A Study on Oscillator-based True Random Number Generator Robust to Environmental Fluctuation

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2013

Takehiko AMAKI

Publication list

Transactions

1. T. Amaki, M. Hashimoto, and T. Onoye, "Jitter Amplifier for Oscillator-based True Random Number Generator," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96-A, no. 3, Mar. 2013.

Conference papers with referee

1. T. Amaki, M. Hashimoto, and T. Onoye, "An Oscillator-based True Random Number Generator with Jitter Amplifier," In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 725-728, May 2011.
2. T. Amaki, M. Hashimoto, and T. Onoye, "Jitter Amplifier for Oscillator-based True Random Number Generator," In *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 81-82, Jan. 2011.
3. T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "A Design Procedure for Oscillator-based Hardware Random Number Generator with Stochastic Behavior Modeling," In *Proceedings of Workshop on Information Security Applications (WISA)*, pp. 107-121, Jan. 2011.
4. T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "An Analysis of Oscillator-based True Random Number Generator with Markov Model," In *Proceedings of IEICE Karuizawa Workshop on Circuits and Systems*, pp. 474–479, Apr. 2009 (in Japanese).

Conference papers without referee

1. T. Amaki, M. Hashimoto, and T. Onoye, "An Oscillator-based True Random Number Generator with Jitter Amplifier," In *IEICE Technical Report*, ICD2011-118, vol. 111, no. 352, pp. 87–92, Dec. 2011 (in Japanese).
2. T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "A Design Procedure for Oscillator-based Physical Random Number Generator with Stochastic Behavior Modeling," In *IEICE Technical Report*, SLDM2010-147, vol. 2010-SLDM-147, no. 19, pp. 1–6, Nov. 2010 (in Japanese).

Abstract

This thesis discusses an oscillator-based true random number generator (TRNG) which is robust to environmental fluctuation. Random number generated from physical random sources is a key component of a security system, such as cryptography and authentication, because of its unpredictability, and many instruments for random number generation have been studied. This thesis focuses on an oscillator-based TRNG, which utilizes random jitter of the oscillator as a random source, since it is easily implemented on silicon and it is inherently robust to deterministic noise compared to a direct amplification method. However, it is difficult to design an oscillator-based TRNG with circuit simulations, because ordinary simulators do not take the jitter into account directly. An efficient design methodology tailored to the oscillator-based TRNG is demanded. In addition, the amount of jitter is generally insufficient while the randomness of output bit stream depends on the jitter amount. Frequency dividers help accumulate the jitter, but they reduce the throughput of the TRNG. Thus, an oscillator which has large jitter yet operates at high frequency is required. Bias of output bits is another critical issue for the TRNG. Probability of '1' occurrence depends on duty cycle of an internal oscillator of the TRNG, whereas the duty cycle fluctuates by environmental variations. Consequently, post-silicon online tuning of duty cycle is indispensable.

This thesis firstly presents a design methodology for the oscillator-based TRNG based on a stochastic behavior model. The model is used to evaluate the randomness of the TRNG. Measurement results of a prototype TRNG fabricated in a 65 nm CMOS process show that the proposed model well reproduces the behavior of the TRNG. The stochastic model also enables a worst-case-aware design of the TRNG. This thesis analytically confirms that the worst case the methodology considers results in the lowest randomness of outputs. The proposed worst-case-aware design methodology determines design parameters according to the worst χ value of a poker test under deterministic noise. Experimental results with a noise-aware gate-level simulator implemented for validation purpose verifies the efficiency of the worst χ evaluation. Also, a design example is presented to exemplify the proposed worst-case-aware methodology.

Secondly, this thesis proposes an architecture of a jitter amplifier to improve the randomness of the oscillator-based TRNG. Jitter of an oscillating signal is intensified by changing a propagation delay of a latency-controllable (LC) buffer. Two types of LC buffer, viz. two-voltage LC buffer and single-voltage LC buffer are presented. This thesis also derives an expression to estimate the gain of the jitter amplifier, and analyzes sufficient conditions for proper amplification. The oscillator-based TRNGs with the jitter amplifiers are implemented with a 65 nm CMOS process. Area of the amplifier with the two-voltage LC buffer is 3,300 μm^2, while the amplifier with the single-voltage LC buffer occupies 1,700 μm^2. Measured jitter gain of the jitter amplifier with the two-voltage LC buffer is 8.4, and that with the single-voltage LC buffer is 2.2. The jitter amplification enhances the entropy of bit streams, and makes the output random bits pass the all tests of the NIST test suite. The proposed jitter

amplifier attains higher throughput per area than frequency dividers in most cases.

Finally, this thesis describes a system that self-calibrates duty cycle to remove the biasing. In the proposed system, a duty cycle monitor measures the duty cycle of an oscillating signal, and a duty cycle corrector adjusts the duty cycle according to the measured value. A TRNG with the proposed monitor and the corrector is fabricated in a 65 nm CMOS process. The duty cycle monitor measures the duty cycle with a resolution of 0.16 % and the duty cycle corrector achieves 0.11 % of resolution in average at 20 °C. In addition, the monitor reduces the necessary time for estimating duty cycle to be 3,500 times smaller than output bit sampling. Employing the self-calibration system, the variation of probability of '1' occurrence due to the temperature fluctuation becomes 1/18 times smaller.

Integrating these accomplishments, this thesis realizes generation of highly random numbers even under environmental fluctuation, and contributes to development of highly secure systems.

Acknowledgments

First of all, I would like to express my deepest gratitude to Professor Takao Onoye in Osaka University for providing me a precious opportunity and an excellent environment to study as a doctoral student in his laboratory.

I have no adequate words to express my heartfelt appreciation to Associate Professor Masanori Hashimoto in Osaka University. All of my productive researches are credited to none other than him. His advanced perspective and thoughtful advises led me to successful achievements.

I would like to express my appreciation to Professor Masaharu Imai, and Associate Professor Katsuyoshi Miura in Osaka University for detailed review and insightful suggestions.

I would like to demonstrate my gratitude to Dr. Mitsuru Harada in Nippon Telegraph and Telephone (NTT) Corporation for helpful discussions in the advisory committee.

I am deeply grateful to Professor Koji Nakamae, Professor Tatsuhiro Tsuchiya, Professor Haruo Takemura, Professor Makoto Nakamura, and Professor Akihisa Yamada in Osaka University for useful advices.

My appreciation also goes to Associate Professor Yuichi Itoh, Assistant Professor Masahide Hatanaka, Dr. Ryusuke Miyamoto in Osaka University, Associate Professor Yukio Mitsuyama (currently working in Kochi University of Technology), Associate Professor Gen Fujita (currently working in Osaka Electro-Communication University), and Assistant Professor Hiroshi Tsutsui (currently working in Kyoto University), for technical and other supports in laboratory.

I would like to express my sincere appreciation to Emeritus Professor Isao Shirakawa of Osaka University.

I also appreciate Mr. Shozo Tsukahara in Synthesis Corporation for providing me a chance to engage on jobs with regard to VLSI circuit designs.

I would like to thank Professor Kazutoshi Kobayashi in Kyoto Institute of Technology, and Associate Professor Toru Nakura in University of Tokyo for their valuable lectures of VLSI design.

The VLSI chip in this thesis has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Semiconductor Technology Academic Research Center (STARC), e-Shuttle, Inc., and Fujitsu Ltd. This work is supported by VDEC, the University of Tokyo in collaboration with Synopsys, Inc., Cadence Design Systems, Inc., and Mentor Graphics, Inc. I demonstrate my gratitude to all members of "Dependable VLSI Platform Using Robust Fabrics" research project in Japan Science and Technology Agency (JST), Core Research of Evolutional Science and Technology (CREST). I would like to appreciate Strategic Information and Communications R&D Promotion Programme (SCOPE) of Ministry of Internal Affairs and Communications.

I would like to thank Dr. Yasuhiro Ogasahara (currently working in National Institute of Advanced Industrial Science and Technology), Assistant Professor Hiroshi Fuketa (currently

working in University of Tokyo), Dr. Ken'ichi Shinkai (currently working in NEC Corporation), Dr. Takashi Enami (currently working in Fujitsu Laboratories, Ltd.), Mr. Shinya Abe (currently working in Renesas Electronics Corporation), Mr. Shinyu Ninomiya (currently working in Renesas Electronics Corporation), and Mr. Koichi Hamamoto (currently working in Mitsubishi Heavy Industries, Ltd.) for helpful advices with regard to VLSI design, measurement, and simulation. I also would like to thank Dr. Ryoji Hashimoto (currently working in Renesas Electronics Corporation) for skillful coaching about Register Transfer Language (RTL) coding and verification. I would like to appreciate Mr. Igors Homjakovs, Mr. Dawood Alnajjar, Mr. Ryo Harada, Mr. Hiroaki Konoura, Mr. Dan Kuroda (currently working in Renesas Electronics Corporation), and Mr. Yasumichi Takai (currently working in Panasonic Corporation) for fruitful discussions. I would like to thank my closest colleagues: Mr. Masashi Okada, Mr. Yu Jaehoon, Mr. Yuuta Asahi, Mr. Kenji Ooga (currently working in Panasonic Corporation), and Mr. Yuki Kawamura(currently working in Japan Broadcasting Corporation).

I would like to thank Mrs. Mayuko Nakamura, Ms. Tomomi Kondo, Mrs. Chika Nomura, Mrs. Yoshimi Fujita, and Ms. Yuiko Shinozaki for their various supports throughout my student life. I also would like to thank all of my friends for having good times.

Finally, I give my thank to my family for supporting my livelihood. I offer my prayer of gratitude to my father Takeshi in heaven.

Contents

List of tables

List of figures

Chapter 1

Introduction

This chapter describes the background and the objectives of this thesis. This thesis focuses on random number generation for security system. Requirements for an on-chip random number generator include robustness to environmental fluctuation, such as power supply noise and temperature fluctuation, as well as high throughput and small area. The purpose of this thesis is to design a hardware random number generator which is tolerant to supply noise and temperature fluctuation. The following sections describe the background of the random number generation, introduce an oscillator-based TRNG, which this thesis focuses on, and finally present the objectives of this thesis.

1.1 Background

This section describes how random number is utilized for security and introduces several kinds of random number generators. Methods to evaluate randomness and to enhance randomness with post processing are also shown.

1.1.1 Random number for security purpose

High-quality random number generation is highly demanded and essential for security [1]. Cryptosystems and authentication systems suppose the usage of random numbers. For example, random numbers are used as the keys and initial vectors for the cipher block chaining (CBC) mode in common key cryptosystems, and Vernam cipher also needs random numbers whose size is more than the plain text [2]. Secure sockets layer (SSL) requires random numbers to generate the keys for data encryption [3]. At the first step of Diffie-Hellman key exchange, a server and a client individually generate random numbers [4]. Challenge-and-response authentication and digital signature also require random numbers [5].

Statistical randomness and unpredictability of the random numbers are important for security applications. For example, in 1995, Goldberg and Wagner [6] pointed out a security flaw of Netscape browser. The cause of this problem was insufficient unpredictability of the secret keys, which was generated from predictable numbers such as time and process ID. Markettos et al. [7] presented a way of attacking the random number generator, and degraded the randomness of bit streams generated from an EMV payment card. The authors claimed that a person could withdraw cash masquerading another person with the attacking method. Thus, deterioration of random numbers degrades the quality of security system.

Pseudo random number, which is easily generated by deterministic calculation [8], is pre-

dictable when a malicious attacker knows the initial state of the pseudo random number generator, and then, it is not suitable for security purposes.

On the other hand, true random numbers are produced from physical random sources, and all bits in bit streams are independent of the other bits and the probabilities of 1/0 occurrences are identical. Note that this thesis hereafter calls a random number generator which utilizes physical random source as a true random number generator (TRNG), while it is sometimes called a physical random number generator or a hardware random number generator in literature. Because true random numbers cannot be predicted by computational methods, they are advantageous for security purposes. True random number generators are implemented on microprocessors as well as smart card ICs [9]. For example, 22 nm Intel microarchitecture implements a true random number generator and supports instructions to utilize the random numbers [10, 11].

1.1.2 True random number generators

There are many kinds of methods for true random number generation. Hu et al. [12] employed unpredictable movement of the mouse. Rohe [13] used radioactive decay of americium as random source. Yoshizawa [14] also employed radioactivity. Recently, chaos in semiconductor laser attracts attention as a source of a high speed TRNG [15–21]. For instance, Kanter et al. [18] demonstrated a TRNG based on a chaotic semiconductor laser, which attained 300 Gbps of throughput. A light quantum is another random source being studied for the TRNG [22–24]. In addition, Yamanashi and Yoshikawa [25, 26] presented a TRNG which utilized a sensitivity of a superconductive circuit to obtain a random bit stream from thermal noise. These TRNGs require special hardware to exploit random sources, which degrades their utilities and availabilities. Accordingly, on-chip TRNGs have been widely studied because of the cost of fabrication and the utility.

Oscillator-based TRNG, which utilizes jitter of oscillators as a random source, is a popular method for generating true random number [27–40]. The TRNG is easy to implement, process-portable, and process-scalable, because it simply consists of digital circuits, i.e. oscillators and a sampler. In addition, the TRNG is tolerant to deterministic noises [44]. This thesis therefore focuses on the oscillator-based TRNG and explains the detail in Sec. 1.2.

Direct amplification is a traditional method for true random number generation [41–45]. This method amplifies internal noise, typically thermal noise from resisters or transistors with amplifiers, and generates a bit stream comparing the voltage of the amplified noise with a threshold voltage. Figure 1.1 illustrates a simple TRNG with direct amplification method. The TRNG in Fig. 1.1 outputs '1' when the amplified noise is more than 0 V and otherwise '0', since the threshold voltage is GND. On the other hand, the amount of thermal noise is generally small [46], and hence the method needs a high-gain wide-band amplifier, which occupies large area and consumes much power. The designers need to at least tune the design parameters of analog circuits, and in some cases, redesign the analog circuits when the TRNG is fabricated in another technology. Although Matsumoto et al. [47] used SiN MOSFET as a source of large noise and shrunk the size of the amplifier, the SiN MOSFET requires the special process. In addition, randomness of bit streams is sensitive to deterministic noises, such as power supply noise, substrate noise, and so on [44], and then, the TRNG should be shielded from the deterministic noises.

Chaos-based TRNG implements a chaotic system and utilizes its sensitivity to the input [48–58]. Figure 1.2 shows an example of chaos-based TRNG which implements Bernoulli

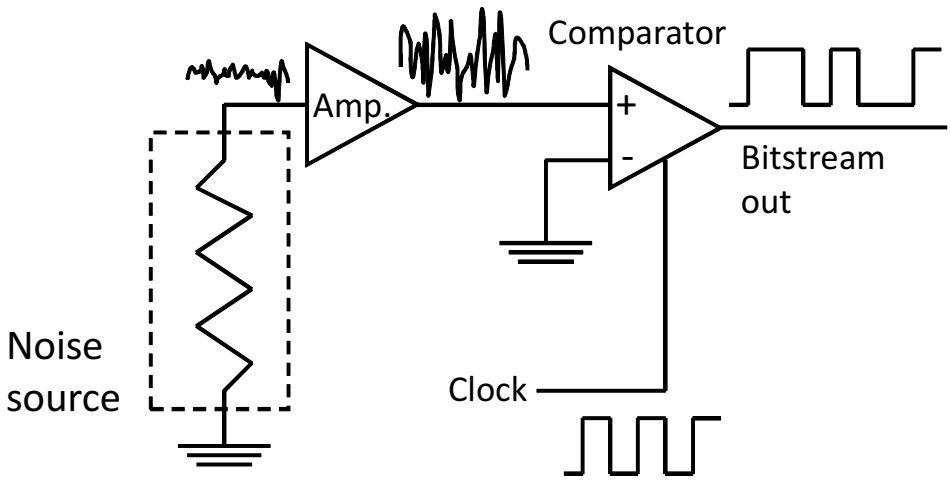

Figure 1.1: Basic TRNG with direct amplification method.

shift map [58]. The chaos system utilized in the TRNG is expressed as follows:

$$X_n = [2(X_{n-1} + e(n))]\text{mod}1.0, \tag{1.1}$$

where X_n is a stochastic variable and $e(n)$ represents random noise. The TRNG in Fig. 1.2 realizes Eq. (1.1) with pipelined analog-to-digital converter. The bit sequence generated from the iterations of Eq. (1.1) is uniformly distributed and its spectrum is flat. Eq. (1.1) includes a term of random noise $e(n)$, and then, the internal states of the system after sufficient number of iterations is unpredictable even if the initial state is known. Since the chaotic system helps improve randomness even through $e(n)$ is quite small, the randomness of output is robust to deterministic noise. The randomness comes from the character of the chaotic system rather than the input noise [58]. The TRNG, however, requires complicated analog circuits and consumes large current.

TRNGs which are based on metastability of the circuits are recently studied [59–71]. Figure 1.3 shows a schematic and simulated voltages of the metastability-based TRNG. In the initial state, CLK is LOW, and therefore both A and B are HIGH. The cross-coupled inverters get into a metastable state and the internal nodes go the intermediate voltages when CLK rises. Then, the metastable state is solved by internal random noise. One of the internal nodes, A or B, is pulled up and the other node is pulled down. Consequently, the cross-coupled inverters get into one of two stable states. Thus, the TRNG generates a random bit every cycle of CLK. Since the core element, which is the cross-coupled inverters, is a latch, the TRNG is implemented with digital circuits. However, the process variation between the two inverters and the deterministic noise bias the final state, and then the TRNG must be accompanied with precise calibration.

In addition, there are other methods studied. Tanamoto et al. utilized MRAM cells [72,73]. Brederlow et al. used random telegraph noise [74]. Xu et al. used hot-electron injection [75, 76]. MOSFET after soft breakdown were utilized as large noise source in [77–80]. Fibonacci and Galois ring oscillator were used as random sources [81, 82]. Modified ring oscillator (MRO) and transition effect ring oscillator (TERO) were proposed in [83, 84].

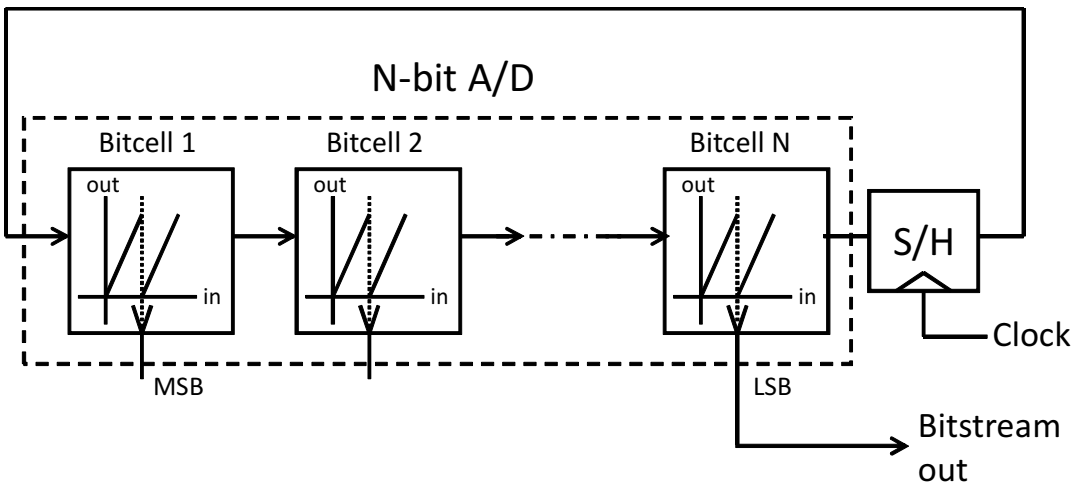

Figure 1.2: An example of chaos-based TRNG.

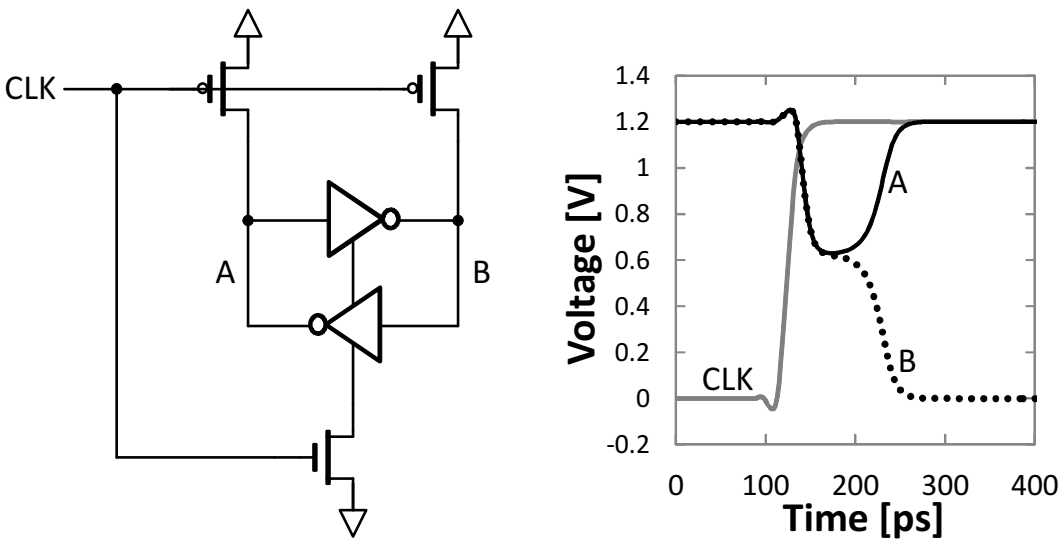

Figure 1.3: An example of metastability-based TRNG and simulated voltages of each node in a 65 nm CMOS process.

1.1.3 Methods of randomness evaluation

Specifications of randomness of TRNGs need to be considered since various security applications utilize the TRNGs. Generally, it is difficult to theoretically demonstrate that a TRNG for cryptography attains enough randomness. The TRNG should prove that the generated bit streams cannot be distinguished from the ideal random numbers. The proof is, however, impossible because it requires trials of an infinity number of judgement algorithms [85]. Con-

sequently, the researchers choose several algorithms to test the randomness of generated bit streams. There are several tests for randomness evaluation, and major tests are introduced in this section. Note that randomness of TRNG in this thesis just means the statistical randomness of output data, and unpredictability is not specifically discussed.

FIPS140-2 [86] provides four statistical randomness tests, i.e. monobit test, poker test, runs test and long runs test. The poker test divides a 20 kbit sequence into 5,000 consecutive 4 bit segments, and counts the number of occurrences of 16 possible 4 bit values. χ value is calculated as following:

$$\chi = \frac{16}{5000} \times \sum_{i=0}^{15} [f(i)]^2 - 5000, \tag{1.2}$$

where $f(i)$ denotes the number of each 4 bit value i, and i is $0 \leq i \leq 15$. Pass range is $2.16 < \chi < 46.17$.

NIST 800-22SP [87] presents a test suite for cryptography application, which consists of 15 tests. The test suite gives sets of p-value and pass proportion. If the p-value is 0.0001 or more and the pass proportion is within $(1-\alpha) \pm 3\sqrt{\alpha(1-\alpha)/n_{seq}}$, then the bit stream passes the test. Here, α is the significance level and n_{seq} is the number of sequences. Pareschi et al. [88] analyzed the testing strategy of the NIST test.

