Applying M-sequences Decimation to Generate Interleaved Sequence

Dang Van Truong, Le Chi Quynh

***Abstract*—M-sequences are widely used in for many purposes, from synchronization, whitening, communications and cryptography. We analyze decimation techniques and introduce two methods to generate decimation sequences which don't have to calculate intermediate states. Then we apply these methods to interleaved sequence as a new method to pre-calculate for set of interleaved order which is more effective in implementation**

***Tóm tắt*—M-dãy đang được sử dụng rất rộng rãi trong nhiều lĩnh vực, từ việc đồng bộ, làm trắng thông tin, viễn thông và kỹ thuật mật mã. Chúng tôi phân tích kỹ thuật phân rã m-dãy theo bước và giới thiệu hai phương pháp sinh dãy phân rã theo bước mà không cần tính các trạng thái trung gian. Áp dụng phương pháp này vào dãy lồng ghép, ta có một phương pháp mới để tính trước tập các thứ tự lồng ghép có tính hiệu quả trong cài đặt thực tế.**

Keywords—decimation, fast decimation, Fibonacci method, interleaved sequence, m-sequence, pseudo-random number generation.

Từ khóa—dãy lồng ghép, m-dãy, phân rã theo bước, phân rã nhanh, phương pháp Fibonacci, sinh dãy giả ngẫu nhiên.

I. INTRODUCTION

M-sequence are widely used in everyday technology. Many methods have been developed to construct pseudo-random sequences with better properties by transforming from m-sequences. One of the earliest of which is the decimation method. We can apply decimation method to generate pseudo-random sequence directly, or we can use this method to improve generating process of interleaved sequence.

This manuscript is received on August 06, 2021. It is commented on August 10, 2021 and accepted on August 30, 2021 by the first reviewer. It is commented on August 10, 2021 and accepted on September 8, 2021 by the second reviewer.

When generate decimation sequences directly, we need to calculate many unnecessary internal states. In this contribution, we introduce two method to compose decimation sequences without calculation of useless states using pre-calculated matrix.

With interleaved sequence [4, 5, 10], we can apply decimation calculus as a new method to determine interleaved order set. We treat the sub-sequences of interleaved sequence as decimation sequences from origin m-sequence. Then we use the new decimation methods to get the first part of interleaved order set.

II. DECIMATION TECHNIQUE

A. Theoretical background

Let A be an m-sequence with periodicity 2^n-1 and root α. We construct new sequence $A(T)$ by sampling every T^{th} bit of A, starting with the first bit of A. $A(T)$ is decimation sequence called decimation of order T from A, with T is the decimation step.

It is known that if the decimation step T and the periodicity $2^n - 1$ are co-prime, $(T, 2^n-1) = 1$, $A(T)$ is also an m-sequence with the root α^T, so-called m-sequence with difference generator polynomial but the same periodicity.

For T equals 2^m, $A(T)$ is a shifted version of A. In case T and $2^n - 1$ are not co-prime, the generated decimation sequence is a LFSR sequence with periodicity [6]

$$\frac{2^n-1}{\gcd(2^n-1,T)} \quad (1)$$

The generator polynomial of this sequence is irreducible, but it is not primitive polynomial in almost case.

If decimation steps and cycle length satisfied interleaved sequence condition [5], then the new generator polynomial is primitive, but new polynomial degree is a divisor of primary degree.

In general application of m-sequences decimation method, step and cycle length must be co-prime in order to have decimation sequence with maximum cycle length and good random properties.

When apply decimation technique on m-sequence, we can change polynomial without changing in algorithm implementation. Decimation progress can be implemented in software only with appreciate calculation, no need to change in hardware design, make it is easier in implementation on small devices.

Demonstration of an m-sequences decimations

Assume that we have an m-sequence with order n = 23 and the generator polynomial:

$$f = x^{23} + x^{18} + x^{15} + x^{14} + x^{11} + x^9 + x^5 + x^2 + 1$$

with initial state *S(0) = (*1*, 0, 0, ..., 0),* or *(000001)* in hexadecimal.

Apply m-sequence generation formula, we can have next 18 internal states of m-sequence below (display in hexadecimal):

{*400000, 200000, 500000, 280000, 548000, 6A0000, 750000, 3A8000, 5D4000, 2EA000, 175000, 4BA800, 25D400, 12EA00, 097500, 44BA80, 625D40, 312EA0*}

We show decimation of this m-sequence with decimation step *T=3* and *T=5* in the figure below.

