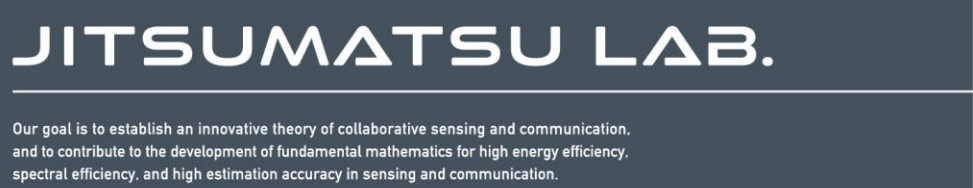
On β-ary to Binary Conversion from an Engineering Point of View

Yutaka Jitsumatsu (Kyushu University, Japan)

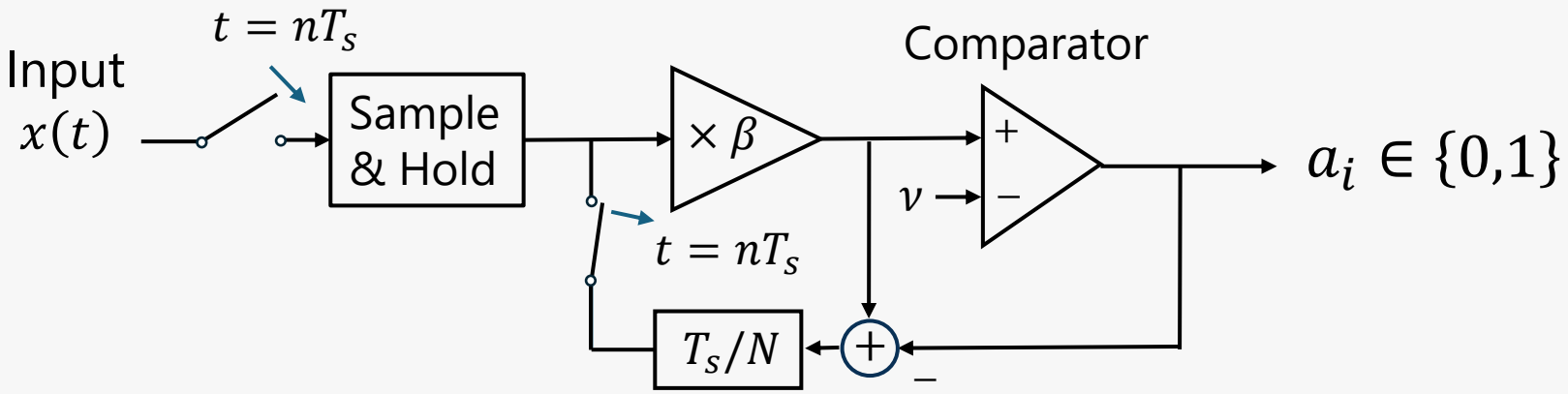


Introduction (1/3)

- The purpose of this talk is to introduce our research on the β encoder, an analog-to-digital converter based on beta transformation.
- My research motivation is an **application of dynamical systems to engineering**.
- I am glad if you are interested in our research.

Introduction (2/3)

❑ A β encoder [DDGV02] is an ADC based on beta transformation.