Marsaglia [89] presented a set of randomness tests, called Diehard tests. AIS31 [90, 91] is an evaluation criteria for TRNGs, in which 9 statistical tests are defined. Knuth [92] described 12 kinds of randomness tests. Crypt-X [93] is a commercial test for randomness evaluation. L'ecuyer and Simard [94] introduced TestU01, which is a C library for empirical testing of random number generators. Udawatta et al. [95] presented a statistical method for TRNG testing.

This thesis employs the randomness tests described in FIPS140-2 and NIST 800-22SP since they are the most popular tests and widely used in TRNG literature.

1.1.4 Post processing

Post processing of the generated bit stream is often employed in order to enhance the randomness, and representative correctors for postprocessing are introduced here.

Von Neumann corrector [96] splits the input bit stream into groups of two bits, and then, the corrector outputs '1' for "01", '0' for "10", and nothing for "00" and "11". The corrector compensates the bias between the occurrences of '1' and '0' since it reduces the number of series of identical bits. On the other hand, the corrector drastically reduces the throughput of the TRNG, since it discards about 3/4 of input bit stream even when the input has sufficient randomness.

XOR corrector [97] outputs XOR of successive two bits of inputs. The corrector adjusts the number of occurrences of '1' and '0' when the input bits are independent of each other while it reduces the throughput by half.

SHA-1 mixer [36] increases the randomness of outputs by mixing the bit stream with a hash function, SHA-1. Mathematical theories for randomness extractions are also presented [85, 98–105]. However, the circuits for the mathematical functions dissipate additional power and occupy large area.

Consequently, this thesis aims at developing a TRNG that can attain sufficient randomness without post processing.

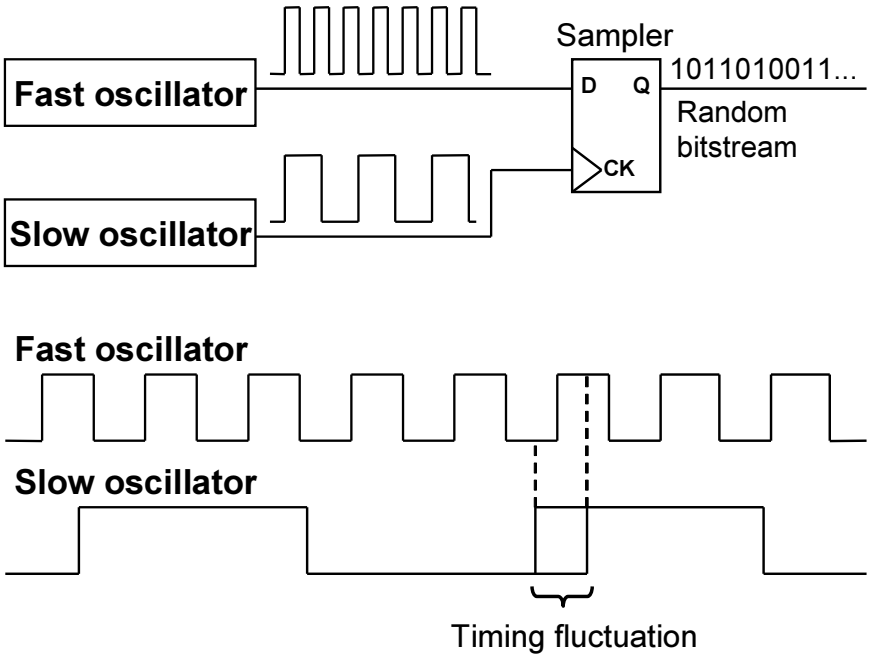

Figure 1.4: Basic oscillator-based TRNG.

1.2 Oscillator-based true random number generator

As described in Sec. 1.1.2, an oscillator-based TRNG is a popular random number generator and is easy to implement. Thus this section focuses on the oscillator-based TRNG. This section reviews the principle and design issues of the oscillator-based TRNG.

1.2.1 Principle of oscillator-based TRNG

Figure 1.4 has a block diagram of a basic oscillator-based TRNG, which consists of a sampler and two distinct oscillators; one is fast and the other is slow. The sampler acquires bits from the fast oscillator (D in Fig. 1.4) using the signal of the slow oscillator as the clock (CK in Fig. 1.4). The oscillators inherently have jitter because of internal noise, and hence the rise timing of the slow oscillator signal fluctuates from the viewpoint of the rising edges of the fast oscillator. The oscillator-based TRNG generates random numbers exploiting this jitter as a random source. Though the throughput of the TRNG is unstable since the frequency of the clock for the sampler is slightly but randomly fluctuated, a first-in first-out (FIFO) interface can easily stabilize the throughput.

Random period jitter, which originates from internal random noise such as thermal, shot, and random telegraph noise, has been called ‘jitter’ for the sake of brevity in this thesis. Then, the amount of the jitter is defined as the standard deviation of periods. Note that the timing fluctuations of the edges of the slow signal relative to the fast oscillator is the source of the randomness in the TRNG, as will be discussed in detail in Sec. 2.2.2.

In addition, the jitter is assumed to be temporally independent in this thesis. For validating this assumption, an example of auto correlation function (ACF) of the jitter of an oscillator is

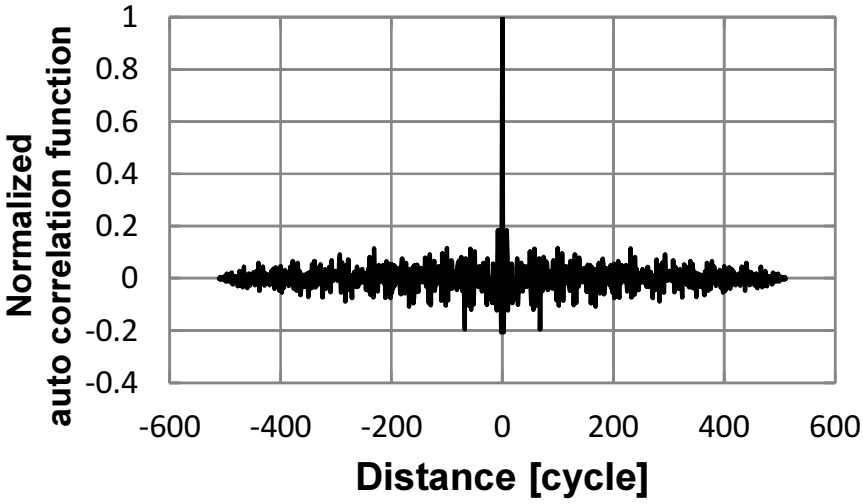

Figure 1.5: Normalized auto correlation function of jitter of a ring oscillator.

given. A normalized ACF of a 251-stage ring oscillator with 16-frequency divider fabricated in a 65 nm CMOS process is shown in Fig. 1.5. The period of 512 cycles were measured with a real-time oscilloscope, the mean of the periods was subtracted from each measured period, and then the normalized ACF was calculated. Figure 1.5 shows that the ACF of the jitter was similar to the Dirac delta function, which means the jitter of the ring oscillator can be assumed to be temporally independent.

1.2.2 Difficulty in designing oscillator-based TRNG

It is necessary to estimate the randomness of TRNGs and obtain appropriate design parameters for designing a TRNG that satisfies given performance specifications. Despite the necessity of randomness evaluation, it is difficult to simply simulate oscillator-based TRNGs because the jitter of oscillators is not directly considered in ordinary circuit simulators such as Synopsys HSPICE [106] and NanoSim [107]. They could simulate oscillator-based TRNGs by modeling jitter with pseudo-random numbers through Verilog-A [108], for example, but it takes an unacceptably long time for simulations, although randomness tests require long bit streams. Furthermore, the oscillation periods for the two oscillators and their jitter are on different orders of magnitude and hence the time steps for transient simulations must be kept small to ensure that the simulations are accurate. Therefore, an efficient behavioral model and a method of evaluating the randomness of oscillator-based TRNGs are necessary to guide explorations into design space and meet the design specifications. Also, a design method that takes into consideration deterministic noise is required because a TRNG should guarantee sufficient randomness even when unwanted noise or malicious attacks occur.

Petrie and Connelly [44, 58] modeled a slow oscillator under random noise and deterministic signals as a voltage-controlled oscillator (VCO). They discussed the required jitter to produce sufficient randomness and the effects of deterministic noise with a poker test [86]. They also discussed the frequency ratio of the two oscillators when it was small (about 15). Although the reference [44] claimed that a larger frequency ratio resulted in better random-

ness, this tendency was from an assumption about the behavior of an oscillator-based TRNG and insufficient quantitative evaluation in terms of frequencies was provided. In addition, their proposed model [44] could not be used to evaluate the effect of deterministic noise accurately when the noise frequencies were higher than that of the slow oscillator, and therefore design with the model could not ensure sufficient randomness under high-frequency noise. Moreover, the model was not validated with hardware measurements. Bucci et al. [39] introduced numerical formulas that gave the transition probability between successive bits as a function of the average and the standard deviation of oscillation periods and the initial phase difference between the two oscillators. However, they did not test randomness [86, 87, 89] rigorously, and did not consider deterministic noise. Bernard et al. [109] proposed a mathematical model of a TRNG using two jittery clocks with rationally related frequencies, and the model could be used to evaluate entropy per bit and bias on the generated bit stream. Their model, however, did not take deterministic noise into consideration. Baudet et al. [110] modeled the oscillators of a TRNG with a phase-oriented approach, and they provided formulas for entropy rates. They also introduced a method of measuring jitter by filtering out deterministic jitter. Ergün [28] modeled a chaotic oscillator that was used as a slow oscillator, and he provided design guidelines based on estimates of entropies. The model was, however, tailored for a chaotic oscillator and a VCO, and hence it could not be used for other types of oscillator-based TRNGs.

1.2.3 Requirement of large jitter

Although jitter is the source of randomness of the oscillator-based TRNG, the amount of internal noise, i.e., jitter is so small that it is very difficult to generate highly random bit streams with a naive circuit structure.

Balachandran and Barnett [37] claimed that the amount of jitter should be at least six times as large as the period of fast oscillator to attain sufficient randomness. For example, the period of a 7-stage ring oscillator implemented with a 65 nm CMOS process is 220 ps from circuit simulation, and then, $220 \times 6 = 1320$ ps of jitter is required. On the other hand, the jitter amount of a 251-stage ring oscillator with 64-frequency dividers is measured as 100 ps, which is quite smaller than the necessary value. Thus, to design an oscillator with large jitter is one of the most challenging subjects for oscillator-based TRNG design.

Frequency dividers help increase the amount of jitter by accumulating the jitter of the oscillator, but they significantly degrade throughput [38, 112]. Bucci et al. [39] utilized a triangular wave oscillator, which is sensitive to random noise and produces large jitter. The oscillator, however, requires analog circuits, and they consume large power and area. Ergün and Özoğuz [27] presented a TRNG with a chaotic oscillator. The chaotic oscillator modulated the slow oscillator of the TRNG, and the timing fluctuations of the slow edges were intensified. The timing fluctuations depend on the employed chaos system rather than the random noise. In addition, the chaotic oscillator needs inductors, and then, the TRNG is not scalable.

1.2.4 Requirement of finely tuned duty cycle

The probability of '1' occurrence of output bit stream is 0.5 for ideal random numbers, and little bias between '1' and '0' occurrences is preferred.

Researchers of TRNGs grappled with the bias issue. Bock et al. [34] and Bucci and

Luzzi [38] adjusted the probability of '1' by controlling the rise timings of the fast signal. Bucci et al. [39] also used T flip-flop to remove the biasing of the output bits. Besides, Srinivasan et al. [64] presented a metastability-based TRNG whose bias of outputs was eliminated by changing the clock timings for the latch in metastable state. Majzoobi et al. [70] adjusted the two rising edges of the signals to a D flip-flop, and thereby probability of output being '1' became 0.5.

Biasing of output bits from an oscillator-based TRNG depends on the duty cycle of fast oscillator [28] as well as the offset of the sampler [111], and therefore, the duty cycle should be finely tuned. The probability of '1' occurrence, p, should be 49.875 % $\leq p \leq$ 50.125 % to pass the frequency test of NIST tests, and the range of acceptable probability is 0.250 %. On the other hand, the probability of '1' occurrence with a 7-stage ring oscillator in a 65 nm CMOS process varies from 31.5 % to 44.7 % because of the process variation, and the range from the largest and the smallest is 13.2 %. Also, temperature also changes the probability of '1' from 37.6 % to 38.0 % when the temperature varies from 0°C to 100°C, and the range is 0.4 %. The simulated probabilities are separated from 50 % because the simulations ignore the offsets of the samplers. Note that careful circuit design and layout can make the probability of '1' 50 % at a typical operating condition in a typical process corner, but the variation after the fabrication still remains. Consequently, the oscillator-based TRNG with sufficient statistical randomness requires post-silicon online tuning of duty cycle.

1.3 Objective of this thesis

For ensuring security systems, TRNGs are required to produce good random numbers robustly even under environmental fluctuation and deterministic noises. The purpose of this thesis is to realize an oscillator-based TRNG which is robust to deterministic noises and fluctuation of temperature. To achieve the purpose, this thesis studies the following subjects;

- a worst-case-aware design methodology based on a stochastic behavior model considering deterministic noise,
- a jitter amplifier to attain the slow oscillator with large jitter, and
- a self-calibration system to finely tune the duty cycle of the fast oscillator.

Figure 1.6 outlines a structure of this thesis in which the subjects correspond to Chapters 2, 3, and 4, respectively.

Firstly, this thesis proposes a stochastic behavior model of the oscillator-based TRNG, and presents a design methodology considering the worst case under deterministic noise with the proposed model. A design methodology is required to determine appropriate design parameters considering deterministic noise, as is described in Sec. 1.2.2. This thesis proposes a stochastic behavior model to efficiently determine the design parameters. Also, the thesis identifies a class of deterministic noise under which the randomness gets the worst. The proposed design methodology can be used to estimate the worst χ value of a poker test directly without generating bit streams. Then, the design space can be efficiently explored and the methodology guarantees sufficient randomness in a hostile environment. The proposed model was validated by hardware measurement.

Secondly, this thesis presents an additional circuit to the oscillator-based TRNG, which amplifies jitter amount of the slow oscillator. As explained in Sec. 1.2.3, oscillators with large jitter and high frequency are required in order to attain highly random bits without decreasing

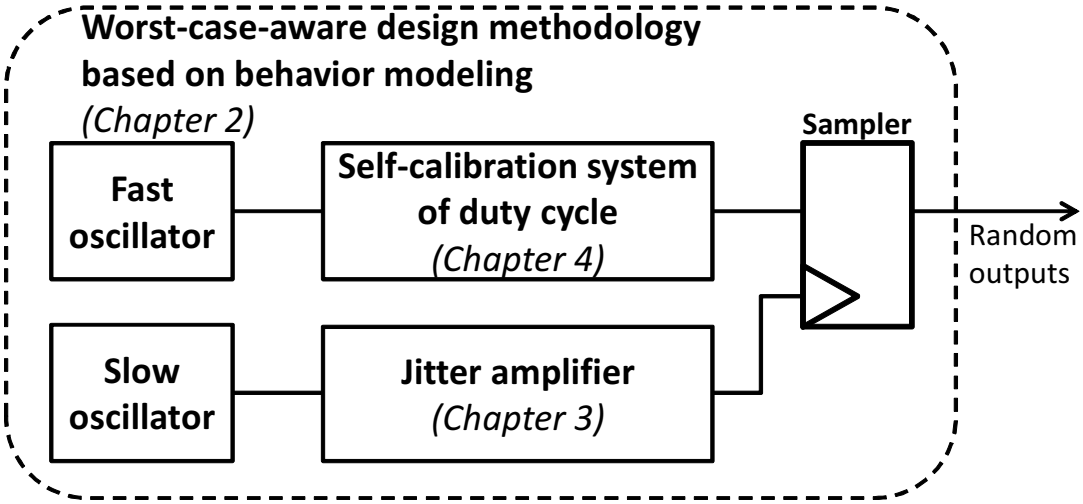

Figure 1.6: General structure of this thesis.

throughput. This thesis proposes an architecture of jitter amplifier circuit for oscillator-based TRNG. The thesis derives an equation for the estimation of the gain of the jitter amplifier, and analyzes sufficient conditions for proper operation.

Thirdly, a self-calibration system for duty cycle of the fast oscillator is presented. Duty cycle of the fast oscillator is an important parameter, which significantly affects the 1/0 ratio of output, and it should be adjusted even under dynamically changing environments, as presented in Sec. 1.2.4. This thesis proposes a duty cycle monitor and a duty cycle adjuster for the fast oscillator. Cooperation of the proposed circuits enables the duty cycle tuning of the fast oscillator under dynamic temperature fluctuation.

The rest of this thesis is organized as follows. Chapter 2 describes the worst-case-aware design methodology for noise-tolerant TRNG with stochastic behavior modeling. Chapter 3 proposes a jitter amplifier, which intentionally amplifies the amount of jitter of the slow oscillator. Chapter 4 presents a self-calibration system, which automatically adjusts duty cycle of the fast oscillator. Chapter 5 gives concluding remarks.

Chapter 2

A Worst-case-aware design methodology with stochastic behavior modeling

This chapter presents a worst-case-aware design methodology for an oscillator-based true random number generator (TRNG) that produces highly random bit streams even under deterministic noise. This chapter proposes a stochastic behavior model to efficiently determine the design parameters, and identifies a class of deterministic noise under which the randomness gets the worst. They can be used to directly estimate the worst χ value of a poker test under deterministic noise without generating bit streams, which enables efficient exploration of design space and guarantees sufficient randomness in a hostile environment. The proposed model is validated by measuring prototype TRNGs fabricated with a 65 nm CMOS process.

2.1 Introduction

TRNG requires to estimate the randomness of a TRNG and find appropriate design parameters, as referred in Sec. 1.2.2. Simulation of an oscillator-based TRNG is difficult, however, because the jitter of oscillators is not directly considered in ordinary circuit simulators and modeling of the jitter with pseudo-random numbers takes an unacceptably long time. Therefore, an behavioral model and a methodology of fast randomness evaluation are necessary to guide explorations into design space. In addition, a design methodology that takes into consideration deterministic noise is required because a TRNG should guarantee sufficient randomness even under unwanted noise.

The author of this thesis [111] proposed a procedure for designing an oscillator-based TRNG with a stochastic behavior model. They determined the design parameters with the model without taking deterministic noise into account, and then evaluated robustness to power-supply noise with bit generation. The procedure, however, required iterations of explorations of design space and checking of robustness until ad hoc design modifications attained sufficient robustness to the supply noise. Moreover, the randomness under deterministic noise was evaluated with only a small subset of possible deterministic noises. In reality, the number of possible deterministic noises is infinite. Thus, the identification of the worst-case through a number of simulations is impossible, and hence the procedure did not guarantee the randomness under deterministic noise.

This chapter proposes a worst-case-aware design methodology using a stochastic behavior model. Key design parameters are explored and determined with the stochastic model. The worst randomness under deterministic noise is quickly evaluated with a number of design parameters using a model without bit generation, and appropriate design parameters are determined so that the TRNG passes tests and satisfies the required specifications, which makes design iterations unnecessary. The proposed worst-case-aware design methodology guarantees enough randomness even under any waveform shapes of deterministic noise, because the worst case derived in this work is the theoretically-proven worst case, and there are no deterministic noises worsen than the worst case. This contribution comes from an identification of the class of deterministic noise which causes the worst situation.

The behavioral model this chapter proposes utilizes a Markov chain, and it is used to quickly estimate the worst χ value of a poker test (defined in the FIPS 140-2 [86]) under any deterministic effects without generating a bit stream. The principal design parameters, which are average periods of oscillators, the duty cycle of the fast oscillator, and the use of correctors are determined guided by the estimated χ values and target χ values. The quality of TRNG outputs could be quantitatively evaluated with other standard randomness tests as well when necessary since the model could also generate a bit stream. It should be noted that the proposed methodology can be applied to all types of oscillators, since the parameters of interest are independent of the topology or type of oscillator. Furthermore, this chapter tests and validates the proposed model with hardware measurements of an oscillator-based TRNG implemented with a 65 nm CMOS process.

The three main contributions are:

- to propose the worst-case-aware design methodology to guarantee sufficient randomness under deterministic noise,
- to verify the efficiency of the worst-case-aware design methodology with gate-level TRNG simulation, and
- to identify the class of deterministic noise under which the randomness gets close to the lowest.

The rest of this chapter is organized as follows. Section 2.2 proposes a behavioral model with a Markov chain and a procedure for evaluating randomness. Section 2.3 explains the validation of the model with hardware measurements. Section 2.4 proposes a methodology to calculate randomness in the worst case without bit generation. The efficiency of the proposed design methodology is proven with gate-level noise-aware TRNG simulation, which is explained in Section 2.5. Section 2.6 presents an example to illustrate how appropriate design parameters are derived and Section 2.7 is the conclusion.

2.2 Proposed stochastic behavior model

This section proposes a behavioral model of oscillator-based TRNGs using a Markov chain.

2.2.1 Behavioral model of oscillator-based TRNG

A Markov chain is a discrete-state/discrete-time stochastic process, $\{X_n\} = \{X_0, X_1, X_2, \ldots\}$, where $\{X_n\}$ is a sequence of random variables, which satisfies, for each r, a Markov property,

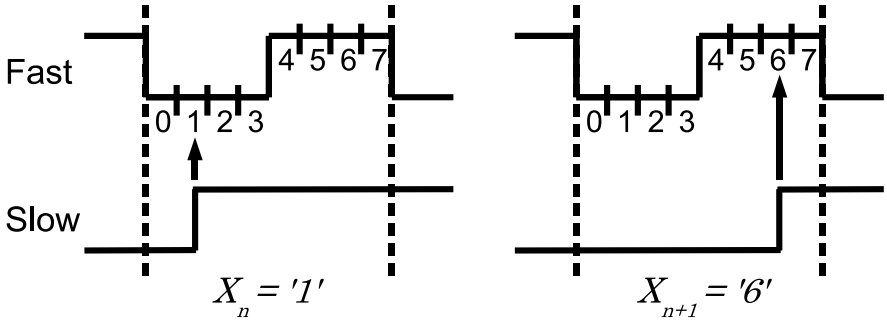

Figure 2.1: Application example.

i.e., [113]

$$P(X_r = x_r | X_{r-1} = x_{r-1}, X_{r-2} = x_{r-2}, \ldots, X_0 = x_0) = P(X_r = x_r | X_{r-1} = x_{r-1}). \tag{2.1}$$

This means that next state X_{n+1} only depends on current state X_n and is independent of past states $X_0, X_1, \ldots, X_{n-1}$.

Before explaining the proposed model, the author will describe some assumptions about the model. This section have assumed that jitter in the oscillators is temporally independent, as described Sec. 1.2.1. This assumption means that the section takes into consideration thermal noise, shot noise, and/or 1/f noise but not deterministic noise (this will be considered in Section 2.4) such as power-supply noise, substrate noise, and external noise. Given this assumption, a Markov chain can be applied to the behavioral modeling of an oscillator-based TRNG.

