A Random Binary Sequence Generator Based on β Encoders

Jitsumatsu, Yutaka
九州大学大学院システム情報科学研究院

Matsumura, Kazuya
Kyushu University

出版情報：MI lecture note series. 62, pp.34-40, 2015-03-10. 九州大学マス・フォア・インダストリ研究所
バージョン：
権利関係：

A Random Binary Sequence Generator Based on β Encoders

Yutaka Jitsumatsu and Kazuya Matsumura
Depart. of Informatics, Kyushu University,
Motooka 744, Nishi-ku, Fukuoka, 819-0395, Japan.
Email: {jitumatu, matsumura}@me.inf.kyushu-u.ac.jp

***Abstract*—A β encoder is an Analog-to-Digital (A/D) converter whose dynamics is governed by (β, α) map. The most important feature of the β encoder is that it is robust to fluctuation of the threshold value for quantization as well as the fluctuation of the value of β. Because of this property, the β encoder can be implemented with low-precision elements and thus realized by an extremely small circuit. Hirata et.al proposed a random number generator using a β encoder followed by a kind of shift register circuit. They showed that random numbers generated by such a circuit can pass the National Institute of Standards and Technology (NIST) Statistical Test Suite. We recently proposed another method for converting the output sequences from the β encoder to binary sequences that can be regarded as independent and identically distributed (i.i.d.) random variables with equal probability. In the proposed method, we try to find a binary expansion of an input value x that is recovered from a β encoder's output sequence. We verified that binary sequences generated from a β encoder followed by the proposed method can pass the NIST Statistical Test Suite. It is shown that the proposed method is robust to the fluctuation of the value of β.**

I. Introduction

The importance of random number generation becomes significant because of the development of information and communication technologies and demand for secure communications. Pseudo-random numbers are generated by deterministic algorithms with seeds and thus completely the same numbers are produced if the same seed is used. On the other hand, there are demands, especially in a security purpose, for physical random number generator that measures some physical phenomenon. For example, a secure key distribution using bit sequences generated by the use of a semiconductor laser [4] and a random number generator based on a chaotic map [5] have been proposed. Many randomness tests have been proposed.

A random number generation method that uses a β encoder as a source of randomness has been proposed [14], [15]. The β encoder is an Analog-to-Digital (A/D) converter that is robust to fluctuation of threshold value of a quantizer [1]. Such a β encoder does not need high-precision circuit elements and is implemented by a complementary metal-oxide-semiconductor (CMOS) circuit that achieves very small area consumption as well as low power consumption [9]. It can be used at from -20 degree to 80 degree Celsius. We can observe chaos attractors in β converters [3]. However, outputs from a β encoder have strong correlations between successive bits. In [14] a random number generation by calculating the exclusive disjunction (exclusive or: EXOR) of several delayed bits of β encoder's outputs was proposed. When we use β encoders for generating random numbers, we generate one million bits, while only the first L bits of β expansion coefficients $b_i \in \{0, 1\}$ are used for expressing approximated input value as $\hat{x} = \sum_{i=1}^{L} b_i \beta^{-i}$, where L is typically less than 20.

A remarkable feature of random number generation using β encoder is that randomness is guaranteed by chaotic behavior of the attractors observed in the β encoder [3]. Hence, basically, thermal noise is not needed for β encoders to generate random numbers and it can work in an extremely low temperature environment. This is a significant difference between the proposed method and those random number generation methods whose randomness is guaranteed by thermal noise. On the other hand, β encoder is robust to fluctuation of threshold. This property makes the β encoder work also at high temperature environment.

After the computer simulation in [14], we performed experiments of random number generation using hardware β encoders implemented by CMOS technologies [15]. We found that more than 10 EXOR operations are needed to make the generated sequences pass the National Institute of Standards and Technology (NIST) statical test suite [11].

In this paper, we propose an algorithm in which the output of β encoder is converted to binary sequences that can be regarded as independent and identically distributed (i.i.d.) sequences with equal probability. The algorithm is based on an idea that using the first n outputs $b_1, b_2, \ldots, b_n$ of a β encoder, we calculate the interval $[l_n, u_n]$ in which the input value x must exist and then obtain the binary expansion $\hat{x} = \sum_{j=1}^{m} \tilde{b}_j 2^{-j}$ where m and $\tilde{b}_j \in \{0, 1\}$ are selected to satisfy $\hat{x} \leq l_n$ and $u_n \leq \hat{x} + 2^{-m}$. Though l_n and u_n are real-valued, we approximate them by fixed-point numbers so that the proposed method can be realized by CMOS circuit. We will show that a fixed-point arithmetic with 15 bit precision is sufficient to make the generated binary sequences pass the NIST test suite.

