A $\beta$-ary to binary conversion for random number generation using a $\beta$ encoder

Yutaka Jitsumatsu$^{1a)}$ and Kazuya Matsumura$^1$

$^1$Department of Informatics, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan

$^a)$jitumatu@inf.kyushu-u.ac.jp

Received May 1, 2015; Revised September 25, 2015; Published January 1, 2016

Abstract: A $\beta$ encoder is an analog-to-digital (A/D) converter, proposed by Daubechies et al. in 2002, that outputs a truncated sequence of $\beta$ expansion of an input value $x$. It is known that the conventional pulse code modulation (PCM) that outputs the binary expansion of $x$ is sensitive to the offset of the threshold voltage, while a $\beta$ encoder is robust to such an offset. We propose an algorithm that calculates the binary expansion of an interval that is identified by an output sequence from a $\beta$ encoder. Such a method is referred to as a $\beta$-ary to binary converter. We generate sequences of random numbers, using a hardware $\beta$ encoder followed by the $\beta$-ary to binary converter. The randomness of the generated binary random numbers is verified by the National Institute of Standards and Technology (NIST) statistical test suite.

Key Words: random number generation, analog-to-digital converter, $\beta$ expansion, $\beta$ transformation, interval algorithm

1. Introduction

The importance of random number generation is expanding because of the development of information and communication technologies and demand for secure communications. Pseudo-random numbers are generated by a deterministic algorithm with a seed, that can be perfectly regenerated if the same algorithm and the same seed are used. On the other hand, there are demands, especially for security purposes, for a physical random number generator that uses measurements of a certain physical phenomenon. Examples of physical random number generation include a random number generator using a semiconductor laser [1] or one that uses an electronic circuit based on a chaotic map [2–9].

An Analog-to-Digital (A/D) converter is an electronic circuit that outputs the digital code (mostly binary code) of a continuous-time real-valued input signal. A Digital-to-Analog (D/A) converter receives the digital code and outputs a real-valued signal that is a distorted representation of the original input signal. In the case of pulse code modulation (PCM), the input signal is sampled at a rate higher than the Nyquist rate and the samples are quantized to binary codes with a finite code length. We may assume that, after a normalization process, the samples are contained in $[0, 1]$. The output of PCM is a truncated binary expansion of the input value. The quantization error is upper bounded by $2^{-(N-1)}$, where $N$ is the number of bits for binary expansion. However, this upper-bound cannot be guaranteed if $N$ becomes very large. One of the reasons is that a comparator (or a
1 bit quantizer) in a PCM circuit (See Fig. 2) makes a false decision when an input voltage is very close to the threshold value. Once a comparator makes an erroneous decision at some bit, the error cannot be recovered by the following bits. Hence, the PCM method is sensitive to the variation in the comparator voltage offset.

In 2002, Daubechies et al. [10–12] proposed a new A/D converter that overcomes the above-mentioned drawback of the PCM, that is based on the \( \beta \) expansion of a real number and, therefore, is called the \( \beta \) encoder. The properties of \( \beta \) expansion and the dynamics of its associated map was first studied by Rényi [13] and then extensively studied by Parry [14–17]. The \( \beta \) expansion of a real number \( x \) is defined by [13]

\[
x = \sum_{i=1}^{\infty} a_i \beta^{-i}, \quad a_i \in \{0, 1, 2, \ldots, \lceil \beta \rceil - 1\} \text{ for } x \in [0, \frac{1}{\beta - 1}],
\]

where \( \beta \) is a positive real number and \( \lceil x \rceil \) denotes the smallest integer greater than or equal to \( x \). Equation (1) simplifies to the binary expansion for \( \beta = 2 \). In the case of \( \beta \) encoder, \( \beta \) is in \((1, 2)\). For a fixed \( x \), \( a_1 a_2 \cdots \) are not unique. In fact, there are infinitely many ways to expand \( x \). This implies that digital codes produced by a \( \beta \) encoder are redundant. Unlike PCM, an erroneous decision made by a \( \beta \) encoder can be recovered by the following bits. Thus, a \( \beta \) encoder is robust to fluctuation of threshold value of a quantizer. Such a \( \beta \) encoder does not need high-precision circuit elements and is implemented by an electronic circuit that achieves very small area consumption as well as low power consumption [19–22].

A \( \beta \) encoder is composed of an amplifier with an amplification factor \( \beta \) and a comparator that outputs a digital code 0 or 1. If the input voltage multiplied by \( \beta \) is greater than a pre-determined threshold, then the output is one. Otherwise, the output is zero. Such an output is subtracted from the input voltage multiplied by \( \beta \) and is fed back to the input. This process is repeated \( N \) times to output \( N \) digital codes. The dynamics of such a repeated process is described by a piecewise linear (PL) map. Practically, \( N < 30 \) bits are sufficient to obtain a distorted representation for \( x \) by \( \hat{x} = \sum_{i=1}^{N} a_i \beta^{-i} \). If the number of iterations, \( N \), is very large, such as one million, we can observe chaos attractors in \( \beta \) converters [23]. This fact motivated Hirata et al. However, outputs from a \( \beta \) encoder have strong correlations between successive bits. Such strong correlations should be eliminated. In [24], the exclusive disjunction (or EXclusive OR: EXOR) of several preceding outputs from a \( \beta \) encoder or the EXOR of several \( \beta \) encoders was taken. It was verified that the resulting random numbers can pass the National Institute of Standards and Technology (NIST) randomness test suite.

There have been many researches on random number generation (RNG) using chaotic dynamics. Among them, RNGs using discrete-time chaotic dynamics with PL maps have attracted much attention. Stojanovsky and Kocarev [2] proposed to use a PL map with two slopes \( 1 < k_1 < 2 \) and \( 1 < k_2 < 2 \). They pointed out that Bernoulli map is unstable because of the non-ideality of circuit element. Using the same PL map as in [2] but with \( k_1 = k_2 (= k) \), Addabbo et al. presented an interesting approach in which the amplification factor \( k \) and a threshold are controlled by utilizing the observed statistics of the binary output codes [5]. Output sequences from such PL maps have bias and correlations that must be eliminated by some post-processing unit. In [3], taking EXOR operation of five consecutive bits is recommended. In [6], using a shift register and an EXOR operation between the current bit and the last issued output bit is proposed. Callegari et al. proposed an RNG
using a 1.5 bit pipelined A/D converter [4], where taking EXOR of consecutive four output bits is recommended so that the resulting random numbers are robust to the non-ideality of circuit element. In [7], the issue of implementation variations of a PL map called zigzag map for chaos-based RNG is discussed. It was proved in [8] that there exists a sequence of post-processors that asymptotically generate independent and identically distributed (i.i.d.) binary random variables if the conversion rate is less than the entropy rate of the raw random numbers. In [9], a notion of binary metric entropy which is a binary version of Kolmogorov-Sinai entropy [25] is introduced and upper and lower bounds of the binary metric entropy are given. A bit-generation function, defined by \( B_i(x) = \lfloor ix \rfloor \mod 2 \) was proposed in [9], where \( i \) is a large integer and \( \lfloor x \rfloor \) denotes the largest integer not greater than \( x \). Such a function was shown to be robust to the implementation variations of a given map.

The throughput, i.e., the number of bits to be generated per second, is one of the most important performance measure for RNGs. An advantage of using simple post-processing method is that it can achieve a high throughput. The conversion rate, defined as the average numbers of output bits per one input bit, of the methods in [3, 4] is 1/4 or 1/5. This makes the throughput reduced to 1/4 or 1/5 of the original throughput but such a loss of throughput is not a problem because the original processing speed is very high. It is claimed in [8] from a viewpoint of the metric entropy that conversion rate 1 cannot make the output sequence to be i.i.d. sequence. One of the disadvantages of the simple post-processing using EXOR operation is that it is often difficult to give a theoretical evaluation of the deviation of the distribution of the generated sequence from an ideal distribution.