The fast oscillator waveform of one cycle is divided into m spans and each span is regarded as a state in the proposed model. Thus, a Markov chain that has m-state space is constructed. Let us suppose the n-th rising edge is the timing of a slow oscillator. Here, the proposed model defines this timing as time n. The fast oscillator at this rise timing stays in one state of the m states defined above. The X_n denotes the state at time n. The TRNG generates the n-th bit corresponding to X_n, since each state corresponds to low or high. Figure 2.1 outlines an example where the model is applied to a TRNG where $m = 8$. The TRNG takes state 1 at time n and state 6 at time $n + 1$, and then $X_n = 1$ and $X_{n+1} = 6$. In this example, since states $0, 1, 2$, and 3 are low and states $4, 5, 6$, and 7 are high, the n-th output is 0 and the $(n + 1)$-th output is 1.

2.2.2 Model construction and use

This subsection explains the process of evaluating randomness with the Markov model. 1) The transition matrix and 2) state probability vector are calculated, and then 3) random bit streams are generated and evaluated with statistical randomness tests. Each step is explained in what follows.

Calculation of transition matrix

This step is to construct transition matrix $\mathbf{P}$ that characterizes the state transition of the Markov chain. The matrix size is $m \times m$ when the model has m-state space. An element of

matrix $p_{i,j}$ is the probability of a transition from i to j ($0 \leq i, j \leq m-1$). Transition step a is the number that the state proceeds to and is defined as $\{(j-i)+m\} \mod m$. Let $q_i(a)$ denote the probability that the next state will advance by a from state i. Assuming a Gaussian distribution, $p_{i,i+a}$ is calculated as:

$$p_{i,i+a} = \sum_{l=-\infty}^{\infty} q_i(a + l \cdot m)$$

$$= \sum_{l=-\infty}^{\infty} \int_{l \cdot t_{\text{fast}} + a \cdot t_{\text{span}}}^{l \cdot t_{\text{fast}} + (a+1) t_{\text{span}}} f_i(x) dx, \tag{2.2}$$

$$f_i(x) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x-\mu)^2}{2\sigma_i^2}\right), \tag{2.3}$$

where t_{fast} is the average period for the fast oscillator, and t_{span} is the time range for one state and is defined as t_{fast}/m. The $f_i(x)$ is the probability density function of a Gaussian distribution whose standard deviation, σ_i, depends on the current state. Note that the proposed model can handle any other distribution shapes as long as they are independent of time, even though a Gaussian distribution has been adopted as a representative shape in this section. The μ is a remainder where the average period for slow oscillator t_{slow} is divided by that of the fast oscillator. It is most likely that the next timing for sampling will advance by μ from the current. The next sampling timing is distributed more uniformly as σ_i increases. The $p_{i,j} (i > j)$ can also be obtained from Eq. (2.2) since $p_{i,j+m}$ is equal to $p_{i,j}$ while extending the maximum range of j. Thus, **P** can be derived with Eq. (2.2).

Let us explain Eq. (2.2) using the situation in Fig. 2.1 as a simple example, where m is 8, X_n is '1', and X_{n+1} is '6'. Figure 2.2 explains the summation and integration in Eq. (2.2). When t_{fast} is sufficiently large, i.e., $t_{\text{fast}} >> \sigma_i$ (top of Fig. 2.2), $p_{1,6}$ is approximately obtained as $q(6-1) = \int_{5 \cdot t_{\text{span}}}^{6 \cdot t_{\text{span}}} f(x)dx$. However, as t_{fast} is comparable to or smaller than σ_i (bottom of Fig. 2.2), $q(5 + 8 \times (-1)), q(5 + 8 \times 1), \cdots$ should not be ignored. As σ_i becomes relatively larger than t_{fast}, more terms of q should be summed up, and finally Eq. (2.2) is obtained.

To easily take jitter from both oscillators into consideration, this section introduces a variance constant and equivalent jitter. An oscillator is composed of stage elements (called gates after this), such as inverters, and the jitter characteristics of gates are an important factor in their design. To discuss this factor, this section defines variance constant r as the variance in the stage delay divided by the average stage delay. Due to this definition, the variance constant of an oscillator composed of n gates with r variable constants is conveniently equal to r. The variance constant characterizes the jitter of an oscillator. The variance constant of the fast oscillator, r_{fast}, and that of the slow oscillator, r_{slow}, can differ. For instance, the variance of periods of the slow oscillator is $r_{\text{slow}} t_{\text{slow}}$.

Next, Fig. 2.3 shows an example of waveforms to explain equivalent jitter. Equivalent jitter σ is the time fluctuation between the rise edge of the slow oscillator and the previous rise edge of the fast oscillator, and is defined as the standard deviation of the time span between the two edges (denoted as T_{diff}). The fluctuation of T_{diff} results from the jitter of the slow rising edge at t_{slow} and of the fast rising edge at $\delta + N t_{\text{fast}}$. Here, δ is the initial time difference between the oscillators, and the jitter for the slow edge is $r_{\text{slow}} t_{\text{slow}}$. N cycles of fast oscillation elapse per cycle of the slow signal where $N = \lfloor \frac{t_{\text{slow}} - \delta}{t_{\text{fast}}} \rfloor$, and then jitter accumulates in the meantime, since t_{slow} is larger than t_{fast}. Then, the variation in the rise edge at $\delta + N t_{\text{fast}}$ is $r_{\text{fast}} (\delta + N t_{\text{fast}})$.

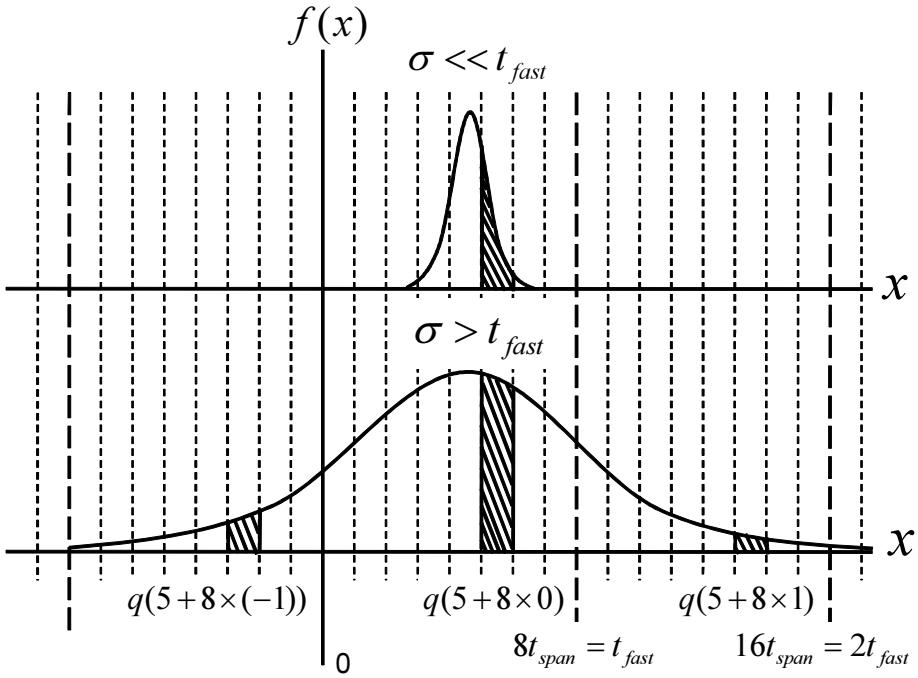

Figure 2.2: Example calculation of transition matrix.

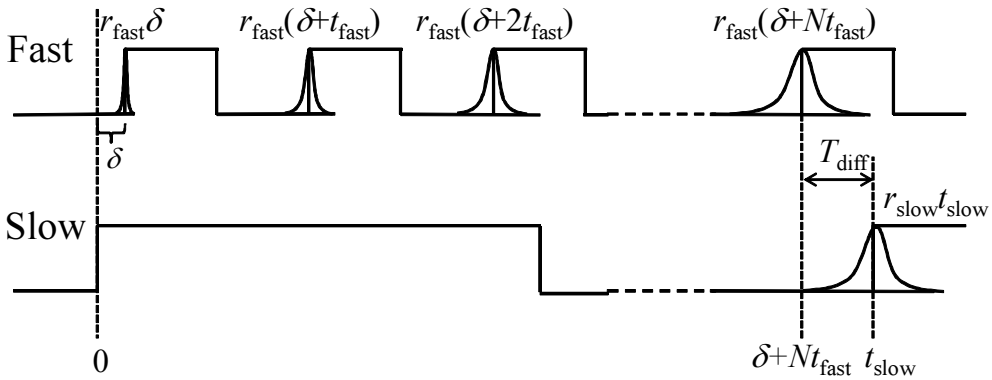

Figure 2.3: Jitter accumulation by fast oscillator. Variance of each rise timing is denoted. ($\lfloor \frac{t_{\text{slow}}-\delta}{t_{\text{fast}}} \rfloor = N$).

The two oscillators have different circuits, and hence the rise edges at t_{slow} and $\delta + Nt_{\text{fast}}$ are independent of each other. Therefore, the variance in T_{diff} is $r_{\text{slow}}t_{\text{slow}} + r_{\text{fast}}(\delta + Nt_{\text{fast}})$. When the current state is i in the Markov chain, the initial time difference is approximated as $\delta \approx t_{\text{fast}} - it_{\text{span}}$. The error in this approximation, which degrades the accuracy of the model, gets smaller as the size of state-space $m = t_{\text{fast}}/t_{\text{span}}$ increases, because the error is always less than t_{span}. Finally, equivalent jitter is expressed as:

$$\sigma_i = \sqrt{r_{\text{slow}}t_{\text{slow}} + r_{\text{fast}}t_{\text{fast}}\left(N + 1 - \frac{i}{m}\right)}. \tag{2.4}$$

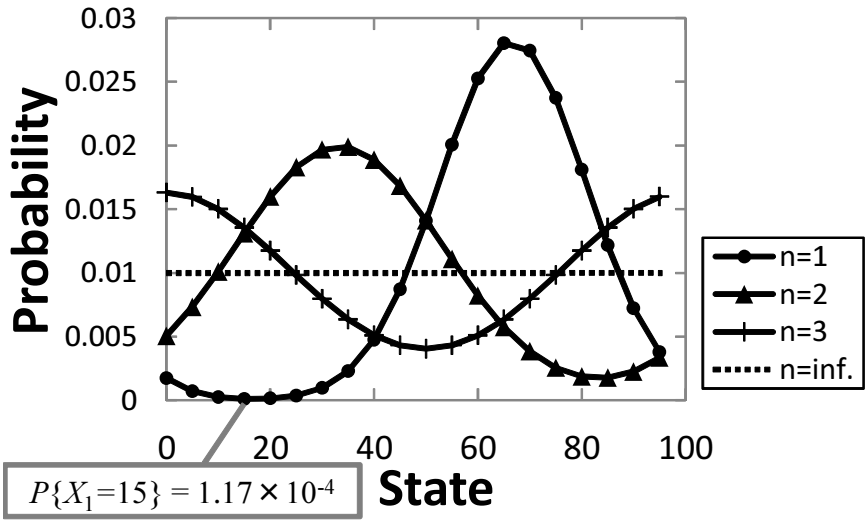

Figure 2.4: State probability vectors with progression of time.

The parameter of m affects the accuracy and run time for evaluation. To precisely the model behavior, $t_{\text{span}}(= t_{\text{fast}}/m)$ should be sufficiently smaller than σ_i. The size of m in the experiments will be discussed in Section 2.6.4.

Calculation of state probability vector

Given the transition matrix, the next state probability vector, $\boldsymbol{\pi_{n+1}}$, is calculated from the current one, $\boldsymbol{\pi_n}$ as:

$$\begin{aligned}\boldsymbol{\pi_{n+1}} &= \begin{pmatrix} P\{X_n=0\} \\ P\{X_n=1\} \\ \vdots \\ P\{X_n=m-1\} \end{pmatrix}^T \begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,m-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m-1,0} & p_{m-1,1} & \cdots & p_{m-1,m-1} \end{pmatrix} \\ &= \boldsymbol{\pi_n}\mathbf{P}. \end{aligned} \tag{2.5}$$

Transition matrix $\mathbf{P}$ is independent of time n because of the Markov property, and hence $\boldsymbol{\pi_n}$ can be calculated with initial state probability vector $\boldsymbol{\pi_0}$; $\boldsymbol{\pi_n} = \boldsymbol{\pi_0}\mathbf{P}^n$. Figure 2.4 plots an example of $\boldsymbol{\pi_n}$ transitions where the initial states are 0s. Because the average periods of the fast and slow oscillators are 0.3 ns and 50 ns, μ is calculated as 50 ns mod 0.3 ns = 0.2 ns. The variance constants of the oscillators are both 1.8×10^{-14} s.

Bit generation and randomness tests

Duty cycle d is defined as the ratio of the number of states that are high to the number of all states m in the model. For example, when states 0 to 29 are low and states 30 to 99 are high ($m = 100$), d is $(70/100) \times 100 = 0.7$. When the next state probability vector, which can be obtained from the current state, and the duty cycle are given, the next state and the next output are stochastically determined with pseudo-random numbers generated by computer. Repeating this process generates a successive bit stream. Randomness is evaluated by testing the generated bit stream with arbitrary statistical tests.

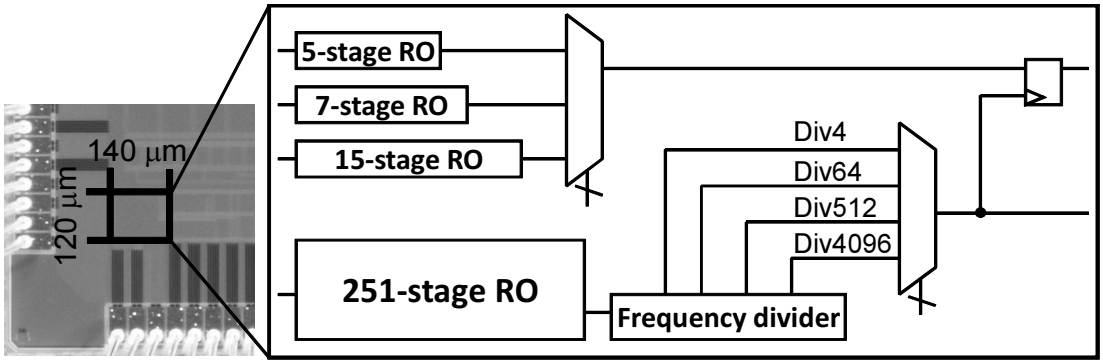

Figure 2.5: Chip photo and block diagram of TRNGs.

Postprocessing with correctors (e.g., the XOR [97] and von Neumann correctors [96]) is a popular technique to improve randomness, as referred in Sec. 1.1.4. When the bit stream is generated by the model, arbitrary correctors can simply be applied to the random numbers and statistical tests are then executed.

Besides, it is difficult to analytically evaluate randomness using probability computations without generating bit streams. For example, calculation of χ of a poker test requires the four successive probabilities of '1' occurrence, $(p_n, p_{n+1}, p_{n+2}, p_{n+3})$. The probabilities, however, depend on the state at $n-1$, which is not uniquely determined. Consequently, bit generation with simulating the state transitions and statistical randomness evaluation are necessary. Note that evaluation of worst-case randomness, which will be presented in Sec. 2.4, does not require bit generation.

2.3 Model validation with hardware measurements

The proposed Markov model is implemented with MATLAB [114], and validated with measurements of a prototype TRNG fabricated with a 65 nm process.

2.3.1 Test structure

Figure 2.5 shows the test TRNG, chip photos, and block diagrams. The test TRNG was fabricated with e-shuttle 65 nm process. This test chip includes 5-, 7-, and 15-stage ring oscillators (ROs) as fast oscillators using standard cells with minimum channel length. A 251-stage RO of the slow oscillator is composed of low-leakage standard cells with 10-nm longer channel length. All stage elements of the ROs are static CMOS inverters and 2-input NAND gates. The periods of the ring oscillators are 178.3 ps for the 5-stage RO, 243.2 ps for the 7-stage RO, 502.6 ps for the 15-stage RO, and 10.0 ns for the 251-stage RO from the circuit simulations. Four-, 64-, 512- and 4096-frequency-dividers are also implemented as slow oscillators.

A body biasing technique is adopted to finely tune the duty cycle of fast oscillators. Although a frequency divider could adjust the duty cycle, a perfect duty cycle of 50% does not necessarily result in a balanced occurrence of 1/0 due to the input offset of the sampler. Figure 2.6 shows an example of duty-cycle adjustment when four body voltages (VNW_A,

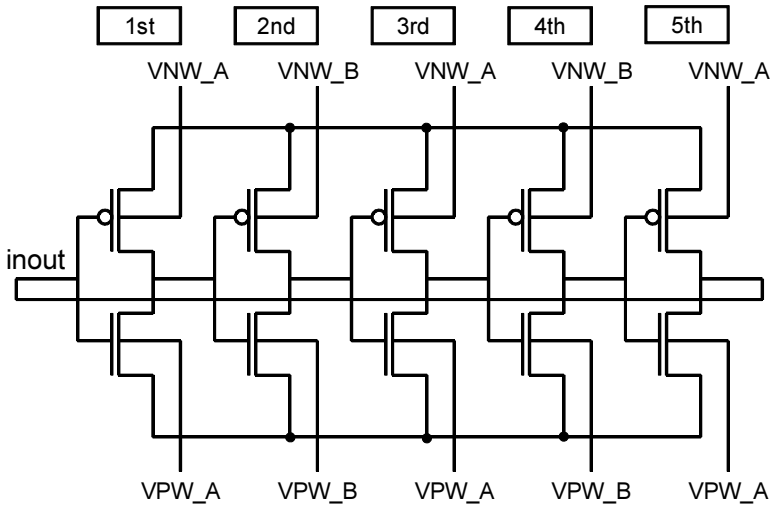

Figure 2.6: Adjustment of duty cycle with body-biasing technique.

VNW_B, VPW_A, and VPW_B) are applied to every other inverter in the 5-stage RO. The time when **inout** is high depends on the delay of the NMOSs in the 1st, 3rd, and 5th inverters, and the PMOSs in the 2nd and 4th inverters. The time for low is complementarily affected by the other MOSs. The duty cycle increases when forward biases are applied to VNW_A and VPW_B, and reverse biases are applied to VNW_B and VPW_A, so that the time for high increases and the time for low decreases. Thus, the duty cycle could be freely chosen by changing the four voltages. This work assumes that the body voltages are provided separately and isolated from VDD, which means deterministic VDD noise does not affect body voltages.

2.3.2 Metric of randomness

The randomness evaluation is mainly carried out with a poker test in this research, because the result of a poker test can easily be calculated in the worst case evaluations of randomness presented in Section 2.4. However, several randomness tests have been proposed such as the NIST test suite, Diehard tests and FIPS140-2 tests, as described in Sec. 1.1.3. Even though the NIST tests and Diehard tests are preferred for testing whether test data are sufficiently random or not, they are difficult to use in comparing multiple test streams because they return many p-values as scores for a test stream. Therefore, entropy of a bit stream is widely used for evaluating randomness variations or differences. Figure 2.7 compares a poker test and an entropy test. The poker test returns χ values as results, and a pass mark of χ is $2.16 < \chi < 46.17$. The vertical axis for the χ value is inversed because smaller χ indicates higher randomness. The figure shows that the χ of the poker test is well correlated with the entropies, which indicates that the poker test can be used for approximate evaluations of randomness.

2.3.3 Validation with poker test

One hundred sequences of 20-k random bit streams were generated with the test chip and measured with a logic analyzer. This section also generated the same number of bits using the

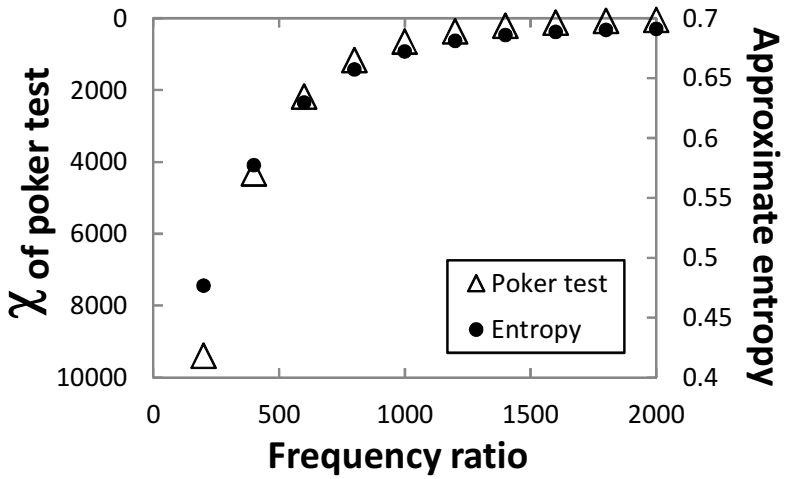

Figure 2.7: χ of poker test vs. approximate entropy.

proposed model. The size of state space was set to 100. The other parameters for the model were determined as follows. This experiment measured the periods of the slow oscillator (the 251-stage ring oscillator with 64-frequency divider) with a real time oscilloscope, and then estimated the variance constant from the measured periods. In this measurement, the fast oscillator was stopped. The trigger jitter of the employed real-time oscilloscope (Tektronix DPO70804 [115]) is 1 ps_{rms}, and it is negligibly small because the measured jitter of 251-stage ring oscillator with 64-frequency divider was 113 ps. The estimated variance constant was 2.6×10^{-14} s*. On the other hand, since it is difficult to directly measure the signal of the fast oscillator, we obtained the average period from circuit simulation and estimated the jitter of the fast signal assuming that the fast and the slow oscillators have the identical variance constant. Namely, the variance of the fast signal was calculated as the variance constant multiplied by the average period which was from circuit simulation. For the same reason, the duty cycle of the fast oscillator was estimated from measurements assuming that 1/0 probability represented the duty cycle [28], instead of direct waveform measurement. Strictly speaking, the probability of '1' occurrence has a little difference from the actual duty cycle because of the input offset of the sampler. On the other hand, they are highly correlated, and then this section regarded the probability of '1' occurrence of the output bit streams as the duty cycle of the fast oscillator.

Figure 2.8 plots the measured and simulation results for the poker test with 5- and 15-stage fast oscillators. The horizontal axis plots the frequency ratio of the oscillators, and it was varied by changing the configuration of the frequency divider. The duty cycle for the fast oscillators were adjusted to within 50±3% by body biasing. The results for both simulations and measurements in Fig. 2.8 show that increasing sampling sparseness s, which means that the sampler captures data once per s rising edges of the clock, viz., enlarging the jitter of the slow oscillator [116], improves the quality of random bit streams. The χ value for the 5-stage ring oscillator estimated by the model satisfies the pass mark when the frequency ratio is

* The variance constant is different from that used in the other sections and the other chapters, since the measured slow oscillator consisted of the low-leakage standard cells while the oscillators in other section consisted of the standard cells with minimum channel length.

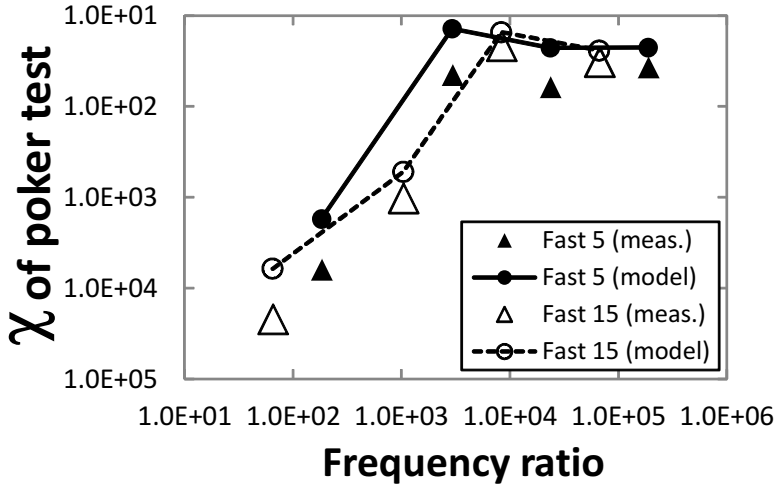

Figure 2.8: Randomness vs. sampling sparseness and fast average period.