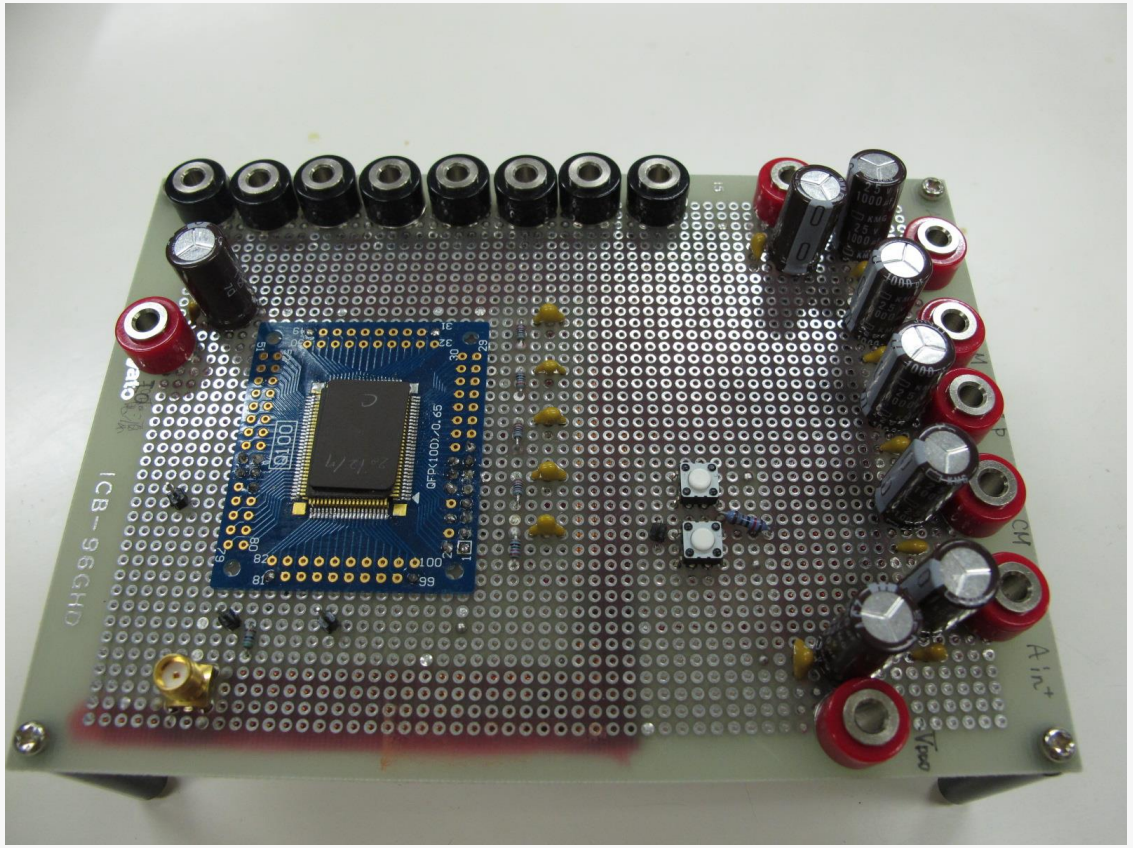


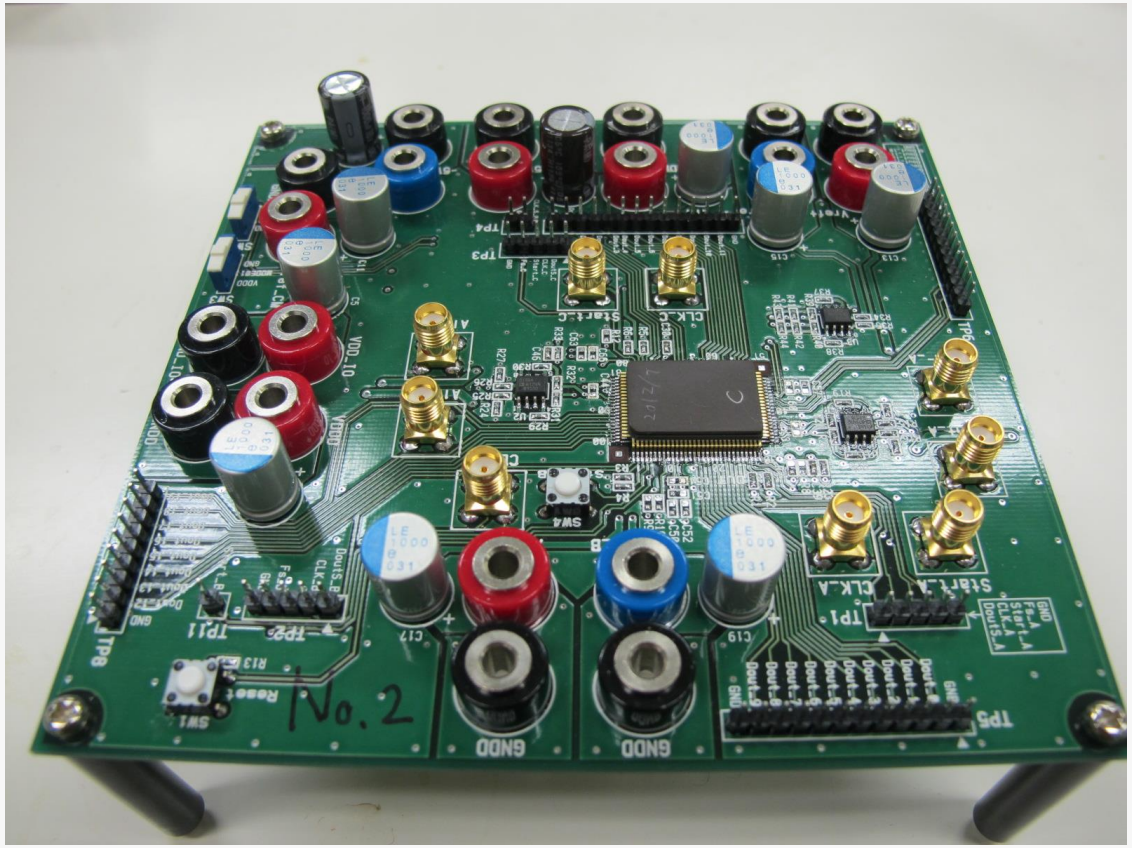
A circuit of beta encoder

$$a_i = Q_\nu(\beta x_{i-1}),$$
$$x_i = \beta x_{i-1} - a_i, x_0 = x,$$
$$Q_\nu(x) = \begin{cases} 0, & x < \nu \\ 1, & x \geq \nu \end{cases}$$

[DDGV02] I. Daubechies; R. DeVore; C.S. Gunturk; V.A. Vaishampayan, "Beta expansions: a new approach to digitally corrected A/D conversion," 2002 IEEE Int. Symp. Circuits and Systems.

Integrated Circuit

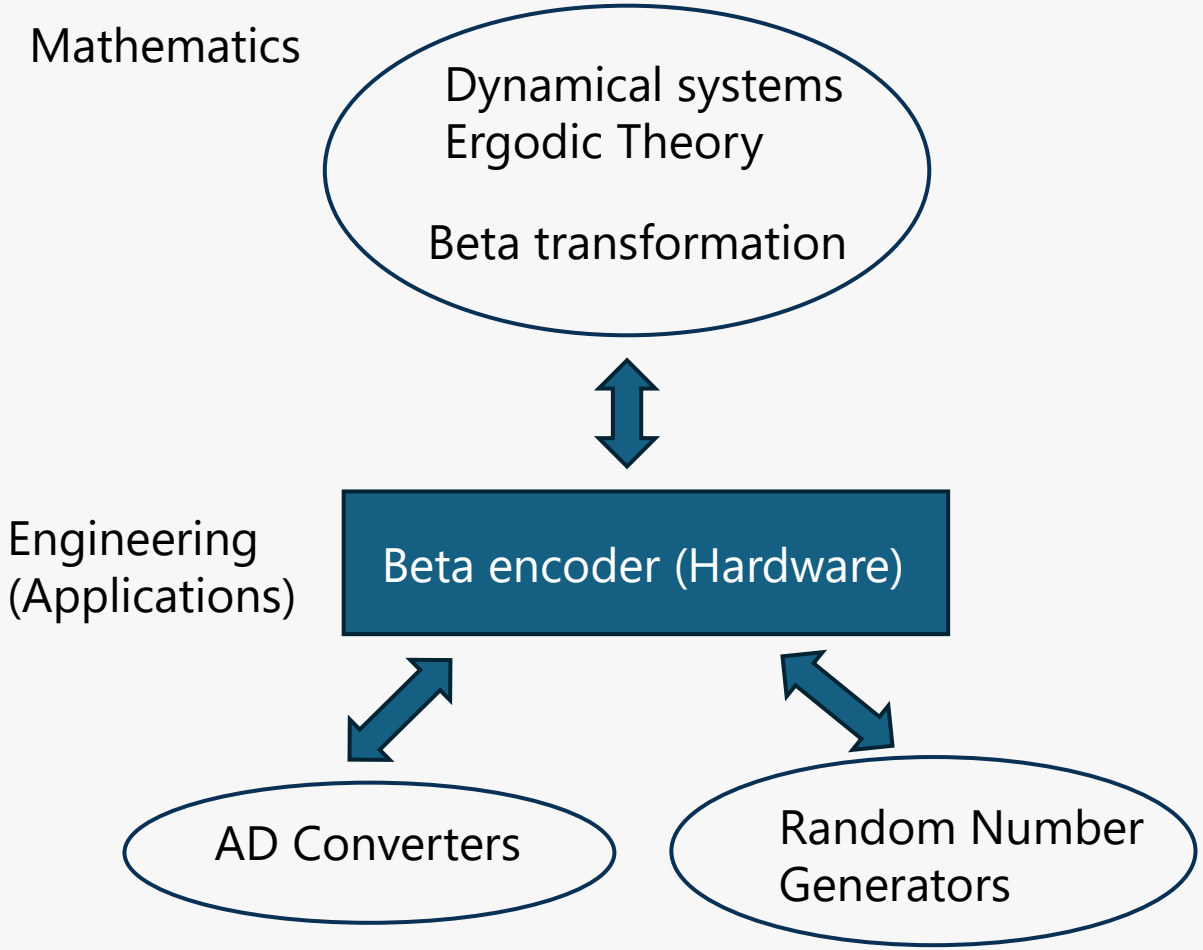




Research Collaborators

- Professor San HAO (Tokyo City University) Expert of analog-to-digital converters

