SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2018, Vol. 61 082304:1–082304:12
https://doi.org/10.1007/s11432-017-9279-3

A pseudo-random sequence generation scheme based on RNS and permutation polynomials

Shang MA[*], Jianfeng LIU, Zeguo YANG, Yan ZHANG & Jianhao HU

National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China

Received 15 June 2017/Revised 24 August 2017/Accepted 10 October 2017/Published online 18 May 2018

Abstract Long period pseudo-random sequence plays an important role in modern information processing systems. Base on residue number system (RNS) and permutation polynomials over finite fields, a pseudo-random sequence generation scheme is proposed in this paper. It extends several short period random sequences to a long period pseudo-random sequence by using RNS. The short period random sequences are generated parallel by the iterations of permutation polynomials over finite fields. Due to the small dynamic range of each iterative calculation, the bit width in hardware implementation is reduced. As a result, we can use full look-up table (LUT) architecture to achieve high-speed sequence output. The methods to find proper permutation polynomials to generate long period sequences and the optimization algorithm of Chinese remainder theorem (CRT) mapping are also proposed in this paper. The period of generated pseudo-random sequence can exceed 2^{100} easily based on common used field programmable gate array (FPGA) chips. Meanwhile, this scheme has extensive freedom in choosing permutation polynomials. For example, 10905 permutation polynomials meet the long period requirement over the finite field F_q with $q \not\equiv 1 (\bmod 3)$ and $q \leqslant 503$. The hardware implementation architecture is simple and multiplier free. Using Xilinx XC7020 FPGA chip, we implement a sequence generator with the period over 2^{50}, which only costs 20 18kb-BRAMs (block RAM) and a small amount of logics. And the speed can reach 449.236 Mbps. The National Institute of Standards and Technology (NIST) test results show that the sequence has good random properties.

Keywords pseudo-random sequence, residue number system (RNS), permutation polynomial, high speed, long period, field programmable gate array (FPGA)

Citation Ma S, Liu J F, Yang Z G, et al. A pseudo-random sequence generation scheme based on RNS and permutation polynomials. Sci China Inf Sci, 2018, 61(8): 082304, https://doi.org/10.1007/s11432-017-9279-3

1 Introduction

Digital pseudo-random sequence is very important and has been widely used in modern information processing systems, including communication, encryption/decryption, scrambling, etc. Methods based on linear or non-linear feedback shift registers, congruential and chaotic mapping are common approaches to generating pseudo-random sequence. The pseudo-random code is not only the basis of multiple access of spread spectrum communication system but also the key of synchronization in 4G mobile communication systems [1]. Feedback shift register has a simple and efficient implementation architecture [2,3]. However, it only has small amount of polynomials and has been widely studied. Thus, it is not an appropriate choice in security communication or encryption systems [1,4–7]. In the past few decades, pseudo-random sequence generation method based on chaotic mapping has been deeply studied. The classical chaotic

[*] Corresponding author (email: mashang@uestc.edu.cn)

mappings include Tent, Chebyshev and Logistic mapping. A method using two chaotic sequences to generate new chaotic mapping was proposed in [6]. Araki et al. [8] analyzed the performance of Logistic maps in integer domain. These methods have similar problems with traditional chaotic mappings. That is, the calculation bit width of its iterative loop will be extremely huge. For example, the output rate can only reach tens of megabits per second of the sequence with period about 2^{50} based on traditional chaotic mapping. Fernando used memristors to generate pseudo-random sequence [9]. Test results based on National Institute of Standards and Technology (NIST) shown the good random properties. However, it is difficult to design the analog circuit for system synchronization. Ref. [10] proposed a method based on semiconductor laser technology to generate the extremely high-speed chaotic sequence (up to 40 Gbps) and analyzed the theoretical limit of the generation rate. However, this method is hard to be implemented in general applications.

Pseudo-random sequence generated by chaotic method has long period property and good randomness, which is a good choice for security and encryption systems. Because of the finite word length effect, the iterative bit width is directly related to the period of classical chaotic mapping. For example, the common-used methods, such as Logistic, Tent and Chebyshev maps can only reach the period of 10^8 at 60-bits width. Hence, period extension is usually required in practice for traditional chaotic mapping and requires more complex computation. On the other hand, excessive calculation bit width will result in the increase of iterative boundaries in the hardware implementation, thus, the high sequence output rate is difficult to be achieved. Residue number system (RNS) is one of the important contributions to the world for ancient China, which uses several parallel and independent smaller calculations to complete large calculations to reduce the hardware complexity. RNS has been deeply studied and widely used in encryption and digital signal processing (DSP) systems. Aiming at the problem of large bit width calculation during the iteration for random sequence generation, Harris Corporation proposed a sequence generation method based on RNS and special chaotic polynomials [11], and pointed out that its information entropy is similar to white noise [12]. However, it only gave the selection method of cubic iterative polynomial with specific form in published literature. And the principles is not analyzed in detailed.

In this paper, we propose a high speed digital pseudo-random sequence generation scheme based on permutation polynomials and RNS. We also present the polynomial selection method and the optimization for Chinese remainder theorem (CRT) mapping. This scheme uses several relatively prime permutation polynomials over the finite field to complete iterative operations independently and parallel. The iterative loop has smaller computation delay. Hence the iterative speed can be ensured. Then, the iterative results of each channel are extended by optimized CRT to a single equivalent integer ring. Thus, we can get an extremely long period and high-speed sequence output. The NIST test results show that the generated sequence can pass all NIST test items and has good randomness. Besides, the proposed scheme has a great degree of freedom in iteration polynomials selection. For example, it has 10905 permutation polynomials that meet the long period requirement just over finite field F_q with $q \not\equiv 1 (\mathrm{mod}\ 3)$ and $q \leqslant 503$. Finally, we propose a multiplier-free architecture and implement the proposed scheme in Xilinx XC7Z020 field programmable gate array (FPGA) chip. The implementation results show that its generation rate can reach up to 449.236 Mbps, and this is about 11 times of traditional method implemented on the same platform. And the sequence period is also 2^{50} times of traditional method. For hardware consumption, it only costs 20 18 kb-BRAMs (block RAM) and a few logic resources.