2933 and higher. On the other hand, the frequency ratio of 8310 is necessary for the 15-stage ring oscillator. Thus, the fast oscillator with higher frequency reduces the frequency ratio to pass the test, and consequently increases the throughput of the TRNG.

Figure 2.9 plots the poker test results obtained by changing the duty cycle for the fast oscillators. This evaluation used the 7-stage ring oscillator as the fast oscillator and employed the 512-frequency-divider. In this experimental configuration, the frequency ratio between the oscillators was large enough to pass the poker test. The duty cycle for the fast oscillator varied from 44 % to 58 %. The same figure indicates that the unbalanced duty cycle for the fast oscillator degrades randomness. The simulations and the measurements are well correlated, which means the analysis using the proposed model is valid. Also, this result exemplifies that a fast oscillator with unbalanced duty cycle limits the randomness of a TRNG even when the frequency ratio is large enough.

2.4 Estimation of worst-case randomness

This section proposes a methodology of evaluating the worst χ value of a poker test under deterministic noise that utilizes a Markov model but does not require any bit generation.

2.4.1 Consideration of deterministic noise

Deterministic noise (e.g., power-supply noise, substrate noise, and external noise) induces deterministic fluctuations in the rise timings of oscillators, and it appears as variations of μ in Eq. (2.3) and representative phases x_0. Here, the representative phase is defined as the time interval from the rising edge of a fast oscillator to the rise timing of a slow oscillator immediately after the fast oscillator without any random jitter, and then $0 \leq x_0 < t_{\text{fast}}$. Assuming that jitter has a Gaussian distribution, the representative phase is equal to the average of T_{diff} in Fig. 2.3. Figure 2.10 illustrates a representative phase for oscillating signals.

A state that contains a representative phase corresponds to the mode of states. For example,

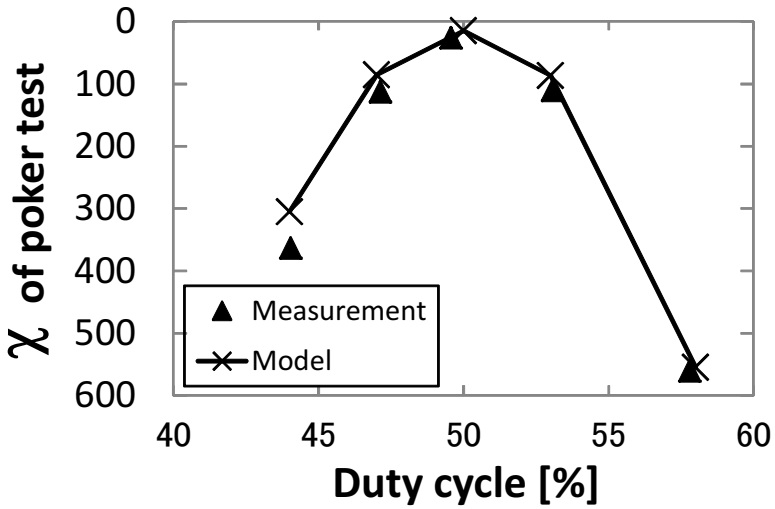

Figure 2.9: Randomness vs. duty cycle.

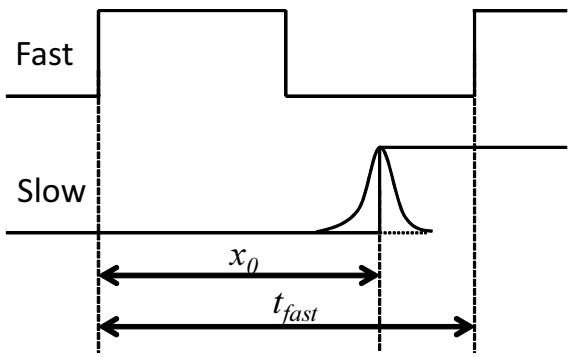

Figure 2.10: Example of representative phase.

$n = 1$ in Fig. 2.4 indicates that the mode value is 65 and the representative phase is $t_{\text{fast}} \times 64/100 \leq x_0 < t_{\text{fast}} \times 65/100$. Further, the 1/0 occurrence of a bit stream is the most biased where the representative phases of cycles are fixed to the same value, which is the center number of high/low states. Thus, the worst case under deterministic noise corresponds to a condition where every representative phase is fixed at the middle state of the low states (duty cycle < 0.5) or high states (duty cycle > 0.5). This middle state will be denoted by s_{middle} after this. The proof can be found in the Appendix A.

The discussion in this section focuses on the class of harmful noise rather than waveform shapes and how to deliver such harmful noise to TRNG. The proposed estimation of the worst-case randomness considers the theoretically worst situation without investigating waveform shapes of the deterministic noise. On the other hand, it is difficult to associate the worst case with the physical attacking method since the noise delivery through physical attacking is totally dependent on the implementation. The mapping of the worst-case to attacking way is another interesting topic to study and one of the future works.

2.4.2 Worst evaluation of χ

Assuming the worst case discussed above, this section estimates the worst χ value under deterministic noise using the Markov model. Additional constraints are given to calculations of the transition matrix and state probability vector to estimate the worst χ value.

First, initial vector π_0 in Eq. (2.5) is set so that the probability of state s_{middle} is 1 and the probabilities of the other states are 0. For example, when states 0 to 29 are low and states 30 to 99 are high ($m = 100$), the initial probability of state 64 (or state 65) is 1 and the others are 0. The representative phases with this constraint are fixed and the bias of 1/0 occurrence becomes the largest, which makes the randomness of output the lowest. Then, μ in Eq. (2.3) is fixed to 0 and does not depend on the periods of oscillators, which means that the representative phase does not change cycle by cycle.

The reason why bit generation is not required is that the temporally-successive probabilities of '1' occurrence can be directly calculated using the proposed model. The computation of χ requires the four successive probabilities of '1', $(p_n, p_{n+1}, p_{n+2}, p_{n+3})$, where the probabilities depend on the state at $n-1$, i_{n-1}. On the other hand, from the discussion in Sec. 2.4.1, for the worst case evaluation, i_{n-1} can be fixed to s_{middle}. In this case, $(p_n, p_{n+1}, p_{n+2}, p_{n+3})$ can be subsequently computed. Note that the representative phase is fixed in the transition matrix computation for the worst case evaluation. Consequently, the consideration of the worst case enables the worst χ value evaluation without bit generation.

Once the state probability vector is computed, the worst χ value can be directly calculated with the proposed model without bit generation. The probabilities of '1' occurring at successive outputs, $p_1, p_2, \cdots$, are calculated with the corresponding state probability vectors and duty cycle. Note that p_n is not independent of $p_1, p_2, \cdots p_{n-1}$ and the correlation with the past bit stream is taken into consideration in calculating the state probability vector. The worst χ is computed from [86] as:

$$\chi = \frac{16}{5000} \times \sum_{i=0}^{15} (5000 \times \xi_i)^2 - 5000, \tag{2.6}$$

where ξ_i is the probability that the four successive bits will be equal to i. For instance, ξ_{10} is the probability of $(1010)_2$ and is described as $p_4(1-p_3)p_2(1-p_1)$. The ξ_i can be calculated with the Markov model from the probabilities of the occurrences of '1' or '0', which differs from the conventional way of counting each i in long generated bit sequences.

2.4.3 Corrector considerations

To estimate the worst χ with a corrector, the probabilities of '1' occurring after correction, p'_n, need to be computed from p_n. The p'_n can be computed as $p'_n = p_{2n-1}p_{2n} + (1-p_{2n-1})(1-p_{2n})$ for the XOR corrector, whereas the Von Neumann corrector is difficult to apply since it may discard bits boundlessly and computing p'_n with it is not easy.

2.5 Validation of worst case-aware design

This section experimentally confirms that the proposed worst case computation guaranteed the worst χ with gate-level TRNG simulations by taking power-supply noise into account.

Here, the discussion on attacks is not supported by measurement, since it is difficult to accurately inject the noise that the author really wants to give due to measurement environment. For example, though on-chip noise generator [117] could generate power supply noise, it is difficult to control the waveform of the noise due to the distortion by the packages, the bonding wires, and the decoupling capacitances. On the other hand, the simulation is preferable to the chip measurement for analyzing the impact of deterministic noise, because the simulation can inject the exact noise. The section therefore has implemented the gate-level TRNG simulator, which enables us to flexibly control the waveform, frequency, and amplitude of the injected noise.

2.5.1 Simulations considering deterministic noise

A gate-level simulator that takes into consideration fluctuations in all gate delays is developed. Each gate delay is denoted as $t_{d,(gate)}$.

$$t_{d,(\mathrm{gate})}(t) = t_{d,\mathrm{offset},(\mathrm{gate})}(\mathrm{Vdd}(t)) + t_{d,\mathrm{random}}, \tag{2.7}$$

$$t_{d,\mathrm{offset},(\mathrm{gate})}(t) = \frac{a_{(\mathrm{gate})}}{(\mathrm{Vdd(t)} - \mathrm{Vth}_{(\mathrm{gate})})^{\alpha_{(\mathrm{gate})}} + b_{(\mathrm{gate})}}, \tag{2.8}$$

where (gate) denotes the types of gates. The $t_{d,\mathrm{offset},(\mathrm{gate})}$ is the gate delay without any random noise. To express the dependence of delays on supply noise, this section uses a gate-delay model (Eq. (2.8)) based on an alpha-power law MOSFET model [118]. Parameters $a_{(\mathrm{gate})}$, $b_{(\mathrm{gate})}$, $\alpha_{(\mathrm{gate})}$, and $\mathrm{Vth}_{(\mathrm{gate})}$ are obtained by fitting them to the results from circuit simulations. $\mathrm{Vdd}(t)$ represents the function of a noise-induced supply voltage waveform. The $t_{d,\mathrm{random}}$ represents a random timing fluctuation originating from random noise, and it is calculated as Gaussian random number whose average is zero and variance is $r \times t_{d,\mathrm{offset},(\mathrm{gate})}$ where r is the variance constant of the oscillators.

Figure 2.11 explains three-step bit generation, denoting the first rise timings of the fast and the slow ROs as $t_{(1)\mathrm{FAST}}$ and $t_{(1)\mathrm{SLOW}}$ and the timings of n-th rising edges as $t_{(n)\mathrm{FAST}}$, $t_{(n)\mathrm{SLOW}}$. 1) Calculate the next timing for the rising edge of slow RO $t_{(2)\mathrm{SLOW}}$ from current rising timing $t_{(1)\mathrm{SLOW}}$. 2) From $t_{(1)\mathrm{FAST}}$ and $t_{(1)\mathrm{SLOW}}$, find $t_{(n)\mathrm{FAST}}$ that satisfies equalities $t_{(n-1)\mathrm{FAST}} < t_{(2)\mathrm{SLOW}} < t_{(n)\mathrm{FAST}}$. 3) Generate one bit from $t_{(2)\mathrm{SLOW}}$, $t_{(n-1)\mathrm{FAST}}$, and $t_{(n)\mathrm{FAST}}$ taking into account the duty cycle of the fast RO.

The time interval between the successive rising edges is the sum of $t_{d,(\mathrm{gate})}(t)$ for two rounds of the slow oscillator. Additionally, when a frequency divider is used for the slow oscillator and the sampling sparseness is s, the gate delays for $2s$ rounds are summed.

2.5.2 Simulation results

Randomness under supply noise of various frequencies is evaluated with the simulator, and compared to the worst randomness estimated with the Markov model.

Figure 2.12 plots the poker test results (a) without any deterministic noise and under power-supply noise. The figure also plots the worst χ values estimated with the Markov model. This section generated 100 sequences of 20-k bits for the test. (b) One hundred kilohertz, (c) 10 MHz, and (d) 100 MHz of sinusoidal noise whose amplitudes were 100 mV were superposed to the DC supply voltages of 1.2 V for ROs. This evaluation used a 5-stage RO whose average period was 178.3 ps and duty cycle was 51% at 1.2 V as a fast RO. The r

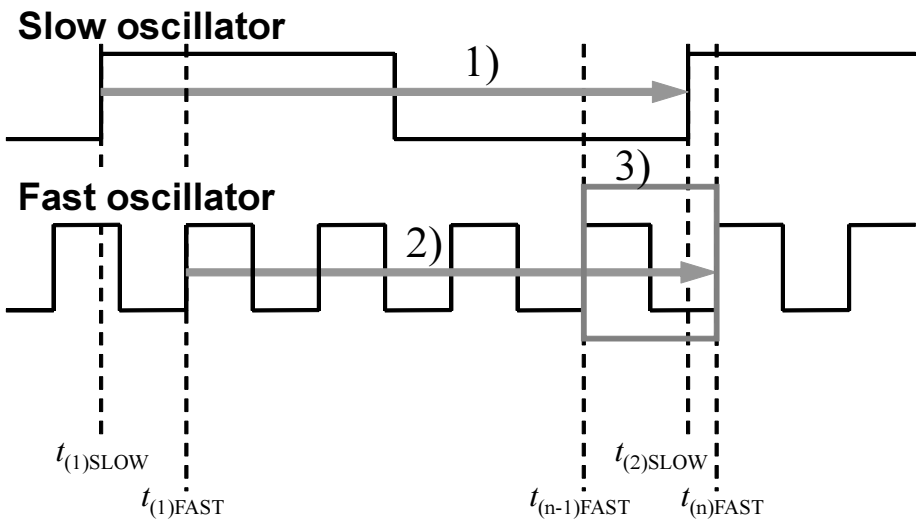

Figure 2.11: Concept behind noise-aware gate-level simulation.

of the ROs was 1.77×10^{-14} s referring to the results obtained from measurements of ROs fabricated with a 65 nm process. Equivalent jitter for the Markov model was calculated with r and the periods of ROs. Figure 2.12 indicates that randomness depends on the frequency of deterministic noise. It can also be seen that randomness under or without deterministic noise is not worse than the results estimated as being the worst case, which verifies the idea of the worst χ evaluation in Section 2.4. The χ values in Fig. 2.12 fluctuate for the low frequency ratios. The fluctuation is caused by μ in Eq. (2.3) and the deterministic noise, since they shift the representative phases x_0. Therefore, the impact of the deterministic noise on the χ value varies depending on the frequency ratio and the noise frequency.

Figure 2.13 shows the χ of a poker test when the period of sinusoidal noise was finely varied. The figure also shows the worst χ value and the pass mark for the poker test. Ten sequences of 20-k bits were generated with the simulation. Five-stage RO, whose duty cycle was 51%, and 251-stage RO with a 9-frequency-divider, whose average period was 73.4 ns, were used as the fast and slow ROs. The r of ROs was 1.77×10^{-14} s. It can be seen that the pass/fail for the poker test depends on the period of deterministic noise. In addition, Fig. 2.13 indicates that the χ values can approach the worst case especially as the period of power-supply noise decreases. Thus, the risk that deterministic noise will degrade randomness to the worst case should not be ignored. The proposed worst case-aware design methodology effectively guarantees randomness even under unwanted noise.

2.6 Exploration of design space with proposed model

This section presents an example to illustrate how to derive appropriate design parameters with the proposed worst-case-aware design methodology. Here, the design space consists of the frequencies of oscillators, the duty cycle of a fast oscillator, and whether the TRNG employs correctors. The design space is explored by evaluating the χ of a poker test when changing the design parameters within feasible values. In Section 2.6.1, only frequencies of

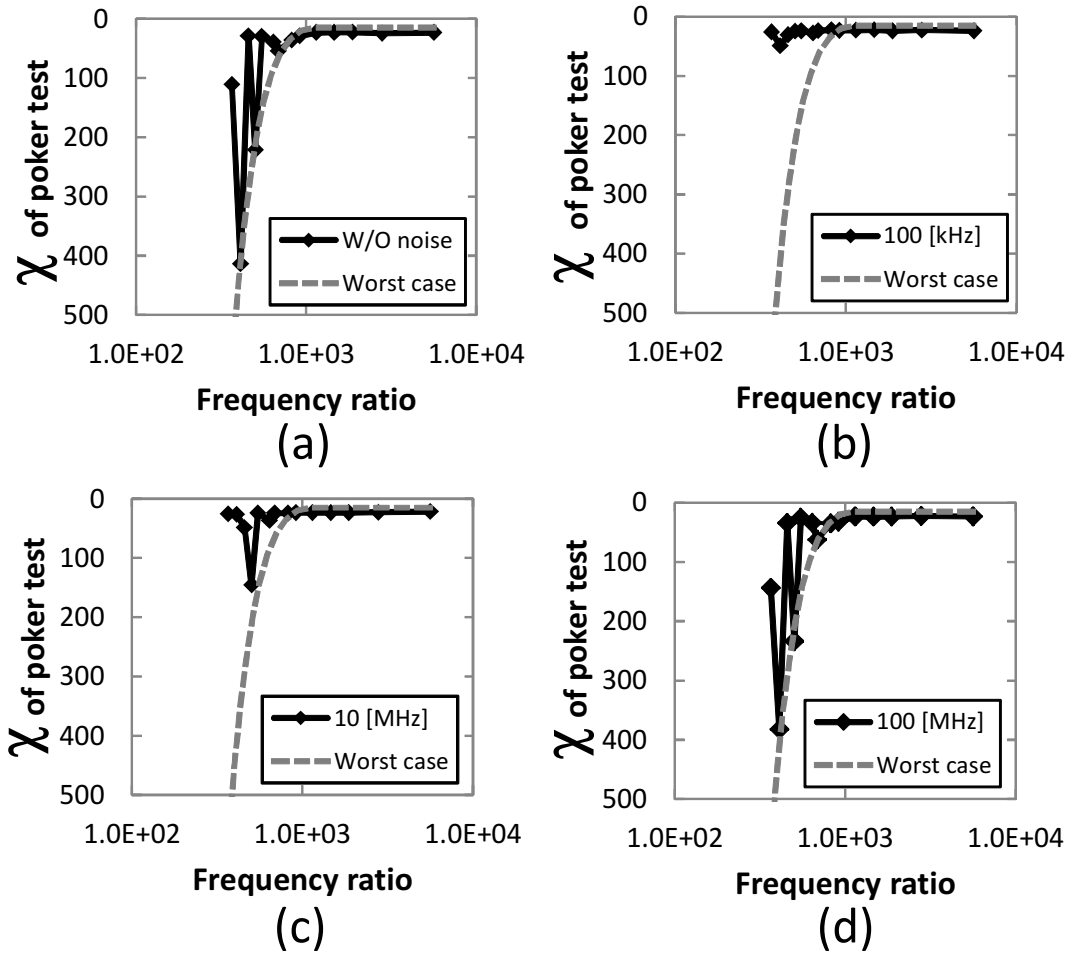

Figure 2.12: Evaluation of randomness under and without deterministic noise.

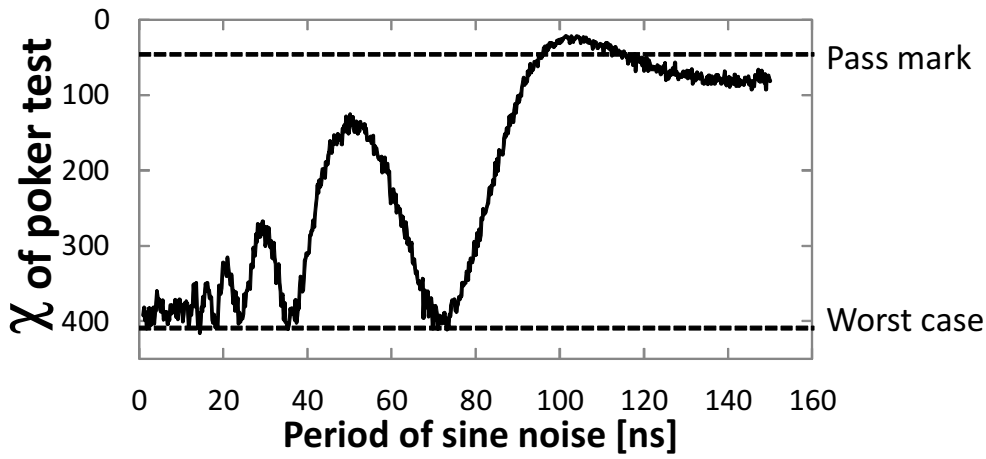

Figure 2.13: Evaluation of randomness under deterministic noise with various periods.

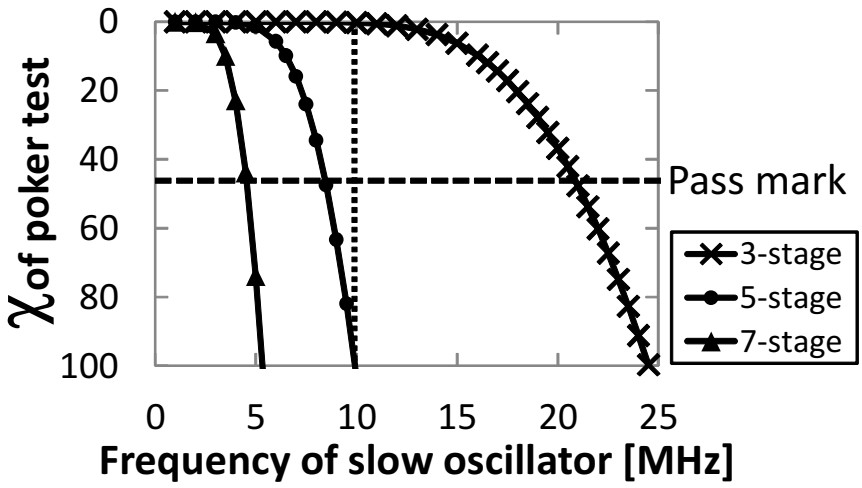

Figure 2.14: Evaluation of randomness to design fast and slow oscillators.

oscillators are explored as a simple example. The allowable shift of duty cycle from 50% can be investigated, though it is not included in this example. Then, Section 2.6.2 shows how to consider the XOR corrector. Section 2.6.3 demonstrates that our design methodology can consider the injection locking attack by changing variance constant. Section 2.6.4 discusses the required size of state space for valuable evaluation as a supplement.

2.6.1 Design of fast and slow oscillators

This subsection explains the design of fast and slow oscillators for TRNG with the given design constraints to follow and circuit information on a 65 nm CMOS process. The variance constant of each gate r is 1.77×10^{-14}s, which was derived from the measurements of ROs in a 65 nm process. Since it is self-evident that the most advantageous duty cycle for the fast oscillator, which is equal to the 1/0 probability here, is 50 %, the duty cycle does not need to be explored. The actual duty cycle of the fabricated oscillator has some error from the optimal value due to process variations. Therefore, the duty cycle of the fast RO is within 50±0.05 % here. To simplify the discussion, no correctors have been employed. Ten million bits per second, which is a typical value in a smart card [47], or higher throughput, are required.