Output sequences from the β encoder are passed through the proposed conversion algorithm. Then, the NIST test suite is applied to these sequences. The value of β used in the β encoder is not precisely known beforehand. We performed numerical experiments of β-ary to binary conversion with estimated $\beta = 1.6, 1.7, 1.8$ and 1.9. It is confirmed that the proposed algorithm is robust to the estimation error for β.

II. Pulse Code Modulation and β encoder

In this section, we briefly review Pulse Code Modulation (PCM) and β encoder.

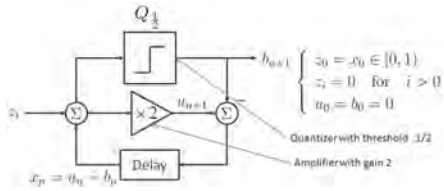

Fig. 1. PCM-based A/D converter

A. Pulse Code Modulation (PCM)

The binary expansion of a real value $z_{\rm i} \in [0,1)$ is given by

$$z_{\rm i} = \sum_{n=1}^{\infty} \tilde{b}_n 2^{-n}, \tag{1}$$

where $\tilde{b}_n \in \{0,1\}$ is given by

$$b_n = Q_{\frac{1}{2}}(x_{n-1}), \quad n = 1,2,\ldots. \tag{2}$$

where $Q_{\frac{1}{2}}(x)$ is a quantizer with a threshold $\frac{1}{2}$, defined by

$$Q_{\frac{1}{2}}(x) = \begin{cases} 0, & (0 \le x < \frac{1}{2}) \\ 1, & (\frac{1}{2} \le x < 1) \end{cases}, \tag{3}$$

and $x_n = B(x_{n-1})$ $(n = 1,2,\ldots)$, where $B(x)$ is Bernoulli shift map defined by

$$B(x) = \begin{cases} 2x, & (0 \le x < \frac{1}{2}) \\ 2x-1, & (\frac{1}{2} \le x < 1). \end{cases} \tag{4}$$

Here, the initial value $x_0 = z_{\rm i}$ is an analog input voltage (See Fig. 1).

A drawback of PCM is that the quantizer $Q(x)$ may make a wrong decision if x_n is very close to the threshold value because a slight fluctuation occurs in the threshold voltage. For example, assume the threshold is changed from $\frac{1}{2}$ to some value ν so that $B(x)$ is replaced by

$$B'(x) = \begin{cases} 2x, & 0 \le x < \nu \\ 2x-1, & \nu \le x < 1 \end{cases} \tag{5}$$

(See Fig. 2). Then, for $0 < \varepsilon < \nu - \frac{1}{2}$, x_n diverges as $B'(\frac{1}{2}+\varepsilon) = 1+2\varepsilon$, $B'(1+2\varepsilon) = 1+4\varepsilon$, $B'(1+4\varepsilon) = 1+8\varepsilon$ $\cdots$. For avoiding such an un-stability of conversion, 1.5 bit encoders [6] and digital calibration techniques [7] have been proposed.

On the other hand, $\Sigma\Delta$ modulators have a good property that they are robust to fluctuations of threshold values in their quantizers. However, they have a drawback that oversampling rate is very high, such as one hundred or one thousand. This implies that $\Sigma\Delta$ modulation can only be used in narrow-bandwidth applications. Moreover, the quantization error of $\Sigma\Delta$ modulation decreases inverse proportionally to the number of bits in contrast to the exponential accuracy of the PCM. β encoders have the two good properties, i.e., robustness against fluctuations of threshold voltages and an exponential accuracy [1]. In the next subsection, a scale-adjusted β encoder [15] is explained.

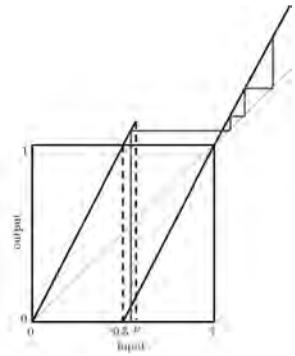

Fig. 2. A map of PCM-based A/D converter with fluctuation of threshold value : $B'(x)$