In this paper, we propose an interval algorithm for converting \( \beta \) expansion coefficients into binary expansion coefficients, by which we eliminate the strong correlation observed in the outputs of \( \beta \) encoder (See Fig. 1). In the \( \beta \)-ary to binary converter, the interval corresponding to the \( \beta \) expansion coefficients is calculated [26] and then binary expansion of the interval is obtained. Such an interval calculation is basically the same as Han and Hoshi [27]'s interval algorithm. The interval algorithm is based on the successive refinement of partitions of the unit interval \([0,1)\). At \( i\)-th step, one of two sub-intervals of \((i-1)\)-th interval is chosen (each sub-interval corresponds to \( a_i = 0 \) or \( 1 \)).

The largest disadvantage of the proposed method is its computational cost. The concept and mathematical structure of the interval algorithm are simple. Thus, it can be efficiently implemented in a modern computer. If we implement the interval algorithm by an electronic circuit its computational cost is much higher than the simple post-processing using EXOR operation [3, 4, 6]. On the other hand, we have a large advantage in using the interval algorithm as a post-processing that distribution of the output sequence can be analyzed theoretically. From a theoretical point of view, it is importance to evaluate the performance of the interval algorithm for \( \beta \)-ary to binary conversion in terms of conversion rate and the variational distance between the distribution of ideal random numbers and that of the generated bits. This subject is discussed in a separate paper [28].

The proposed \( \beta \)-ary to binary converter should be implemented in a digital circuit. Therefore, we have developed a fixed-point arithmetic with limited precision for the interval algorithm, which is similar to the methods discussed in [29, 30]. The differences between the proposed algorithm and the conventional algorithms [27, 29–32] are

1. Coin random numbers are the outputs from a hardware \( \beta \) encoder [20, 21]. The source of the randomness is the uncertainty in the initial state of the system. The random number generation problem in information theory discusses how we can simulate a prescribed distribution using given random numbers, where randomness of the information source is given a priori.

2. The \( \beta \) encoders use imperfect quantizers that allow threshold voltage offsets. This implies the two subintervals corresponding to the next input symbol overlap each other. On the other hand, intervals calculated in Han and Hoshi’s algorithm do not overlap.

3. The PL map for the \( \beta \) encoder does not satisfy the Markov partition condition [33] and thus cannot be expressed by a finite-state Markov chain.

4. The radix \( \beta \) is realized as an amplification factor of an amplifier in an analog circuit. Therefore the value of \( \beta \) may fluctuate.
Kanaya discussed how to construct a PL map for a given discrete memoryless source \[34\] and a given finite-order discrete Markov source \[35\] and discussed the connection between such chaos models and the arithmetic code. See also \[33\] for the connection between information sources and the chaotic dynamics.

The contribution of this paper is twofold: One is that we give a $\beta$-ary to binary conversion method based on the implementation of an interval calculation for $\beta$ expansion with an imperfect quantizer by a finite fixed-point arithmetic. The other is that we examine the validity of the proposed method by use of a hardware $\beta$ encoder implemented by an electronic circuit. The validity of the proposed random number generator is examined by the National Institute of Standards and Technology (NIST) test suite. The word length of the fixed-point arithmetic required for the generated sequences to pass the NIST statistical test suite is investigated. Robustness of the proposed method to mismatches of the $\beta$ value is also examined.

This paper is organized as follows: In Section 2, we review the relation between binary expansion used in PCM and Bernoulli shift map and explain the instability issue of Bernoulli maps realized by electronic circuits. Then, the fundamental property of $\beta$-encoder and its associated $\beta$ transform are explained. A flaky quantizer model introduced by Daubechies et al. \[11, 12\] to describe the instability of a quantizer is explained. Advantage of $\beta$ encoders over PCM as well as 1.5 bit encoder is discussed.

In Section 3, we introduce RNG using $\beta$ encoder together with a post-processing unit called $\beta$-ary to binary conversion. Strong correlations observed in the outputs of $\beta$ encoder are mitigated by the post-processing. The $\beta$-ary to binary conversion is based on the interval algorithm for RNGs. Fundamental limits of an RNG using the interval algorithm are reviewed.

In Section 4, the performance of the generated random binary sequence is examined by the NIST randomness test suite. A hardware $\beta$ encoder designed by San et al. is used for generating raw binary sequence which is processed by $\beta$-ary to binary conversion. Section 5 summarizes this paper.

2. Pulse code modulation and $\beta$ encoder

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

![Fig. 2. PCM-based A/D converter](image)

### 2.1 Pulse code modulation (PCM)

Consider a continuous-time real-valued bandlimited signal $s(t)$ whose highest frequency is $W$, i.e., the Fourier transformation of $s(t)$, denoted by $S(\omega) = \int_{-\infty}^{\infty} s(t)e^{-j\omega t}dt$ vanishes for $|\omega| > 2\pi W$. In a Pulse Code Modulation (PCM), the signal $s(t)$ is sampled at some rate higher than the Nyquist rate, denoted by $s(kT_s)$, $k = 0, 1, \ldots$, where $T_s = 1/f_s$ is the sampling interval and $f_s > 2W$ is the sampling rate. After a normalization process, samples $s(kT_s)$ are assumed to be in $[0, 1)$. Then, analog-to-digital conversion of PCM gives an $N$-bit truncated binary expansion of $x = s(kT_s)$.

The binary expansion of a real value $x \in [0, 1)$ is defined by

$$x = \sum_{i=1}^{\infty} b_i 2^{-i}, \quad b_i \in \{0, 1\}. \quad (2)$$

The binary expansion of a real number $x$ is unique except for the case when $x$ is a dyadic number, i.e., except for $x$ with two expansions $b_1b_2 \ldots b_n1000 \cdots$ and $b_1b_2 \ldots b_n0111 \cdots$. Normally the former
expression is adopted. Define \( x_{i+1} = B(x_i) \) \((i = 0, 1, 2, \ldots)\) with an initial value \( x_0 = x \), where \( B(x) \) is a Bernoulli map (or a dyadic map), defined by

\[
B(x) = 2x - Q_1(2x),
\]

where \( Q_1(x) \) is a one-bit quantizer (or a comparator) with a threshold 1, defined by

\[
Q_1(x) = \begin{cases} 
0, & \text{if } x < 1, \\
1, & \text{if } x \geq 1.
\end{cases}
\]

Then, the coefficients \( \{b_i\} \) are determined recursively by

\[
b_{i+1} = Q_1(2x_i), \quad i = 0, 1, \ldots
\]

Such a process to determine \( b_i \) is implemented in a PCM by an electronic circuit, as depicted in Fig. 2.

The most critical drawback of a PCM that degrades the accuracy of the quantization is that the comparator \( Q_1(x) \) may make a wrong decision if \( x_i \) is very close to the threshold value because of a slight fluctuation of the threshold voltage. For example, assume the threshold is changed from 1 to some value \( \nu > 1 \) so that \( B(x) \) is replaced by

\[
\tilde{B}(x) = 2x - Q_\nu(2x),
\]

where \( Q_\nu(x) = \begin{cases} 
0, & \text{if } x < \nu, \\
1, & \text{if } x \geq \nu.
\end{cases}
\]

Then, if \( x_i \in \left[\frac{3}{2}, \nu \right) \) for some \( i \geq 1 \), we have \( x_{i+n} = \tilde{B}^n(x_i) = \tilde{B}^{n-1}(2x_i - 1) = \tilde{B}^{n-2}(4x_i - 3) = \cdots = \frac{2^n(x_i - \frac{1}{2}) + 1}{2^n} \rightarrow +\infty \) \((n \rightarrow \infty)\). For avoiding such un-stability of conversion, 1.5 bit encoders [39] and digital calibration techniques [40] have been proposed.

On the other hand, \( \Sigma \Delta \) modulators [41] have a good property in that they are robust to fluctuations of threshold values in their quantizers. However, they have a drawback in that the 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 in inverse proportion to the number of bits in contrast to the exponential accuracy of the PCM. \( \beta \) encoders have the two good properties of PCM and \( \Delta \Sigma \), i.e., robustness against fluctuations of the threshold voltage and exponential accuracy [11]. In the next subsection, a scale-adjusted \( \beta \) encoder [23] is explained.