First, the periods for several oscillators that could be used as fast ROs were estimated by simulating the circuits. Here, fast oscillators with different numbers of stages (3, 5, and 7) were evaluated for the sake of simplicity, and their periods corresponded to 113.5 ps, 178.3 ps, and 243.2 ps. Second, the worst χ values for TRNGs with each of the fast ROs were evaluated with the Markov model varying the frequencies of slow ROs.

Figure 2.14 plots the estimated χ values. The duty cycle was set to 50.05% assuming the least preferable case. This section estimated achievable throughputs for all fast ROs. Now that the required throughput is 10 Mbps, the number of stages of fast ROs should not exceed five. When a 3-stage RO is adopted, the number of stages of slow ROs is determined so that the slow oscillator frequency is 20.5 MHz or less. However, for a 7-stage RO, the frequency should not exceed 4.5 MHz, which means more stages, i.e., a larger area is necessary, and furthermore multiple TRNGs are needed to satisfy these requirements.

To further investigate randomness, 100 Mbits of bit streams from the model were evaluated

Table 2.1: NIST randomness test results. p-value/pass proportions have been listed in each cell. Bold fonts have been used for passed tests.

Test name	21 [MHz]	10 [MHz]
Frequency	**0.1281 / 0.99**	**0.0051 / 0.98**
BlockFrequency	**0.0037 / 0.99**	**0.1626 / 0.99**
CumulativeSums	**0.0805 / 0.99**	**0.0010 / 0.97**
Runs	0.0000 / 0.00	**0.0478 / 0.99**
LongestRun	0.0000 / 0.04	**0.1917 / 0.96**
Rank	**0.3669 / 0.98**	**0.8514 / 1.00**
FFT	0.0000 / 0.70	**0.9781 / 0.99**
NonOverlappingTemplate	0.0000 / 0.00	**0.0060 / 0.98**
OverlappingTemplate	0.0000 / 0.00	**0.0118 / 0.99**
Universal	0.0000 / 0.03	**0.3345 / 0.98**
ApproximateEntropy	0.0000 / 0.00	**0.6163 / 0.99**
RandomExcursions	0.0000 / 0.95	**0.1088 / 1.00**
RandomExcursionsVariant	**0.0106 / 0.98**	**0.0352 / 1.00**
Serial	0.0000 / 0.00	**0.1453 / 1.00**
LinearComplexity	**0.9241 / 1.00**	**0.1223 / 0.99**

by using the NIST test program. The fast oscillators were 3-stage ROs and the frequencies of the slow oscillators were 10 MHz and 21 MHz. As Fig. 2.14 shows, the former parameter set achieves sufficient randomness and the latter does not. Table 2.1 summarizes the results obtained from the NIST tests. The randomness failed nine tests with the 21 MHz slow oscillator, which demonstrates the insufficiency of randomness. However, the 10 MHz slow oscillator attained such a high degree of randomness that it passed all the tests. The efficiency of the proposed design methodology was verified since these results are consistent with those in Fig. 2.14.

Different oscillator topologies and logic styles, such as the current mode logic for faster ROs, can be also explored in actual designs. Here, power consumption, in addition to area, becomes a key performance metric and a more complex design space has to be explored. The proposed evaluation of randomness using the worst χ is effective in terms of CPU time to achieve such purposes.

2.6.2 Effect of XOR corrector

Figure 2.15 plots variations in the worst χ with the XOR corrector as the frequency ratio of the oscillators changes. The r is 1.77×10^{-14} s, and the fast oscillators are 5-stage ROs (average period and frequency are 178.3 ps and 5.6 GHz) whose duty cycle is set to 50%. Figure 2.15 indicates that the XOR corrector improves the estimated χ values. The XOR corrector, however, reduced the throughput of the TRNG by half. The minimum frequency ratios that pass the poker test are 701 without the corrector and 244 with the XOR corrector. Consequently, the throughputs without a corrector and with the XOR corrector correspond to 5.6 Gbps / 701 = 8 Mbps and (5.6 Gbps / 204) / 2 = 11.5 Mbps. This means that the XOR corrector is effective even when the duty cycle of the fast oscillator is balanced.

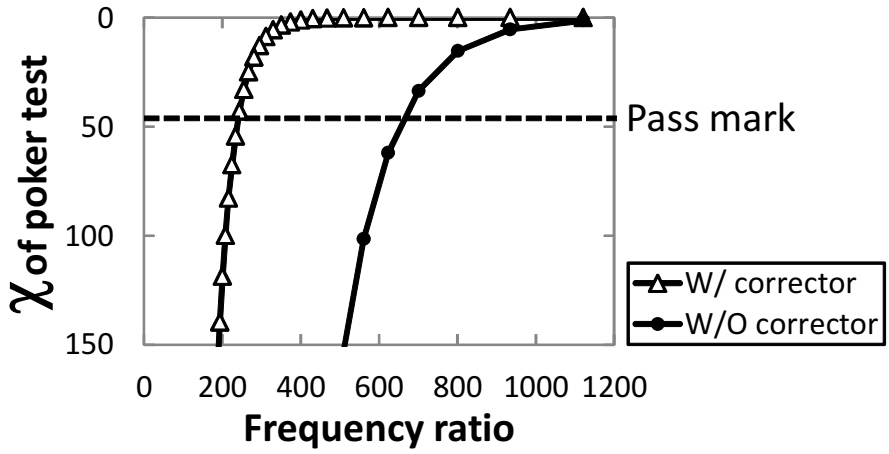

Figure 2.15: Improvement in χ value with XOR corrector.

2.6.3 Injection locking attack

Frequency injection [7] is a state-of-the-art attack on oscillator-based TRNGs that utilizes injection locking in ring oscillators to reduce the amount of jitter and degrade randomness. The proposed model can deal with injection locking attacks by decreasing variance constant. In this section, the slow oscillator is injection-locked and its variance constant r_{slow} is reduced while the jitter of the fast oscillator is constant, because an oscillator with high frequency is difficult to be injection-locked. As a preliminary experiment using another small test structure fabricated in the same process, the author measured the variance reduction of a ring oscillator under injection locking. A noise generator attacked a 2-input NAND gate in 293-stage ring oscillator. The measurement showed that the variance of the periods decreased to 1/16, and thus this section employed 16 as a factor of variance reduction. In addition, this section evaluates the randomness of the output when the variance constant for the slow oscillator is zero as an extreme case.

Figure 2.16 plots the worst χ as a function of the frequency ratio of oscillators and the three curves correspond to $r_{\text{slow}} = 1.77 \times 10^{-14}$ s, $r_{\text{slow}}(= 1.77 \times 10^{-14}/16) = 1.11 \times 10^{-15}$ s, and $r_{\text{slow}} = 0$ s. Here, the fast oscillator is a 5-stage RO and its duty cycle is 50%. Figure 2.16 indicates that small r_{slow} requires a low frequency for the slow oscillator to pass the randomness test. If the injection locking reduces r_{slow} from 1.77×10^{-14} s to 1.11×10^{-15} s or 0 s, the throughput decreases by half to sustain sufficient randomness.

Let us examine the throughput reduction above. When the frequency of fast oscillator is constant, the required equivalent jitter to attain sufficient randomness is almost constant. As Eq. (2.4) shows, equivalent jitter, σ_i, originates from slow oscillator, $r_{\text{slow}}t_{\text{slow}}$, and fast oscillator, $r_{\text{fast}}t_{\text{fast}}$. When the slow oscillator is under ideal injection locking, the variance constant for slow oscillator becomes zero and $r_{\text{slow}}t_{\text{slow}} = 0$ while r_{fast} and jitter component from fast oscillator is unchanged. The variance of fast oscillator during a period of slow oscillator, $r_{\text{fast}}t_{\text{fast}}(N + 1 - i/m)$, is approximately proportional to frequency ratio, and hence increasing frequency ratio can sustain equivalent jitter. For example, if $r_{\text{slow}}t_{\text{slow}}$ without injection locking is equal to $r_{\text{fast}}t_{\text{fast}}(N + 1 - i/m)$, increasing the frequency ratio by a factor

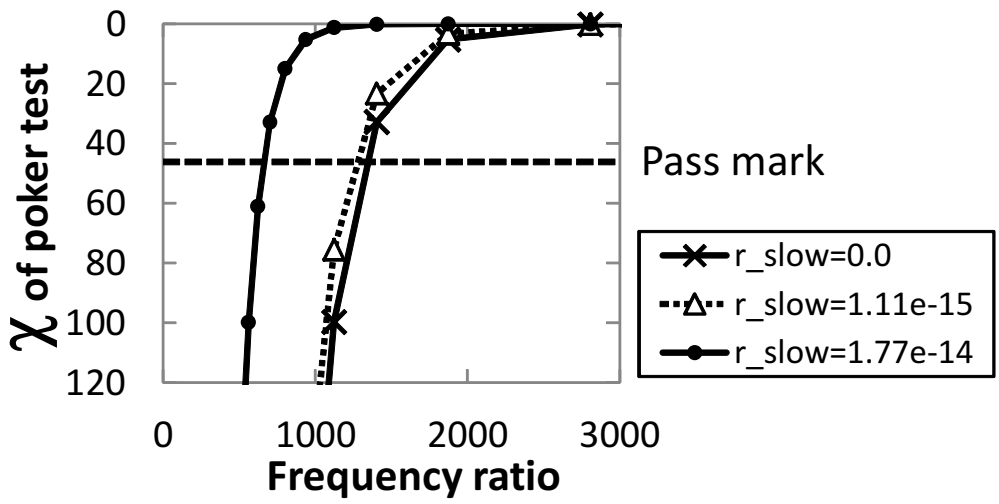

Figure 2.16: χ value vs. frequency ratio with different variance constants.

of 2 is reasonable with the slow oscillator under ideal injection locking.

2.6.4 Size of state space

The size of state space m affects the accuracy of the model as explained in Section 2.2.2. Figure 2.17 plots the estimated χ values when m is varied. The r is 1.77×10^{-14} s. The fast oscillator is a 5-stage RO and its duty cycle is 50%. These are typical settings in the experiments of this chapter. The frequencies of the slow oscillator are 5, 10, and 20 MHz. Figure 2.17 shows that as m becomes larger, the χ value converges, and a large state space is necessary for the conversion when the estimated χ is high (viz., randomness is low). In the range of χ being below 1000, an m of 100 enables an approximate estimate of randomness and an m of 1000 is sufficient for precise analysis. This chapter therefore employed 100 or 1000 of m in the experiments discussed in this section.

In the case that m is 100 and the slow frequency is 20 MHz, $t_{\mathrm{span}} = t_{\mathrm{fast}}/m$ is 1.8 ps, which is $1/23$ of the equivalent jitter σ. This result also suggests a guideline that t_{span} should be less than about $\sigma/25$.

2.7 Conclusion

This chapter presented a worst-case-aware design methodology for oscillator-based TRNGs. A behavioral model of a TRNG and a methodology of evaluating the worst randomness under deterministic noise were proposed. This chapter confirmed the effectiveness of the proposed model through hardware measurements and comparisons obtained with a gate-level noise-aware TRNG simulator, which was tailored to evaluate randomness under deterministic noise. The proposed design methodology aided us in designing an oscillator-based TRNG that satisfied performance specifications even under a hostile environment.

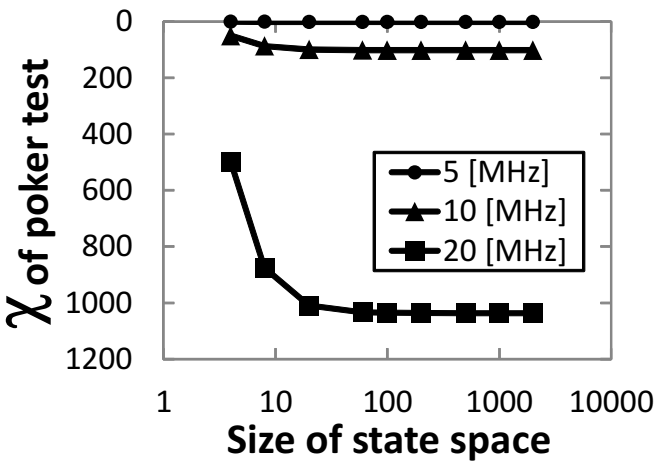

Figure 2.17: χ value vs. sizes of state space.

Chapter 3

Jitter amplifier for slow oscillator

This chapter proposes a jitter amplifier for an oscillator-based true random number generator (TRNG). Two types of latency-controllable (LC) buffer, which are the key components of the proposed jitter amplifier, are presented. This study derives an equation to estimate the gain of the jitter amplifier, and analyzes sufficient conditions for the proposed circuit to work properly. The proposed jitter amplifier was fabricated with a 65 nm CMOS process. The jitter amplifier with the two-voltage LC buffer occupies 3,300 μm^2 and attains 8.4x gain, and that with the single-voltage LC buffer achieves 2.2x gain with an 1,700 μm^2 area. The jitter amplification of the sampling clock increases the entropy of a bit stream and improves the results of the NIST test suite so that all the tests pass whereas TRNGs with simple correctors fail. The jitter amplifier attains higher throughput per area than a frequency divider when the required amount of jitter is more than two times larger than the inherent jitter in our test-chip implementations.

3.1 Introduction

As discussed in Sec. 1.2.3, in general, the jitter of the oscillator is not sufficient for generation of highly random number. Frequency dividers accumulate the jitter of the oscillator, reducing the throughput of the TRNG [38, 112]. Bucci et al. [39] utilized a triangular wave oscillator for a jittery oscillator, though it consumes large power and area. Ergün and Özoğuz [27] presented a TRNG with a chaotic oscillator. The timing fluctuations of the oscillator depend on the employed chaos system rather than the random noise. The chaotic oscillator required inductors, which reduces its scalability.

This chapter proposes a jitter amplifier for an oscillator-based TRNG [119,120]. Figure 3.1 illustrates the structure and the operation of a TRNG employing the jitter amplifier. The fast oscillating signal (D in Fig. 3.1) is sampled with a jittery slow clock whose jitter is amplified with the jitter amplifier (CK in Fig. 3.1), which results in a random bit stream. This timing jitter increases as the period jitter of the slow oscillator, and hence this chapter focuses on the period jitter of the slow oscillator and discusses how to amplify it. Note that the jitter of the oscillator is assumed to be temporally independent in this thesis, as discussed in Sec. 1.2.1. This assumption is used to analyze the behavior of the jitter amplifier in Sec. 3.2. As is referred in Sec. 1.2.1, FIFO stabilizes the throughput of the oscillator-based TRNG though the throughput is unstable due to the jitter of the oscillator. This chapter therefore focuses on the randomness of the bitstream from the sampler to clarify the efficiency of the proposed jitter amplifier. To obtain the theoretical substantiation of jitter amplification, the

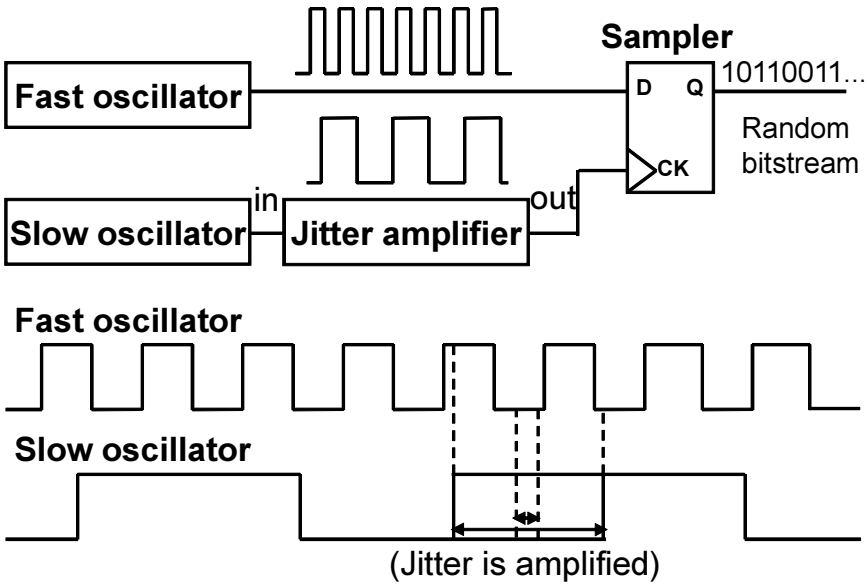

Figure 3.1: Oscillator-based TRNG with jitter amplifier.

gain of the jitter amplification is analytically estimated. Furthermore, sufficient conditions for proper amplification is analyzed, which helps the jitter amplifier to be integrated with the TRNGs. Two kinds of test chips were fabricated with a 65 nm process to validate the proposed amplifier. The measurements demonstrate that the proposed circuit improves randomness with a small increase in area without degrading throughput.

The reminder is organized as follows. Section 3.2 presents the proposed jitter amplifier and analyzes its behavior. Section 3.3 presents and compares two types of implementations. Section 3.4 explains the results obtained from measurements. Section 3.5 concludes this chapter.

3.2 Behavior of jitter amplifier

3.2.1 Concept behind jitter amplification

Figure 3.2 shows a block diagram of the jitter amplifier. The proposed jitter amplifier consists of an LC buffer and a timing generator. The LC buffer is designed so that each buffer delay t_d could be changed by ctrl from t_{df} to t_{ds}, where $t_{df} < t_{ds}$. That is, the buffer operates in fast mode until the ctrl rise edge arrives, and after that it works in slow mode. Details on the implementations of the LC buffer and the timing generator will be given in Sec. 3.3.

Figure 3.3 illustrates the concept behind jitter amplification, where the jitter of the input oscillating signal in is amplified. The timings of rise edges of in fluctuate since the input signal has jitter. To exemplify how the temporally fluctuated signals are processed in the jitter amplifier, let us consider an early rising signal ine and a late rising signal inl, and their outputs oute and outl, respectively. ine rises at t_{ine} and inl rises at t_{inl} ($t_{ine} < t_{inl}$), and oute rises at t_{oute} and outl rises at t_{outl}. The total latencies of the LC buffer for ine and inl are $d_{bufe} = t_{oute} - t_{ine}$ and $d_{bufl} = t_{outl} - t_{inl}$. Here, ctrl rises while in is propagating through the

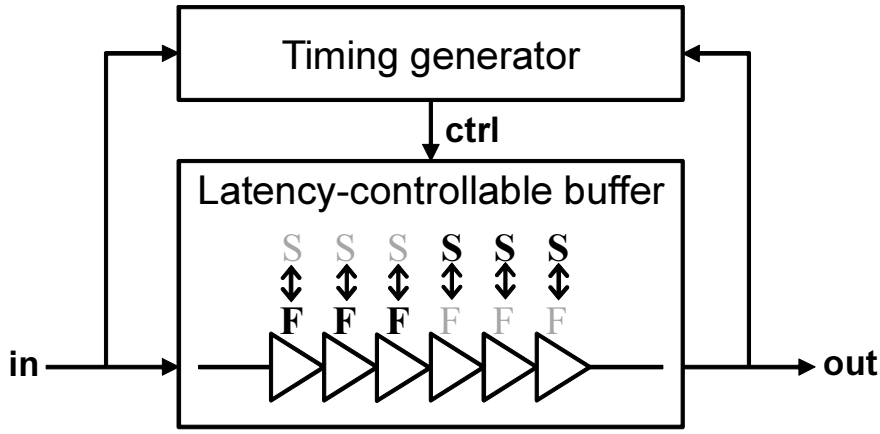

Figure 3.2: Block diagram of jitter amplifier.

LC buffer, namely, the rise timing of ctrl t_{ctrl} is $t_{ine} < t_{ctrl} < t_{oute}$ and $t_{inl} < t_{ctrl} < t_{outl}$. Note that t_{ctrl} is constant for both t_{ine} and t_{inl} since ctrl is generated by the timing generator which is distinct from the slow oscillator. The time intervals of fast mode are $d_{faste} = t_{ctrl} - t_{ine}$ for ine and $d_{fastl} = t_{ctrl} - t_{inl}$ for inl, where $d_{faste} > d_{fastl}$ from $t_{ine} < t_{inl}$. Longer time for fast mode reduces time for slow mode and results in smaller total latency of LC buffer, and therefore, $d_{bufe} < d_{bufl}$. This means that the later rising edge of in causes larger latency in the LC buffer. In order to validate the jitter amplification, time differences at in and out are compared as follows:

$$\begin{aligned} t_{outl} - t_{oute} &= (t_{inl} + d_{bufl}) - (t_{ine} + d_{bufe}) \\ &= (t_{inl} - t_{ine}) + (d_{bufl} - d_{bufe}) \\ &> (t_{inl} - t_{ine}). \end{aligned} \tag{3.1}$$

Therefore, the time difference between the early and the late rise timings at in, which indicates the input jitter, is intensified at out by the variable latency of the LC buffer. Thus, the jitter of in is amplified.

3.2.2 Analysis of behavior

Preparation

In this section, the behavior of the jitter amplifier is analyzed and an equation to estimate the gain of the jitter amplification is presented. Figure 3.4 shows a timing chart to explain the behavior of the jitter amplifier. A timing of the n-th rising edge of in is defined as $t_{in(n)} = 0$ ($n \in \mathbb{N}$), and the corresponding rise edges of ctrl and out are $t_{crise(n)}$ and $t_{out(n)}$, where $t_{in(n)} < t_{crise(n)} < t_{out(n)}$. Here, a rise edge of ctrl is generated by the timing generator from a previous edge of in, and $t_{crise(n+1)}$ depends on $t_{in(n)}$. The amount of input jitter given to the jitter amplifier is represented as the standard deviation during the time interval of $t_{in(n+2)} - t_{in(n+1)}$ and output jitter is the standard deviation of $t_{out(n+2)} - t_{out(n+1)}$, and gain of the jitter amplifier is output jitter divided by input jitter. $t_{in(n)}$ needs to be considered to derive the output jitter since $t_{crise(n+1)}$ depends on $t_{in(n)}$ and $t_{crise(n+1)}$ affects the rise timing of out, $t_{out(n+1)}$. Also, timing information before $t_{in(n)}$ is not needed since $t_{in(n+1)}$ and $t_{crise(n+1)}$, which are the necessary

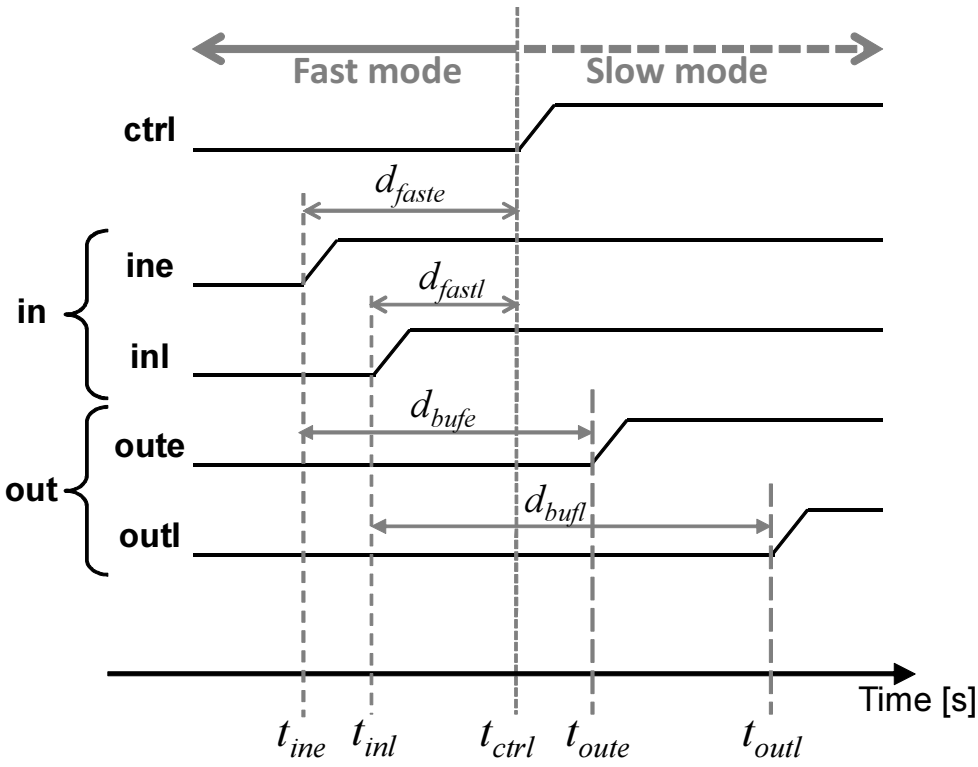

Figure 3.3: Timing chart explaining concept behind jitter amplification.