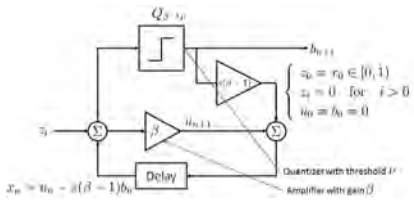

Fig. 3. Scale-adjusted β-encoder

B. β encoder

Define a scale-adjusted β map for $0 \le x \le s$ with a scale parameter s as (See Fig.3),

$$S_{\beta,\nu,s}(x) = \begin{cases} \beta x, & (0 < x < \nu\beta^{-1}) \\ \beta(x-s)+s, & (\nu\beta^{-1} < x < s) \end{cases}. \tag{6}$$

The ordinary β encoder corresponds to the case $s = \frac{1}{\beta-1}$. Note that the domain of the map is $[0,s]$ irrespective of β, while that of the ordinary β map depends on β. The output of the scale-adjusted β encoder for the input value $z_{\rm i} = x_0$ is

$$b_n = Q_{\nu\beta^{-1}}(x_{n-1}), \tag{7}$$

where

$$x_n = S_{\beta,\nu,s}(x_{n-1}), \quad n = 1,2,\ldots, \tag{8}$$

and

$$Q_\nu(x) = \begin{cases} 0, & (x < \nu) \\ 1, & (x \ge \nu) \end{cases} \tag{9}$$

is a quantization function with threshold ν. Let

$$l_n = s(\beta-1)\sum_{i=1}^{n} b_i \beta^i \tag{10}$$

$$u_n = l_n + s\beta^n. \tag{11}$$

Then, given the first n outputs, $b_1, b_2, \ldots, b_n$, we know that x_0 must exist in $[l_n, u_n]$.

We hereafter assume $s = 1$ for simplicity. For almost all initial value x_0, x_n generated by Eq. (8) does not fall into

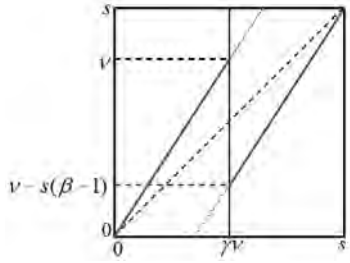

Fig. 4. A map of β-expansion

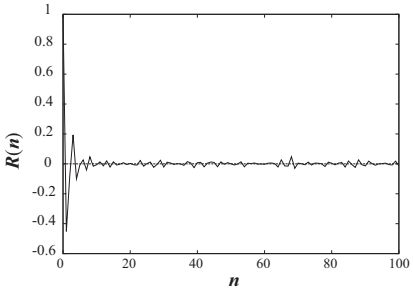

Fig. 5. Autocorrelation of β-encoder's output ($L = 10000,\ 0 \le n < 100$)

some periodic points but stays in some range. Attractors are observed in dynamics of β encoders. They are referred to as β expansion attractors [3]. We consider such attractors are used as sources of randomness for generating sequences of random numbers.

III. Random Number Generation using β encoders

Bernoulli shift map is an ideal model for fair coin tossing, i.e., a model for generating independent and identically distributed (i.i.d.) random variables. Thus, a binary expansion of an arbitrarily chosen input voltage x_0 is considered as a sequence of ideal binary i.i.d. random variables. However, the dynamics of PCM modulation is unstable because of the issue mentioned in Section II-A. In this paper, we show a method that approximately converts a sequence of β expansion coefficients for an input voltage x_0 to a sequence of binary expansion coefficients for the same initial value [16]. Such a conversion is referred to as β-ary to binary (β-ary/binary) conversion. Since β converters can be implemented in very small CMOS circuits, random number generators using a β encoder should also be implemented in CMOS circuits. Therefore, we consider a digital calculation with a finite precision to perform the proposed β-ary/binary conversion.

A. $\beta_{\rm D}$ sequences

It is easily verified that the consecutive outputs from a β encoder have a negative correlation, i.e., $\frac{1}{L}\sum_{i=1}^{L} b_i b_{i+1}$ tends to take a negative value. Fig. 5 shows an autocorrelation function of an output sequences from a β encoder with $\beta = 1.8$ and $\nu = \frac{\beta}{2(\beta-1)} = 1.125$, where a $\{0,1\}$-valued sequence is converted to a $\{-1,+1\}$-valued sequence, i.e., the graph shows $R(n) = \frac{1}{L}\sum_{i=1}^{L}(2b_i - 1)(2b_{i+n} - 1)$. Fig. 5 shows that the output sequence from the β encoder has a negative autocorrelation of delay $n = 1$, which implies that some additional process is needed to generate sequences whose distribution is close to that of i.i.d. random variables.