### 2.2 1.5 bit encoder

The 1.5 bit encoder is defined as follows [39]: Consider a quantizer with two thresholds and three digital output codes, i.e.,

\[
Q_{1.5}^{(\nu_0, \nu_1)}(x) = \begin{cases} 
00, & \text{if } x \leq \nu_0, \\
01, & \text{if } \nu_0 < x < \nu_1, \\
10, & \text{if } x \geq \nu_1,
\end{cases}
\]

where \( \nu_0 \) and \( \nu_1 \) are the thresholds. The dynamics of the 1.5 bit encoder is described by \( x_{i+1} = \tau(x_i) \) with the input value \( x_0 = x \), where

\[
\tau(x) = 2x - \frac{1}{2}Q_{1.5}^{(\nu_0, \nu_1)}(x).
\]

The digital codes, 00, 01, and 10 are understood as binary numbers, and equal to 0, 1, and 2 in decimal numbers. Let \( \tilde{b}_{i+1} = Q_{1.5}^{(\nu_0, \nu_1)}(x_i) \) for \( i = 0, 1, \ldots \). We observe that the orbit \( \{x_i\} \) does not
diverge if \( \nu_0 \) and \( \nu_1 \) satisfy \( \frac{1}{6} < \nu_0 < \frac{1}{2} \) and \( \frac{1}{2} < \nu_1 < \frac{3}{4} \). In such a case, the distorted representation of \( x \) from \( \tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_N \), given by

\[
x_N^{(1.5)} = \sum_{i=1}^{N} \tilde{b}_i 2^{-i-1}.
\]  

(10)
gives the exponential accuracy. Therefore, the 1.5 bit encoder is robust to the fluctuation of the thresholds \( \nu_0 \) and \( \nu_1 \). Note that 1.5 bit encoder is reduced to the binary expansion if \( \nu_0 = \nu_1 = 1/2 \).

The expression of Eq. (10) is redundant because in addition to the two choices for binary expansion, we have a third choice “01” with a region \([\nu_0, \nu_1]\). A redundant expression enables us to recover an erroneous decision on the comparator caused by the fluctuations of \( \nu_0 \) and \( \nu_1 \), by using the following bits.

### 2.3 \( \beta \) encoder

The \( \beta \) expansion of a real number \( x \in [0, 1/(\beta - 1)] \) with \( \beta \in (1, 2) \) is defined by

\[
x = \sum_{i=1}^{\infty} a_i \beta^{-i}, \quad a_i \in \{0, 1\}.
\]

(11)

Unlike the binary expansion, almost all \( x \in (0, 1/(\beta - 1)) \) can be expanded as (11) in infinitely many ways. One way to determine \( a_i \) is greedy expansion [18] defined by

\[
a_i = \begin{cases} 1, & \text{if } \sum_{j<i} a_j \beta^{-j} + \beta^{-i} \leq x, \\ 0, & \text{if } \sum_{j<i} a_j \beta^{-j} + \beta^{-i} > x. \end{cases}
\]

(12)

Another way to determine \( a_i \), called lazy expansion [18], is defined by

\[
a_i = \begin{cases} 1, & \text{if } \sum_{j<i} a_j \beta^{-j} + \sum_{j>i} \beta^{-j} < x, \\ 0, & \text{if } \sum_{j<i} a_j \beta^{-j} + \sum_{j>i} \beta^{-j} \geq x. \end{cases}
\]

(13)

More generally, \( \{a_i\} \) are called cautious expansion if \( a_i \) is determined as follows:

**Definition 1** Cautious expansion with a threshold \( \nu \in (1, 1/(\beta - 1)) \) of a real number \( x \in [0, 1/(\beta - 1)] \) is defined recursively by

\[
a_{i+1} = Q_\nu(\beta x_i), \quad i = 0, 1, \ldots,
\]

(14)

\[
x_{i+1} = C_{\beta, \nu}(x_i),
\]

(15)

where \( C_{\beta, \nu} \) is the cautious \( \beta \) map, defined by

\[
C_{\beta, \nu}(x) = \beta x - Q_\nu(\beta x).
\]

(16)

**Definition 2** The \((\beta, \alpha)\)-transformation \( T_{\beta, \alpha} : [0, 1) \mapsto [0, 1) \) is defined by [15]

\[T_{\beta, \alpha}(x) = \beta x + \alpha \mod 1, \beta \geq 1, 0 \leq \alpha \leq 1.\]

(17)

T. Kohda et al. showed the following lemma [26].

**Lemma 1** The dynamical system \(((\nu - 1, \nu), C_{\beta, \nu}(x))\) with \( \nu = 1 + \alpha/(\beta - 1) \) is topologically conjugate to the dynamical system \(((0, 1), T_{\beta, \alpha}(x))\) via the conjugacy \( \varphi^{-1}(x) = x + \alpha/(\beta - 1) \), i.e.,

\[
\varphi(C_{\beta, \nu}(\varphi^{-1}(x))) = T_{\beta, \alpha}(x).
\]

(18)

The properties of the \( \beta \) expansion and its associated map were first studied by Rényi [13], then extensively studied by Parry [14–17]. The most important property of the \( \beta \) map is that there is a unique absolutely continuous invariant probability measure [13]. The explicit form of the invariant measure is given by Parry [15] as follows:
Theorem 1 (Parry [15]) An unnormalized invariant measure for $T_{\beta,\alpha}(x)$ is given as

$$h(x) = \sum_{x<T_{\beta,\alpha}^n(1)} \beta^{-n} - \sum_{x<T_{\beta,\alpha}^n(0)} \beta^{-n}.$$  \hspace{1cm} (19)

Remark 1 Suppose that we ignore the first $a_n$ and consider $a_{n+1}a_{n+2}a_{n+3} \cdots$. It should be noted that from this sequence we obtain $x_n = \sum_{i=n+1}^{\infty} a_i \beta^{-i}$. Theorem 1 implies that $\{x_n\}$ tends to follow Parry’s absolutely continuous invariant density $h(x)$ for almost all initial values $x = x_0$.

In what follows, we consider a scale-adjusted $\beta$ expansion of $\tilde{x} = s(\beta - 1)x \in [0,s]$, where $s$ is a scale parameter and $\beta \in (1,2)$.

Definition 3 The scale-adjusted $\beta$ map, $S_{\beta,\nu,s} : [0,s) \mapsto [0,s]$ for $\nu \in [s(\beta - 1), s]$ is defined by

$$S_{\beta,\nu,s}(x) = \beta x - s(\beta - 1)Q_{\nu}(\beta x),$$  \hspace{1cm} (20)

where $s$ is a scale parameter. The scale-adjusted map is called greedy if $\nu = s(\beta - 1)$ and lazy if $\nu = s$.

Let the input for the scale-adjusted $\beta$ encoder be $\tilde{x} \in [0,s)$, which is a scaled version of the input for the original $\beta$ encoder $x \in [0,1/(\beta - 1))$. The output sequence for $\tilde{x}$ is (See Fig. 3)

$$a_{i+1} = Q_{\nu}(\beta x_i), \quad i = 0, 1, \ldots$$  \hspace{1cm} (21)

where

$$x_{i+1} = S_{\beta,\nu,s}(x_i), \quad x_0 = \tilde{x}. \hspace{1cm} (22)$$

Note that the original $\beta$ map corresponds to the case where $s = 1/(\beta - 1)$. Note also that we have

$$\tilde{x} = s(\beta - 1) \sum_{i=1}^{\infty} a_i \beta^{-i}$$  \hspace{1cm} (23)

As shown in Fig. 4(a), $x_n = S_{\beta,\nu,s}^n(x_0)$ does not diverge if the threshold value $\nu$ satisfies $\nu \in [(\beta - 1)s,s]$. This property makes the $\beta$ encoders robust to the fluctuations of the threshold.

Daubechies et al. [11] pointed out that the value $\nu$ may differ from one application to the next and introduced a flaky version of an imperfect quantizer, defined by

$$Q_{\nu_0,\nu_1}^f(x) = \begin{cases} 0, & \text{if } x \leq \nu_0, \\ 1, & \text{if } x \geq \nu_1, \\ 0 \text{ or } 1, & \text{if } x \in (\nu_0, \nu_1). \end{cases} \hspace{1cm} (24)$$

This notation means that for $x \in (\nu_0, \nu_1)$ we do not know which value in $\{0, 1\}$ the flaky quantizer will assign.