2 RNS and permutation polynomial

2.1 Residue number system

RNS is a non-weight numerical representation system and defined by a set of relatively prime radix $\{m_1, m_2, \ldots, m_L\}$ [13]. An integer X can be represented in RNS as $\{x_1, x_2, \ldots, x_L\}$, in which x_i is the residue of X mod m_i and denoted as $x_i = \langle X \rangle_{m_i}$. The integers in the range of $[0, M)$ can be uniquely represented in this RNS, where $M = \Pi_{i=1}^{L} m_i$. Let the representations of integers A, B and C in RNS be $\{a_1, a_2, \ldots, a_L\}$, $\{b_1, b_2, \ldots, b_L\}$ and $\{c_1, c_2, \ldots, c_L\}$ respectively. According to the rules of modulo operations, if $c_i = (a_i \Delta b_i) \bmod m_i$, then $C = \langle A\Delta B\rangle_M$, where "$\Delta$" can be the addition, subtraction or multiplication. The RNS integer $\{x_1, x_2, \ldots, x_N\}$ can be converted to the corresponding integer X by the celebrated CRT,

$$X = \left\langle \sum_{i=1}^{L} M_i \langle M_i{}^{-1}\rangle_{m_i} x_i \right\rangle_M , \tag{1}$$

where $M_i = M/m_i$, $\langle M_i^{-1}\rangle_{m_i}$ is the modulo multiplicative inverse of M_i mode m_i and $\langle\langle M_i^{-1}\rangle_{m_i} M_i\rangle_{m_i} = 1$. CRT is one of the fundamental theorems in RNS and plays an important role in RNS for R/B (residue to binary) conversion, scaling, magnitude comparison, and overflow detection, etc.

In RNS, modulo multiplications and additions in residue channels are mutually independent, which embodies the parallelism property of RNS and can achieve better performance of area, latency, and power consumption. On the other hand, small integers in RNS can be mapped to traditional binary system by CRT. That is, we can use low complex computation in RNS to complete extremely large number computation in the traditional binary system and achieve high-speed output. In this paper, we just use these properties of RNS to get long period and high speed sequence output.

2.2 Permutation polynomial

Permutation polynomial is the polynomial that can express complete residue classes [14,15]. A polynomial $f(x) \in F_q(x)$, if the associated function $f: a \to f(a)$ from F_q and F_q is a permutation of F_q, then $f(x)$ is a permutation polynomial of F_q [16]. Equally, permutation polynomial also can be defined as that $f(x)$ is permutation polynomial of F_q if and only if one of the following conditions holds:

(1) function $f: c \to f(c)$ is one-to-one (an injection);
(2) function $f: c \to f(c)$ is onto (a surjection);
(3) $f(x) = a$ has a solution in F_q for each a in F_q;
(4) $f(x) = a$ has a unique solution in F_q for each a in F_q.

However, these four conditions can only be used to verify whether a polynomial is a permutation polynomial, but cannot generate a permutation polynomial. Dickson polynomial can be used to generate permutation polynomial on given finite fields, this polynomial is as

$$g_k(x,a) = \sum_{j=0}^{[k/2]} \frac{k}{k-j} \begin{pmatrix} k-j \\ j \end{pmatrix} (-a)^j x^{k-2j}, \tag{2}$$

where k is the order of the polynomial and $a \in F_q \backslash \{0\}$. By using Dickson polynomial, we can find all permutation polynomials of degree at most five over all finite fields shown in Table A1 in Appendix A. With careful selection of the constant term of permutation polynomial, the iteration period of each finite field will be the longest. This is the foundation of long period sequence scheme proposed in this paper.

3 Pseudo-random sequence generation method based on RNS and permutation polynomial

3.1 Generation method

3.1.1 *Iterative method and iteration period of permutation polynomial over finite field*

Assuming $f_l(x)$ is a permutation polynomial over finite fields F_{q_l}, and the order of finite field is m_l, $l = 1, 2, \ldots, L$. According to Subsection 2.2, if $x \in [0, m_l)$, then $f_l(x) \in [0, m_l)$, they satisfy the injection and surjection relation. The iterative process of permutation polynomials $f_l(x)$ over finite field F_{q_l} is defined as

$$x^l_{k+1} = f_l(x^l_k), \tag{3}$$

the iterative period p_l of (3) is defined as follows: assuming that the initial iterative value is x^l_0 and $\{x^l_1, x^l_2, \ldots, x^l_N\}$ is a set of N $(N = 1, 2, 3, \ldots)$ times iterative results, if the iterative result is one of the