- Professor Katsutoshi SHINOHARA (Hitotsubashi Univ.) Expert of dynamical systems
- I am an engineer in communication theory and information theory.



Introduction (3/3) The purpose of my research

- Can we use the output of beta encoder a_i's as a random number?
  - They have bias and correlations.
- Can we construct a randomness extractor for beta encoders?

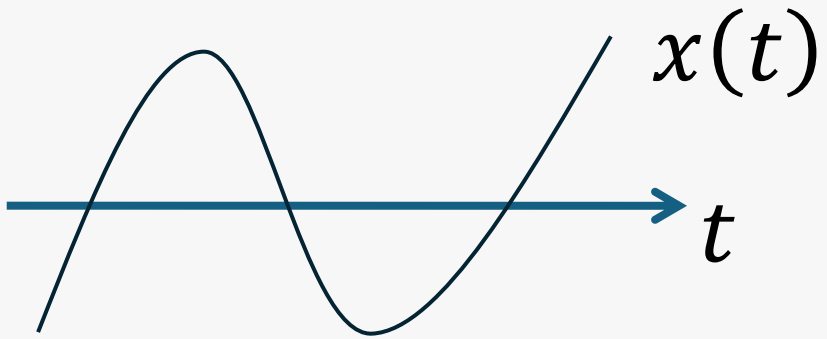
input $a_1 a_2 \ldots a_k$ **non i.i.d.** → **randomness extractor** → output $b_1 b_2 \ldots b_n$ **i.i.d. binary**

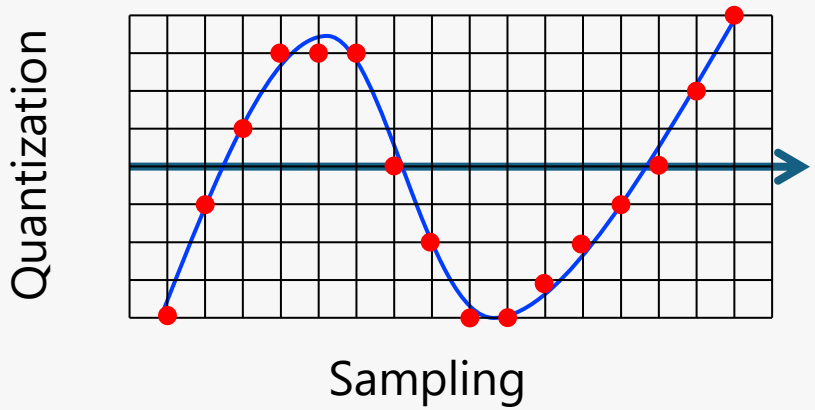
Note: a_i's are physically generated.

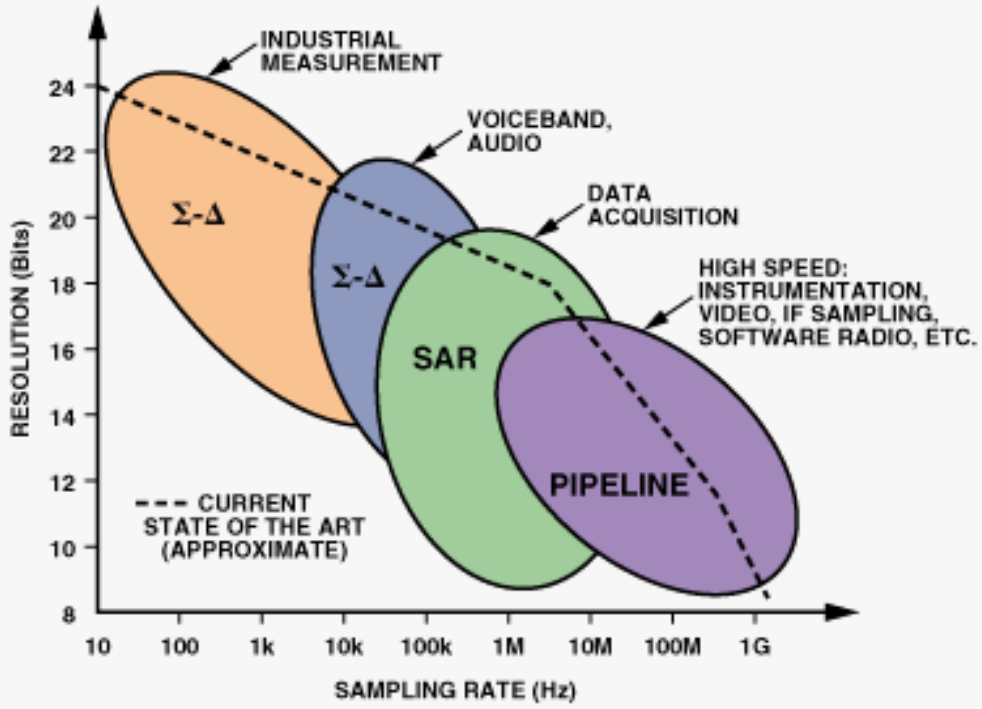
- It is important to find **an appropriate mathematical model** that can properly describe the behavior of the β encoder.

ADCs

- An analog-to-digital converter (ADC) is a device that converts an input analog signal, $x(t)\colon \mathbb{R} \to \mathbb{R}$, into a binary sequence that gives its finite precision approximation.