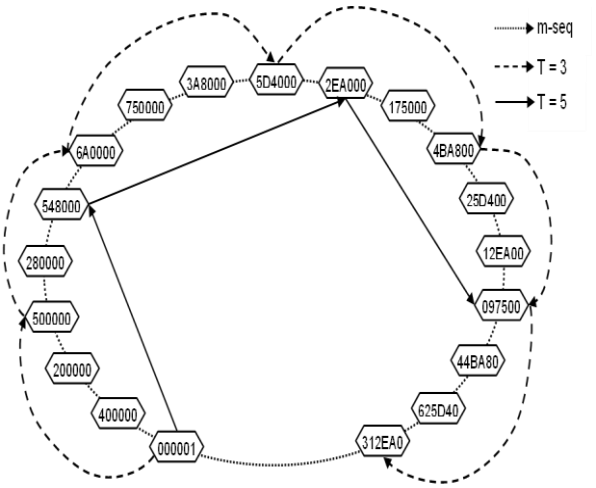

Fig. 1. Decimation of m-sequence with decimation step *T=3* and *T=5*.

We create small software to simulate decimation operation. In this software, we use Belekamp-Massey algorithm to determine the generator polynomial of newly generated decimation sequence. Running the software with different polynomials and decimation steps, we have the following result in Table I.

TABLE I. RESULT OF ANALYSE DECIMATION SEQUENCES BY APPLICATION

No	Deg	Polynomial	Step	New Polynomial
1	24	$x^{24} + x^{20} + x^{19} + x^{17} + x^{15} + x^{13} + x^{12} + x^{10} + x^8 + x^4 + 1$	11	$x^{24} + x^{22} + x^{20} + x^{17} + x^{14} + x^{11} + x^9 + x^8 + x^5 + x^3 + 1$
2	23	$x^{23} + x^{18} + x^{15} + x^{14} + x^{11} + x^9 + x^5 + x^2 + 1$	5	$x^{23} + x^{22} + x^{18} + x^{17} + x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^6 + x^5 + x^3 + x^2 + x + 1$

3	23	$x^{23} + x^{18} + x^{15} + x^{14} + x^{11} + x^9 + x^5 + x^2 + 1$	3	$x^{23} + x^{20} + x^{19} + x^{18} + x^{17} + x^{15} + x^{14} + x^{12} + x^9 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$
4	17	$x^{17} + x^{10} + x^6 + x + 1$	19	$x^{17} + x^{11} + x^{10} + x^9 + x^6 + x + 1$
5	14	$x^{14} + x^{12} + x^9 + x^7 + x^6 + x + 1$	8	$x^{14} + x^{12} + x^9 + x^7 + x^6 + x + 1$
6*	21	$x^{21} + x^{18} + x^{16} + x^{13} + x^{10} + x^7 + x^4 + x^3 + x^2 + x + 1$	7	$x^{21} + x^{17} + x^{16} + x^{15} + x^{14} + x^{10} + x^5 + x^3 + 1$
7**	21	$x^{21} + x^{18} + x^{16} + x^{13} + x^{10} + x^7 + x^4 + x^3 + x^2 + x + 1$	16513	$x^7 + x^6 + x^5 + x^3 + x^2 + x + 1$

*Decimation step and cycle length are not co-prime, so the new generator polynomial is irreducible, but not primitive

** Decimation step and cycle length are not co-prime and satisfied interleaved sequence condition, so the new polynomial is primitive, but new polynomial degree is divisor of primary degree

In the case of 5^{th} sequence, when the new polynomial coincides with the old polynomial, or decimation sequence is a shifted version of primary sequence.

III. SPEEDING UP CREATION PROCESS OF DECIMATION SEQUENCES

We concern on generation performance: we need to calculate for T internal states to generate only one output bit of decimation. There are T-1 inter-mediate internal states those are discarded but has to be calculated.

If we used d-Transform to calculate new state (or Gaussian method) [2], we can calculate internal state in each step:

$$S_1(d) = \frac{S_0(d) \cdot d}{g(d)} \qquad (2)$$

We can directly calculate internal state after T step in which current internal state multiply by d^T

$$S_T(d) = \frac{S_0(d) \cdot d^T}{g(d)} \qquad (3)$$

Using this method, we don't have to save T-1 intermediate internal state, but when calculating polynomial remainder by $g(d)$ we still have to calculate remainder for T co-efficient. This computation take some works, especially when T grow large.

When generate sequence by Fibonacci method, which is common way to generate sequence in micro controller

We store internal state of m-sequence in a shift register denote by a_i (i=0..n-1)

To operate an m-sequence, each step require calculation of "feedback" bit a_n:

$$a_n = \sum_{i=0}^{n-1} a_i \cdot f_{n-i} \qquad (4)$$

Then we shift all bits of the shift register to the right position, and put newly calculated bit a_n in bit n-1. We can take the discarded bit a_0 as output bit of this step.

