Home

Performance analysis for a new CDMA long code fast computing method

Jiaming He, Xingbin Zeng, Bensong Xu
Institute of Communication Technologies
Ningbo University, 315211, P.R.China
jmhe@mail.nbptt.zj.cn

Abstract: In this article, we present a new CDMA long code fast computing method. Simulation and analysis shows that, compared with MSRG configuration, the new method requires no more hardware RAM, and it can turn the vast long code generating operation to no more than 42×42 times long code generating operation and 42 times 42×42 matrix mode 2 multiplication operations. So, the new method greatly accelerates the CDMA long code generating speed, and improves the system performance.

Key Words: long code state, fast computing

I. Introduction

Long code is very important in a CDMA communication system. For the forward link, long code is used for scrambling; for the reverse link, it is used for spreading, and uniquely identifies a mobile station. In CDMA system, long code is a PN sequence with period 2^{42}-1, and it is easy to obtain the current chip number. The difficulty lies in that, if we compute the current long code by the long code generator, the number of operations will increase greatly with the chip number.

II. Principles

The long code characteristic polynomial in IS-95 standard is as follows [1]:

$f(x)=1+x^7+x^9+x^{11}+x^{15}+x^{16}+x^{17}+x^{20}+x^{21}+x^{23}+x^{24}+x^{25}+x^{26}+x^{32}+x^{35}+x^{36}+x^{37}+x^{39}+x^{40}+x^{41}+x^{42}$

A. Long Code Generation Steps

The following four steps are needed to generate long code:

1) Compute the time period between current time and the starting time of GPS;

2) Compute the corresponding chip number of current time;

3) Through the chip number, deduce the current long code state;

4) Driven by the system time clock, the long code generator generates long code.

In this article, we focus in step 3: with known chip offset number n, we present a new fast long code computing method. All the addition and multiplication operations are mode 2 operations.

B. Current Long Code Computing Methods

Denote $\mathbf{S}(n)=[s_{41}(n)\ s_{40}(n)\ \dots\ s_0(n)]^T$ as the long code generator state at chip offset n, $\mathbf{A}$ as the 42×42 transform matrix, $\mathbf{Y}(n)=[y(n+41)\ y(n+40)\ \dots\ y(n)]^T$ as a 42×1 vector formed by 42 concecutive long code generator output, C as the mask of 1×42 vector.

According to the long code generate polynomial, the state equation is:

$$\mathbf{S}(n)=\mathbf{AS}(n-1)$$

The output of long code generator is:

$$y(n)=\mathbf{CS}(n+1)$$

B.1. Classical Methods

For MSRG Configuration[2]:

$$\mathbf{A}=\begin{bmatrix} a_{41} & 1 & 0 & \cdots & 0 & 0 \\ a_{40} & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

For SSRG Configuration [2]:

$$\mathbf{A}=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & a_{41} & a_{40} & \cdots & a_2 & a_1 \end{bmatrix}$$

B.2. Fast Computing Method [3]

From the state equation, we get

$$\mathbf{Y}(n)=\begin{bmatrix} y(n+41) \\ \vdots \\ y(n+1) \\ y(n) \end{bmatrix}=\begin{bmatrix} \mathbf{CA}^{41} \\ \vdots \\ \mathbf{CA} \\ \mathbf{C} \end{bmatrix}\mathbf{S}(n)=\mathbf{FS}(n),$$

where $\mathbf{F}=[\mathbf{CA}^{41}\ \dots\ \mathbf{CA}\ \ \mathbf{C}]^T$ ($\mathbf{CA}^i$ means mode 2 multiplication).

Choose an appropriate $\mathbf{C}$ to make matrix $\mathbf{F}$ invertible, and denote the inverse matrix as $\mathbf{G}$, then $\mathbf{GFS}(n)=\mathbf{GY}(n)$, where $\mathbf{GF}=\mathbf{I}$(mode 2). So $\mathbf{S}(n)=\mathbf{GY}(n)$, which means, by consecutive 42 long code generator output $\mathbf{Y}$, we can get the long code generator state $\mathbf{S}(n)$.