Daubechies et al. [11] gave the following theorem that guarantees the exponential accuracy of the quantization error made by a $\beta$ encoder with a flaky quantizer.
Fig. 4. (a) The scale-adjusted $\beta$ map with scale $s$. (b) Daubechies’s map defined by $y = \beta x - Q f_{[\nu_0,\nu_1]}(\beta x)$.

Theorem 2 (Daubechies et al. [11]) For $\beta \in (1, 2)$, $x \in [0, 1)$, $1 < \nu_0 < \nu_1 < 1/(\beta - 1)$, define

$$b_{i+1}^f = Q f_{[\nu_0,\nu_1]}(\beta x_i^f), \quad i = 0, 1, 2, \ldots, \quad (25)$$

$$x_{i+1}^f = \beta x_{i+1}^f - Q f_{[\nu_0,\nu_1]}(\beta x_i^f), \quad x_0^f = x \quad (26)$$

Then for all $N \in \mathbb{N}$,

$$0 \leq x - \sum_{i=1}^{N} b_i^f \beta^{-i} \leq \nu_1 \beta^{-N}. \quad (27)$$

The map $y = \beta x - C^f_{[\nu_0,\nu_1]}(x) \stackrel{\text{def}}{=} Q f_{[\nu_0,\nu_1]}(\beta x)$ is shown in Fig. 4 (b), where $y$ takes blue solid line if $x \leq \nu_0$, red solid line if $x \geq \nu_1$, and we do not know which one of the two green dashed lines $y$ will take if $x \in (\nu_0, \nu_1)$.

There is another mathematical model of a $\beta$ map with an imperfect quantizer called random $\beta$ transform, introduced by Dajani and Kraaikamp [42]. They introduced a binary sequence $\omega = \omega_1 \omega_2 \cdots$. Let $\sigma$ denote the left shift on the set of $\{0, 1\}^\mathbb{N}$. The random $\beta$ transform $K$ is defined as the set $\{0, 1\}^\mathbb{N} \times [0, 1/(\beta - 1)]$ to itself as follows:

$$K(\omega, x) = \begin{cases} 
(\omega, \beta x), & \text{if } x < \frac{1}{\beta}, \\
(\omega, \beta x - 1), & \text{if } x > \frac{1}{\beta(\beta - 1)}, \\
(\sigma \omega, \beta x - \omega_1), & \text{if } x \in \left[\frac{1}{\beta}, \frac{1}{\beta(\beta - 1)}\right].
\end{cases} \quad (28)$$

In [43], Dajani and de Vries showed that $K$ has an invariant probability measure $\mu_\beta = m_1 \times \mu_\beta$, where $\mu_\beta$ is absolutely continuous with respect to the Lebesgue measure and $m_1$ is the $(\frac{1}{2}, \frac{1}{2})$ Bernoulli measure on $\{0, 1\}^\mathbb{N}$.

Bernoulli map, which corresponds to the case $\beta = 2$ in $\beta$ encoder, is unstable in the sense that small perturbation of the map makes the orbit $x_n$ to diverge to $+\infty$ or $-\infty$, as discussed in Section 2.1. This phenomenon has been mentioned in [2, 4, 5]. Thus, $\beta = 2$ cannot be used for an RNG and $\beta$ smaller than 2 but close to 2 is recommended. 1.5 bit A/D converter is a strong candidate for an RNG since it is robust to a perturbation of thresholds in quantizers and thus instability issue of Bernoulli map is solved in 1.5 bit A/D converter. The advantage of $\beta$ encoder over 1.5 bit A/D converter is that $\beta$ encoders can employ very small capacitors with relatively large manufacturing error in capacitance and low gain amplifiers because $\beta$ encoder allows a perturbation of the value...
of $\beta$. Moreover, the effective $\beta$ is estimated by a simple algorithm [12]. These characteristics of $\beta$ encoder enable us to implement a $\beta$ encoder in a very small CMOS circuit [19–21]. This is the main advantage of $\beta$ encoder over 1.5 bit A/D converter. On contrary, slopes of three linear regions of an 1.5 bit A/D converter must be maintained to be 2. This requires a relatively large capacitors with small manufacturing error and high gain amplifiers.

For almost all initial values $x_0 \in [0, s)$, $x_n$ generated by Eq. (22) does not fall into periodic points but stays within a range. Attractors are observed in the dynamics of $\beta$ encoders and are referred to as $\beta$ expansion attractors [23]. Hirata et.al considered that such attractors are used as sources of randomness for generating sequences of random numbers [24]. It was reported that consecutive outputs from a $\beta$ encoder have a strong negative correlation [23, 26], which implies that some additional process is needed to generate sequences whose distribution is close to that of i. i. d. random variables.

3. Random Number Generation using $\beta$ encoders

The interval algorithm for converting $\beta$-ary code to binary code is explained in this section. The proposed random number generator uses a hardware $\beta$ encoder as an entropy source and uses the interval algorithm as a post-processor. A review of the fundamental limits of random number generation using interval algorithm is also presented.

3.1 The proposed method

The proposed method consists of two parts (See Fig. 1). The first part is a hardware $\beta$ encoder that generates a $\beta$ expansion of input voltage $x$, denoted by $a^m = a_1 a_2 \cdots a_m$. The second part is a post-processor that calculates the binary expansion $b^n = b_1 b_2 \cdots b_n$ of the interval identified by $a^m$, denoted by $I(a^m)$. The definition of the binary expansion of an interval is given later. For a fixed $n$, we continue obtaining $a_1 a_2 \cdots$ until $b^n = b_1 b_2 \cdots b_n$ are generated. The performance of the generated sequence is examined by the NIST statistical test suite. The NIST test suite can not be used for claiming that the generated sequence is truly random. We can only say that the generated sequence is not random if it fails the tests. Performance analysis of the proposed method in terms of variational distance as well as the expected length of input sequence per an output symbol will be reported in [28].

We recently proposed a method for converting a binary sequence generated from a $\beta$ encoder to another binary sequence that is approximately regarded as i.i.d. random variables [49, 50]. We hereafter assume $s = 1$ for simplicity.

Definition 4 (Finite binary expansion of an interval) : The finite binary expansion of an interval $[\ell, u) \subset [0, 1)$ is $b_1 b_2 \cdots b_n \in \{0, 1\}^n$ satisfying

$$\sum_{i=1}^{n} b_i 2^{-i} \leq \ell, \quad u < \sum_{i=1}^{n} b_i 2^{-i} + 2^{-n}, \quad (29)$$

where $n$ is the maximum integer for which the above inequality holds.

Let $x \in I_0 = [0, 1)$. Express $x$ as

$$x = \sum_{i=1}^{\infty} b_i 2^{-i} = (\beta - 1) \sum_{i=1}^{\infty} a_i \beta^{-i}, \quad (30)$$

where $a_i, b_i \in \{0, 1\}$. For given first $m$ $\beta$-expansion coefficients, $a^m = a_1 a_2 \cdots a_m$, $a_i \in \{0, 1\}$, $m \geq 1$, defines its associated interval as

$$I(a^m) = [\hat{x}^{(\beta)}(a^m), \hat{x}^{(\beta)}(a^m) + \beta^{-m}], \quad (31)$$

where $\hat{x}^{(\beta)}(a^m) = (\beta - 1) \sum_{i=1}^{m} a_i \beta^{-i}$.

Let $\gamma = \beta^{-1}$. Note that $I(a^m)$ can be determined iteratively by
\[ I(a^{i+1}) = \begin{cases} (l_i, l_i + (u_i - l_i) \gamma i), & \text{if } a_{i+1} = 0, \\ (u_i - (u_i - l_i) \gamma i, u_i), & \text{if } a_{i+1} = 1. \end{cases} \] # (32)

with \([l_0, u_0) = [0, 1)\).

**Lemma 2** Let \(a^m\) be the first \(m\) outputs from a scale-adjusted \(\beta\) encoder with an input value \(x\) and with an unknown \(\nu \in [s(\beta - 1), s]\). Then, we have \(x \in I(a^m)\) for all \(m = 1, 2, \ldots\).