Apply decimation method with Fibonacci method, we have to calculate (4) for T times to generate one output bit. We can perform decimation method faster by pre-calculate in such way:

For example, we re-use m-sequence degree n = 23 with generator polynomial

$$f = x^{23} + x^{18} + x^{15} + x^{14} + x^{11} + x^9 + x^5 + x^2 + 1$$

The decimation step is $T = 5$

Analyze by application we found the new generated polynomial:

$$g = x^{23} + x^{22} + x^{18} + x^{17} + x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^6 + x^5 + x^3 + x^2 + x + 1$$

From the initial state $S_{(0)} = \{a_0, a_1, \ldots, a_{22}\}$, we can build formula for next 5 bits:

$$\boldsymbol{a_{23}} = a_0 \wedge a_5 \wedge a_8 \wedge a_9 \wedge a_{12} \wedge a_{14} \wedge a_{18} \wedge a_{21}$$

$$\boldsymbol{a_{24}} = a_1 \wedge a_6 \wedge a_9 \wedge a_{10} \wedge a_{13} \wedge a_{15} \wedge a_{19} \wedge a_{22}$$

$$\boldsymbol{a_{25}} = a_2 \wedge a_7 \wedge a_{10} \wedge a_{11} \wedge a_{14} \wedge a_{16} \wedge a_{20} \wedge a_{23}$$

$$= a_2 \wedge a_7 \wedge a_{10} \wedge a_{11} \wedge a_{14} \wedge a_{16} \wedge a_{20} \wedge a_0 \wedge a_5 \wedge a_8 \wedge a_9 \wedge a_{12} \wedge a_{14} \wedge a_{18} \wedge a_{21}$$

$$= a_0 \wedge a_2 \wedge a_5 \wedge a_7 \wedge a_8 \wedge a_9 \wedge a_{10} \wedge a_{11} \wedge a_{12} \wedge a_{16} \wedge a_{18} \wedge a_{20} \wedge a_{21}$$

$$\boldsymbol{a_{26}} = a_3 \wedge a_8 \wedge a_{11} \wedge a_{12} \wedge a_{15} \wedge a_{17} \wedge a_{21} \wedge a_{24}$$

$$= a_1 \wedge a_3 \wedge a_6 \wedge a_8 \wedge a_9 \wedge a_{10} \wedge a_{11} \wedge a_{12} \wedge a_{13} \wedge a_{17} \wedge a_{19} \wedge a_{21} \wedge a_{22}$$

$$\boldsymbol{a_{27}} = a_4 \wedge a_9 \wedge a_{12} \wedge a_{13} \wedge a_{16} \wedge a_{18} \wedge a_{22} \wedge a_{25}$$

$$= a_0 \wedge a_2 \wedge a_4 \wedge a_5 \wedge a_7 \wedge a_8 \wedge a_{10} \wedge a_{11} \wedge a_{13} \wedge a_{20} \wedge a_{21} \wedge a_{22}$$

So the internal state of register after 5 step is:

$$S_{(5)} = \{a_5, a_6, \dots, a_{22}, \boldsymbol{a_{23}, a_{24}, a_{25}, a_{26}, a_{27}}\} \quad (5)$$

Display in matrix form:

$$A = \begin{bmatrix} 0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ \vdots \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,1\,0\,1\,0\,1\,1\,0 \\ 0\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,1\,0\,1\,0\,1\,1 \\ 1\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,1\,1 \end{bmatrix}$$

Then

$$S_{(5)} = A \bullet S_{(0)}{}^T \quad (6)$$

We need all values of state $S_{(5)}$ because in order to calculate a_{32} (next bit of decimation follow $S_{(5)}$ at state $S_{(10)}$), we need all values of state $S_{(5)}$. We may calculate the formula for a_{32} directly from $S_{(0)}$, but we cannot pre-calculate all the formula for every bit of decimation.

When $T<m$, we can calculate all bit of decimation recursively with need to store only *5* equation in (5), not the whole matrix *A*. In this case, using (5) is faster than doing the matrix multiply in (6).

In practice, calculation works are the same between traditional method (directly compute step by step using (4)) and our new method (using (5) and (6)). But in our method, we only shift the internal register by *T* position onetime only. In traditional method, we have to shift the internal register *T* times, each time it shifts one position. When using micro controller, this operation takes time to move data in memory. In hardware implementation, it takes *T* clocks to complete this operation.

IV. Applying decimation method to interleaved sequence

Interleaved sequence [5] is construct from m-sequence with degree *n*, which $n = m.l$

Choose the interleaved step *T* satisfied:

$$T = \frac{N}{L} \text{ where } L = p^m\text{-}1,\ N = p^n\text{-}1$$

Interleaved sequence is linked from sub-sequences, the k^{th} sub-sequence is a decimation of order *T* from the origin m-sequence shifted by *k* step. As show in [4], each sub-sequence is shifted version of full cycle m-sequence with degree *m*.