Performance comparison of ADCs

Binary Expansion

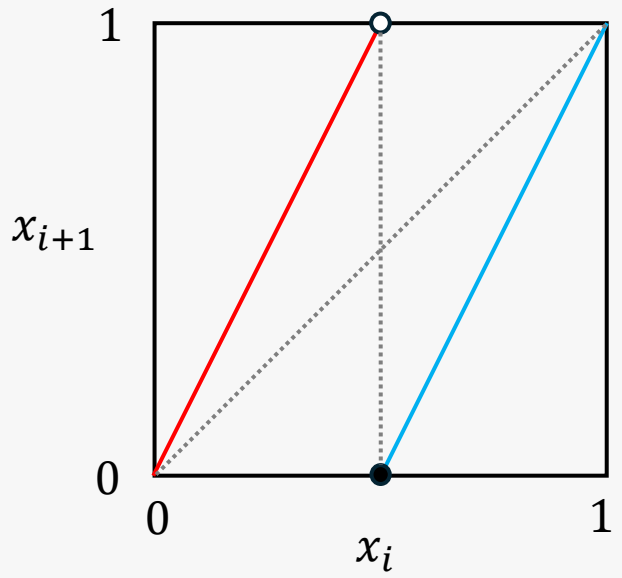
- Consider a binary expansion of a sample $x \in [0,1] \subset \mathbb{R}$

$$x = \sum_{i=1}^{+\infty} b_i 2^{-i}, \qquad b_i \in \{0,1\},$$

- b_i is determined recursively as

$$b_i = Q_1(2x_{i-1}),$$
$$x_i = 2x_{i-1} - b_i, x_0 = x,$$

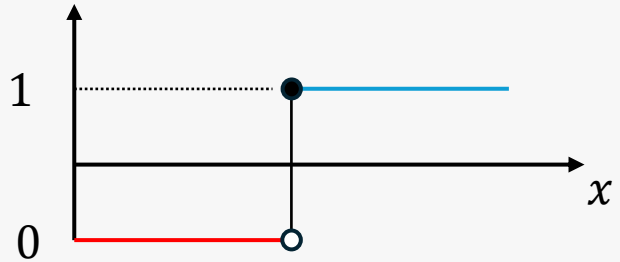
$$Q_1(x) = \begin{cases} 0, & x < 1, \\ 1, & x \geq 1. \end{cases}$$



- The precision of an ADC is finite. N-bit representation of x is $\hat{x}^{(N)} = \sum_{i=1}^{N} b_i 2^{-i}$
- The output binary sequence of the ADC beyond the precision is considered random.

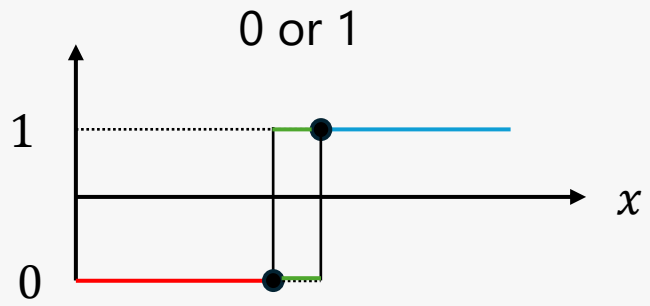
Comparator's precision

- The threshold can only be controlled with some finite precision.
- The threshold is affected by thermal noise.



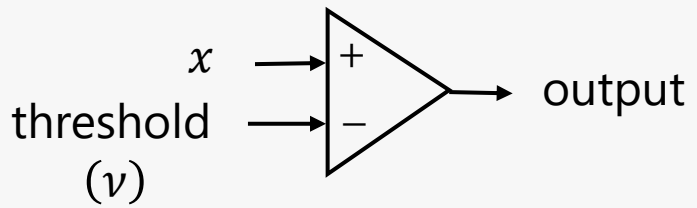
A hard threshold

Input/Output of Ideal comparator



Flaky comparator model

$$\nu_L = \nu - \delta,$$
$$\nu_H = \nu + \delta,$$
$$\delta > 0$$



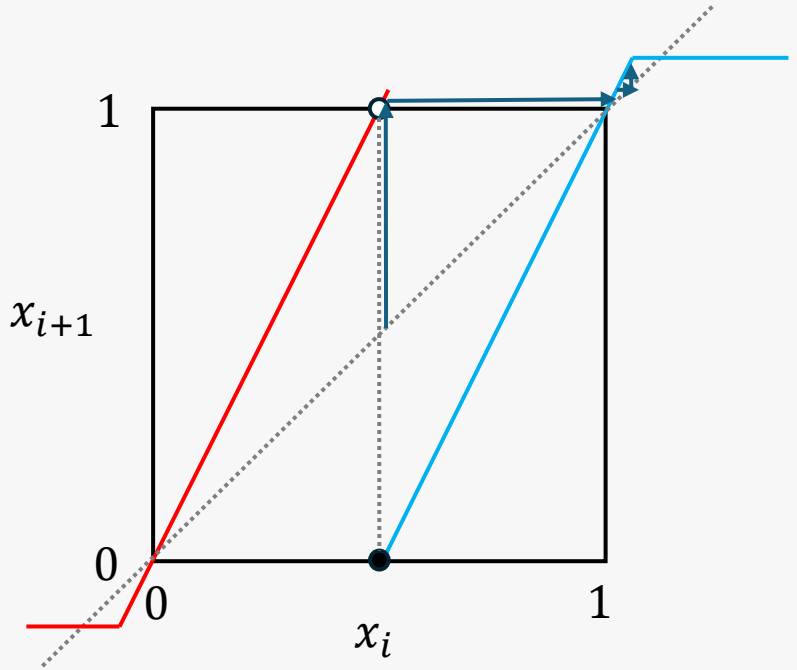
Circuit diagram

$$Q_{\mathrm{f}}(x) = \begin{cases} 0, & x \le \nu_L, \\ 0 \text{ or } 1, & \nu_L < x < \nu_H, \\ 1, & x \ge \nu_H. \end{cases} \qquad 1 < \nu_L < \nu_H < \frac{1}{\beta - 1}$$

Q_{f} is a model of the quantizer such that its output around the threshold is unreliable.

The effect of mismatch of the threshold

- We can only expect that the comparator threshold is guaranteed to be within some interval, e.g., $[\nu_L, \nu_H]$.
- The recursive formula for the binary expansion does not work if the quantizer is imperfect.