Hirata et.al have proposed [14] the β_D sequences that are generated by taking EXOR of several outputs. Specifically, the β_D sequence is defined as

$$b_D(k) = \bigoplus_{i=1}^{N_{xor}} b(k - d_i), \quad k \ge d_{N_{xor}} \tag{12}$$

where $d_1 < d_2 < \ldots, d_{N_{xor}}$, N_{xor} is the number of bits of which EXOR is taken.

B. β-ary/binary conversion

We recently proposed a method for converting a binary sequence generated from β encoder to another binary sequence that is approximately regarded as i.i.d. random variables [16].

Let $\{b_i\}_{i=0}^{n}$ be a β expansion, determind by Eq.(7), of some initial value $x_0 = z_{\rm i}$. In the proposed method, we calculate the interval $[l_n, u_n]$ in which the input value $z_{\rm i}$ exists. Then we obtain the binary expansion

$$\hat{z}_{\rm i} = \sum_{j=1}^{m} \tilde{b}_j 2^{-j} \tag{13}$$

where m and $\tilde{b}_j \in \{0,1\}$ are selected to satisfy $\hat{x} \le l_n$ and $u_n \le \hat{x} + 2^{-m}$.

If the perfect knowledge of β is available, then we find an interval $[l, u]$ that includes $z_{\rm i}$ in Method 1, explained later (See Fig.6). However, the proposed algorithm should be implemented in digital circuit. Thus, u and l should be expressed by integers.

In order to make the explanation easy, we suppose u and l are real-valued for a while. An integer implementation is explained later. The goal of the proposed method is to make a generated binary sequence that can pass the NIST statistical test suite. A binary sequence generated by the proposed method, denoted by c_j, is different from the true binary expansion $\tilde{b}_i$ because of two reasons. One is that u and l are expressed by integer numbers. The other is that there is a difference between the true β and the β used for the β-ary/binary conversion.

[Method 1]

1) Initialize: $i = j = 1, [l, u] = [0, 1], \gamma = \frac{1}{\beta}$.
2) Read b_i.
   If $b_i = 0$, then u is updated to $l + (u - l) \times \gamma$.
   If $b_i = 1$, then l is updated to $u - (u - l) \times \gamma$.
3) If $u < \frac{1}{2}$, then output $c_j = 0$ and update $j = j + 1$, $l = 2l$, and $u = 2u$.
   If $l \ge \frac{1}{2}$ then output $c_j = 1$ and update $j = j + 1$, $l = 2l - 1$, and $u = 2u - 1$.

TABLE I. VOLTAGE PARAMETERS

	$V_{\rm DDA}$	$V_{\rm DDD}$	$V_{\rm DD\_IO}$	$V_{\rm ref+}$	$V_{\rm ref-}$	$V_{\rm ref\_CM}$	$A_{\rm in+}$	$A_{\rm in-}$
A	1.20	1.20	1.20	0.85	0.35	0.60	0.80	0.40
B	1.20	1.20	1.20	0.85	0.35	0.60	0.60	0.60
C	1.40	1.20	1.20	0.95	0.45	0.70	0.70	0.70

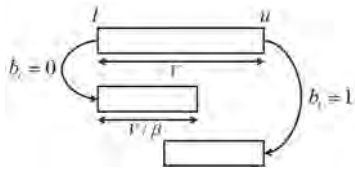

Fig. 6. How to update an interval $[l, u]$.

4) If b_i is EOF(End Of File), then quit. Otherwise, update $i = i + 1$ and go back to Step 2.

After we obtain the first n outputs from β encoder, the width of interval $[l_n, u_n]$ is exactly β^{-n}. This fact implies that the number of outputs from the converter is approximately $m = n \log_2 \beta < n$. Hence, the proposed converter does not always generate one output bit per one input bit.

In Method 1, the interval $[l, u]$ may become very small after some updates. This situation happens if the input value x subtracted by $\sum_{i'=1}^{i} b_{i'} 2^{-i'}$ exists in $[l, u]$. Hence this situation implies that the next output sequence is $0 \cdots 01$ or $10 \cdots 0$. When we express l and u by some integers, this situation makes approximation error very large. In Method 2 below, we void such a situation by doubling the size of $|u - l|$ without making decision $c_j \in \{0, 1\}$. This makes $|u-l|$ always greater than $\frac{1}{4}$. Method 2 is based on arithmetic codes [10], but the way to expand the interval is slightly different since subintervals for expressing 0 and 1 are overlapping each other. The algorithm of Method 2 is also similar to a 1.5 bit quantizer [6]. A new parameter k expresses the number of undecided output bits.