To generate interleaved sequence we need to determine the phase shift of each sub-sequence, called interleaved order set. There are 3 methods to pre-calculate the interleaved order set [5], using d-Transform, trace function and direct computation.

We can apply decimation generation methods above to get the first sub-sequence, which is the decimation of order *T* from the origin m-sequence. But we know than this sub-sequence is a full cycle m-sequence with degree m, so we only need to compute the first m bit of this sub-sequence. Using (4) with the generator polynomial of sub-sequence and the degree change to m, we have the remaining bits.

When we compute the first m bit of the sub-sequence, we also have m internal states of origin m-sequence at position *0, T, 2T .. (m-1)T*. From these states, using (4) we can determine the first m-bit of the second sub-sequence, and so on…

By using this method, we can calculate all the initial states of all sub-sequences. From these initial states, we can build interleaved order set of interleaved sequence. In memory usage, we need memory space to store *m* states of origin m-sequence with memory size *m.n*.

In practice, we often generate the first part of interleaved sequence for a given length. In this case, we don't have to determine all the interleaved order set. We just calculate which initial states as needed to generate such given length output.

CONCLUSION

From definition of decimation sequence, we introduce two method to speed up generating decimation sequence, even with a large step: using d-Transform and using pre-calculated transition matrix. The matrix method has the advantage of register shifting, so it is more effective in practice.

Applying these methods to interleaved sequence, we can directly determine the beginning part interleaved order set, with a little memory store needed. Using the introduced methods, we can effectively generate interleaved sequence for a given length, especially when the origin degree is large.

There is still another problem to be concerned, when the value of *T* is very large, these methods still need to be improved.

REFERENCES

[1] G. Gong, *New design for signal sets with low cross correlation, balance property and large linear span - GF(p) case*, IEEE Trans. Inform. Theory, vol 48, no. 11, pp.2847-2867, Nov. 2002.

[2] G. Gong, *Theory and application of q-ary interleaving sequences*, IEEE Trans. Inform. Theory, vol41, pp. 400-41l, March 1995.

[3] Jing He, *Interleaved Sequences Over Finite Fields*, Carleton University, Ottawa 2013.

[4] L.M. Hieu and L.C. Quynh, *Design and Analysis of Sequences with Interleaved Structure by d-Transform*, IETE Journal of Research, vol. 51, no. l, pp.61-67, Jan-Feb. 2005.

[5] Hieu Le Minh, Truong Dang Van, Binh Nguyen Thanh and Quynh Le Chi, *Design and Analysis of Ternary m-sequences with Interleaved Structure by d-Transform*, Journal of Information Engineering and Applications, vol.5, no.8, pp.93-101, 2015.

[6] R.J. McEliece, Finite fields for computer scientists and engineers, Springer, 1987.

[7] J.S. No, P-ary unified sequences: *P-ary extended d form sequences with the ideal autocorrelation property*, IEEE Trans. Inform. Theory, vol 48, no. 9, pp. 2540-2546, Sept 2002.

[8] L.C.Quynh, S.Prasad, *A class of binary cipher sequences with best possible correlation funtion*, IEEE Proceeding Part F .Dec 1985. vol 132.pp.560-570.

[9] Quynh Le Chi, Cuong Nguyen Le, Thang Pham Xuan, *A hardware oriented method to generate and evaluate nonlinear interleaved sequences with desired properties*, Journal of Information Engineering and Applications, Vol.6, No.7, 2016.

[10] Truong Dang Van, Binh Nguyen Thanh, Hieu Le Minh and Quynh Le Chi, *Construction of Nonlinear q-ary m-sequences with Interleaved Structure by d-Transform*, IEEE ICCE 2018, pp.389-392, 2018.

[11] Nguyen Van Son, et. al, *FPGA Implementation of Optimal PN-Sequences by Time-Multiplexing Technique*, International Conference on Engineering Research and Applications (ICERA), 2019.

[12] O.W. Yeung and K.M. Chugg, *An iterative algorithm and low complexity hardware architecture for fast acquisition of long PN codes in UWB systems*, The Journal of VLSI Signal Processing, vol. 43, no. 1, pp. 25–42, 2006.

ABOUT THE AUTHORS

Dang Van Truong

Workplace: Institute of Science and Technology of Cryptography

Email: truongdv@gmail.com

Education: He received a bachelor's degree in cryptography technology in 1996, received a master's degree in computer science in 2003 from the Le Quy Don University, doctoral student at Posts and Telecommunications Institute of Technology from 2014.

Recent research direction: Computer science, Information security.

Le Chi Quynh

Workplace: Electric Power University

Email: quynh.lechi@gmail.com

Education: He received a doctor's degree in Electronics in 1987.

Recent research direction: Number theory, Electronic and Communication, Information security.