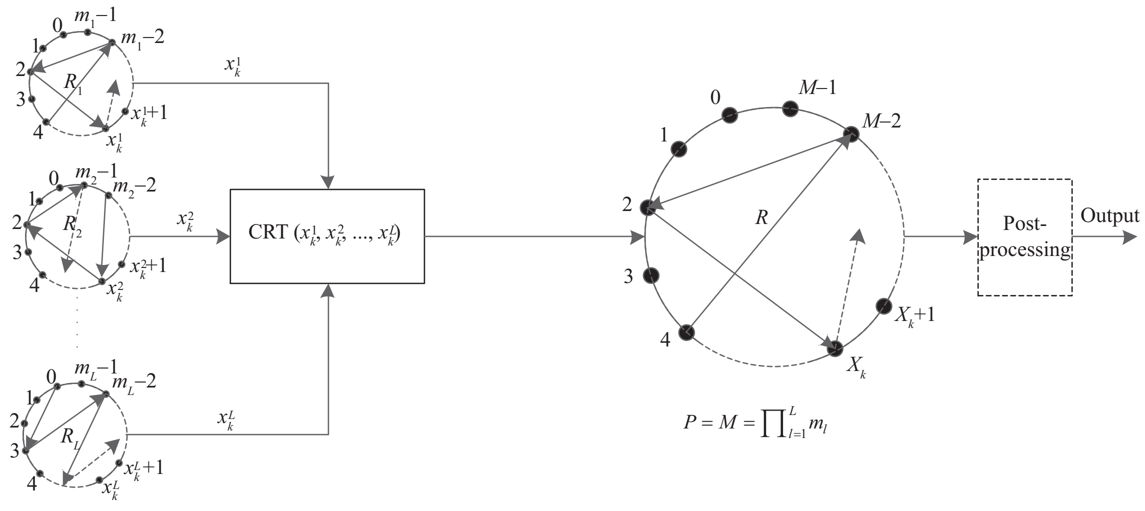

Figure 1 The proposed pseudo-random sequence generation method base on permutation polynomial and RNS.

elements of $\{x_0^l, x_1^l, x_2^l, \ldots, x_N^l\}$ by the $(N+1)$th iteration, then iterative period is N. Because $f_l(x)$ is permutation polynomial, obviously, $0 < p_l \leqslant m_l$.

3.1.2 *Iterative period extension based on CRT*

Assuming that $p_l = m_l$, $\mathrm{GCD}(m_i, m_j) = 1$ and $i \neq j$, ($\mathrm{GCD}(m_i, m_j)$ is the greatest common divisor of m_i and m_j), where m_j is the order of each finite field. In other words, the orders of finite field are relatively prime. Obviously, iterative polynomials generate no more than $\Pi_{l=1}^{L} m_l$ different residue vectors $\{x_k^1, \ldots, x_k^l, \ldots, x_k^L\}$, where $k \in [0, \Pi_{l=1}^{L} m_l)$. According to CRT, residue vectors $\{x_k^1, \ldots, x_k^l, \ldots, x_k^L\}$ generated by L iterative processes over finite field are one to one mapping with an integer X_k. Because $p_l = m_l$ and $\mathrm{GCD}(m_i, m_j) = 1$, the period of generated integer X_k also is $\Pi_{l=1}^{L} m_l$.

The above procedure can be described in Figure 1. The iterations of L polynomials over L finite fields generate L integer rings, denoted as R_i $(i = 1, 2, \ldots, L)$. Then, CRT is used to convert residue vector $\{x_k^1, \ldots, x_k^l, \ldots, x_k^L\}$ to its corresponding integer X_k. The CRT converting results will generate a new integer ring over the finite fields $[0, \Pi_{l=1}^{L} m_l)$. If the iterative polynomial $f_l(x)$ is nonlinear, the output sequence of ring R has random property. In Figure 1, the post-processing module may contain Gauss mapping, sequence balance adjustment, bit mapping in terms of different applications.

Figure 1 shows that several small iterative loops are mapped to an equivalent iterative loop with extremely long period by using CRT. In turn, this method also replaces the finite field iteration which has large dynamic range with parallel iteration on several small range finite fields. As a result, the iterative boundary of loops is decreased and the generation rate is improved. In Figure 1, the method to select permutation polynomial is the key of randomness and long period of output sequence. And the optimization of CRT is the basic for reducing complexity.

3.2 Polynomial selecting method

According to Subsection 2.2, the mapping from x to $f(x)$ is one-to-one if $f(x)$ is a permutation polynomial. However, the iterative period of permutation polynomial cannot always reach its order when the iteration is defined as (3). The precondition of $p_l = m_l$ is the basic to get extremely long period sequence for our method in this paper. So, we need to select permutation polynomial carefully to guarantee this requirement.

3.2.1 *lost state and multiple loops problem*

For permutation polynomial $f_l(x^l)$, if $x_a^l = f_l(x_a^l)$ and the initial iteration value is x_a^l, then

$$x_{k+1}^l = f_l(x_k^l) = x_k^l. \tag{4}$$

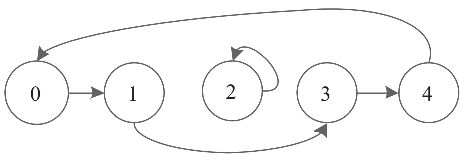

Figure 2 The state transition diagram of the iteration of permutation polynomial.

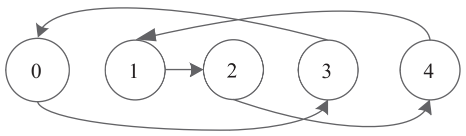

Figure 3 The state transition diagram of the iteration of $f(x) = \langle x^3 + x^2 + 2x + 3\rangle_5$.

Thus, each iteration has the same result and $p_l = 1$, that is so-called lost state. Even though x_a^l is not chosen as the initial iterative value, the iterative period cannot reach up to m_l because of the lost state. Consequently, the longest period characteristic cannot be guaranteed. For example, the iterative process according to (3) of permutation polynomial $f(x) = \langle x^3 + 3x^2 + 3x + 1\rangle_5$ on finite field F_q and $q \not\equiv 1 (\text{mod } 3)$ can be shown in Figure 2.