[Method 2]

1) Initialize: $i = j = 1$, $k = 0$, $[l, u] = [0, 1]$, and $\gamma = \frac{1}{\beta}$.
2) The same as Step 2. in Method 1.
3)
   - If $u < \frac{3}{4}$ and $l \geq \frac{1}{4}$, then update $l = 2l - \frac{1}{2}, u = 2u - \frac{1}{2}$, and $k = k + 1$.
   - If $u < \frac{1}{2}$, then output $c_j = c_{j+1} = \cdots = c_{j+k-1} = 1, c_{j+k} = 0$. (If $k = 0$, then output $b_j = 0$) and update $k = 0$, $l = 2l$, $u = 2u$, and $j = j + k + 1$.
   - If $l \geq \frac{1}{2}$, then output $b_j = b_{j+1} = \cdots = b_{j+k-1} = 0$, and $b_{j+k} = 1$ (If $k = 0$, then output $b_j = 1$) and update $k = 0, l = 2l - 1, u = 2u - 1$, and $j = j + k + 1$.
4) If b_i is EOF, then quit. Otherwise, update $i = i + 1$ and go back to Step 2.

Finally, we give Method 3 which is an integer calculation version of Method 2. A real number $r \in [\frac{i}{2^w}, \frac{i+1}{2^w})$ is approximated by $\frac{i}{2^w}$, where w is referred to as a window size. Real numbers l and u in $[0, 1]$ in Method 2 are replaced by integers l' and u' in $\{0, 1, \ldots, 2^w - 1\}$ in Method 3.

[Method 3]

1) Initialize: $i = j = 1$, $k = 0$, $[l', u'] = [0, 2^w - 1]$, and $\gamma = \lfloor \frac{2^w}{\beta} \rfloor$.
2) Read b_i.
   If $b_i = 0$, then update
   $$u' = l' + \lfloor \frac{(u' - l') \cdot \gamma}{2^w} + \frac{1}{2} \rfloor \quad (14)$$
   If $b_i = 1$, then update
   $$l' = u' - \lfloor \frac{(u' - l') \cdot \gamma}{2^w} + \frac{1}{2} \rfloor \quad (15)$$
3)
   - If $u' < \frac{3}{4} \times 2^w$ and $l' \geq \frac{1}{4} \times 2^w$, then update $l' = 2l' - 2^{w-1}, u' = 2u' - 2^{w-1}$, and $k = k + 1$.
   - If $u < \frac{2^w}{2}$, then output $b_j = b_{j+1} = \cdots = b_{j+k-1} = 1$, and $b_{j+k} = 0$ and update $k = 0, l' = 2l', u' = 2u'$, and $j = j + k + 1$.
   - If $l \geq \frac{2^w}{2}$, then output $b_j = b_{j+1} = \cdots = b_{j+k-1} = 0$, and $b_{j+k} = 1$ and update $k = 0$, $l' = 2l' - 2^w$, $u' = 2u' - 2^w$, and $j = j + k + 1$.
4) If b_i is EOF, then quit. Otherwise, update $i = i + 1$ and go back to Step 2.

Here, $\lfloor x \rfloor$ is the largest integer not greater than x. Since the calculation of Eqs.(14) and (15) are approximation of those of Method 2, the interval $[u, l]$ is not exact. Note that this algorithm has an internal state that is specified by (u', l', k), where $l' \in \{0, 1, \ldots, 2^{w-1} - 1\}$, $u' \in \{2^{w-1}, 2^{w-1} + 1, \ldots, 2^w - 1\}$, and $k \in \{0, 1, \ldots\}$.

IV. RESULTS OF EXPERIMENT

Experiments were carried out to show the validity of the proposed method. San et.al have manufactured CMOS circuits in which β encoders are embedded [8], [9]. We use the same β encoder implemented in CMOS circuit as the one used in [15]. The parameter β of the β encoder is designed to 1.83 but its effective value is slightly fluctuated and not known precisely beforehand. We generated 125 sequences; the length of each sequence is 1.05×10^6. Method 3 was applied to the output sequences from such a β encoder. The window size is $w = 20$ and $\beta = 1.8$ unless otherwise specified.

The NIST test suite [11] was applied to β_D sequences [14] and sequences obtained by β-ary/binary conversions. In a randomness test for the β_D sequence, the delay parameters in Eq.(12) were $d_1 = 6$, $d_2 = 6 + 7$, ..., $d_i = d_{i-1} + i + 5$ $(i \geq 3)$ and $N_{xor} = 4, 8$, and 16.