Since $\mathbf{S}(n)=\mathbf{AS}(n-1)=(\mathbf{A}^n \mod 2)\mathbf{S}(0)$, if we turn n into binary, $n=\sum_{i=0}^{41} n_i\times 2^i$, where n_i=0 or 1, then $\mathbf{A}^n=\prod_{i=0}^{41}(\mathbf{A}^{2^i})^{n_i}$.

Denote $\mathbf{s}_0(n)=\mathbf{A}^{n_0}\mathbf{S}(0)$, $\mathbf{s}_i(n)=(\mathbf{A}^{2^i})^{n_i}\mathbf{s}_{i-1}(0)$, i=0, ⋯, 41, then $\mathbf{S}(n)=\mathbf{s}_{41}(n)$.

Suppose we take $\mathbf{C}_0\mathbf{A}^{2^i}$ as the long code mask and $\mathbf{S}_{i-1}(n)$ as the long code generator state, where $\mathbf{C}_0$ is the initial mask. Then the consecutive 42 output of the long code generator will be:

This paper is sponsored by Natural Science Fund (No.601018), Zhejiang Province, P.R.China

Top

$$\mathbf{Y}_i = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{41} \end{bmatrix} \times \mathbf{S}_{i-1}(n) = \begin{bmatrix} \mathbf{C}_0 \\ \mathbf{C}_0\mathbf{A} \\ \vdots \\ \mathbf{C_0A}^{41} \end{bmatrix} \times \mathbf{A}^{2^i} \mathbf{S}_{i-1}(n)$$

$$= \begin{bmatrix} \mathbf{C}_0 \\ \mathbf{C}_0\mathbf{A} \\ \vdots \\ \mathbf{C_0A}^{41} \end{bmatrix} \mathbf{S}_i(n) = \mathbf{F_0 S}_i(n)$$

where $\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 & a_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & a_{40} \\ 0 & 0 & \cdots & 0 & 1 & a_{41} \end{bmatrix}$, $\mathbf{C}_0 = [0 \;\; 0 \;\; \cdots \;\; 0 \;\; 1]$, then

$\mathbf{S}_i(n) = \mathbf{G_0 Y}_i$.

So, if we store $\mathbf{G}_0$ and $\mathbf{C}_0\mathbf{A}^{2^i}$, i=0⋯41, knowing n, we can compute $\mathbf{S}(n)$ by deduction, and the output of long code generator $y(n)$ can be given by $y(n)=\mathbf{CS}(n)$.

III. Simulation Models

A. Method one: Deduction method

Due to the 42×42 characteristic matrix **A**, the long code initial state **S**(0), and the long code chip offset n, we have:

$\mathbf{S}(1)=\mathbf{AS}(0)$

$\mathbf{S}(2)=\mathbf{AS}(1)=\mathbf{A}^2\mathbf{S}(0)$

…

$\mathbf{S}(n)=\mathbf{A}^n\mathbf{S}(0)$

To get **S**(n), we need n times 42×42 matrix mode 2 multiplication with a 42×1 vector, the number of operation is as follows:

Mode 2 multiplication: $M_1=42\times42\times n=1764n$;

Mode 2 addition: $A_1=42\times41\times n=1722n$.

B. Method two: x^k *mode* $f(x)$ [2]

According to MSRG configuration, the output sequence satisfies $\frac{\mathbf{g}(x)}{f(x)} = \frac{\mathbf{S}^*(x)}{f(x)}$, where $\mathbf{S}^*(x)$ is the conjugate polynomial for $\mathbf{S}(x)$. To get the current long code state $\mathbf{S}(x)$, we should do as follows:

$\mathbf{g}(x)=x^k \bmod f(x)$

$\mathbf{S}(x)=\mathbf{g}^*(x)$

k is the current chip offset, that means, $k=n$. So, the operation mainly depends on x^k mod f(x). If ignoring the optimizing of operator 0, the number of operation is as follows:

Mode 2 multiplication: $M_2=43\times n$.