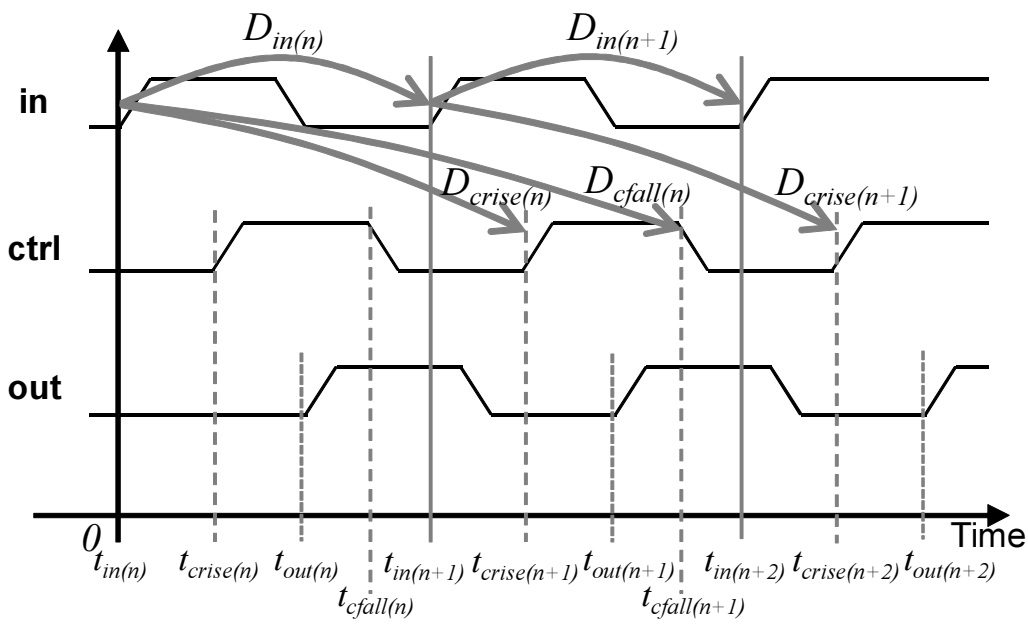

Figure 3.4: Behavior of jitter amplifier.

timings for deriving the output jitter, are both generated from the same timing $t_{in(n)}$.

The n-th period of in is represented as a stochastic variable $D_{in(n)} = t_{in(n+1)} - t_{in(n)}$. The time interval from the n-th rising edge of in to the $(n+1)$-th rise edge of ctrl is a stochastic variable $D_{crise(n)} = t_{crise(n+1)} - t_{in(n)}$. $D_{in(n)}$ represents the period of the slow oscillator and $D_{crise(n)}$ represents the latency of the timing generator. The stochastic process $\{D_{in(n)}\}$ is assumed to be independent and identically distributed, and its element $D_{in(n)}$ is assumed to be normally distributed. $\{D_{crise(n)}\}$ also assumed to be independent and identically distributed, and $D_{crise(n)}$ is assumed to be normally distributed. The above assumptions are reasonable because the delay elements which construct $D_{in(n)}$ and $D_{crise(n)}$ are fluctuated by the internal noises and the noises are temporarily independent. The assumption is also supported by the measured ACF of the jitter of the ring oscillator in Fig. 1.5. The mean of $D_{in(n)}$ is μ_{in} and its variance is σ_{in}^2, and the mean of $D_{crise(n)}$ is μ_{crise} and its variance is σ_{crise}^2. That is, $D_{in(n)} \sim N(\mu_{in}, \sigma_{in}^2)$ and $D_{crise(n)} \sim N(\mu_{crise}, \sigma_{crise}^2)$. $D_{in(n_1)}$ is independent of $D_{crise(n_2)}$ for arbitrary n_1 and n_2 $(n_1, n_2 \in \mathbb{N})$ since the slow oscillator is distinct from the timing generator.

Figure 3.5 shows the behavior of the LC buffer for n-th rising edge of in. Here, let us introduce an analogy that a signal is propagating on a line at a certain speed. The length of the line, that is the distance between start and end points, is l and it corresponds to the length of the LC buffer. Note that the length l is an abstract length rather than a concrete size of the buffer chain. The latencies for a sufficiently small length Δl are $\Delta D_{bf(n)}$ for fast mode and $\Delta D_{bs(n)}$ for slow mode. The stochastic process $\left\{\Delta D_{bf(n)}\right\}$ is assumed to be independent and identically distributed, and $\{\Delta D_{bs(n)}\}$ are also assumed to be independent and identically distributed. $\Delta D_{bf(n)}$ and $\Delta D_{bs(n)}$ are assumed to be normally distributed, their means are μ_{bf} and μ_{bs}, and their variances are σ_{bf}^2 and σ_{bs}^2, namely, $\Delta D_{bf(n)} \sim N(\mu_{bf}, \sigma_{bf}^2)$ and $\Delta D_{bs(n)} \sim N(\mu_{bs}, \sigma_{bs}^2)$. This is because the delay elements which constructs $\Delta D_{bf(n)}$ and $\Delta D_{bs(n)}$ are fluctuated by the internal noise, and the noise is temporarily independent. Then, for example, the latency of LC buffer in fast mode is calculated as $l\Delta D_{bf(n)}/\Delta l$. Practically, the amount of jitter is much smaller than the periods, therefore, σ_{bf} and σ_{bs} are much smaller than μ_{bf}.* Here, $D_{in(n_1)}$, $D_{crise(n_2)}$, $\Delta D_{bf(n_3)}$ and $\Delta D_{bs(n_4)}$ are independent of each other for arbitrary n_1, n_2, n_3 and n_4 $(n_1, n_2, n_3, n_4 \in \mathbb{N})$.

Gain derivation

From now, an analytical expression of gain is derived. Firstly, the amount of input jitter is σ_{in} since $t_{in(n+2)} - t_{in(n+1)}$ is $D_{in(n+1)}$.

Here, $\Delta D_{bf(n)}$ is rewritten as $\mu_{bf} + D_{bfr(n)}$, where $D_{bfr(n)} \sim N(0, \sigma_{bf}^2)$. Now that σ_{bf} is much smaller than μ_{bf}, $\left|-D_{bfr(n)}/\mu_{bf}\right| << 1$ holds. Though, strictly speaking, $\left|D_{bfr(n)}\right|$ is not always smaller than μ_{bf} due to the normal distribution, the probability of $\left|D_{bfr(n)}\right| \geq \mu_{bf}$ is so small that it can be ignored in actual situations. According to Taylor expansion, $1/\Delta D_{bf(n)}$

* If the jitter amount is sufficiently large, the jitter amplification itself is not required.

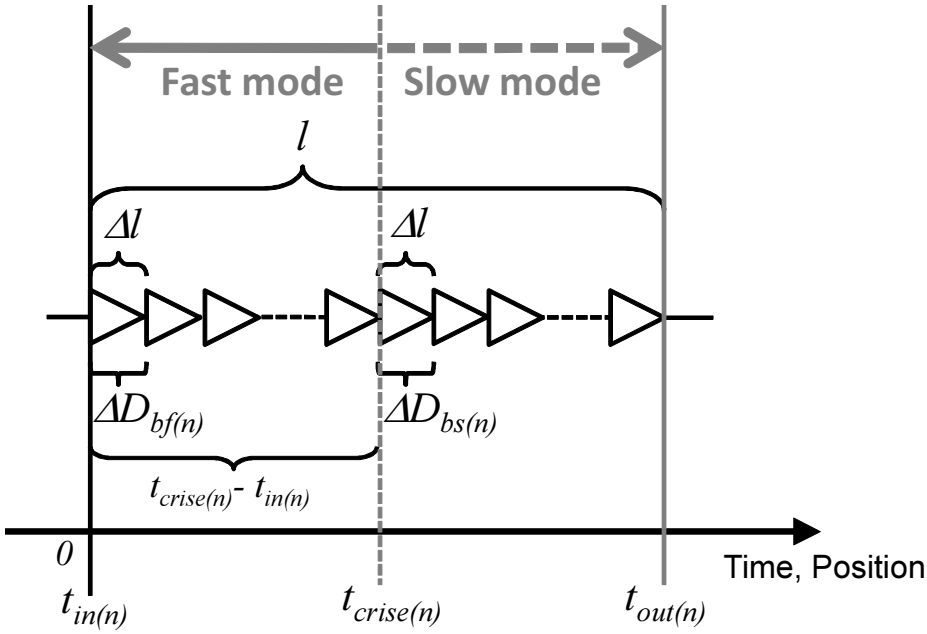

Figure 3.5: Fast and slow modes of LC buffer.

can be approximated as follows;

$$\begin{aligned}\frac{1}{\Delta D_{bf(n)}} &= \frac{1}{\mu_{bf}}\frac{1}{1+\frac{D_{bfr(n)}}{\mu_{bf}}} \\ &= \frac{1}{\mu_{bf}}\left\{1-\frac{D_{bfr(n)}}{\mu_{bf}}+\sum_{k=2}^{\infty}\left(-\frac{D_{bfr(n)}}{\mu_{bf}}\right)^k\right\} \\ &\approx \frac{1}{\mu_{bf}^2}\left(\mu_{bf}-D_{bfr(n)}\right). \end{aligned} \tag{3.2}$$

In order to identify the dominant factors, σ_{bf}^k/μ_{bf}^k and σ_{bs}^k/μ_{bf}^k $(k \geq 2)$ are approximated as zeros since σ_{bf} and σ_{bs} are much smaller than μ_{bf}.

Under these conditions, the variance of $t_{out(n+2)} - t_{out(n+1)}$ is calculated as;

$$\begin{aligned}\mathrm{Var}[t_{out(n+2)}-t_{out(n+1)}] \approx &\left\{2(a+1)x^2-2(2a+1)x+(2a+1)\right\}\sigma_{in}^2 \\ &+ 2\sigma_{bs}^2\left\{\frac{l}{\Delta l}-\frac{\mu_{crise}-\mu_{in}}{\mu_{bf}}\right\}, \end{aligned} \tag{3.3}$$

where $a = \sigma_{ctrl}^2/\sigma_{in}^2$ and $x = \mu_{bs}/\mu_{bf}$ $(a > 0, x > 0)$. The detailed derivation can be found in Appendix B. The first term of Eq. (3.3) shows that the input jitter is magnified by the mechanism explained in Sec. 3.2.1 This magnification of the input jitter is independent of the rise timing of ctrl. The second means the additional jitter appended during the slow mode.

From Eq. (3.3), the gain of the jitter amplifier is calculated as follows;

$$Gain^2 \approx \left\{2(a+1)x^2 - 2(2a+1)x + (2a+1)\right\} + 2\frac{\sigma_{bs}^2}{\sigma_{in}^2}\left\{\frac{l}{\Delta l} - \frac{\mu_{crise} - \mu_{in}}{\mu_{bf}}\right\}. \tag{3.4}$$

Because the second term of Eq. (3.4) is positive, a sufficient condition for $Gain > 1$ is;

$$2(a+1)x^2 - 2(2a+1)x + (2a+1) > 1, \quad 0 < x < \frac{a}{a+1}, 1 < x. \tag{3.5}$$

In actual situations, $Gain > 1$ is attained since μ_{bs} is larger than μ_{bf}, i.e., $x > 1$. In addition, in case of $x > 1$, $Gain$ becomes larger monotonically as x increases. On the other hand, when x is $0 < x < a/(a+1)$, the circuit amplifies the jitter in the inverse way. In this case, μ_{bs} is smaller than μ_{bf}, and then the early rising edge at in rises late at out and the late edge of in rises early at out. Though the analysis suggests such an implementation, the following discussion focuses on $1 < x$.

3.2.3 Constraints on LC buffer and input signal

In the discussion so far, it is assumed that $t_{in(n)} < t_{crise(n)} < t_{out(n)}$ holds for arbitrary n. In addition, the circuit should be initialized without hindering the amplifying operation. Thus, the timing generator should adjust the timings of ctrl appropriately. However, the precision of the adjustment is limited and furthermore the signals have jitter, which restricts the length of the LC buffer and the frequency of input signal. The conditions for the proposed circuit to amplify the jitter properly are explained here.

In Fig. 3.4, the timing of the n-th falling edge of ctrl is defined as $t_{cfall(n)}$. The time interval from the n-th in to the $(n+1)$-th fall edge of ctrl, which represents the latency in the timing generator, is $D_{cfall(n)} = t_{cfall(n+1)} - t_{in(n)}$. $D_{cfall(n)}$ is normally distributed, and its mean is μ_{cfall} and its variance is σ_{cfall}^2, that is, $D_{cfall(n)} \sim N(\mu_{cfall}, \sigma_{cfall}^2)$. Here, $D_{cfall(n_1)}$ is independent of $D_{in(n_2)}$, $D_{crise(n_3)}$, $\Delta D_{bf(n_4)}$ and $\Delta D_{bs(n_5)}$ for arbitrary n_1, n_2, n_3, n_4 and n_5 ($n_1, n_2, n_3, n_4, n_5 \in \mathbb{N}$).

The sufficient condition for the proper function is $t_{in(n+1)} < t_{crise(n+1)} < t_{out(n+1)} < t_{cfall(n+1)} < t_{in(n+2)}$. If this condition is satisfied, a rise edge of in propagates through the LC buffer in fast mode firstly, and then propagates in slow mode until the edge goes through the buffer.

In actual design, μ_{rise} and μ_{fall}, which are the average latencies in the timing generator and correspond to μ_{crise} and μ_{cfall}, are discretely controlled rather than continuously. Therefore, μ_{rise} and μ_{fall} are expressed as $\mu_{rise} = \mu_{rise\_offset} + s\Delta\mu_{rise}$ and $\mu_{fall} = \mu_{fall\_offset} + t\Delta\mu_{fall}$ ($s, t \in \mathbb{Z}, s, t \geq 0$), where $\Delta\mu_{rise}$ and $\Delta\mu_{fall}$ represent the adjustment steps of μ_{rise} and μ_{fall}. For example, as will be discussed in Sec. 3.3.1, our implemented timing generator employs counters whose clock signal is given by internal ring oscillators, and then $\Delta\mu_{rise}$ and $\Delta\mu_{fall}$ are equal to the periods of the clocks. Note that, with $\Delta\mu_{rise} \to 0$ and $\Delta\mu_{fall} \to 0$, the following discussion can be applied to an ideal timing generator which can control μ_{rise} and μ_{fall} continuously.

Here, let us suppose $\mu_{rise\_offset} < \mu_{in} - m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} + (l/\Delta l)\mu_{bf} - m\sigma_{bf}\sqrt{(l/\Delta l)}$ and $\mu_{fall\_offset} < \mu_{in} - m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} + (\mu_{in} - m\sigma_{in})$ are satisfied, as will be derived in Appendix C.

These conditions mean that the offsets of the timing generator, μ_{rise_offset} and μ_{fall_offset}, are small enough for the rise and fall edges of ctrl to be adjusted into the proper range. Under these conditions, the sufficient conditions are expressed as the following two equations;

$$\Delta\mu_{rise} + 2m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} < \frac{l}{\Delta l}\mu_{bf} - m\sigma_{bf}\sqrt{\frac{l}{\Delta l}}, \tag{3.6}$$

$$\Delta\mu_{fall} + 2m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} < (\mu_{in} - m\sigma_{in}) - \left(\frac{l}{\Delta l}\mu_{bs} + m\sigma_{bs}\sqrt{\frac{l}{\Delta l}}\right). \tag{3.7}$$

The derivations will be presented in Appendix C. Here, a coefficient m ($m > 0$) is introduced to bound the normal distribution. To be more precise, the upper bound of a normal distribution $N(\mu, \sigma^2)$ is defined as $\mu + m\sigma$ and the lower bound is $\mu - m\sigma$. Intuitively, Eq. (3.6) means that the range of $t_{crise(n+1)}$ added by the step of μ_{rise} is smaller than the minimum latency of the LC buffer. Eq. (3.7) means that the range of $t_{cfall(n+1)}$ added by the step of μ_{fall} is smaller than the minimum time interval between the rise edges of out and the next in. The number of stages of LC buffer is limited because the length of LC buffer is restricted from the Eqs. (3.6)(3.7) and l is proportional to the number of stages.

Eqs. (3.6)(3.7) represent the constraints for designing the jitter amplifier. For obtaining a proper jitter amplification, the timing generator should control the timings of ctrl rise and fall edges, as will be shown in Sec. 3.3.1. The edges of ctrl, however, cannot be adjusted into the ranges of proper function if Eqs. (3.6)(3.7) are not satisfied. Thus, the designer should confirm that the constraints are satisfied in designing the jitter amplifier.

Also, with rearranging the Eq. (3.7) for μ_{in}, the constraint on the period of input signal is derived;

$$\Delta\mu_{fall} + 2m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} + m\sigma_{in} + \left(\frac{l}{\Delta l}\mu_{bs} + m\sigma_{bs}\sqrt{\frac{l}{\Delta l}}\right) < \mu_{in}. \tag{3.8}$$

Thus, the input frequency, $1/\mu_{in}$, is limited by Eq. (3.8).

As an example, the constraints are verified for a jitter amplifier this section has implemented, which will be shown in Fig. 3.11(a). μ_{in} and σ_{in} are from measurement results of a 251-stage ring oscillator with a 64-frequency divider, employing 1.2 V and 0.7 V of supply voltages. $\Delta\mu_{rise}$, $\Delta\mu_{fall}$, σ_{rise}^2, σ_{fall}^2, $l\mu_{bf}/\Delta l$, $l\mu_{bs}/\Delta l$, $l\sigma_{bf}^2/\Delta l$ and $l\sigma_{bs}^2/\Delta l$ are calculated from the measurement results considering that the average and the variance of the latency are proportional to the number of stages. m is set to 3. Then, the left and right terms of Eq. (3.6) are calculated as 3.7×10^{-9} and 1.5×10^{-8}, and those of Eq. (3.7) are 3.8×10^{-9} and 4.1×10^{-7}. Consequently, our implemented circuit can amplify the jitter properly.

3.3 Implementation

3.3.1 Implementation of timing generator

The timing generator is responsible for generating ctrl, and its implementation is illustrated in Fig. 3.6. When in rises, the edge detector generates a negative pulse, and the 1-bit

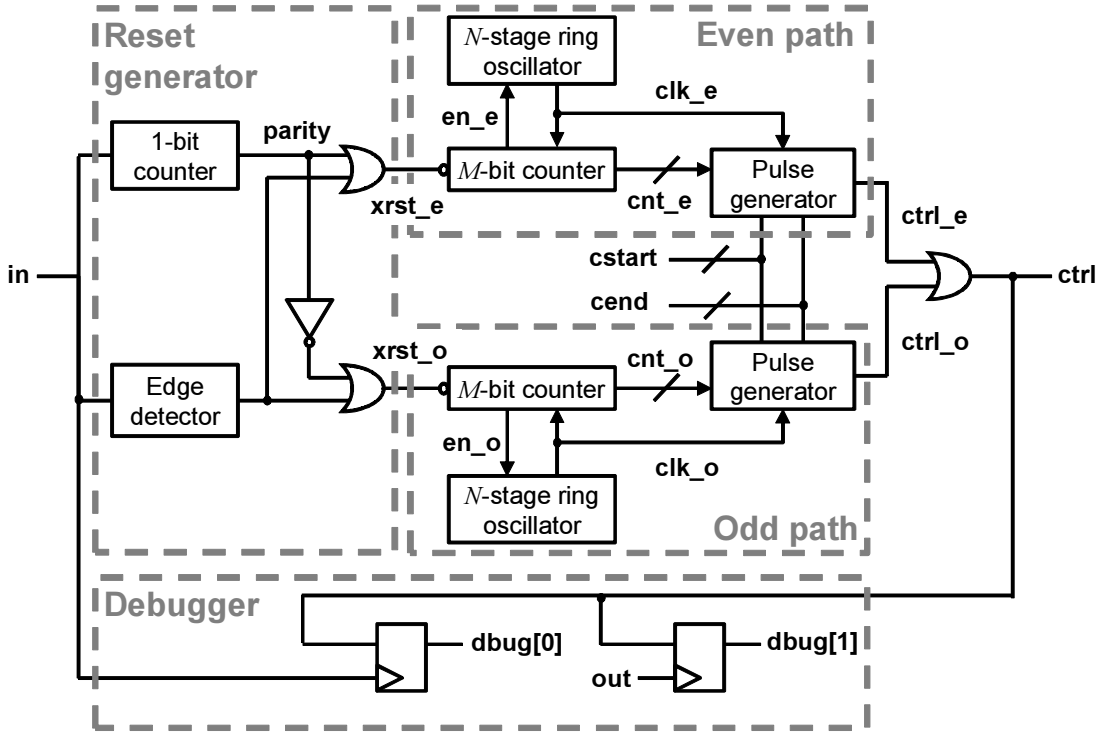

Figure 3.6: Block diagram of timing generator implemented in this section.