There are fifteen tests for the NIST test suite. A result of each test is expressed by P (Pass) or F (Fail) except for non-overlapping template matching test for which the number of templates that passes the test is shown [12].

β encoders have voltage parameters. Three patterns of voltage parameters shown in Table I are used, where $V_{\rm DDA}$, $V_{\rm DDD}$, and $V_{\rm DDD\_IO}$ are drive voltages for analog, digital and Input/Output (I/O) purposes, $A_{\rm in+}$ and $A_{\rm in-}$ are differential input voltage which expresses the initial value of an input voltage for A/D conversion, and $V_{\rm ref+}$, $V_{\rm ref-}$, and $V_{\rm ref\_CM}$ are reference voltages which are upper limit, lower limit, and threshold voltage. The last three values are designed to satisfy $V_{\rm ref\_CM} = \frac{1}{2}(V_{\rm ref-} + V_{\rm ref+})$.

The difference between Pattern A and Pattern B is that $A_{\rm in+} = 0.80$ and $A_{\rm in-} = 0.40$ for the former and $A_{\rm in+} = 0.60$ and $A_{\rm in-} = 0.60$ for the latter. This comparison shows the effect of input voltage on the randomness of generated sequences. The difference between Pattern B and Pattern C is that $V_{\rm DDA} = 1.20$ for the former and $V_{\rm DDA} = 1.40$ for the latter. In general, a CMOS circuit has a range of $V_{\rm DDA}$ that the circuit can properly work. $V_{\rm DDA}$ for the CMOS-implemented β encoders is designed to be 1.20, but the circuit is more stable if $V_{\rm DDA} = 1.40$.

Tables II and III show results of the NIST test suite for β_D sequences and the sequences obtained by β-ary/binary conversion, respectively. These results show that EXOR operation is needed for the former sequences to pass the NIST test suite, but is not needed for the latter sequences. Table IV shows the effect of window size w on the results of NIST test. It is shown that $w \geq 15$ is required to guarantee the generated sequences pass the NIST test and that if $w = 10$ the generated sequences do not pass most of the tests.

Since the effective value of β is not known beforehand, we verified the robustness of the proposed method to the mismatch of the values of β used in the encoder and the β-ary/binary converter. The value of β is designed to be $\beta = 1.83$. Denote the β used in the β-ary/binary converter by β'. Table.V shows that the results for $\beta' = 1.7, 1.8$, and 1.9 are almost the same. However, the result for $\beta' = 1.6$ is very poor. We conclude that the proposed method can allow a fluctuation of β at most 0.1.

V. Conclusion

In this paper, a β-ary/binary conversion method for generating random numbers using a β encoder, proposed in [16], has been explained. Experimental results have shown that sequences obtained by the β-ary/binary conversion can pass the NIST statistical test suite. The proposed method is implemented by integer calculations. It has been shown that the necessary window size is 15 and the converter is robust to the mismatch of β.

Acknowledgment

This research was supported by the Aihara Project, the FIRST program from JSPS, initiated by CSTP and JSPS KAKENHI Grant Number 25820162.

References

[1] I. Daubechies, R.A. DeVore, C.S. Güntürk, and V. A. Vaishampayan, "A/D Conversion With Imperfect Quantizers,"*IEEE Trans. Inform. Theory*, vol.52, no.3, March 2006.

[2] I. Daubechies and O. Yilmaz, "Robust and Practical Analog-to-Digital Conversion With Exponential Precision," *IEEE Trans. Inform. Theory*, vol.52, no.8, pp. 3533-3545, 2006.

[3] T. Kohda, Y. Horio, K. Aihara, "β-Expansion Attractors Observed in A/D Converters," *Chaos: An Interdisciplinary J. of Nonlinear Science*, Vol. 22, 047512, 2012.

[4] K. Yoshimura, J. Muramatsu, P. Davis, A. Uchida and T. Harayama, "Secure Key Distribution Using Correlated Randomness in Optical Devices,"*IEICE Tech. Rep.*, NLP2011-106, vol. 111, no. 276, pp. 81-84, 2011.

[5] S. Callegari, R. Rovatti, and G. Setti, "Embeddable ADC-Based True Random Number Generator for Cryptographic Applications Exploiting Nonlinear Signal Processing and Chaos," *IEEE Trans. Signal Proc.*, vol. 53, no. 2, Feb. 2005.