C. Method three: Fast computing

The fast computing algorithm has been described in section 2.2.2. The computing steps are as follows:

1) Turn chip offset n into binary: $n = \sum_{i=0}^{41} n_i \times 2^i$, where n_i=0 or 1;
2) i=0;
3) If n_i=1 then jump to step 6;
4) $\mathbf{Y}_i=\mathbf{F}_0\mathbf{S}_i(n)$;
5) $\mathbf{S}_{i+1}(n)=\mathbf{G}_0\mathbf{Y}_i$;
6) $i=i+1$;
7) If i<42 then back to step 3;
8) $\mathbf{S}(n)=\mathbf{S}_{41}(n)$.

To get **S**(n), there will be $42\times\sum_{i=0}^{41} n_i$ times long code generator operation and $\sum_{i=0}^{41} n_i$ times vector mode 2 multiplication of 42×42 matrix with 42×1 vector. The numbers of operation are:

Mode 2 multiplication:

$$M_3 = [42\times(42\times42+42)+42\times42]\times\sum_{i=0}^{41} n_i = 77616\sum_{i=0}^{41} n_i$$

Mode 2 addition:

$$A_3 = [42\times(42\times41+41)+42\times41]\times\sum_{i=0}^{41} n_i = 75768\sum_{i=0}^{41} n_i$$

IV. Simulation Results and Analysis

As shown in Fig.1, Fig.2 and Fig.3, the numbers of operation for method one and method two increases linearly with chip offset n, the slope for method two is less than that for method one, while the number of operation for method three wavers in a small range, with its local average increases slightly. Specially, when $n=2^i$(i=0, 1, …, 41), the number of operation for method three will be the minimum, 77616 for addition and 75768 for multiplication, respectively. As shown in Fig.4, when $n\geq2^8$, the number of operation for method three will be less than that for method one; when $n\geq3\times2^{14}$, the number of operation for method three will be less than that for method two. For all three methods, the number of operation will reach the maximum when $n=2^{42}$-1. Using TMS320C5420, whose speed reaches 200MIPS, the time needed to perform the operation at different n will be as shown in table 4-1.

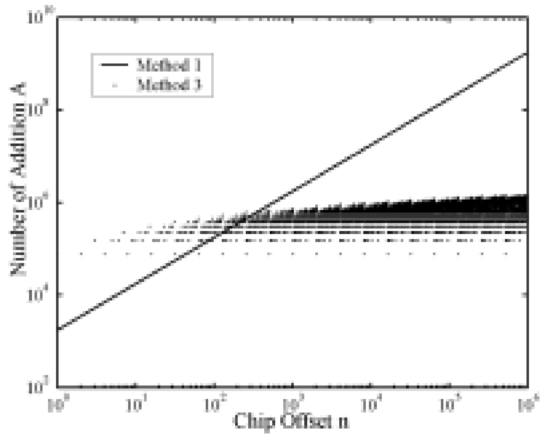

Fig.1 Numbers of operation for Addition ($1\leq n\leq2^{20}$)

Top

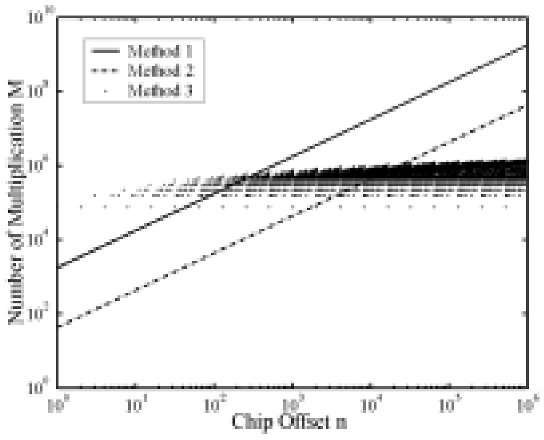

Fig.2 Number of operation for Multiplication ($1 \le n \le 2^{20}$)

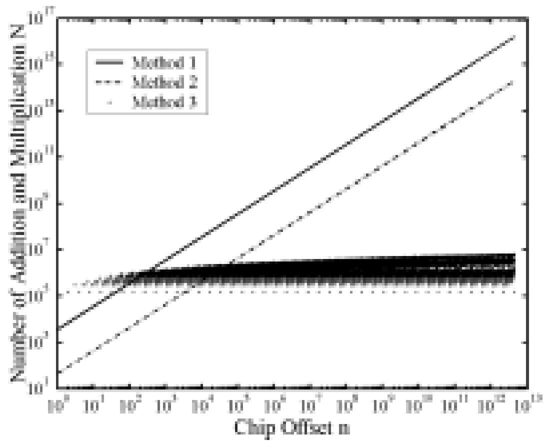

Fig.3 Numbers of operation for Addition and Multiplication ($1 \le n \le 2^{42}$)