The dynamics of binary expansion with imperfect quantizer

The β-encoder

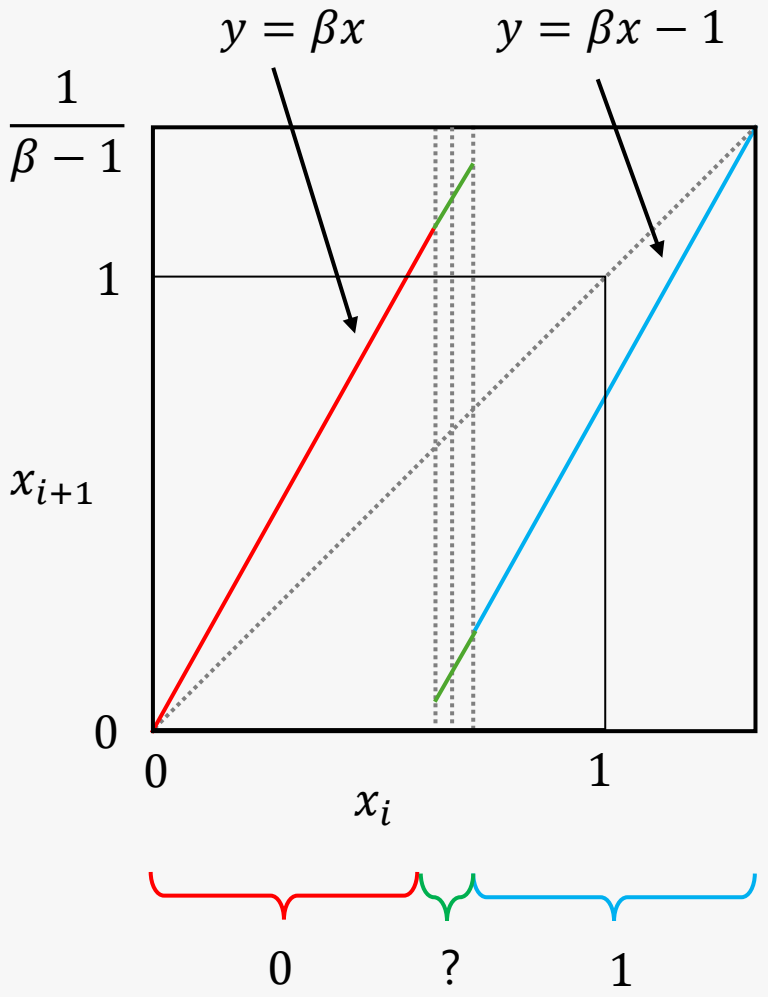
- β-encoder is an ADC that produces a β expansion of x with $1 < \beta < 2$

$$x = \sum_{i=1}^{+\infty} a_i \beta^i, \qquad b_i \in \{0,1\}$$

- a_i is determined recursively as

$$a_i = Q_{\mathrm{f}}(\beta x_{i-1}),$$
$$x_i = \beta x_{i-1} - a, x_0 = x,$$
$$Q_{\mathrm{f}}(x) = \begin{cases} 0, & x \le \nu_L, \\ 0 \text{ or } 1, & \nu_L < x < \nu_H, \\ 1, & x \ge \nu_H. \end{cases}$$

- **Advantage**: x_i does not diverge, if $1 < \nu_L < \nu_H < \frac{1}{\beta - 1}$ is satisfied. **Imperfect quantizer can be used.**



Use of a beta encoder as a random number generator.

- Beta encoder output is biased and correlated.

Input $a_1 a_2 \ldots a_k$ → **randomness extractor** → Output $b_1 b_2 \ldots b_n$

Non i.i.d.

i.i.d. binary

$n < k$

Some basic ideas about randomness extractors

- Von Neumann debiaser algorithm [vN51].
  - HH and TT are discarded
  - HT and TH are converted to 0 and 1.
- Hash function can be a randomness extractor
  - Cryptographic hash function
- We borrow an idea from [HH'97]'s interval algorithm.

[vN51] J. von Neumann "Various techniques used in connection with random digits," Applied Math Series, 12:36–38, 1951.

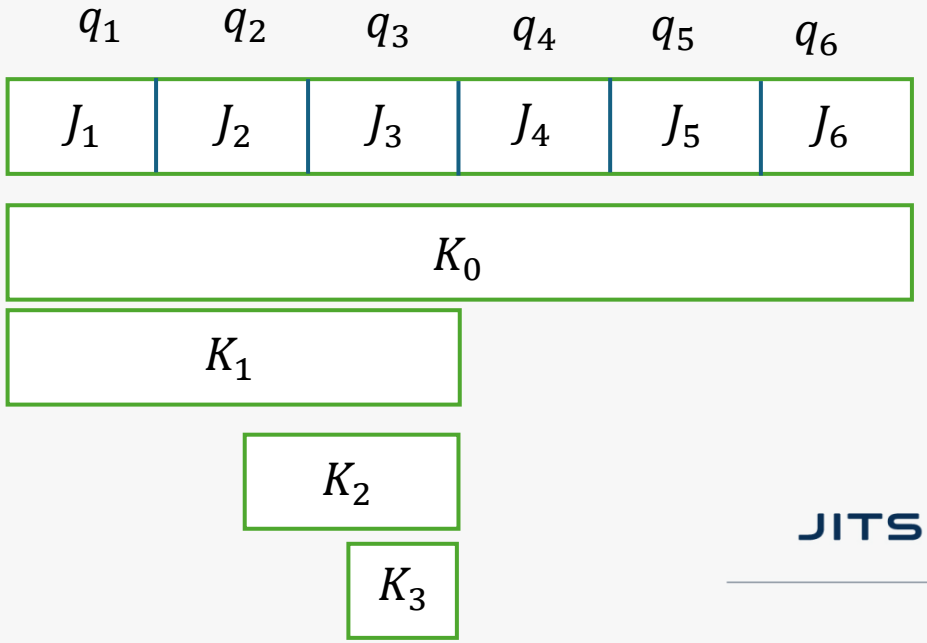
[HH'97]: T. S. Han and M. Hoshi, "Interval algorithm for random number generation," IEEE Trans. Inform. Theory, vol. 43, no. 2, pp. 599–611, March 1997

Han and Hoshi's interval algorithm

- The distributions for "coin" and "target" rvs. p_m and q_n are given.
- The unit interval is partitioned into J_ns according to q_n (See the Fig.)
- Toss a coin many times to simulate one "target" rv.
  1. Set $t = 0$ and $K_0 = [0,1)$;
  2. if $K_t \subset J_n$ for some n, then set the target $X = n$ and quit.
  3. else flip the coin, $k := k + 1$ and update K_t.

Example

- A dice simulated by flips of a fair coin.
- This figure shows the case where the outcome is 011.
- The output is $X = 3$.