Determine \(b^j = b_1b_2 \cdots b_j \ (j \geq 0)\) for which the following hold

\[ I(a^m) \subseteq J(b^j), \quad I(a^m) \not\subseteq J(b^0), \quad I(a^m) \not\subseteq J(b^1), \] # (33)

where \(b^0\) and \(b^1\) denote the sequences \(b_1b_2 \cdots b_j\) 0 and \(b_1b_2 \cdots b_j\) 1 and

\[ J(b^j) = [\hat{x}_{(2)}(b^j), \hat{x}_{(2)}(b^j) + 2^{-j}), \] # (34)

where \(\hat{x}_{(2)}(b^j) = \sum_{k=1}^{j} b_k 2^{-k}\).

Figure 5 shows an example of subdivisions \(I(a^m)\) and \(J(b^n)\) of the unit interval. The unit interval \([0, 1)\) is divided into two subintervals \(I(0) = [0, \gamma)\) and \(I(1) = (1 - \gamma, 1)\). Then, \(I(0) = [0, \gamma)\) is divided into its two subintervals \(I(00) = [0, \gamma^2)\) and \(I(01) = [\gamma - \gamma^2, \gamma)\) and so on. The output for \(a_1a_2a_3a_4 = 0001\) with \(m = 4\) is \(b_1 = 0\) and \(b_2 = 0\) with \(n = 2\) since we have \(I(0001) \subset J(00), \ I(0001) \not\subset J(00)\) and \(I(0001) \not\subset J(001)\).

![Fig. 5. Overlapped subintervals I(a^m) for a \(\beta\) expansion with an unknown threshold and disjoint subintervals J(b^n) for binary expansion.](image)

We can make a decision on the first bit \(b_1 = 0\) or 1 immediately after the interval \(I(a^m)\) becomes a subset of \([0, 1/2)\) or \([1/2, 1)\). The decision on the second bit \(b_2\) can be done in the same manner.

We give the following algorithm for generating the binary expansion of \(x\) from the outputs from the scale-adjusted \(\beta\) encoder.

[**Method 1**]

1. Initialize: \(i = j = 1, (\ell, u) = [0, 1), \gamma = \frac{1}{\beta}\).

2. Read \(a_i\).

   If \(a_i = 0\), then \(u\) is updated to \(\ell + (u - \ell) \times \gamma\).

   If \(a_i = 1\), then \(\ell\) is updated to \(u - (u - \ell) \times \gamma\).

3. If \(u < \frac{1}{2}\), then output \(b_j = 0\) and update \(j = j + 1, \ell = 2\ell,\) and \(u = 2u\).

   If \(\ell \geq \frac{1}{2}\) then output \(b_j = 1\) and update \(j = j + 1, \ell = 2\ell - 1,\) and \(u = 2u - 1\).

4. If \(j = n\), then quit. Otherwise, update \(i = i + 1\) and go back to Step 2.
The number of outputs depends on $a^n$, as shown in Fig. 5. However, after we obtain the first $m$ outputs from the $\beta$ encoder, the width of interval $I(a^n)$ is $\beta^{-n}$ irrespective of $a^n$. This fact together with the condition $I(a^n) \subset J(b^n)$ implies that the number of outputs from the converter satisfies $n \leq m \log_2 \beta$. A theoretical analysis of the expected length of the input sequence of the proposed method remains to be studied.

**Remark 2** The threshold value $\nu$ is not needed for the proposed method. Overlapped subintervals $I(a^n)$ can accommodate a fluctuation of $\nu \in [(\beta - 1), 1]$. Hence, the proposed method is robust to the offset of $\nu$.

Suppose that we have already obtained $b_1 b_2 \ldots b_{j-1}$ ($j \leq n$). The current interval $[\ell, u)$ in Method 1 may become very small during the update. This situation happens if the input value $x$ subtracted by $\sum_{k=1}^{j-1} b_k 2^{-k}$ is close to 1/2. For example, the interval $I(011)$ in Fig. 5 contains 1/2 and thus we cannot output $b_j$ until the lower bound of $I(0111a_5a_6 \ldots)$ becomes greater than 1/2 or the upper bound of $I(0111a_5a_6 \ldots)$ becomes smaller than 1/2. In this case, the next output sequence is $b_j b_{j+1} \ldots = 0 \cdots 01$ or $1 \cdots 0$. When we express $\ell$ and $u$ as integers, this situation makes the approximation error very large. In Method 2 below, we avoid such a situation by doubling the size of $|u - \ell|$ without making decision $b_j \in \{0, 1\}$. This results in $|u - \ell|$ always being greater than $\frac{1}{4}$. A new parameter $k$ expresses the number of undecided output bits.

[ **Method 2** ]

1. Initialize: $i = j = 1$, $k = 0$, $[\ell, u) = [0, 1)$, and $\gamma = \frac{1}{3}$.
2. The same as Step 2. in Method 1.
3. (a) If $\frac{1}{4} \leq \ell < \frac{1}{2}$ and $\frac{1}{2} \leq u < \frac{3}{4}$, then update $\ell = 2\ell - \frac{1}{2}, u = 2u - \frac{1}{2}$, and $k = k + 1$.
   (b) If $u < \frac{1}{2}$, then output $b_j = b_{j+1} = \cdots = b_{j+k-1} = 1, b_{j+k} = 0$. (If $k = 0$, then output $b_j = 0$) and update $k = 0, \ell = 2\ell, u = 2u$, and $j = j + k + 1$.
   (c) 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 $k = 0, \ell = 2\ell - 1, u = 2u - 1$, and $j = j + k + 1$.
4. If $j = m$, then quit. Otherwise, update $i = i + 1$ and go back to Step 2.

It may be worthwhile to mention that the update rule of $u = 2u - \frac{1}{2}$ and $\ell = 2\ell - \frac{1}{2}$ in 3. (a) is similar to the dynamics of the 1.5 bit quantizer [39].

Finally, we give Method 3, which is an integer arithmetic version of Method 2. A real number $r \in \left[\frac{1}{2w}, \frac{1}{2w+1}\right)$ is approximated by $\frac{r}{2^w}$, where $w$ is referred to as the word length. Real numbers $\ell$ 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.

Uyematsu and Li [29] proposed two algorithms for random number generation using finite precision arithmetic and analyzed their performances. The first algorithm generates an i.i.d. sequence of random numbers with arbitrarily given distribution using an input sequence of random numbers with an i.i.d. uniform distribution. The second algorithm generates a sequence of random numbers with uniform distribution using a sequence from a given i.i.d. source. The way to partition the interval of integers is based on the Jones method [51]. They proved that the variational distance between the desired distribution and the distribution of the output sequence is upper bounded by $C \cdot 2^{-w}$, where $C$ is a constant and $w$ is the word length. This implies that the approximation error vanishes as the word length goes to infinity. Moreover, they gave upper and lower bounds for the expected length of the input sequence.

Method 3 below is similar to the second algorithm of the Uyematsu and Li method [29]. The main difference is that the subintervals for expressing 0 and 1 are overlapping in the proposed method.

[ **Method 3** ]

1. Initialize: $i = j = 1$, $k = 0$, $l' = 0, u' = 2^w - 1$, and $\gamma = \left\lfloor \frac{2^w}{\beta} \right\rfloor$. 

48
2. Read $a_i$.
   If $a_i = 0$, then update
   \[ u' = l' + \left(\frac{(u' - l') \cdot \gamma}{2^w} + \frac{1}{2}\right) \]  
   If $a_i = 1$, then update
   \[ l' = u' - \left(\frac{(u' - l') \cdot \gamma}{2^w} + \frac{1}{2}\right) \]  

3. (a) If $\frac{1}{2} \leq \frac{l'}{2^w} < \frac{1}{2}$ and $\frac{1}{2} \leq \frac{u'}{2^w} < \frac{3}{4}$, then update $l' = 2l' - 2^{w-1}$, $u' = 2u' - 2^{w-1}$, and $k = k + 1$.
   (b) If $\frac{u'}{2^w} < \frac{1}{2}$, then output $b_j^{+k} = 1$...10 and update $l' = 2l'$, $u' = 2u'$, $j = j + k + 1$, and $k = 0$.
   (c) If $\frac{l'}{2^w} \geq \frac{1}{2}$, then output $b_j^{+k} = 0$...01 and update $l' = 2l' - 2^w$, $u' = 2u' - 2^w$, $j = j + k + 1$ and $k = 0$.