[6] S. Domichi, "Impacts and Countermeasures in Device Mismatches in Analog Circuits," *the Journal of IEICE*, vol. 92, no. 6, pp. 446-451, 2009.

[7] T. Matsuura, "Technical Trend of Digitally Assisted A/D Converters," *IEICE Technical Report*, CAS2011-104, vol. 111, no. 377, pp. 103-108, Jan. 2012.

[8] H. San et. al, "Non-binary Pipeline Analog-to-Digital Converters Based on β-Expansion," *IEICE Trans. Fundamentals*, vol.E96-A, no.2, pp. 415-421, Feb. 2013.

[9] R. Sugawara, H. San, K. Aihara and M. Hotta, "Experimental Implementation of Non-binary Cyclic ADCs with Radix-value Estimation Algorithm," *IEICE Trans on Electronics*, vol.E97-C, no.4, pp.308–315, April 2014.

[10] T. S. Han and K. Kobayashi, *Mathematics of Information and Coding*, Amer Mathematical Society, 2007.

[11] A. Rukhin, et.al, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST, Special Publication 800-22, May 2001.

[12] H. Yoshida, T. Murakawa, and S. Kawamura, "Study on Testing for Randomness of Pseudo-Random Number Sequence with NIST SP800-22 rev.1a," *IEICE Technical Report*, vol. 112, no. 301, NLP2012-78, pp. 13-18, Nov. 2012.

[13] T. Makino, Y. Iwata, K. Shinohara, Y. Jitsumatsu, M. Hotta, H. San and K. Aihara, *Rigorous Estimates of Quantization Error for A/D Converters Based on Beta-Map*, *NOLTA Journal*, vol.6, no.1, pp.99-111, Jan. 2015.

[14] Y. Hirata, Y. Jitsumatsu, T. Kohda, and K. Aihara, "Pseudo-Random Number Generator Using β-Expansion Attractors in A/D Converters," *Proc. of the 30th Sympo. on Cryptography and Information Security (SCIS2013)*, Jan. 2013.

[15] K. Matsumura, Y. Jitsumatsu, and T. Kohda, "Implementation of Pseudo-Random Number Generator Based on β-Expansion Attractor," *IEEJ Technical Report*, ECT-14-026, pp. 135-140, Jan. 2014.

[16] K. Matsumura, T. Teraji, K. Oda, and Y. Jitsumatsu, "Random Number Generation Using β Encoder," *Proc. of the 32nd Sympo. on Cryptography and Information Security (SCIS2015)*, Jan. 2015.

TABLE II. RESULTS OF THE NIST STATISTIAL TEST SUITE FOR β_D SEQUENCES

	β-ary/binary conversion is NOT applied											
Voltage Pattern	A				B				C			
The number of EXORs	1	4	8	16	1	4	8	16	1	4	8	16
Frequency (Monobits) Test	F	P	P	P	F	P	P	P	F	P	P	P
Frequency Test within a Block	F	F	F	P	F	F	F	P	F	F	P	P
Cummulative Sums (Cusum) Test	F	P	P	P	F	P	P	P	F	P	P	P
Runs Test	F	F	F	P	F	F	F	P	F	F	F	P
Test for the Longest Run of Ones in a Block	F	F	P	P	F	F	P	P	F	F	P	P
Binary Matrix Rank Test	P	P	P	P	P	P	P	P	P	P	P	P
Discrete Frourier Transform Test	F	F	F	P	F	F	P	P	F	F	P	P
Non-Overlapping Template Matching Test	0	12	128	157	0	12	135	P	0	19	156	155
Overlapping Template Matching Test	F	F	F	P	F	F	F	P	F	F	P	P
Maurer's "Universal Statistical" Test	F	F	P	P	F	F	P	P	F	F	P	P
Approximate Entropy Test	F	F	F	P	F	F	F	P	F	F	P	P
Random Excursion Test	F	F	P	P	F	F	F	P	F	F	P	P
Random Excursions Variant Test	F	P	P	F	F	P	P	P	F	P	P	P
Serial Test	F	F	P	P	F	F	P	P	F	F	P	P
Linear Complexity Test	P	P	P	P	P	P	P	P	P	P	P	P