Theorem (HH'97): The interval algorithm satisfies

$$\frac{H(q)}{H(p)} \leq L^* \leq \frac{H(q)+f(p)}{H(p)}, f(p) = \ln\big(2(M-1)\big) + \frac{h(p_{\max})}{1-p_{\max}},$$

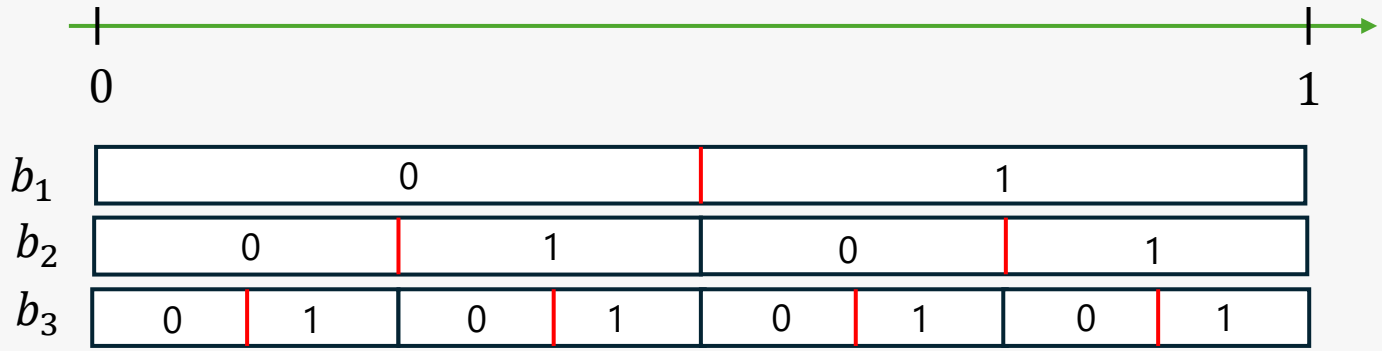
where L^* is the expected number of tosses, $H(p) = \sum_i p_i \log \frac{1}{p_i}$

A randomness extractor for β encoder

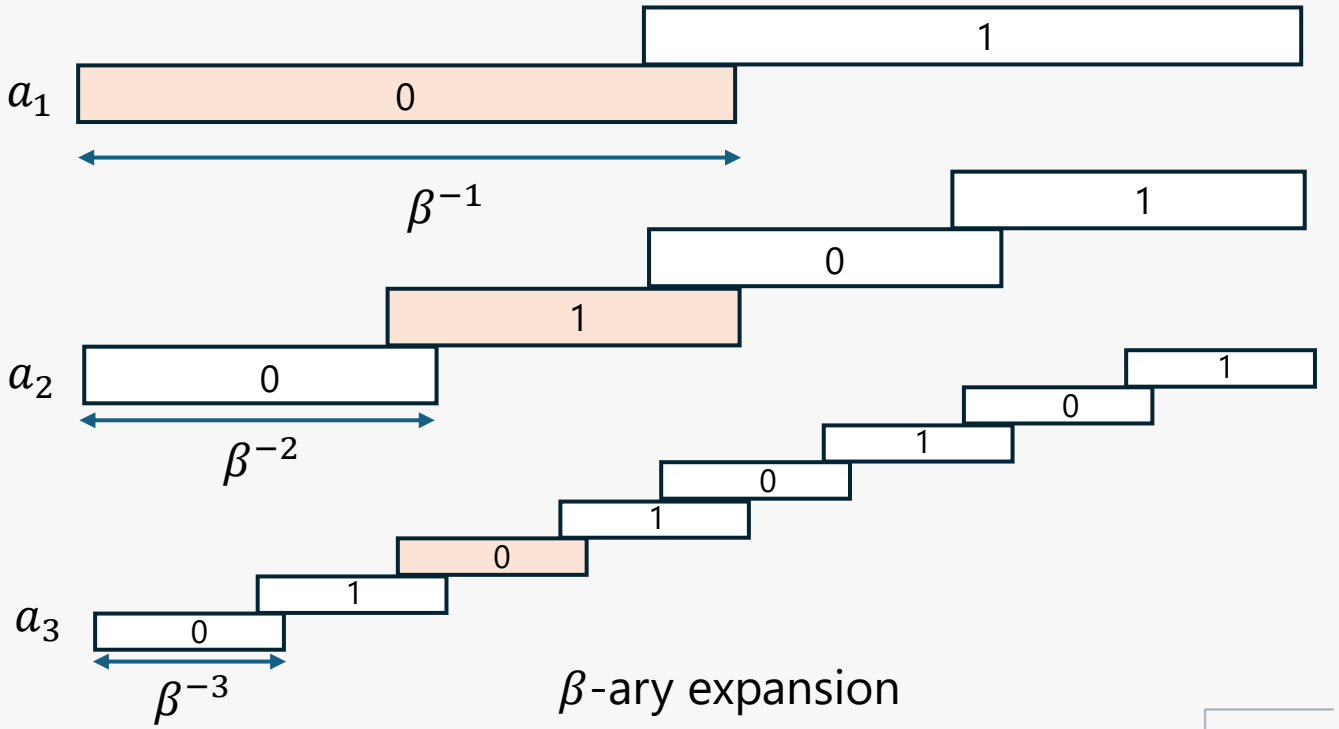
- Let a_i be the sequence of scale-adjusted beta expansion of a x.

$$x = (\beta - 1)\sum_{i=1}^{\infty} a_i \beta^{-i}$$

- Compute the binary expansion b_j of x sequentially from the series a_i.
- **Can we regard b_j as a sequence of i.i.d. random numbers?**
- x is set to some value, e.g. $0.5 \pm \epsilon$, where ϵ is an additive noise.