4. If $j = m$, then quit. Otherwise, update $i = i + 1$ and go back to Step 2.

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 $k \in \{0, 1, \ldots\}$. Since the interval $[\frac{l'}{2^w}, \frac{u'}{2^w}]$ is an approximation of $[\ell, u]$ in Method 2, the distribution of the generated sequence is deviated from the target distribution. The effect of the word length $w$ is examined by the experiments shown in the next section.

**Remark 3** The subintervals in Uyematsu and Li’s algorithm are defined so that the inclusion relationship between intervals is maintained. Thus, the sequence generated by their second algorithm for any input sequence is the same as the sequence generated by the original interval algorithm. In the proposed method, the subintervals overlap. Hence, the inclusion relationship is not maintained.

### 3.2 Fundamental Limits of the Interval Algorithm

When raw random numbers from a physical RNG have bias, i.e., imbalance of the rate of occurrence of 0s and 1s, or have correlations among the successive bits, post-processing is applied to improve the randomness of the random numbers [3, 4, 6–8]. The problem of converting from a sequence of numbers generated from a random process with known or unknown probability distribution to another sequence of random numbers with a prescribed target distribution is known as a random number generation problem in information theory [44]. The origin of the random number generation problem dates back to von Neumann [45]. Knuth and Yao [46] have investigated the problem of random number generation for simulating an arbitrary target distribution by successive tosses of an unbiased coin where the target random numbers should be generated exactly according to the prescribed distribution. A theoretical framework of random number generation was established by Han and Verdú [47] and its extensions were studied by Vembu and Verdú [48]. In these two works, asymptotic approximation problems are considered in which the target random numbers are generated approximately within an arbitrarily small tolerance in terms of variational distance or normalized divergence distance.

Han and Hoshi [27] proposed a simple deterministic algorithm, called the interval algorithm. They studied the method of generating target random sequences of fixed length from a prescribed information source by use of random coin sequences of variable length from a given information source. The interval algorithm was originally developed for arithmetic coding in data compression. They gave a tight upper bound on the expected number of tosses. Let $A = \{0, 1, \ldots, |A| - 1\}$ and $B = \{0, 1, \ldots, |B| - 1\}$ be finite sets. Let $p = \{p(a)\}$, and $q = \{q(b)\}$ be probability distributions on $A$ and $B$. Let $L$ be a random variable denoting the number of tosses required to simulate a random number with probability $q$. Han and Hoshi [27] gave the following theorem:

**Theorem 3 (Han and Hoshi [27])** Let $E(L)$ be the expected number of tosses of an $|A|$-sided coin with probability $p$ required to simulate a random number with probability $q$. Then, the following inequality holds.
\[
\frac{H(q)}{H(p)} \leq E(L) \leq \frac{H(q)}{H(p)} + \log(2(|A| - 1)) + \frac{h(p_{\text{max}})}{(1 - p_{\text{max}})H(p)}, \tag{37}
\]

where \( p_{\text{max}} = \max_{a \in A} p(a) \) and \( h(p) \) is the binary entropy.

It was also shown in [27] that the interval algorithm can also be used for generating a random process. Let us consider the generation of an i.i.d. random sequence \( Y^n = (Y_1, \ldots, Y_n) \) of length \( n \) subject to the distribution \( q \). Then, the upper bound is replaced by [27]

\[
E(L) \leq n\frac{H(q)}{H(p)} + \log(2(|A| - 1)) + \frac{h(p_{\text{max}})}{(1 - p_{\text{max}})H(p)}. \tag{38}
\]

Studies of random number generation are found in the works of Uyematsu and Kanaya [31], Uyematsu and Li [29], Oohama [32] and Fujisaki [38]. Oohama [32] derived an explicit form of the generating tree describing the interval algorithm for Han and Hoshi’s random number generation and gave a tighter upper bound on the expected number of coin tosses in the interval algorithm than Han and Hoshi’s bound. Fujisaki [38] gave upper and lower bounds for the expected number of coin tosses when the input and the target random variables follow uniform distributions. In these works, the distribution of a biased coin is assumed to be known. In [30] and [36], the problem of generating random numbers with a prescribed distribution by use of a biased coin with an unknown distribution was studied.

The interval algorithm is based on the successive refinement of partitions of the unit interval \([0, 1]\). Thus, the required precision for partitioning the unit interval becomes higher as the lengths of the input and output sequences get longer. This implies that the interval algorithm requires unlimited precision of the arithmetic. In [29, 30], implementation of the interval algorithm by use of integer arithmetic with limited precision was provided, where partitions of the unit interval are determined by the same method developed by Jones [51] for the arithmetic code. Uyematsu and Li gave the performance analysis in terms of the variational distance between the target distribution and the distribution of the output sequence as well as the expected length of the input sequence per one output symbol [29].

4. Results of experiment

Experiments were carried out to show the validity of the proposed method. San et al. have manufactured CMOS circuits in which \( \beta \) encoders are embedded [19–21]. The parameter \( \beta \) of the \( \beta \) encoder is designed to satisfy \( 1.7869 \leq \beta \leq 1.9000 \) [21]. Its effective value is not known precisely beforehand.

Using the \( \beta \) encoder, we generated 125 sequences. A constant voltage source is connected to the input port of the \( \beta \) encoder. We discard the first several outputs from the \( \beta \) encoder and obtain the following output sequence denoted by \( a_1, a_2, \ldots \), to which Method 3 is applied. The effective value of \( \beta \) is not known, but we set \( \beta = 1.8 \) for the \( \beta \)-ary/binary conversion. The threshold value \( \nu \) is also not known, but Method 3 does not need the knowledge for \( \nu \). When the length of the output sequence from the \( \beta \)-ary/binary converter reaches \( n = 8 \times 10^5 \), we are finished with the conversion. The word length is \( w = 20 \) unless otherwise specified.

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

### Table I. Voltage Parameters.

| \( V_{\text{DDA}} \) | \( V_{\text{DDD}} \) | \( V_{\text{DDJIO}} \) | \( V_{\text{ref}+} \) | \( V_{\text{ref}-} \) | \( V_{\text{ref,CM}} \) | \( A_{\text{in}+} \) | \( A_{\text{in}-} \) |
|---|---|---|---|---|---|---|---|
| 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 |
Remark 4 For every trial to generate a sequence of random numbers, the input value to the \( \beta \) encoder is fixed to \( x = \frac{A_{in+} - A_{in-}}{V_{ref+} - V_{ref-}} \). If \( x \) for each trial and the threshold \( \nu \) in the quantizer \( Q_\nu \) are the same, the generated sequence \( a^m \) should be the same. However, the value of \( \nu \) can vary at every application and \( x \) contains small fluctuation because \( V_{ref+}, V_{ref-}, A_{in+} \) and \( A_{in-} \) are realized at an analog level. Moreover, we ignore the first several outputs from the \( \beta \) encoder. This situation makes the generated sequence different for each trial.

Remark 5 Randomness of the sequences from a \( \beta \) encoder is theoretically guaranteed by the chaotic behavior of the attractors observed in the \( \beta \) encoder [23]. Hence, basically, thermal noise is not needed for \( \beta \) encoders to generate random numbers. However, the hardware \( \beta \) encoder is an analog circuit and is not free from background noise. Such background noise introduces an offset of the threshold. The \( \beta \) encoder is robust to such an offset and thus can work properly under the background noise. Further discussion is needed to describe the behavior of the \( \beta \) encoder with flaky quantizer.

The difference between Pattern A and Pattern B is \( A_{in+} = 0.80 \) and \( A_{in-} = 0.40 \) for the former and \( A_{in+} = 0.60 \) and \( A_{in-} = 0.60 \) 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 \( V_{DDA} = 1.20 \) for the former and \( V_{DDA} = 1.40 \) for the latter. In general, a CMOS circuit has a range of \( V_{DDA} \) within which the circuit can properly work. The standard analog drive voltage for the \( \beta \) encoders is designed to be \( V_{DDA} = 1.20 \), but the circuit becomes more stable when \( V_{DDA} = 1.40 \).