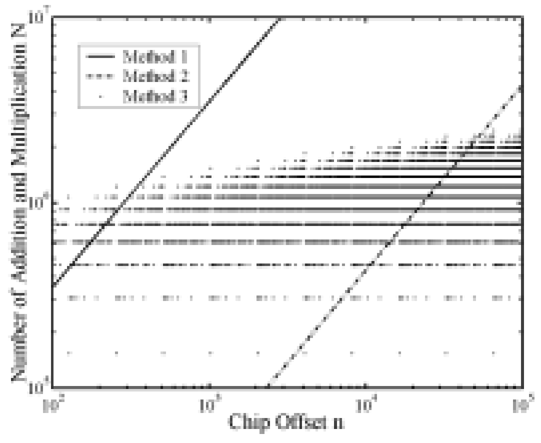

Fig.4 Numbers of operation for Addition and Multiplication ($10^2 \le n \le 10^5$)

As chip offset number n lies between 1 and 2^{42}-1, for method three, the number of mode 2 addition lies between 75768 and 3182256, mode 2 multiplication lies between 77616 and 3259872. Using TMS320C5420, all the operation can be done between 0.767ms and 32.21ms. As a comparison, the time lies between 0.017ms and 21294h for method one and 0 to 263h for method two. Using some other skills, the new method could be optimized further.

TABLE I Numbers of operation and time needed to perform the operations

Chip Offset n	Computing Method	Number of Addition	Number of Multiplication	Time
2^8-1	Method 1	439110	449820	4.445ms
	Method 2	NA	10965	0.055ms
	Method 3	606144	620928	6.136ms
2^8	Method 1	440832	451584	4.462ms
	Method 2	NA	11008	0.055ms
	Method 3	75768	77616	0.767ms
3×2^{14}-1	Method 1	84638022	86702364	856.702ms
	Method 2	NA	2113493	10.567ms
	Method 3	1136520	1164240	110503ms
3×2^{14}	Method 1	84639744	86704128	856.719ms
	Method 2	NA	2113536	10.567ms
	Method 3	75768	77616	0.767ms
2^{42}-1	Method 1	7.5734×10^{15}	7.7582×10^{15}	≈21294h
	Method 2	NA	1.8912×10^{14}	≈263h
	Method 3	3182256	3259872	32.21ms

V. Summary and Conclusion

Using the fast computing method, the CDMA long code state can be determined within 4ms by TMS320C5x DSP chip, and it needs only 1764B to store the 1×42 binary vector $\mathbf{C}_0\mathbf{A}^{2^i}$ (i=0~41). All this shows that the fast computing method is reliable to engineering applications. This method has been used in the "High Speed CDMA BTS Demo System" made by Institute of Communication Technology, Ningbo University.

Reference

[1] http://www.3gpp2.org, "3GPP2.S0002-A-1 Version 1.0", Sep.12, 2000.

[2] J.S.Lee, L.E.Miller, CDMA Systems Engineering Handbook, Translated by Xibin Xu, etc., Beijing: People's Post & Telecommunications Publish House, 2000.11.

[3] Xuequan Xiong, Xiaojun Wang, Jinxiang Li, Xiaoming Chen, Xingbin Zeng, "A fast CDMA long code computing method," Chinese patent number 01132029.X.

Top

Title: Performance Analysis for a New CDMA Long Code Fast Computing Method
Version: #2
Area: WCDMA, CDMA2000, EDGE
Date of Submission: 2002-12-20
Authors: Dr. Jiaming He, Dr. Xingbin Zeng, Bensong Xu
Mail: jmhe@mail.nbptt.zj.cn
Daytime telephone: 0086-574-87908556
Fax: 0086-574-87908161
Contact: Xingbin Zeng, znxb@sina.com

Top