Binary expansion



β-ary expansion

[JM16] Y. Jitsumatsu, K. Matsumura, "A β-ary to binary conversion for random number generation using a β encoder," Nonlinear Theory and Its Applications, IEICE, 2016, vol.7, no.1, p. 38-55,

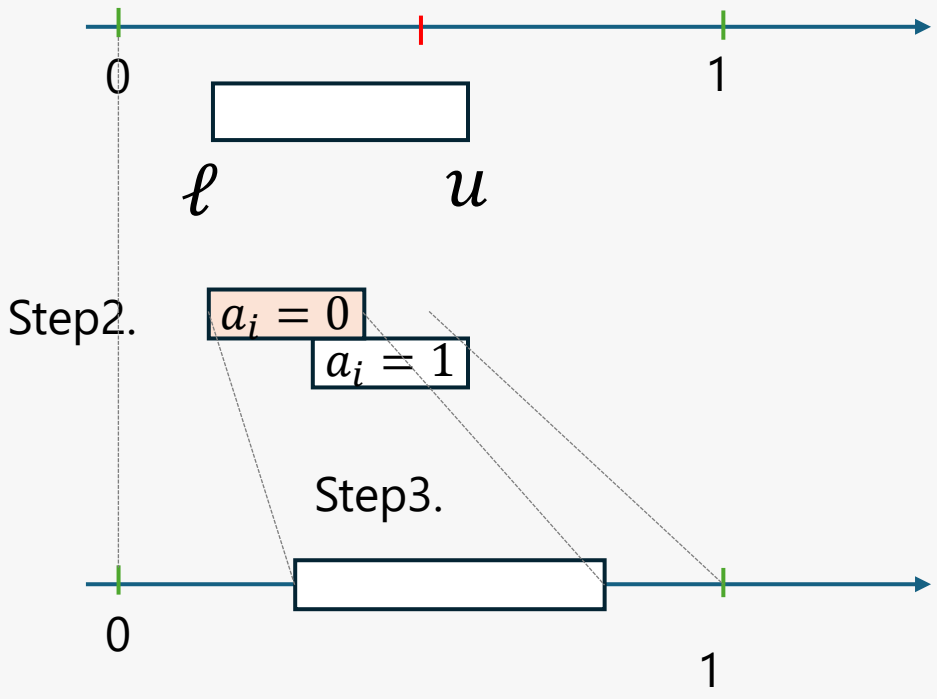
JITSUMATSU LAB.

The β-ary to binary conversion [1]

$a_1, a_2, \ldots, a_i$
$b_1, b_2, \ldots, b_j$

Proposed method: (1st version)

1. Initialize $i = j = 1, \ell = 0, u = 1$, and $\gamma = \beta^{-1}$
2. Read a_i.
   If $a_i = 0$, then $u := \ell + \gamma(u - \ell)$.
   If $a_i = 1$, then $\ell := u - \gamma(u - \ell)$.
3. (a) If $u < \frac{1}{2}$, then output $b_j = 0$ and update $j = j + 1, \ell = 2\ell$ and $u = 2u$
   (b) 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.



Fundamental question on the proposed method

- How many a_is are needed to obtain n bits of b_j?

We see that $\beta^{-k} < 2^{-n}$ holds.→ $k > \dfrac{n}{\log_2 \beta}$

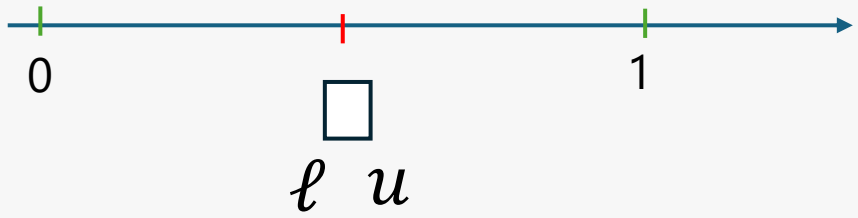
Theorem(C.Kalle, E.Verbitskiy, and B.Zeegers, 2023) Let $k(n, \boldsymbol{\nu}, x)$ be the number of bits of a_i necessary to obtain n bits of b_j. For each $\varepsilon \in (0, 1)$ and $\boldsymbol{\nu} = (\nu_i)_{i \geq 1}$, $\nu_i \in [1, 1/(\beta - 1)]$, there exists a constant $C(\varepsilon)$ such that for all $n \in \mathbb{N}$,

$$\lambda\left(\left\{x \in [0,1] : k(n, \boldsymbol{\nu}, x) - \frac{n}{\log_2 \beta} > C(\varepsilon)\right\}\right) < \varepsilon, \tag{1}$$

where λ is the one-dimensional Lebesgue measure.

→ $\dfrac{k}{n} \approx \dfrac{1}{\log_2 \beta}$

Drawback of Version 1



- The interval $[\ell, u]$ may become very small.
- This happens if ℓ is smaller than ½ but close to ½ and u is greater than ½ but close to ½.
- Version 1 is a prototype where u and ℓ are calculated using real values but should eventually be implemented digitally.
  If $[\ell, u]$ becomes very small, it will lead to a degradation of the calculation accuracy.

- This drawback is overcome in Version 2.

The β-ary to binary conversion [1]

Proposed method (2nd version)

1. Initialize $i = j = 1, \ell = 0, k = 0, u = 1$, and $\gamma = \beta^{-1}$

   k expresses the number of undecided output bits.

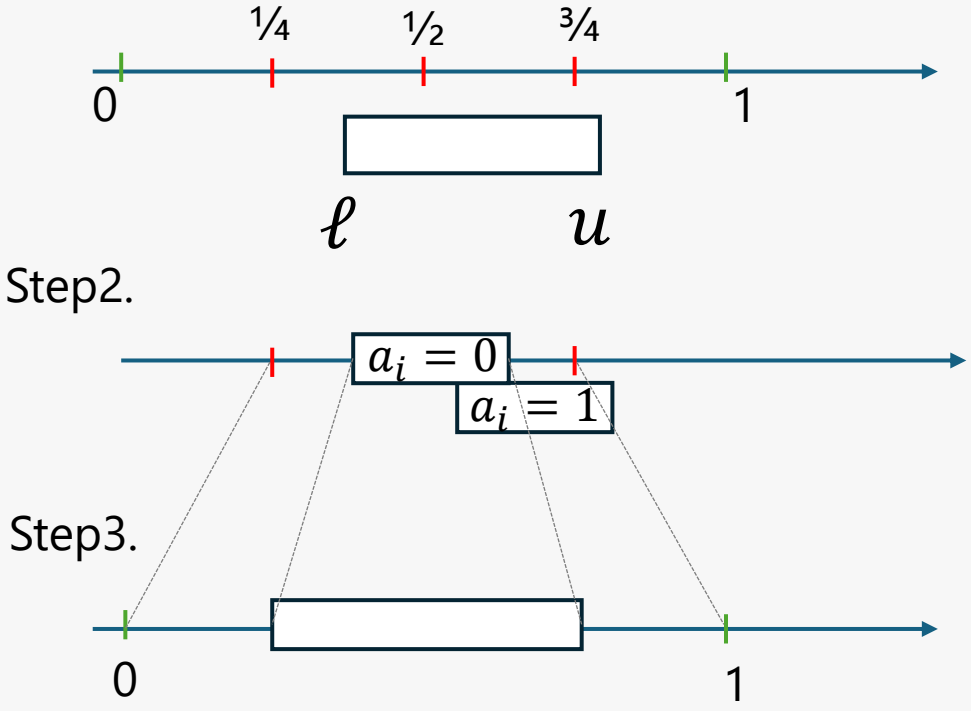
2. (The same as ver. 1) Read a_i.