The NIST test suite [52] was applied to sequences obtained by \( \beta \)-ary/binary conversions. There are fifteen tests for the NIST test suite. The result of each test is expressed as P (Pass) or F (Fail) except for the non-overlapping template matching test for which the number of templates among 148 templates that pass the test is shown. It is known that the non-overlapping template matching test is the most difficult test among the 15 NIST tests [53].

Table II shows the results of the NIST test suite for the sequences obtained by \( \beta \)-ary/binary conversion. The table shows that the generated sequences with voltage parameter C pass all the tests and that with parameter A and B the sequences pass all the tests except for the non-overlapping template matching test. Therefore, we recommend parameter C. However, only one test for A and one for B did not pass the test. These results are also fairly good.

Table II. Results of the NIST statistical test suite for \( \beta \)-ary/binary conversion.

| Voltage Pattern | A | B | C |
|-----------------|---|---|---|
| Frequency (Monobits) Test | P | P | P |
| Frequency Test within a Block | P | P | P |
| Cumulative Sum (Cusum) Test | P | P | P |
| Runs Test | P | P | P |
| Test for the Longest Run of Ones in a Block | P | P | P |
| Binary Matrix Rank Test | P | P | P |
| Discrete Fourier Transform Test | P | P | P |
| Non-Overlapping Template Matching Test | 147 | 147 | P |
| Overlapping Template Matching Test | P | P | P |
| Maurer’s "Universal Statistical" Test | P | P | P |
| Approximate Entropy Test | P | P | P |
| Random Excursion Test | P | P | P |
| Random Excursions Variant Test | P | P | P |
| Serial Test | P | P | P |
| Linear Complexity Test | P | P | P |

Table III shows the effect of window size \( w \) on the results of the 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 \( \beta \) is not known beforehand, we verified the robustness of the proposed method to the mismatch of the values of \( \beta \) used in the encoder and the \( \beta \)-ary/binary converter. The value of \( \beta \) is designed to satisfy \( 1.7869 \leq \beta \leq 1.9000 \). Denote the \( \beta \) used in the \( \beta \)-ary/binary converter by \( \beta' \). Table IV 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 allows a fluctuation of 0.1 at most for $\beta$.

5. Conclusion

In this paper, a physical random number generator using a $\beta$ encoder followed by a $\beta$-ary to binary conversion has been proposed. In the $\beta$-ary to binary conversion method, an interval in which the input value $x \in [0,1)$ should exist is iteratively calculated by use of the outputs from the $\beta$ encoder and then the binary expansion of the interval is calculated. The proposed $\beta$-ary to binary conversion is similar to Han and Hoshi’s interval algorithm. The most significant difference between Han and Hoshi’s algorithm and the proposed method is that sub-intervals determined by the former are disjoint, while sub-intervals of the latter method are overlapped. The computational cost of such a $\beta$-ary to binary conversion is considerably higher than that of post-processings using EXOR operations. However, in case of the interval algorithm, it is possible to evaluate the variational distance between the distribution of generated random numbers and the ideal distribution. Evaluation of the distribution of the generated bits in terms of the conversion rate and the variational distance is reported in [28].

Experimental results have shown that sequences obtained by $\beta$-ary to 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 that the converter is robust to mismatches of $\beta$.

| Table III. Results of the NIST statistical test suite: Comparison of window size. |
|---------------------------------------------|---|---|---|---|---|---|
| Voltage Pattern                           | A | B | C |
| The number of EXORs                      | 10 | 15 | 20 | 10 | 15 | 20 |
| Frequency (Monsobis) Test                | F | P | P | P | P | P |
| Frequency Test within a Block            | F | P | P | P | P | P |
| Cumulative Sums (Casum) Test             | F | P | P | P | P | P |
| Runs Test                                | F | P | P | P | P | P |
| Test for the Longest Run of Ones in a Block | P | P | P | P | P | P |
| Binary Matrix Rank Test                  | P | P | P | P | P | P |
| Discrete Fourier Transform Test          | P | F | P | P | P | P |
| Non-Overlapping Template Matching Test   | 96 | P | 147 | 105 | P | 147 |
| Overlapping Template Matching Test       | F | P | P | F | P | P |
| Maurer’s "Universal Statistical" Test    | P | P | P | P | P | P |
| Approximate Entropy Test                 | F | P | P | P | P | P |
| Random Excursion Test                    | F | P | P | P | P | P |
| Random Excursions Variant Test           | P | P | P | P | P | P |
| Serial Test                              | F | P | P | F | P | P |
| Linear Complexity Test                   | P | P | P | P | P | P |

| Table IV. Results of the NIST statistical test suite: Comparison of $\beta$ used in the converter. |
|---------------------------------------------|---|---|---|---|---|---|
| Voltage Pattern                           | A | B | C |
| The number of EXORs                      | 1.6 | 1.7 | 1.8 | 1.9 | 1.6 | 1.7 | 1.8 | 1.9 |
| Frequency (Monsobis) Test                | F | P | P | P | P | P | P | P |
| Frequency Test within a Block            | P | P | P | P | P | P | P | P |
| Cumulative Sums (Casum) Test             | F | P | P | P | P | P | P | P |
| Runs Test                                | F | P | P | P | P | P | P | P |
| Test for the Longest Run of Ones in a Block | P | F | P | P | P | P | P | P |
| Binary Matrix Rank Test                  | P | P | P | P | P | P | P | P |
| Discrete Fourier Transform Test          | P | P | P | P | P | F | P | P |
| Non-Overlapping Template Matching Test   | 142 | P | 147 | P | 137 | 145 | 147 | P |
| Overlapping Template Matching Test       | P | P | P | P | P | P | P | P |
| Maurer’s "Universal Statistical" Test    | P | P | P | P | P | P | P | P |
| Approximate Entropy Test                 | F | P | P | P | P | P | P | P |
| Random Excursion Test                    | P | P | P | P | F | P | P | P |
| Random Excursions Variant Test           | F | P | P | P | P | F | P | P |
| Serial Test                              | P | P | P | P | P | P | P | P |
| Linear Complexity Test                   | P | P | P | P | P | P | P | P |
Acknowledgments

The authors wish to thank Professor Masao Hotta and Professor Hao San for their help in experiments using a hardware beta encoder and also wish to thank Professor Evgeny Verbitskiy, and Professor Kirill Morozov for their helpful comments and discussions. The authors wish to thank anonymous reviewers for their valuable comments and Professor Tohru Kohda for giving us the starting point of this research. This research was supported in part by the Aihara Project, the FIRST program from JSPS, initiated by CSTP and JSPS KAKENHI Grant Number 25820162.