In which, the state transition loops are $0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 0$ and $2 \rightarrow 2$, the longest iterative loop period is state 4, and state 2 is the lost state.

On the other hand, if $1 < p_l < m_l - 1$ for any initial iterative value, the multi-loop problem will exist so that any iterative loop $f_l(x)$ cannot achieve the longest period. Thus, the longest period characteristic cannot be ensured. Take the permutation polynomial $f(x) = \langle x^3 + x^2 + 2x + 3\rangle_5$ on finite field F_q $q \not\equiv 1 (\text{mod } 3)$ as an example, its state transition is shown as Figure 3.

There are 2 iterative loops in Figure 3, $0 \rightarrow 3 \rightarrow 0$ and $1 \rightarrow 2 \rightarrow 4 \rightarrow 1$. Obviously, the period of two loops cannot reach up to 5 and the long period characteristics cannot be achieved. Therefore, permutation polynomial with the longest period has only one iterative loop.

3.2.2 *Permutation polynomial selection*

In order to ensure the randomness of the sequence, the permutation polynomial should have nonlinear characteristic. Since permutation polynomial cannot ensure the longest period characteristics in iterations, thus, permutation polynomial should be carefully selected to eliminate the above problems and make the random sequences after CRT extension have the longest period $\Pi_{l=1}^{L} m_l$ (i.e., the iterative period of each iteration loop is m_l).

Using the standard permutation polynomial in Table A1, we can generate many permutation polynomials. In fact, the constant term of permutation polynomial means its mapping offset on finite field. Different mapping offset leads to different number of iterative loops, but it would not change the permutation characteristic of permutation polynomial. We proposed Algorithm 1 for searching the constant terms of permutation polynomial so that the iterative period is the longest.

According to the definition of iterative period, the basic idea of this algorithm is as follows. Firstly, choose a permutation polynomial $\langle f(x)\rangle_m$ without constant term. Then, choose an initial value $x_0 = k$ to perform the iteration $x_{k+1} = \langle f(x_k) + c\rangle_m$, where c is the constant term and its value range is 0 to $m - 1$. When current iterative result equal to initial iterative result at the first time, stop the iterating. Finally, judge whether the number of iteration equals to the longest period. If condition is satisfied, the constant c satisfies the condition of single loop iteration. Otherwise, lost state or multiple-loop situation is existed.

Using this algorithm, we have searched the single loop permutation polynomial over finite field F_q $q \not\equiv 1 (\text{mod } 3)$ and $q \leqslant 503$. There are 10905 polynomials meeting requirement. Table A2 in Appendix A gives some of them on this condition. Therefore, the method proposed in this paper has extremely large freedom in polynomial selection.

3.3 Optimization for CRT

CRT is the most important method for converting RNS to the binary system. By using CRT, the value of integer and the arithmetic operations in RNS are one-to-one with those in traditional weighted systems. However, the large modulo operation in (1) is inefficient in practice. On the other hand, the output

Algorithm 1 An algorithm for selecting the constant term of permutation polynomial with the longest iterative period

```
1: Choosing a permutation polynomial $\langle f(x)\rangle_m$;
2: $x_0 = k$, $x_1 = \langle f(x_0) + c\rangle_m$    %setting iterative initial value and computing the corresponding results;
3: for $c = 0, \ldots, m-1$    %the value range of constant term $c$ in permutation polynomial;
   do
4:   for $i = 1, \ldots, m$    %the number of iteration is increasing;
     do
5:     $x_{i+1} = \langle f(x_i) + c\rangle_m$;    %iterative polynomial computation;
6:     if $x_{i+1} = x_1$    %if the current result equal to initial result;
       then
7:       break;
8:       if $i \neq m$    %if the current result equals to initial result and the number of iteration does not equal to $m$;
         then
9:         loop period is $i$;    %multiple-loop situation are existing, constant $c$ does not satisfy the longest period property;
10:      else
11:        loop period is $m$;    %iterative period reached maximum, constant $c$ satisfy the longest period property;
12:      end if
13:    end if
14:  end for
15: end for
```

of the scheme in this paper is a random sequence. If the mapping between RNS and binary system is unique, the output sequence will still have long period property. Thus, we do not need to guarantee the one-to-one property in arithmetic operations. As for the randomness of sequences, regardless of whether they are accurately numerical value mapped by CRT, special tests still are necessary. According to the above analysis, the computation of (1) can be optimized as

$$X_k = \sum_{l=1}^{L} M_l x_k^l. \tag{5}$$

Compared with (1), Eq. (5) removes the production of $\langle {M_i}^{-1}\rangle_{m_i}$ and the last modulo M operation. We will prove that the mapping of (5) is one-to-one between $\{x_k^1, \ldots, x_k^l, \ldots, x_k^L\}$ and X_k as following.

Proof. Let X_i and X_j be the results of remainder vector $\{x_i^1, \ldots, x_i^l, \ldots, x_i^L\}$ and $\{x_j^1, \ldots, x_j^l, \ldots, x_j^L\}$ mapped by (5) respectively. Assuming $X_i = X_j$, if we can prove that $\{x_i^1, \ldots, x_i^l, \ldots, x_i^L\}$ is equal to $\{x_j^1, \ldots, x_j^l, \ldots, x_j^L\}$, we can conclude that the map of (5) is injective.

According to (5), we have

$$X_i - X_j = \sum\nolimits_{l=1}^{L} M_l(x_i^l - x_j^l) = \sum\nolimits_{l=1}^{L} M_l\alpha_l, \tag{6}$$

where $\alpha_l = x_i^l - x_j^l$. Because $x_i^l \in [0, m_i)$, so $\alpha_l \in (-m_i, m_i)$.