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 $k = k + 1$.

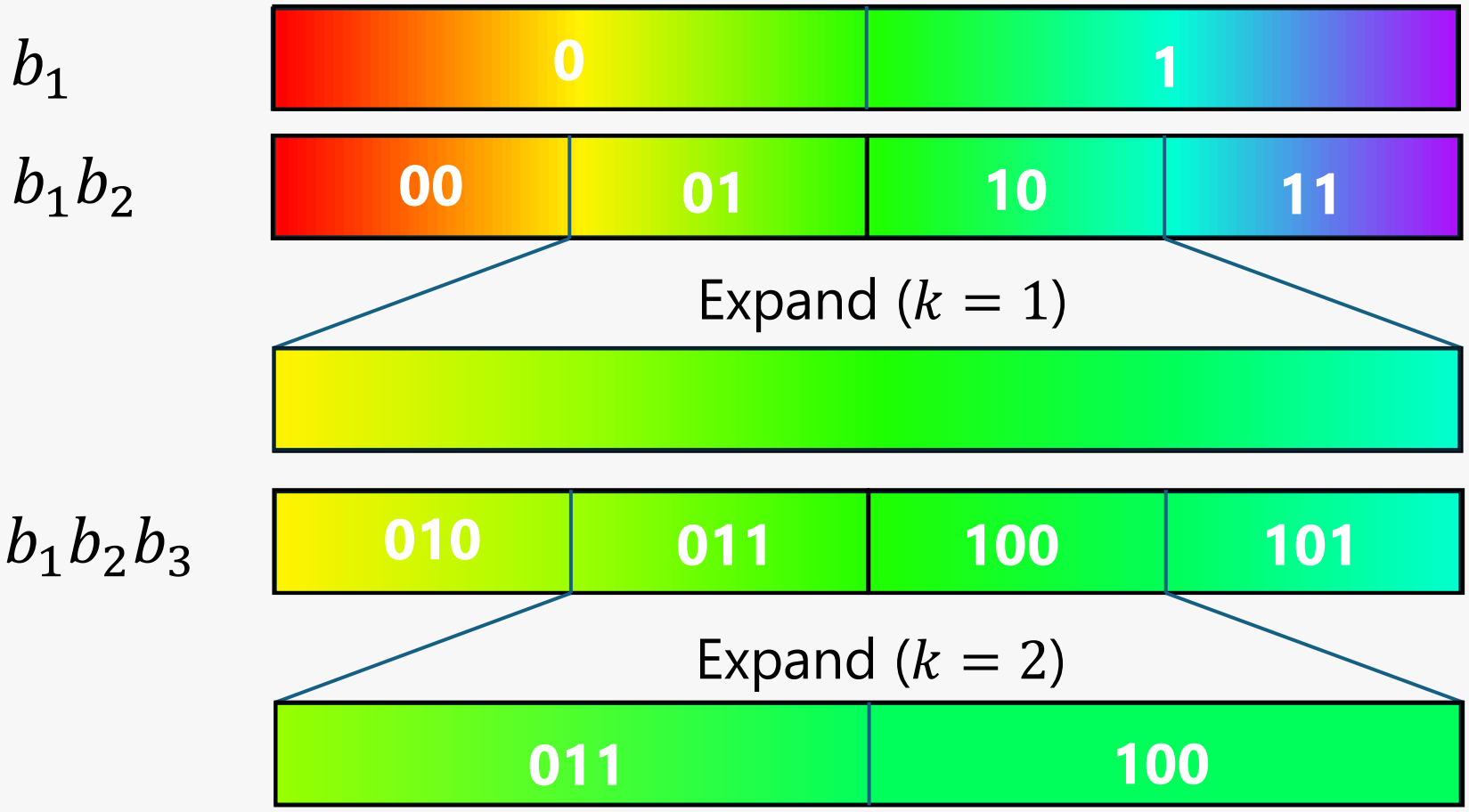
   (b) If $u < \frac{1}{2}$, then output $01 \ldots 1$ and update $\ell = 2\ell, u = 2u, j = j + k + 1$, and $k = 0$.

   (c) If $\ell \geq \frac{1}{2}$, then output $10 \ldots 0$ and update $\ell = 2\ell - 1, u = 2u - 1, j = j + k + 1$, and $k = 0$.

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



The binary expansion slightly smaller than ½ is 0.0111…
and the one lightly greater than ½ is 0.1000…



The β-ary to binary conversion [1]
The proposed method (the 3rd version)

- All calculations must be performed digitally.
- In the 3rd version, u, ℓ and $\gamma = \beta^{-1}$ are expressed by fixed-point numbers (or integers).
- The error caused by the finite precision calculation has not been evaluated. →Future work

A remaining problem

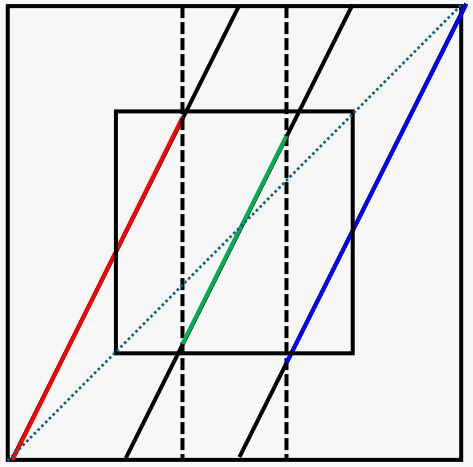
- Mismatch of the value of β.
  - We have assumed that the exact β is available.
  - The exact value of β is **unknown**.
  - It is desirable to obtain a binary sequence of i.i.d. from the output sequence of the β encoder even if the exact β is not known.

Extensions and future works

❑ Extension to the case of $\beta \geq 2$.

- The ADC with three branches is called 1.5 bit encoder and is commercially often used.
- Other beta value is also possible. (We get more bits at one cycle)

$\beta = 2$



$3 \leq \beta < 4$