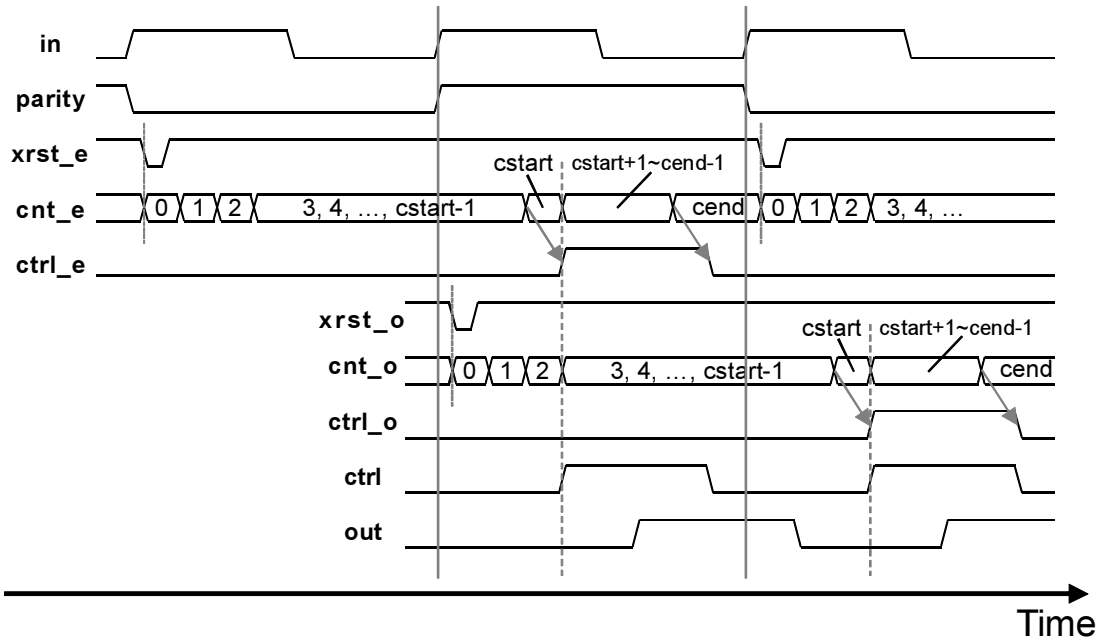

Figure 3.7: Timing chart of timing generator. Clock signals (clk_e/o) are omitted.

counter selects the path which delivers the reset signal (xrst_e/xrst_o). When a negative pulse is generated, a ring oscillator is enabled (en_e/o) and its corresponding counter starts to in-

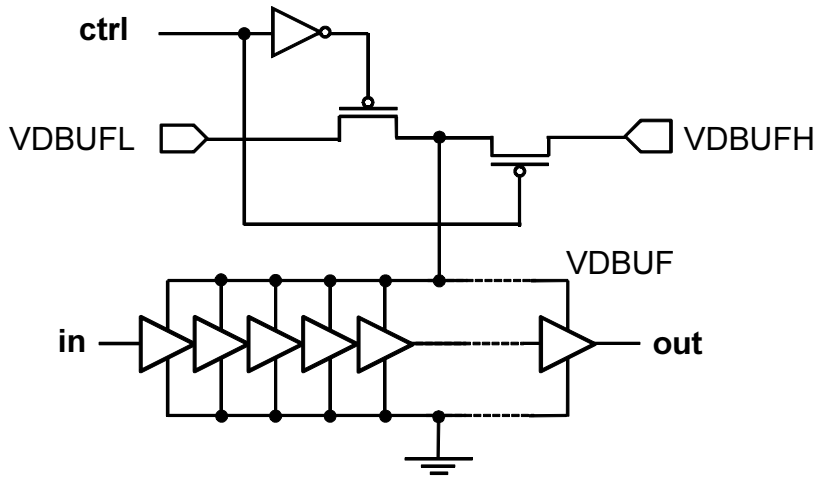

Figure 3.8: Implementation of two-voltage LC buffer.

crement after being initialized. Every time the counter value (cnt_e/o) exceeds predefined values (cstart/cend), a pulse generator produces rise and fall edges. The timing generator has two (even and odd) paths to generate ctrl for every in rise edge because the increment of the counter starts at the rising edge of in and ends after the next rising edge. The mismatch between the even and odd paths due to process variation affects the timing of ctrl rising edge, μ_{crise}. However, the impact on the gain of jitter amplifier is limited, since μ_{crise} affects only the second term of Eq. (3.4) and the second term can be ignored with large x. Debug signals (dbug) output ctrl when in and out rise. dbug must be 2'b10 since ctrl is low at the rising edge of in and high at the rising edge of out for proper function. If dbug is 2'b11, for example, it means that ctrl rises before the edge of in, and therefore the rise timing of ctrl should be delayed by cstart. Thus, the debug signal can be used for adjusting the rise/fall timings of ctrl.

Figure 3.7 shows the behavior of the signals in the timing generator. When in rises, the 1-bit counter flips parity. If parity is zero, a negative pulse is generated as xrst_e, and the M-bit counter cnt_e is initialized and starts increment. While cnt_e is between cstart and cend-1, the pulse generator makes a pulse ctrl_e. In the same way, when parity is one, xrst_o resets cnt_o, and ctrl_o is produced. Finally, ctrl_e and ctrl_o are ORed and its output becomes ctrl. With properly selected values of cstart and cend, ctrl rises between rise edges of in and out, and falls between out and the next in, namely, Eqs. (3.6) and (3.7) are satisfied.

3.3.2 Implementations of LC buffers

Various LC buffer implementations are possible that change the element delay depending on the control signal. This section presents two implementations that use voltage scaling, i.e., two-voltage and single-voltage implementations. Other implementations of the LC buffer, for example, could change the loading and/or the drive strength of the buffer elements.

Figure 3.8 shows the two-voltage LC buffer. The VDD of the buffer (VDBUF) is varied to change the buffer delay from high voltage (VDBUFH) to low (VDBUFL) by using PMOS switches according to ctrl. The jitter gain varies depending on what voltages are selected for VDBUFH and VDBUFL. The sizes of the PMOSs should be determined so that the switch-

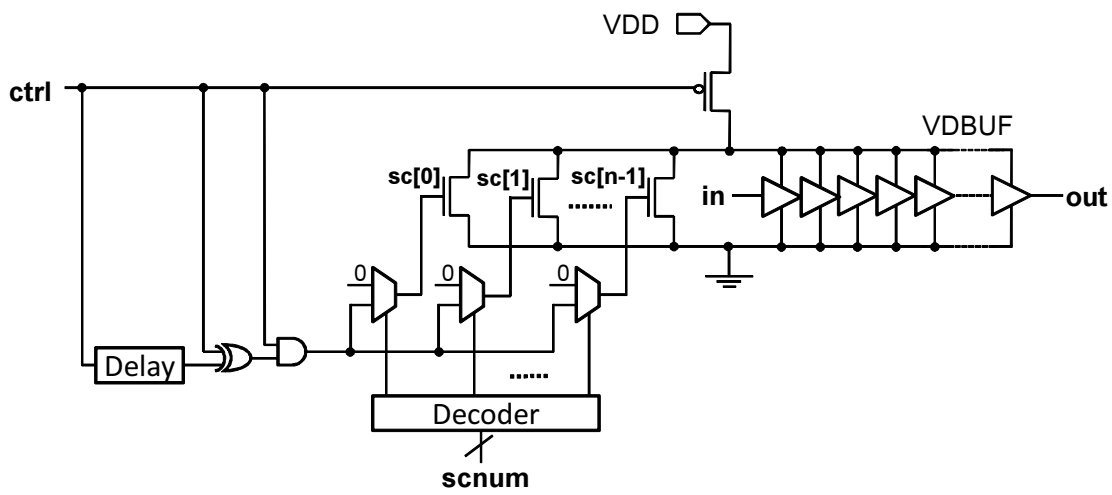

Figure 3.9: Implementation of single-voltage LC buffer.

ing time while VDBUF changes from VDBUFH to VDBUFL should be sufficiently small comparing to the latency of the LC buffer.

Figure 3.9 depicts the single-voltage LC buffer. The VDD of the buffer (VDBUF) can be gated from global VDD with PMOS, and can be shorted to the ground by the NMOS transistors. The decoder generates select signals of the multiplexers, where the external signal scnum determines the number of HIGH select signals. The delay element, XOR and AND make a short pulse whose width is equal to the delay of the delay element. Each multiplexer passes the input pulse signal when its select signal is HIGH. The ctrl and sc are LOW in fast mode, and VDBUF is close to VDD. When the operation mode is switched to slow mode by the rising edge of ctrl, the HIGH signal is first input to the PMOS, which makes the VDBUF float. Parasitic capacitances connecting to VDBUF are discharged, VDBUF is lowered, and consequently, the buffer element delay increases.

Figure 3.10 shows simulated waveforms of the single-voltage LC buffer. Parasitic capacitances and resistances of the wire and the diffusion layers were extracted from the layout, and the well capacitances were also taken into account in the simulation. The number of HIGH sc is three. out_buffered is out signal in Fig. 3.9 after propagating through a buffer. It can be seen that VDBUF drops when pulses are input to sc, and VDBUF recovers after the fall edge of ctrl. The edges of sc are sharp enough because sufficiently large buffers were inserted after the multiplexers in Fig. 3.9. In contrast to the two-voltage LC buffer, after VDBUF gets float and dropped, the voltage of the LC buffer is decreasing gradually as the rise and the fall edges of in propagate through the buffer. As Eq. (3.1) suggests, on the other hand, the single-voltage LC buffer amplifies the jitter when the delay in the LC buffer increases from fast mode to slow mode. Since the discussion in Sec. 2.2 assumes that the average delay in slow mode, μ_{bs}, is constant, the gain estimation with Eq.(3.4) is not accurate for the single-voltage LC buffer. The jitter gain changes depending on the number of shorted NMOSs, which can be changed by scnum, and how long the duration of sc is, since they affect voltage drop at the beginning of slow mode. The pulse width, which is determined by the delay element in Fig. 3.9, and the sizes of the switching transistors should be specified considering parasitic capacitance of

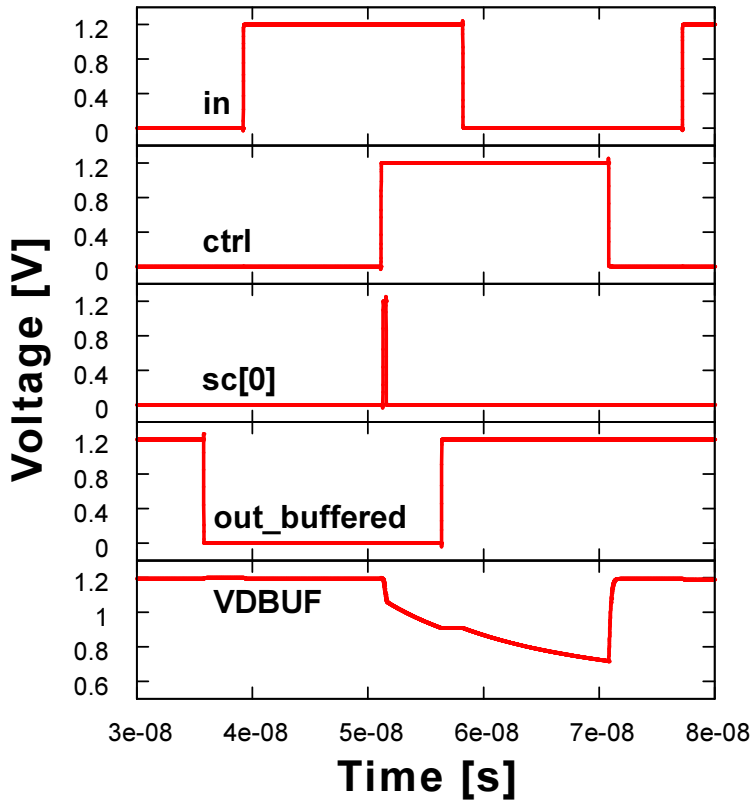

Figure 3.10: Waveform example of single-voltage LC buffer.

VDBUF since the time interval during changing VDBUF should be sufficiently small.

Even though the two-voltage LC buffer requires an additional one or two analog pins for VDBUFH and VDBUFL, they can provide stable VDBUF, which makes the estimate of gain Eq. (3.4) reasonably accurate. The single-voltage LC buffer, on the other hand, is suitable for low cost implementation since no additional pins are necessary. The jitter gain, however, is difficult to accurately estimate due to the gradual decrease in VDBUF in slow mode. Another issue is the difficulty of predicting the amount of potential drop because it is not easy to accurately estimate parasitic capacitance such as well junction capacitance at the design time.

3.4 Results from experiments

3.4.1 Implementation of chips

Prototype oscillator-based TRNGs with a two-voltage LC buffer (chip A) and with a single-voltage LC buffer (chip B) were fabricated with a 65 nm CMOS process (Fig. 3.11). A 31-stage ring oscillator and a 251-stage ring oscillator with a 64-frequency divider are implemented as fast and slow oscillators in chip A. A 7-stage ring oscillator and a 251-stage ring oscillator with a four-frequency divider are the fast and slow oscillators of chip B. The number of frequency division for the slow oscillator in chip B was set to four so that the oscillating frequency became lower than 50 MHz taking into account some safety margin, since the signal with more than 100 MHz cannot be delivered to the outside of the chip due

to bonding wire inductance. On the other hand, the slow oscillator in chip A is accompanied with 64-frequency divider, since the internal ring oscillators in the timing generator is slower than that of chip B and then lower frequency was required to achieve jitter amplification.[†] Basically, higher frequency of fast oscillator is desirable for randomness [39], and then 7-stage ring oscillator was selected for chip B. On the other hand, the slow oscillator of chip A was slower as mentioned above and then it had larger intrinsic jitter. To clearly demonstrate the contribution of the jitter amplification to randomness improvement, a slower fast oscillator was selected in chip A. Note that the slower slow oscillator means lower throughput, and hence it is not desirable. For attaining higher throughput, a faster slow oscillator with insufficient jitter should be adopted and in this case jitter amplification becomes necessary to achieve sufficient randomness. The duty cycles of the fast oscillators, which determines the probabilities of 1/0 occurrences, could be finely adjusted by using body biasing technique [111]. The VDD for the fast oscillators is supplied through a dedicated external pin. A 1,000-stage inverter chain and a 800-stage inverter chain were employed for the LC buffers of chip A and chip B, respectively. The number of stages of LC buffer was determined to satisfy the constraints in Eqs. (3.6) and (3.7). Since the slow oscillator of chip B is faster than chip A, the buffer in chip B was set smaller. The areas of the jitter amplifiers are 3,300 μm^2 for chip A and 1,700 μm^2 for chip B. This area difference mainly comes from the implementations of the fast oscillator. In chip A, P-wells of every gate are separated to supply distinct body voltages. On the other hand, the gates share the identical body voltage in chip B, and therefore, the area is smaller than chip A.

In the following, this section first evaluates the jitter gain of two implementations of the jitter amplifier with two-voltage and single-voltage LC buffers. On the other hand, now that the implementations of the timing generators and the oscillators as well as the LC buffers are different between the Chip A and the Chip B, the impact of the jitter amplification on the improvement in randomness cannot be directly compared. Therefore, the randomness after the jitter amplifiers will be discussed separately for each chip.

3.4.2 Jitter gain

The gains of the jitter amplifiers were measured using a real-time oscilloscope. Figure 3.12 plots the measured jitter gains for the two-voltage LC buffer under different temperatures. The gains estimated with Eq. (3.4) are also shown. Here, assuming that the variance of the delay is proportional to the number of gates, a is calculated as the number of gates through which the rising edge of in propagates until the rise edge of ctrl divided by that during a cycle of the slow oscillator. The internal ring oscillator was 83-stage ring oscillator, cstart was 195, and the slow oscillator was 251-stage ring oscillator with 64-frequency divider. Then, a is $\{83 \times 2 \times (195 + 1)\}/(251 \times 2 \times 64) = 1.01$. x was computed from simulation results of 251-stage ring oscillator at various VDDs. For example, because the periods of the ring oscillator were 8.2 ns with 1.2 V of VDD and 62.8 ns with 0.6 V, then x at 1.2 V of VDBUFH and 0.6 V of VDBUFL was $62.8/8.2 = 7.7$. The second term of Eq. (3.4) is ignored because it gets sufficiently small comparing to the first term as x increases. The X-axis is the difference in voltage defined as VDBUFH - VDBUFL, where VDBUFH is fixed to 1.2 V. Figure 3.12 shows that a larger difference in voltage achieves higher gain. This is

[†] The timing generator of Chip B is the revised version of Chip A, and therefore its internal oscillator was designed faster in order to decrease $\Delta\mu_{rise}$ and $\Delta\mu_{fall}$.

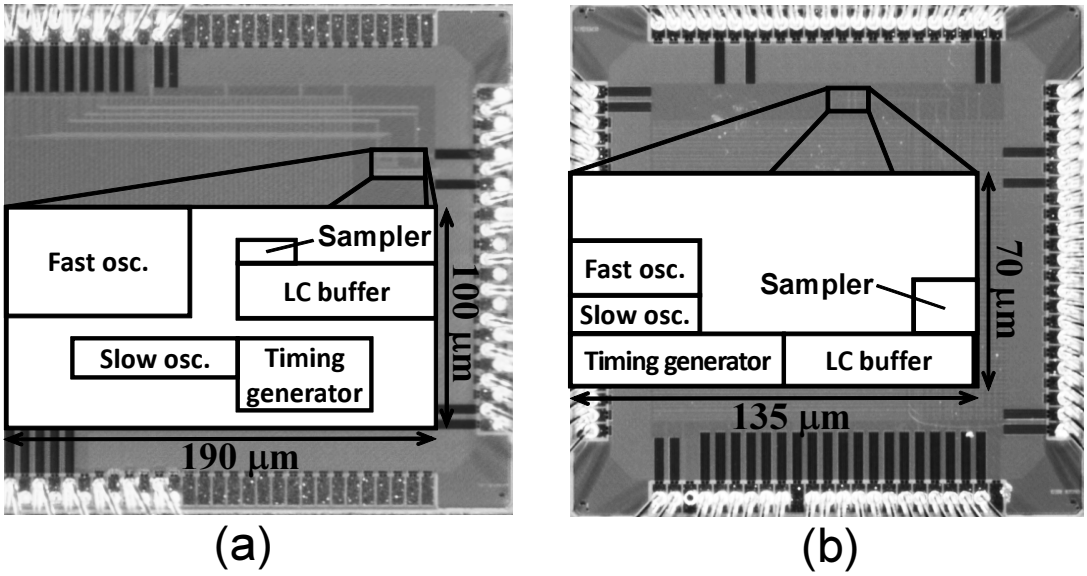

Figure 3.11: Chip photos. (a) Chip A employing two-voltage LC buffer. (b) Chip B using single-voltage LC buffer.

consistent with Eq. (3.4), since the larger voltage difference increases x. Also, decreasing temperature increases jitter gain because the sensitivity of the buffer delay to supply voltage becomes larger and x increases at lower temperatures. It attains 8.4 times gain at 25 °C. The estimated gain agrees well with the measurements.

Figure 3.13 shows the measured gain for the single-voltage LC buffer, where the number of shorted NMOSs is varied. The estimation is not shown since, as referred in Sec. 3.3.2, it is difficult to calculate the gain of the single-voltage LC buffer. Larger numbers of shorted NMOSs yield higher gain of jitter amplification. The gain increases as temperature decreases, which is consistent with the results in Fig. 3.12. It should be noted that randomness monotonously improves as the jitter of the slow clock increases, and hence the magnitude of jitter amplifier gain is important yet its stability is not required.

3.4.3 Approximate entropy

Approximate entropies were calculated for the output bit streams of 16 Mbits measured by a logic analyzer at 25 °C, following the NIST SP800-22 [87]. Figure 3.14 shows the approximate entropies when (a) the difference in voltage in the two-voltage LC buffer and (b) the number of shorted NMOSs in the single-voltage LC buffer were changed. The pass marks for the NIST tests (= 0.69099) are also plotted, where the entropy of an ideal RNG is $\log_e 2 = 0.693$. Fast oscillators for (a) and (b) are the 31-stage ring oscillator whose VDD is 0.9 V and the 7-stage ring oscillator whose VDD is 1.2 V, respectively, and their duty cycles are adjusted within 50 ± 0.8 %. Figure 3.14 clearly demonstrates that the proposed jitter amplifiers improve randomness and enable sufficient entropies. The approximate entropy of Fig. 3.14(a) without jitter amplification (leftmost point) is relatively high compared to Fig. 3.14(b) and is improved less significantly than (b), because the slow oscillator of (a) had lower frequency and its intrinsic jitter was larger.

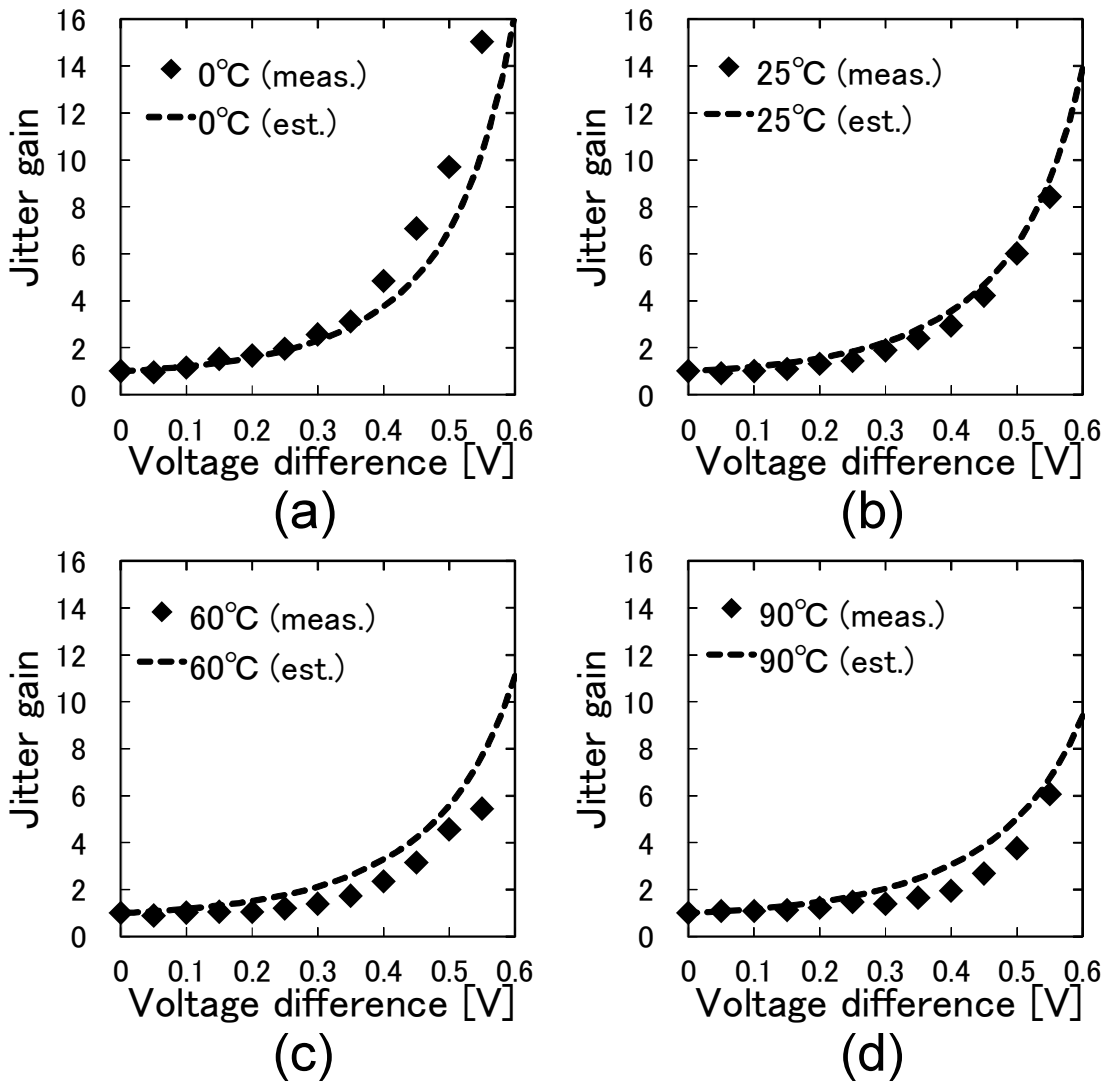

Figure 3.12: Jitter gain for two-voltage LC buffer. Temperatures are (a) 0°C, (b) 25°C, (c) 60°C and (d) 90°C.