Firstly, consider the two-channel residue vectors $\{x_i^1, x_i^2\}$ and $\{x_j^1, x_j^2\}$, their mapping results are X_i and X_j respectively. If mapping results by using (5) are equal, then

$$X_i - X_j = M_1\alpha_1 + M_2\alpha_2 = m_2\alpha_1 + m_1\alpha_2 = 0. \tag{7}$$

Thus,

$$\frac{m_2}{m_1} = -\frac{\alpha_2}{\alpha_1}. \tag{8}$$

Note that $\mathrm{GCD}(m_1, m_2) = 1$, the condition of $\alpha_1 \geqslant m_1$, $\alpha_2 \geqslant m_2$ is necessary for (8) to be established. But it does not satisfy the requirement of $\alpha_l \in (-m_l, m_l)$. So, $\alpha_1 = \alpha_2 = 0$ is necessary for (7).

For three-channel residue vectors $\{x_i^1, x_i^2, x_i^3\}$ and $\{x_j^1, x_j^2, x_j^3\}$, the mapping results by (5) are X_i and X_j respectively. If the computation results of (5) are equal, then

$$X_i - X_j = m_2m_3\alpha_1 + m_1m_3\alpha_2 + m_1m_2\alpha_3 = (m_2\alpha_1 + m_1\alpha_2)m_3 + m_1m_2\alpha_3 = 0. \tag{9}$$

Thus,

$$\frac{m_2\alpha_1 + m_1\alpha_2}{\alpha_3} = -\frac{m_1m_2}{m_3}. \tag{10}$$

Because $\mathrm{GCD}(m_3, m_1m_2) = 1$ and $\alpha_3 \in (-m_3, m_3)$, similarly, Eq. (11) is necessary for (9) to be established,

$$\alpha_3 = m_2\alpha_1 + m_1\alpha_2 = 0. \tag{11}$$

Considering the condition of (7) and (11) to be established, we can get $\alpha_1 = \alpha_2 = \alpha_3 = 0$. Thus, the mapping of (5) is injective for three-channel mapping.

Let X_i and X_j be the mapping results of L-channel remainder vectors $\{x_i^1, \dots, x_i^l, \dots, x_i^L\}$ and $\{x_j^1, \dots, x_j^l, \dots, x_j^L\}$ by (5). If $X_i = X_j$, then

$$\begin{aligned} X_i - X_j = m_L(m_{L-1}(\cdots m_3(m_2\alpha_1 + m_1\alpha_2) + m_1m_2\alpha_3) \\ + m_1m_2\cdots m_{L-2}\alpha_{L-1})\cdots + m_1m_2\cdots m_{L-1}\alpha_L = 0. \end{aligned} \tag{12}$$

Similarly, due to $\mathrm{GCD}(m_L, m_1m_2\cdots m_{L-1}) = 1$ and $\alpha_L \in (-m_L, m_L)$, we have

$$\begin{aligned} \alpha_L = m_L(m_{L-1}(\cdots m_3(m_2\alpha_1 + m_1\alpha_2) \\ + m_1m_2\alpha_3) + m_1m_2\cdots m_{L-2}\alpha_{L-1}) = 0. \end{aligned} \tag{13}$$

The rest can be done in the same manner, we have

$$\alpha_l = 0 \quad (l = 1, 2, \dots, L). \tag{14}$$

This means that if the mapping results of residue vectors $\{x_i^1, \dots, x_i^l, \dots, x_i^L\}$ and $\{x_j^1, \dots, x_j^l, \dots, x_j^L\}$ by (5) are equal, they must be equal. So, the mapping of (5) is one-to-one.

By improving the CRT mapping as shown in (5), the large modulo M operation is eliminated. Meanwhile, the multiplication of $\langle {M_i}^{-1} \rangle_{m_i}$ is also eliminated. Thus, the complexity of implementation can be significantly simplified.

4 Random test and analysis

The most important characteristic of pseudo-random sequence is randomness. In this section, we perform detailed randomness test and analysis for the sequence generated by the proposed scheme.

4.1 Test environment setting

4.1.1 *polynomial selection*

From Table A1 in Appendix A and the constant term searching algorithm, we select six permutation polynomials on the finite field F_q with $q \not\equiv 1 (\mathrm{mod}\ 3)$ and show them as

$$\begin{cases} f_1(x) = \langle x^3 + 3x^2 + 3x + 59 \rangle_{251}, \\ f_2(x) = \langle x^3 + 6x^2 + 12x + 77 \rangle_{347}, \\ f_3(x) = \langle x^3 + 9x^2 + 27x + 111 \rangle_{443}, \\ f_4(x) = \langle x^3 + 12x^2 + 48x + 42 \rangle_{467}, \\ f_5(x) = \langle x^3 + 15x^2 + 75x + 288 \rangle_{479}, \\ f_6(x) = \langle x^3 + 21x^2 + 147x + 8 \rangle_{503}. \end{cases} \tag{15}$$

The period of sequence X_k generated by this group polynomials is $M = 251 \times 347 \times 443 \times 467 \times 479 \times 503$ (about 2^{52}). If the generation speed is 100 Mbps, the period cycle is about 1.38 year.

4.1.2 *Test tool*

In this paper, the common used test tool, NIST, is adopted to evaluate the randomness of sequences generated by the proposed scheme. NIST test standard contains 15 test items, including frequency detection, discrete fourier transform test, linear complexity test, etc. This paper takes the latest software STS-2.1.2 with the standard of NIST SP800-22. In NIST test standard, each test item will generate a p_value. If $p\text{\_value} \geqslant 0.01$, the corresponding test item is regarded as having passed the test [17].