References

[1] K. Yoshimura et al., “Secure key distribution using correlated randomness in lasers driven by common random light,” Phys. Rev. Lett., vol. 198, 070602, 2012.
[2] T. Stojanovskv and L. Kocarev, “Chaos-based random number generators –Part I: analysis,” IEEE Trans. Circ. Syst., I, vol. 48, pp. 281–288, 2001.
[3] T. Stojanovskv, J. Pihl, and L. Kocarev, “Chaos-based random number generators –Part II: practical realization,” IEEE Trans. Circ. Syst., I, vol. 48, pp. 381–385, 2001.
[4] S. Callegari, R. Rozvatti, and C. 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, pp. 793–805, February 2005.
[5] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, “A feedback strategy to improve the entropy of a chaos-based random bit generator,” IEEE Trans. Circ. Syst., I, vol. 53. pp. 326–337, 2006.
[6] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, “Efficient post-processing module for a chaos-based random bit generator,” in Proc. IEEE ICECS 2006, pp. 1224–1227, 2006.
[7] H. Nejati, A. Beirami, and W.H. Ali, “Discrete-time chaotic-map truly random number generators: design, implementation, and variability analysis of the zigzag map,” Analog Integrated Circuits and Signal Processing, vol. 73, no. 1, pp. 363–374, 2012.
[8] A. Beirami and H. Nejati, “A framework for investigating the performance of chaotic-map truly random number generators,” IEEE Trans. Circ. and Syst. II: vol. 60, no. 7, pp. 446–450, 2013.
[9] A. Beirami, H. Nejati, and S. Callegari, “Fundamental performance limits of chaotic-map random number generators,” IEEE 52nd Annual Allerton Conf. on Commun., Control, and Computing, pp. 1126–1131, 2014.
[10] I. Daubechies, R.A. DeVore, C.S. Güntürk, and V.A. Vaishampayan, “Beta expansions: a new approach to digitally corrected A/D conversion,” Proc. IEEE Int. Symp. Circ. Syst. 2002 (ISCAS2002), vol. 2, pp. 784–787, 2002.
[11] 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, pp. 874–885, 2006.
[12] 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.
[13] A. Rényi, “Representations for real numbers and their ergodic properties,” Acta Math. Hungar., vol. 8, pp. 477–493, 1957.
[14] W. Parry, “On the β-expansions of real numbers,” Acta Math. Acad. Sci. Hung., vol. 11, pp. 401–416, 1960.
[15] W. Parry, “Representations for real numbers,” Acta Math. Acad. Sci. Hung., vol. 15, pp. 95–105, 1964.
[16] W. Parry, “Intrinsic Markov chains,” Trans. Amer. Math. Soc., 112, pp. 55–66, 1964.
[17] W. Parry, “Symbolic dynamics and conversions of the unit interval,” Trans. Amer. Math. Soc., 122, pp. 368–378, 1966.
[18] P. Erdös, I. Joó, and V. Komornik, “Characterization of the unique expansions 1 = ∑∞i=1 qni and related problems,” Bull. Soc. Math. France, 118, pp. 377–390, 1990.
[19] 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, February 2013.

[20] R. Sugawara, H. San, K. Aihara, and M. Hotta, “Experimental implementation of nonbinary cyclic ADCs with radix-value estimation algorithm,” IEICE Trans on Electronics, vol. E97-C, no. 4, pp. 308–315, April 2014.

[21] H. San, R. Sugawara, M. Hotta, T. Matsuura, and K. Aihara, “An area-efficient 12-bit 1.25MS/s radix-value self-estimated non-binary ADC with relaxed requirements on analog components,” 2014 IEEE Proc. of Custom Integrated Circuits Conference (CICC), September 2014.

[22] T. Makino, et al., “Rigorous estimates of quantization error for A/D converters based on beta-map,” NOLTA, vol. 6, no. 1, pp. 99–111, January 2015.

[23] 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.

[24] 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), January 2013.

[25] E. Ott, Chaos in dynamical systems, 2nd. Edt., Cambridge university press, 2002.

[26] T. Kohda, Y. Horio, Y. Takahashi, and K. Aihara, “Beta encoders: Symbolic dynamics and electronic implementation,” Int. J. of Bifurcation and Chaos, 22(09), 2012.

[27] T.S. Han and M. Hoshi, “Interval algorithm for random number generation,” IEEE Trans. on Inform. Theory, vol. 43, pp. 599–611, 1997.

[28] Y. Shu, Y. Jitsumatsu, and K. Oda, “Performance evaluation of a random number generation using a beta encoder,” Proc. of Nonlinear Theory and Its Applications 2015 (NOLTA2015), pp. 511–514, December 2015.

[29] T. Uyematsu and Y. Li, “Two algorithms for random number generation implemented by using arithmetic of limited precision,” IEICE Trans. Fundam., vol. E86-A, no. 10, pp. 2542–2551, 2003.

[30] T. Taniguchi and T.S. Han, “Random number generation of adaptive iterative algorithm and its performance,” Proc. the 21st Symposium on Inform. Theory and Its Applications (SITA’98), pp. 61–64, 1998 (in Japanese).

[31] T. Uyematsu and F. Kanaya, “Channel simulation by interval algorithm: A performance analysis for interval algorithm,” IEEE Trans. Inform. Theory, vol. 45, pp. 2121–2129, 1999.

[32] Y. Oohama, “Performance analysis of the interval algorithm for random number generation based on number systems,” IEEE Trans. Inform. Theory, vol. 57, no. 3, pp. 1177–1185, 2011.

[33] T. Kohda, “Information sources using chaotic dynamics,” Proc. IEEE, vol. 90, no. 5, pp. 641–661, 2002.

[34] F. Kanaya, “A chaos model of a stationary discrete memoryless sources and arithmetic coding,” Proc. of 18th Sympo. on Inform. Theory and Its Applications (SITA95), pp. 361–364, 1995. (in Japanese)

[35] F. Kanaya, “A chaos model of finite-order discrete Markov sources and arithmetic coding,” Proc. of 19th Sympo. on Inform. Theory and Its Applications (SITA96), pp. 81–84, 1996. (in Japanese)

[36] Y. Oohama and H. Sugano, “Adaptive random number generation using arithmetic coding and its performance analysis,” IEICE Tech. Rep., vol. IT98-60, pp. 31–36, 1999. (in Japanese)

[37] H. Fujisaki, “Generating stochastic process based on the finitary interval algorithm,” IEICE Trans. Fundam., vol. E91-A, no. 9, pp. 2482–2488, 2008.

[38] H. Fujisaki, “Entropy of the induced transformations associated with the interval algorithm,” NOLTA, vol. 5, no. 2, pp. 127–139, 2014.

[39] A.M. Abo and P.R. Gray, “A 1.5-V, 10-bit, 14.3-MS/s CMOS pipeline analog-to-digital converter,” IEEE J. Solid-State Circ., vol. 34, no. 5, pp. 599–606, 1999.

[40] T. Matsuura, “Technical trend of digitally assisted A/D converters,” IEICE Technical Report, CAS2011-104, vol. 111, no. 377, pp. 103–108, January 2012 (in Japanese).

[41] H. Inose, Y. Yasuda, and J. Murakami, “A telemetering system by code manipulation –ΔΣ
modulation,” *IRE Trans on Space Electronics and Telemetry*, pp. 204–209, September 1962.

[42] K. Dajani and C. Kraaikamp, “Random $\beta$-expansions,” *Ergodic Theory Dynam. Systems*, vol. 23, no. 2, pp. 461–479, 2003.

[43] K. Dajani and M. de Vries, “Invariant densities for random $\beta$-expansions,” *J. Eur. Math. Soc. (JEMS)*, vol. 9, pp. 157–176, 2007.

[44] T.S. Han, *Information-Spectrum Methods in Information Theory*, Springer, 2003.

[45] J. von Neumann, “Various techniques used in connection with random digits,” *Notes by G.E. Forsythe, Appl. Math. Ser.*, vol. 12, pp. 36–38, 1951.

[46] D. Knuth and A. Yao, “The complexity of nonuniform random number generation,” *Algorithm and Complexity, New Directions and results*, pp. 357–428, ed. by J.F. Traub, Academic Press, New York, 1976.

[47] T.S. Han and S. Verdú, “Approximation theory of output statistics,” *IEEE Trans. Inform. Theory*, 41, pp. 1322–1332, 1995.

[48] S. Vembu and S. Verdú, “Generating random bits from an arbitrary source: Fundamental limits,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 63–86, 1996.

[49] K. Matsumura, T. Teraji, K. Oda, and Y. Jitsumatsu, “Random number generation using $\beta$ encoder,” Proc. of the 32nd Sympo. on Cryptography and Information Security (SCIS2015), January 2015 (in Japanese).

[50] Y. Jitsumatsu and K. Matsumura, “A random binary sequence generator baesd on $\beta$ encoders,” Workshop on “$\beta$-transformation and related topics” MI Lecture Note, Inst. of Math. for Industry, Kyushu Univ., vol. 62, pp. 34–40, March 2015.

[51] C.B. Jones, “An efficient coding system for long source sequences,” *IEEE Trans. on Inform. Theory*, vol. IT-27, no. 3, pp. 490–201, May 1968.

[52] A. Rukhin, et al., “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” NIST, Special Publication 800-22, May 2001.

[53] H. Yoshida, T. Murakawa, and S. Kawamura, “Study on testing for randomness of pseudorandom number sequence with NIST SP800-22 rev.1a,” *IEICE Technical Report*, vol. 112, no. 301, NLP2012-78, pp. 13–18, November 2012 (in Japanese).