TABLE III. RESULTS OF THE NIST STATISTIAL TEST SUITE FOR β-ARY/BINARY CONVERSION

	β-ary/binary conversion is applied											
Voltage Pattern	A				B				C			
The number of EXORs	1	4	8	16	1	4	8	16	1	4	8	16
Frequency (Monobits) Test	P	P	P	P	P	P	P	P	P	P	P	P
Frequency Test within a Block	P	P	P	P	P	P	P	P	P	P	P	P
Cummulative Sums (Cusum) Test	P	P	P	P	P	P	P	P	P	P	P	P
Runs Test	P	P	P	P	P	P	P	P	P	P	P	P
Test for the Longest Run of Ones in a Block	P	P	P	P	P	P	P	P	P	P	P	P
Binary Matrix Rank Test	P	P	P	P	P	P	P	P	P	P	P	P
Discrete Frourier Transform Test	P	P	P	P	P	P	P	P	P	P	P	F
Non-Overlapping Template Matching Test	157	157	P	P	157	P	156	156	P	P	157	P
Overlapping Template Matching Test	P	P	P	P	P	P	P	P	P	P	P	P
Maurer's "Universal Statistical" Test	P	P	P	P	P	P	P	P	P	P	P	P
Approximate Entropy Test	P	P	P	P	P	P	P	P	P	P	P	P
Random Excursion Test	P	P	P	P	P	P	P	P	P	P	P	P
Random Excursions Variant Test	P	P	P	P	P	P	P	P	P	P	P	P
Serial Test	P	P	P	P	P	P	P	P	P	P	P	P
Linear Complexity Test	P	P	P	P	P	P	P	P	P	P	P	P

TABLE IV. RESULTS OF THE NIST STATISTIAL TEST SUITE: COMPARISON OF WINDOW SIZE

	β-ary/binary conversion is applied								
Voltage Pattern	A			B			C		
window size	10	15	20	10	15	20	10	15	20
Frequency (Monobits) Test	F	P	P	P	P	P	F	P	P
Frequency Test within a Block	F	P	P	F	P	P	P	P	P
Cummulative Sums (Cusum) Test	F	P	P	P	P	P	F	P	P
Runs Test	F	P	P	F	P	P	F	P	P
Test for the Longest Run of Ones in a Block	P	P	P	P	P	P	P	P	P
Binary Matrix Rank Test	P	P	P	P	P	P	P	P	P
Discrete Frourier Transform Test	P	F	P	P	P	P	P	P	P
Non-Overlapping Template Matching Test	106	P	157	115	P	157	147	P	P
Overlapping Template Matching Test	F	P	P	F	P	P	P	P	P
Maurer's "Universal Statistical" Test	P	P	P	P	P	P	P	P	P
Approximate Entropy Test	F	P	P	F	P	P	F	P	P
Random Excursion Test	P	F	P	P	P	P	P	P	P
Random Excursions Variant Test	P	P	P	P	P	P	P	P	P
Serial Test	F	P	P	F	P	P	F	P	P
Linear Complexity Test	P	P	P	P	P	P	P	P	P

TABLE V. RESULTS OF THE NIST STATISTIAL TEST SUITE: COMPARISON OF β USED IN THE CONVERTER

	β-ary/binary conversion is applied											
Voltage Pattern	A				B				C			
β'	1.6	1.7	1.8	1.9	1.6	1.7	1.8	1.9	1.6	1.7	1.8	1.9
Frequency (Monobits) Test	P	P	P	P	F	P	P	P	P	P	P	P
Frequency Test within a Block	P	P	P	P	P	P	P	P	P	P	P	P
Cummulative Sums (Cusum) Test	F	P	P	P	P	P	P	P	P	P	P	P
Runs Test	P	P	P	P	P	P	P	P	P	P	P	P
Test for the Longest Run of Ones in a Block	F	P	P	P	P	P	P	P	P	P	P	P
Binary Matrix Rank Test	P	P	P	P	P	P	P	P	P	P	P	P
Discrete Frourier Transform Test	P	P	P	P	P	P	P	P	F	P	P	P
Non-Overlapping Template Matching Test	152	P	157	P	147	155	157	P	152	157	P	157
Overlapping Template Matching Test	P	P	P	P	P	P	P	P	P	P	P	P
Maurer's "Universal Statistical" Test	P	P	P	P	P	P	P	P	P	P	P	P
Approximate Entropy Test	P	P	P	P	P	P	P	P	P	P	P	P
Random Excursion Test	P	P	P	P	P	F	P	P	F	P	P	P
Random Excursions Variant Test	F	P	P	P	P	P	P	P	F	P	P	P
Serial Test	P	P	P	P	P	P	P	P	P	P	P	P
Linear Complexity Test	P	P	P	P	F	P	P	P	P	P	P	P