Table 1 Test results for the proposed pseudo-random sequence generation scheme

Test item	Passing rate p_value > 0.01	p_value Average	p_value Std	Passing rate [6] p_value > 0.01	p_value [6] Average
Frequency test	99.20%	0.4994	0.1372	98.00%	0.1538
Frequency test within a block	99.15%	0.4948	0.0182	99.00%	0.7792
Runs test	98.65%	0.5065	0.3433	100.00%	0.4944
Test for the longest run	99.05%	0.4971	0.1705	99.00%	0.5141
Binary matrix rank test	98.70%	0.4985	0.0935	99.00%	0.8832
Discrete Fourier transform	99.00%	0.4842	0.2865	98.00%	0.0428
Non-overlapping template	99.20%	0.5043	0.0986	98.38%	0.4794
Overlapping template	98.90%	0.5249	0.1426	100.00%	0.2493
Maurer's universal	98.75%	0.4960	0.6358	100.00%	0.1917
Linear complexity test	99.00%	0.5034	0.0218	98.00%	0.1154
Serial test	98.85%	0.4936	0.4220	98.50%	0.2523
Approximate entropy test	98.85%	0.5012	0.4215	99.00%	0.4373
Cumulative sums test	99.35%	0.5012	0.1212	98.00%	0.5087
Random excursions test	97.75%	0.5183	0.3868	98.89%	0.4506
Random excursions variant	98.95%	0.5134	0.3046	99.01%	0.2544
The number of success items	15/15	–	–	15/15	–

Table 2 The number of items which failed in individual sequence set testing

The number of items p_value < 0.001 in each 15 test items	The number of sequence sets
5	2
4	3
3	7
2	37
1	215
0	1736
Total	2000

4.2 Results and analysis

Because the length of the sequence of each test in STS-2.1.2 is no more than one million. We perform 2000 tests with sequence length of 10^6 and set the initial iterative value of (15) at random for each test. The test results are shown in Table 1.

In Table 1, the 1st and 4th column are passing rate when significance level α is 0.01 [18], the former is our test results and the latter is from reference [6]. The 2nd and last column are average of p_value, the former is our test results and the latter is from reference [6]. The 3rd column is standard error of our p_value test results. Table 1 shows that the passing rate of all test items is over 96.5%, which meets the threshold requirement of passing rate of NIST SP800-22 standard [18]. Besides, Table 1 shows that most of p_value averages value are more than that in [6], which indicates that the randomness of sequences generated by the method proposed in this paper is better than [6]. Table 2 gives the number of sequences corresponding to the number of items with p_value < 0.01.

In Table 2, 1736 of 2000 sequences can pass all of the test items. 215 sequences only fail in one test item. Only 7 sequences fail in 3 test items and only 2 sequences fail in 5 test items. It shows that there is not any sequence generated by the method proposed in this paper cannot pass the random number testing standard in all of items, and there is only minority sequence failling in a few test items. This indicates that the generated sequences have good randomness.

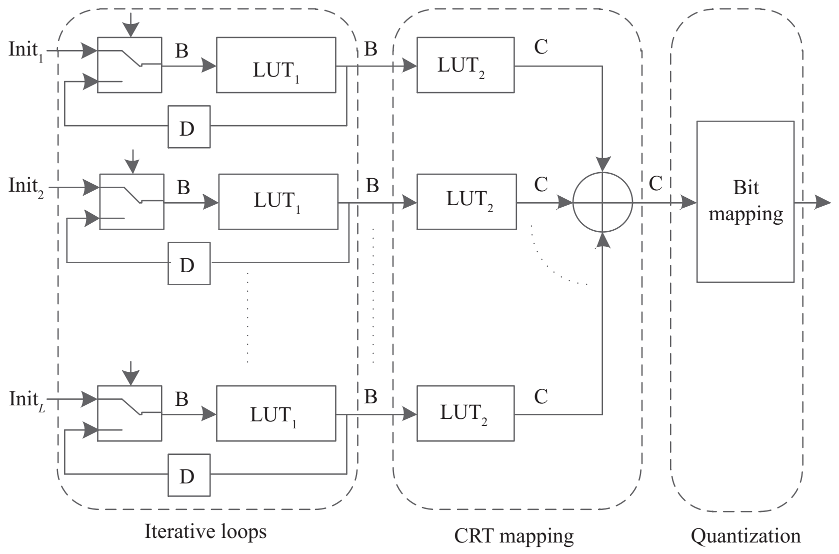

Figure 4 The FPGA implementation architecture.

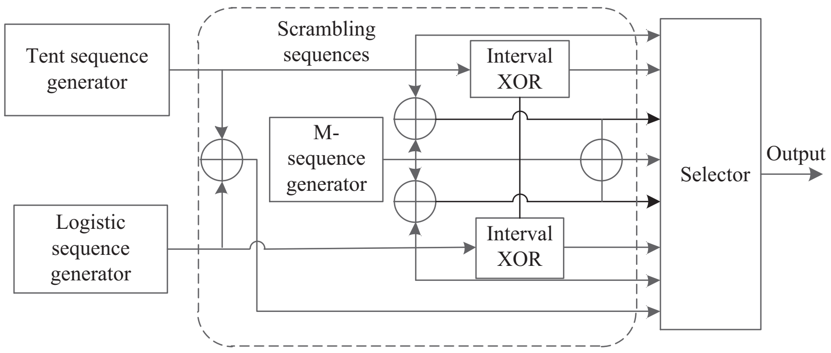

Figure 5 The FPGA implementation architecture of traditional method.

