ziedan, "Gnss receivers for weak signals," *Artech House, London*, 2006.