5 Implementation based on FPGA and performance analysis

The main advantage of the proposed pseudo-random sequence generation method is that the pseudo-random sequence with extremely long period can be generated by several small iterative loops. Thus, each iterative loop has small iterative boundary, and it is easy to achieve high-speed output in hardware implementation. Figure 4 is the FPGA implementation architecture based on the proposed scheme. Large dynamic range and long period output can be obtained by the combination of several channels due to each remainder channel can choose the finite field which has smaller dynamic range, such as smaller than 10 bits. As a result, the iterative computation of each channel can be implemented by look-up table (LUT). In addition, in order to further improve the generation speed, the CRT-based mapping part also uses LUT. The LUT depth of each iterative loop is the order of the finite field of the channel, and the corresponding bit width is B bits. Based on topology transitivity, take low C-bit data of the CRT extended output as input of the binary bit mapping model. If there are 10 channels, each channel has 10 bits iterative width and each CRT mapping LUT has 16 bits width, a total of 260 kb LUTs is needed. It is easy to be implemented in modern mainstream FPGA chips by Block Memory. Another benefit of this architecture based on LUT is that the data in LUT can be pre-computed and configured by the selected finite field permutation polynomial. Meanwhile, the iterative polynomials of each iterative loop have a great degree of selection freedom. Thus, the proposed method and architecture is more flexible in practice.

Figure 5 is the FPGA implementation architecture of traditional method. Table 3 is the comparison results of hardware consumption, speed and sequence period between our method and traditional chaotic

Table 3 Implementation comparison based on FPGA

	Traditional architecture	Proposed architecture
LUT	681 (1%)	245 (0%)
Register	181 (0%)	52 (0%)
BRAM	0 (0%)	20 (14%)
DSP	27 (12%)	0 (0%)
Generated speed	39.853 Mbps	449.236 Mbps
Period	2^{40}	2^{90}

mapping. In hardware implementations, the proposed architecture has 10 iterative loops. For traditional methods, the long-period characteristics obtained only by increasing the iterative loop width. This must cause the extremely slow output speed. Therefore, we use a common used periodic expansion method in conventional method based architecture. In which, it includes a Logistic mapping module, a Tent mapping module and an M-sequence generating module. The mutual scrambling result of them is taken as the final output sequence. Both of them is implemented by Xilinx XC7Z020 chip. As one of SOC chips, XC7Z020 is a commonly used low-cost SOC chip but it is not very good at area and speed. However, the speed of the proposed architecture still reaches up to more than 449.236 Mbps and is about 11 times as great as traditional architecture. Meanwhile, the sequence period of our method is about 2^{50} times of that of traditional method. In terms of the FPGA resource consumption, our method mainly costs 14% of the on-chip BRAM resources and only one third of the traditional method of LUT resources. Furthermore, the traditional method needs about 12% on-chip multipliers and our method is multiplier-free. Therefore, the method proposed in this paper has remarkable advantages no matter in the long period characteristics of sequences, hardware resource cost or generating speed.

6 Conclusion

Based on RNS and permutation polynomial over finite fields, a long period, high speed, and low complexity pseudo-random sequence generation scheme is proposed in this paper. We also propose a permutation polynomial selection method to ensure long period characteristics. In addition, an optimization of CRT mapping is presented and proved in detail. This scheme combines the non-linear iterative results on several small finite fields into an extremely large number ring and gets extremely long period pseudo-random sequence. The main advantages of the proposed generation scheme are as follows: (1) the iterative boundary of each iterative loop is very small, hence the speed can be improved significantly; (2) it is much easier to implement the extremely long period expansion only by increasing the iterative channels; (3) polynomial selection has very large freedom; (4) the implementation complexity is simplified and LUT can be used to implement this calculation, the generating speed can be further increased and flexible configurable capabilities can be provided. A special test based on NIST is taken in this paper. The results show that the generated sequences have good randomness. The implementation based on FPGA shows that the speed of our architecture is about 11 times of traditional ones and the period is about 2^{50} times of that generated by traditional methods. Compared with traditional architecture, our architecture mainly costs BRAM resources and is multiplier-free.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61571083).

References

1 Figueiredo F A P D, Mathilde F S, Cardoso F A C M, et al. Efficient FPGA-based mplementation of a CAZAC sequence generator for 3GPP LTE. In: Proceedings of International Conference on ReConFigurable Computing and FPGAs (ReConFig14), Cancun, 2014. 1–6

2 Nawkhare R, Tripathi A, Pokle P. DS-SS communication system using pseudo chaotic sequences generator. In: Proceedings of International Conference on Communication Systems and Network Technologies, Gwalior, 2013. 78–82

3 Peinado A, Munilla J, Fúster-Sabater A, et al. Improving the period and linear span of the sequences generated by DLFSRs. In: Proceedings of International Joint Conference SOCO'14-CISIS'14-ICEUTE'14, Bilbao, 2014. 397–406

4 Mandal K, Gong G. Feedback reconstruction and implementations of pseudorandom number generators from composited De Bruijn sequences. IEEE Trans Comput, 2016, 65: 2725–2738

5 Yang B, Liao X F. Period analysis of the Logistic map for the finite field. Sci China Inf Sci, 2017, 60: 022302

6 Zhou Y C, Hua Z Y, Pun C-M, et al. Cascade chaotic system with applications. IEEE Trans Cybern, 2015, 45: 2001–2012

7 Taleb F. A new chaos based image encryption scheme using chaotic logistic maps. In: Proceedings of International Conference on Multimedia Computing and Systems (ICMCS), Marrakech, 2014. 1222–1228

8 Araki S, Muraoka H, Miyazaki T, et al. A design guide of renewal of a parameter of the logistic map over integers on pseudorandom number generator. In: Proceedings of International Symposium on Information Theory and Its Applications (ISITA), Monterey, 2016. 781–785

9 Corinto F, Krulikovskyi O V, Haliuk S D. Memristor-based chaotic circuit for pseudo-random sequence generators. In: Proceedings of the 18th Mediterranean Electrotechnical Conference (MELECON), Lemesos, 2016. 1–3

10 Oliver N, Soriano M C, Sukow D W, et al. Fast random bit generation using a chaotic laser: approaching the information theoretic limit. IEEE J Quantum Electron, 2013, 49: 910–918

11 Chester D B, Michaels A J. Digital generation of a chaotic numerical sequence. US Patent, US2008/0263119

12 Michaels A J. A maximal entropy digital chaotic circuit. In: Proceedings of IEEE International Symposium of Circuits and Systems (ISCAS), Rio de Janeiro, 2011. 717–720

13 Hu J H, Ma S. High Speed Digital Signal Processing Application Based on Residue Number System. Beijing: Chinese Science Publishing & Media Ltd., 2012. 1–25

14 Khachatrian G, Kyureghyan M. A new public key encryption system based on permutation polynomials. In: Proceedings of IEEE International Conference on Cloud Engineering, Boston, 2014. 540–543

15 Sun Q, Wan D Q. Permutation Polynomials and Their Applications. Harbin: Harbin Institute of Technology Press, 2012. 22–55

16 Lidi R, Niederreiter H, Cohn F M. Finite Fields. London: Cambridge University Press, 1984. 347–389

17 Rukhin A, Soto J, Nechvatal J, et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology (NIST), Special Publication 800-22 Revision 1a, 2010

18 Xu D, Xue X X, Wang T, et al. Research on chaotic pseudo random bit generator based on logistic map. Microelectron Comput, 2016, 2: 1–6

Appendix A

Table A1 The standard permutation polynomials of degree at most five over all finite fields

The standard permutation polynomial over finite field F_q	q
x	Any value
x^2	$q \equiv 0 (\mathrm{mod}\ 2)$
x^3	$q \not\equiv 1 (\mathrm{mod}\ 3)$
$x^3 - ax$ ($a \in F_{\mathrm{q}}$ a a not a square)	$q \equiv 0 (\mathrm{mod}\ 3)$
$x^4 \pm 3x$	$q = 7$
$x^4 + a_1x^2 + a_2x$ (if $x = 0$ is the exclusive root in F_q)	$q \equiv 0 (\mathrm{mod}\ 2)$
x^5	$q \not\equiv 1 (\mathrm{mod}\ 5)$
$x^5 - ax$ ($a \in F_{\mathrm{q}}$) (a not a fourth power)	$q \equiv 0 (\mathrm{mod}\ 5)$
$x^5 - ax$ ($a^2 = 2$)	$q = 9$
$x^5 \pm 2x^2$	$q = 7$
$x^5 + ax^3 \pm x^2 + 3a^2x$ ($a \in F_{\mathrm{q}}$ a not a square)	$q = 7$
$x^5 + ax^3 + 5^{-1}a^2x$ (a arbitrary)	$q \equiv \pm 2 (\mathrm{mod}\ 5)$
$x^5 + ax^3 + 3a^2x$ ($a \in F_{\mathrm{q}}$ a not a square)	$q = 13$
$x^5 - 2ax^3 + a^2x$ ($a \in F_{\mathrm{q}}$ a not a square)	$q \equiv 0 (\mathrm{mod}\ 5)$

Table A2 A few permutation polynomials over finite field F_q and $q \not\equiv 1(\text{mod } 3)$ and $q \leqslant 503$ with single iterative loop

m	$\{q, r, s, c\}$
11	$\{1,1,4,1\},\{1,6,1,2\},\{1,4,9,3\},\{1,3,3,4\},\{1,2,5,5\},\{1,10,4,10\}$
47	$\{1,22,36,1\},\{1,31,7,2\},\{1,7,32,3\},\{1,24,5,4\},\{1,25,36,46\}$
59	$\{1,24,15,2\},\{1,40,22,3\},\{1,18,49,4\},\{1,25,51,5\},\{1,5,28,57\}$
71	$\{1,39,10,1\},\{1,67,29,2\},\{1,12,48,3\},\{1,19,2,4\},\{1,32,10,70\}$
83	$\{1,21,64,1\},\{1,51,37,2\},\{1,6,12,3\},\{1,32,37,4\},\{1,62,64,82\}$
131	$\{1,27,112,1\},\{1,62,15,2\},\{1,63,13,3\},\{1,11,84,5\},\{1,16,129,130\}$
251	$\{1,12,48,1\},\{1,103,106,2\},\{1,15,75,3\},\{1,57,79,4\},\{1,239,48,250\}$
311	$\{1,84,175,1\},\{1,23,280,2\},\{1,36,121,3\},\{1,40,15,4\},\{1,227,175,310\}$
347	$\{1,244,182,1\},\{1,33,16,3\},\{1,115,13,6\},\{1,38,250,8\},\{1,103,182,346\}$
443	$\{1,163,144,1\},\{1,240,151,2\},\{1,25,356,3\},\{1,57,197,5\},\{1,280,144,442\}$
467	$\{1,396,435,1\},\{1,288,95,2\},\{1,82,62,3\},\{1,31,9,5\},\{1,71,435,466\}$
479	$\{1,69,150,1\},\{1,324,25,2\},\{1,67,219,3\},\{1,53,138,5\},\{1,410,150,478\}$
491	$\{1,21,147,1\},\{1,127,139,2\},\{1,33,363,3\},\{1,10,197,5\},\{1,470,147,490\}$
503	$\{1,109,104,1\},\{1,267,122,2\},\{1,271,1,3\},\{1,8,189,5\},\{1,394,104,502\}$