3.4.4 Comparison with post-processors using NIST test

This section will discuss the advantages of the proposed jitter amplifier here by comparing it with simple post-processors, i.e., a XOR corrector [97] and a von Neumann corrector [96].

After a sufficient number of bits were generated from the TRNG without any correctors or jitter amplifiers, 100 Mbits of streams were obtained with the XOR corrector and von Neumann corrector. The same amount of random bit stream was also generated by the TRNG with jitter amplification with the two-voltage LC buffer. Then, their qualities were evaluated with the NIST test suite. The 31-stage ring oscillator at 0.9 V was the fast oscillator for the TRNG, and its duty cycle was adjusted within 50±0.1%. Depth of the XOR corrector is one. This evaluation applied 1.2 V of VDBUFH and 0.7 V of VDBUFL for the LC buffer, and then, the voltage difference was 0.5 V. Temperature was 25 °C. Table 3.1 lists the test parameters

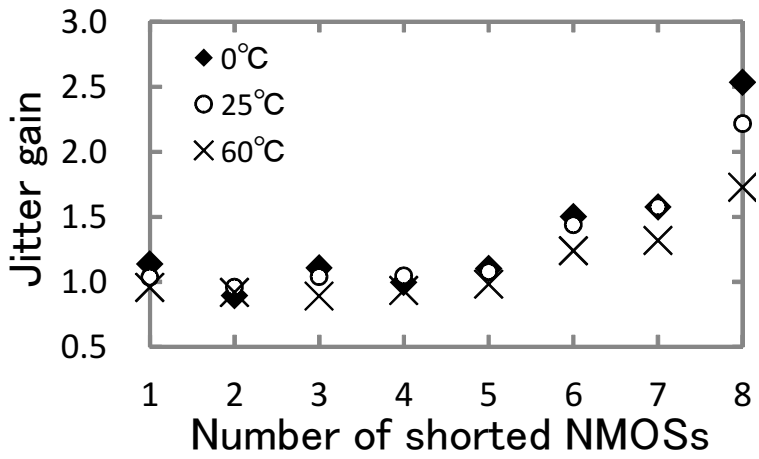

Figure 3.13: Measured jitter gain for single-voltage LC buffer.

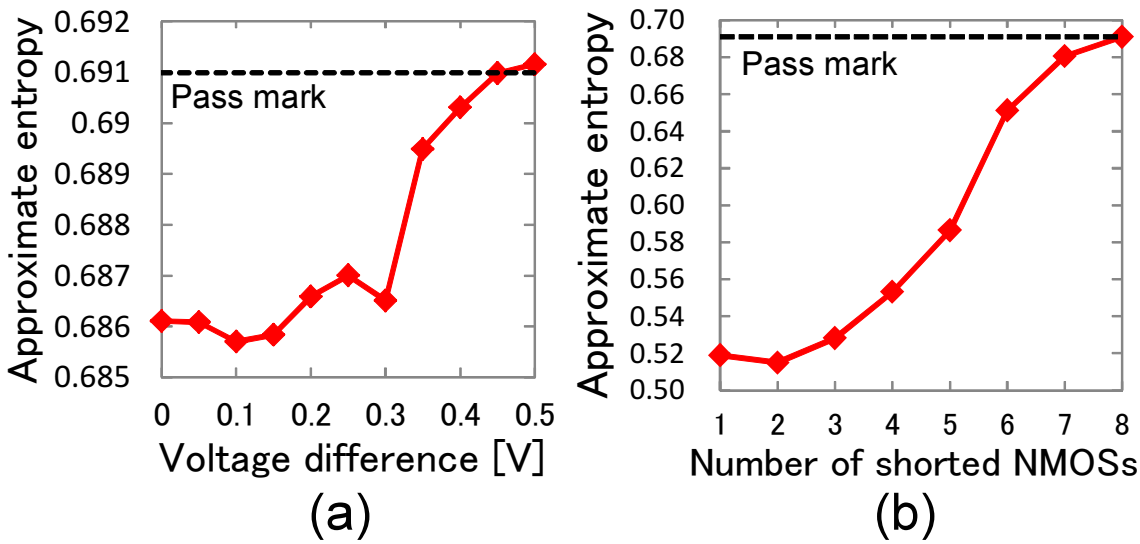

Figure 3.14: Approximate entropies. Pass marks (=0.69099) are also given. Temperature was 25°C. (a) Two-voltage LC buffer. (b) Single-voltage LC buffer.

this section employed, and the parameters satisfy recommendations given in NIST SP800-22. From Table 3.1, the pass range of the pass proportion for runs test is between 0.986 and 0.990, and the pass range for the other tests is between 0.961 and 1.000.

Table 3.2 summarizes the NIST test results. With neither a corrector nor a jitter amplifier (plain), seven tests failed, which evidences the low randomness of the raw outputs. The XOR corrector degraded the results for the NIST tests because XOR operation for a poorly random bit stream unbalanced its occurrences of 1/0, and what is worse, the corrector reduced throughput by half. Though the results can be improved with employing the depth of two or more, it unacceptably decreases throughput. Even though the von Neumann corrector increased the number of passed tests, six tests still failed. Additionally, the throughput after the corrector was 0.26 times smaller than that of “plain” in this case, where the reduction in

Table 3.1: Setup for NIST randomness tests. The other necessary parameters are automatically decided by the testing program provided by NIST.

Name	Value
Block length for BlockFrequency	20000
Block length for NonOverlappingTemplate	9
Block length for OverlappingTemplate	9
Block length for ApproximateEntropy	10
Block length for Serial	16
Sequence length for LinearComplexity	500
Significance level	0.01
Test data for Runs	20 Kbits × 5000 seqs.
Test data for the other tests	1 Mbits × 100 seqs.

Table 3.2: NIST randomness test results. P-value / pass proportion have been listed in each cell. Bold fonts indicate passed tests.

Test name	Plain	XOR corrector	Von Neumann corrector	Jitter amplifier
Frequency	**0.6993 / 0.99**	0.0000 / 0.00	**0.0270 / 0.99**	**0.1296 / 0.97**
BlockFrequency	**0.0095 / 1.00**	0.0000 / 0.00	**0.8832 / 0.98**	**0.2133 / 0.99**
CumulativeSums	**0.4944 / 0.98**	0.0000 / 0.00	**0.2248 / 0.99**	**0.0032 / 0.97**
Runs	0.0000 / 0.26	0.0000 / 0.02	0.0000 / 0.94	**0.1376 / 0.99**
LongestRun	0.0000 / 0.01	0.0000 / 0.00	0.0000 / 0.91	**0.9558 / 1.00**
Rank	**0.0156 / 1.00**	**0.3838 / 1.00**	**0.1917 / 1.00**	**0.6163 / 1.00**
FFT	**0.8165 / 1.00**	0.0000 / 0.79	**0.7981 / 1.00**	**0.3669 / 0.99**
NonOverlappingTemplate	0.0000 / 0.00	0.0000 / 0.00	0.0000 / 0.15	**0.0072 / 1.00**
OverlappingTemplate	0.0000 / 0.00	0.0000 / 0.00	0.0000 / 0.28	**0.1626 / 1.00**
Universal	0.0000 / 0.00	0.0000 / 0.00	**0.0028 / 0.98**	**0.3041 / 0.99**
ApproximateEntropy	0.0000 / 0.00	0.0000 / 0.00	0.0000 / 0.31	**0.8514 / 0.98**
RandomExcursions	**0.0267 / 1.00**	- / -	**0.0805 / 0.98**	**0.0554 / 1.00**
RandomExcursionsVariant	**0.0190 / 1.00**	- / -	**0.0127 / 0.98**	**0.0909 / 0.98**
Serial	0.0000 / 0.00	0.0000 / 0.00	0.0000 / 0.94	**0.3669 / 0.97**
LinearComplexity	**0.4559 / 1.00**	**0.3838 / 1.00**	**0.3191 / 0.97**	**0.7981 / 0.99**

throughput depended on the original bit stream. The jitter amplifier significantly improved the randomness of TRNG output to pass all the tests. Note that the deteriorations in p-values found in the frequency and FFT tests could be ignored since they were within the pass range. In this experimental setup, the voltage difference higher than or equal to 0.5 V is necessary to pass all NIST tests, and below 0.5 V some NIST tests failed. Because the LC buffer keeps the sampling frequency of TRNG unchanged, the same throughput as that for "plain" could be achieved. Thus, the jitter amplifier enables the target TRNG to generate a sufficiently random bit stream without degrading throughput.

3.4.5 Comparison with frequency divider

Dividing the slow oscillator output with a frequency divider, which often consists of serially-connected two-frequency dividers, is a simple solution to obtain large jitter. The two-frequency divider means the simplest asynchronous frequency divider which consists of an inverter and a DFF. 2^n-frequency divider is easily constructed by connecting n two-frequency dividers in series. For example, 16-frequency divider consists of a series of four

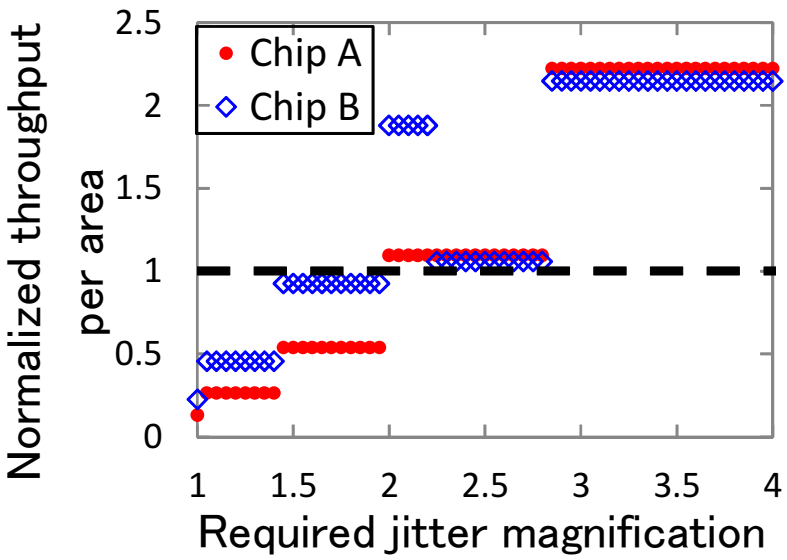

Figure 3.15: Normalized throughput per area vs. required magnification.

two-frequency dividers. Though there are many frequency dividers whose factors of divisions are not 2^n, their areas depend on the implementations. Thus, this section considers only 2^n-frequency dividers which consist of n two-frequency dividers. Here, the p-th period of the slow oscillator is $t_{slow(p)} \sim N(\mu_{slow}, \sigma^2_{slow})$, where μ_{slow} is the average of the slow periods and σ^2_{slow} is the variance. $t_{slow(p)}$ is independent of $t_{slow(p+k)}$, for $k \in \mathbb{Z}$. When the slow oscillator is divided by m_d, the variance is accumulated during m_d cycles of the slow oscillator. And then, the average period of the divided signal is $m_d\mu_{slow}$ and the variance is $m_d\sigma^2_{slow}$. Since the jitter is defined as a standard deviation of the periods, the jitter of the slow oscillator is σ_{slow} and that after a m_d-frequency divider is $\sqrt{m_d}\sigma_{slow}$, and thus the amount of the jitter is multiplied by $\sqrt{m_d}$ with a m_d-frequency divider, whereas the frequency becomes $1/m_d$ times smaller.

The jitter amplifier and the frequency divider are compared here using throughput per area of TRNG as a metric. Let us improve the jitter of a slow oscillator whose area is A_{osc}, frequency is F_{osc}, and jitter is σ_{osc}. When the required jitter is σ_{req}, the required magnification of jitter is $M_{req} = \sigma_{req}/\sigma_{osc}$. To attain M_{req} with two-frequency dividers whose area is A_{div}, $\lceil \log_2 M^2_{req} \rceil$ dividers are necessary. The required area is $A_{osc} + A_{div}\lceil \log_2 M^2_{req} \rceil$ and the throughput is $F_{osc}/2^{\lceil \log_2 M^2_{req} \rceil}$. On the other hand, when a jitter amplifier is used whose area is A_{ja} and achievable jitter gain is G_{ja}, $\lceil M_{req}/G_{ja} \rceil$ amplifiers are necessary. The area is $A_{osc} + A_{ja}\lceil M_{req}/G_{ja} \rceil$ and the throughput is F_{osc}. Here, the throughput per area of the jitter amplifier is divided by that of the frequency divider, and this value indicates whether the jitter amplifier is superior or not. The condition under which the throughput per area of the jitter amplifier is superior to that of the frequency divider is:

$$1 < \frac{F_{osc}}{A_{osc} + A_{ja}\lceil \frac{M_{req}}{G_{ja}} \rceil} \Bigg/ \frac{F_{osc}/2^{\lceil \log_2 M^2_{req} \rceil}}{A_{osc} + A_{div}\lceil \log_2 M^2_{req} \rceil}. \tag{3.9}$$

This means the jitter amplifier is superior in throughput per area, when the right hand term of Eq. (3.9), called as the normalized throughput per area, is more than one.

Figure 3.15 shows the normalized throughput per area as the required magnification M_{req} changes. The normalized throughput per area was calculated from the right term in Eq. (3.9) using the parameters value below. For chip A, A_{osc} = 495.0 μm^2, A_{ja} = 3300.0 μm^2, A_{div} = 8.0 μm^2, and G_{ja} = 8.4. As for chip B, A_{osc} = 478.0 μm^2, A_{ja} = 1663.0 μm^2, A_{div} = 8.0 μm^2, and G_{ja} = 2.2. The areas for the slow oscillator, two-frequency divider, and jitter amplifier are based on the layouts of chips A and B. The gain of the jitter amplifier was set to the largest value observed in the measurement. Figure 3.15 indicates an oscillator-based TRNG should employ a jitter amplifier when required jitter magnification is larger than 2.0. With this jitter magnification, chip A with the proposed jitter amplifier passed all NIST tests as shown in Sec 4.4. On the other hand, TRNGs with the same magnification by frequency divider and chip B are assumed to pass NIST tests while NIST tests were not carried out for them. Large jitter magnification is required when designing a good TRNG that generates highly random numbers at high throughput yet occupies a small area. The jitter amplifier is more suitable than the frequency divider for such purposes.

3.5 Conclusion

A jitter amplifier for an oscillator-based TRNG has been developed with a 65 nm CMOS process. The results from measurements demonstrated that it efficiently amplified the jitter of the sampling signal and enabled a TRNG with high throughput yet a small area with sufficient random bit stream quality. One of the fabricated jitter amplifier attained 8.4x of jitter gain occupying 3,300 μm^2, the other amplifier achieved 2.2x gain with area of 1,700 μm^2. In addition, this chapter tested and confirmed that the jitter amplifier was better than simple correctors and a frequency divider in most cases.

Chapter 4

Self-calibration system of duty cycle under dynamic environmental variation

This chapter describes a self-calibration system of duty cycle for fast oscillator. A duty cycle monitor and a duty cycle corrector are presented. A prototype oscillator-based TRNG employing the proposed system is developed with a 65 nm CMOS process. The monitor estimates the duty cycle with an average resolution of 0.16 % and the duty cycle corrector tunes the duty cycle with resolution of 0.11 % in average at 20 °C. The duty cycle monitor accelerate the estimation of duty cycle compared to output bit sampling, and the required time for estimation is 3,500 times smaller. The proposed system reduces the variation of probability of '1' occurrence from 0.90 % to 0.05 % even when the temperature varies from 5 to 60 °C.

4.1 Introduction

Removal of biasing between the probabilities of '1' and '0' is a critical for the oscillator-based TRNG design, as is discussed in Sec. 1.2.4. Online tuning of the duty cycle of the fast oscillator is required to overcome temperature fluctuation. This section does not consider the process variation, which is another concern about the biasing, since it is a static variation and is compensated before shipment. Bhatti et al. [121] presented a method of duty cycle measurement and correction using a random sampling technique. This method samples the objective signal with a random clock from a chaotic oscillator, counts the numbers of '1' and '0', and calculates the proportion of '1' as the duty cycle. The method, however, takes a lot of time to count the number of '1' and '0'.

This chapter presents a self-calibration system of duty cycle for the oscillator-based TRNG. The system includes a duty cycle monitor and a duty cycle adjustment circuit. When the duty cycle is changed by environmental fluctuation, the proposed monitor measures the duty cycle of the fast oscillator and the information is fed back to the duty cycle corrector. The duty cycle corrector tunes the duty cycle of the fast oscillator, and thus, the bias between the occurrences of '1' and '0' is removed.

The reminder of this chapter is organized as follows. Section 4.2 describes the behavior of the self-calibration system. Section 4.3 proposes the monitor of duty cycle, and Section 4.4

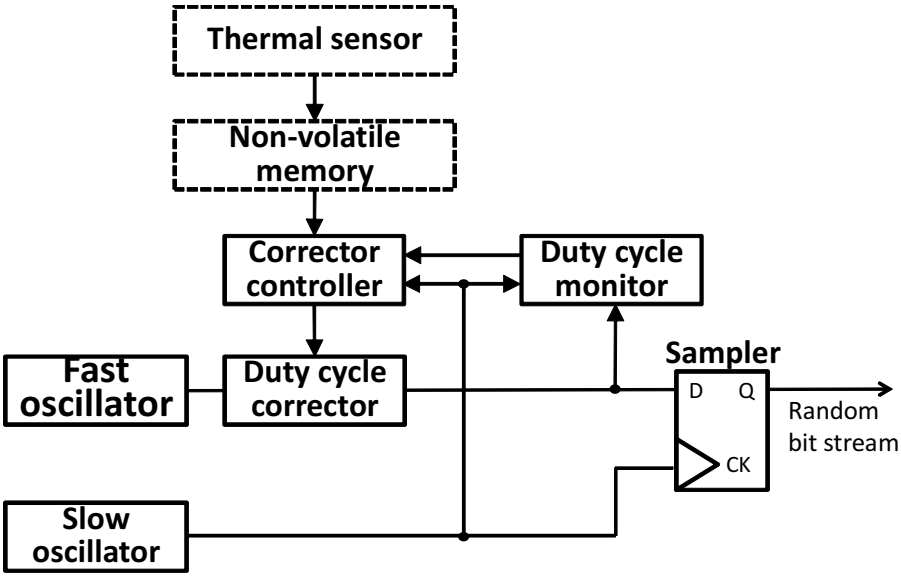

Figure 4.1: An oscillator-based TRNG with the proposed self-calibration system.

explains the duty cycle corrector. Section 4.5 shows the experimental results. Conclusions of this chapter are given in Section 4.6.

4.2 Behavior of self-calibration circuit

Figure 4.1 shows an oscillator-based TRNG with the self-calibration system for the fast oscillator. The proposed system consists of the duty cycle monitor, the duty cycle corrector, and the corrector controller. Temperature information, which is provided by an external thermal sensor, is also given to the corrector controller.

The duty cycle monitor receives the oscillating signal of the fast oscillator and measures the duty cycle. The outputs of the monitor are digital signals, which enables easy processing in the corrector controller. Note that the duty cycle monitor and the sampler are also affected by environmental fluctuations such as temperature variation, and therefore, the same digitized monitor outputs does not always mean the same probability of '1' when the temperature changes. Accordingly, this thesis preliminarily measures target digitized monitor outputs, which produces the target probability of '1', in various temperatures. The non-volatile memory contains a table of the target monitor outputs. The controller gets the monitor output, and look up the target monitor output from the table whose key is temperature. Then, the controller outputs the control signals to the duty cycle corrector so that the measured output gets close to the target. The duty cycle corrector varies the duty cycle of fast oscillating signal according to the control signals. Note that the thermal sensor and the non-volatile memory can be fabricated on the same chip though the TRNG in Fig. 4.1 implements them as external components.

4.3 Duty cycle monitor

Figure 4.2 illustrates a concept of the proposed duty cycle monitor. Output bit sampling, which is a method of duty cycle estimation, is also shown for comparison. This method, whose principle is similar to the random sampling method, gathers the output bits from the sampler and calculates the proportion of '1' as the duty cycle. Note that output bit sampling uses jittery slow oscillator as the random clock while the random sampling method employs the chaotic oscillator, and then output bit sampling requires several cycles of the slow oscillator. On the other hand, the proposed monitor directly measures time differences between HIGH and LOW of the fast signal and obtains the duty cycle. The monitor estimates the duty cycle within one cycle of the slow oscillator, which is much smaller than output bit sampling.

Figure 4.3 has a block diagram of the duty cycle monitor and schematics of the internal circuits. The monitor circuit mainly consists of two duration to delay converter (DDC) and a Vernier time to digital converter (TDC) [122, 123]. The DDC includes P-type DDC and N-type DDC. The DDCs receive the oscillating signals from the fast oscillators through the duty cycle corrector and output rising edges. The output signal from P-type DDC rises earlier as the duty cycle of the fast oscillator gets lower, and that from N-type DDC rises earlier as the duty cycle gets higher. Consequently, the time difference between the two rising edges from the P-type DDC and the N-type DDC becomes larger as the duty cycle increases, where the edge from the N-type DDC is earlier than the P-type. The Vernier TDC digitizes the time difference between the rise edges from the two DDCs and transmits the information to the controller. In this work, 63 bit Vernier TDC is implemented. The variable delay after N-type DDC in Fig. 4.3 adjusts the time difference into the input range of the TDC.

The DDC includes serially-connected DDC units whose schematics are also shown in Fig. 4.3. This work implements six DDC units. This section here explains the behavior of the P-type DDC unit, and the similar explanation is valid for N-type DDC unit. The P-type DDC unit gets the signal from the fast oscillator (fast), and positive and negative resets (rst and xrst), which are generated with the signal of the slow oscillator. The voltage of the negative input node (xin) is equal to rst for the first DDC unit and otherwise the node is connected to the output node (out) of previous unit. The P-type DDC unit consists of five transistors. C_{M2} and C_{M5} represent the parasitic capacitances of M2 and M5 whose gate areas are larger than the other transistors. Figure 4.4 illustrates waveforms at the nodes in the P-type DDC unit. $V(C_{M5})$ and $V(C_{M2})$ mean the input voltages to M5 and M2. In an initial state, xrst is LOW, and rst and xin of the first unit are HIGH. C_{M2} is charged through M4 and C_{M5} is discharged through M3. After that, xrst is pulled up and rst and xin of the first unit are pulled down while the fast oscillating signal is input to fast. M1 and M2 charge C_{M5} while fast is LOW. The transistors do not fully charge C_{M5} within a cycle of fast and $V(C_{M5})$ in Fig. 4.4 increases gradually rather than stepwise, because the frequency of fast is high, and RC product of C_{M5} and on-resistance of M2 is much larger than the cycle time of the fast oscillator. This enhances the resolution of the proposed monitor since the difference between the time of HIGH and LOW accumulates during several cycles. M5 is turned on when C_{M5} is sufficiently charged up. C_{M2} is discharged through M5, and then, LOW signal is output to the next DDC unit. The NOT gate in Fig. 4.3 inverts the fall edge from the last DDC unit and generates a rise signal as an output of the P-type DDC. A propagation delay of the DDC unit depends on the time for charging C_{M5}, and therefore, the delay gets smaller as the duty cycle

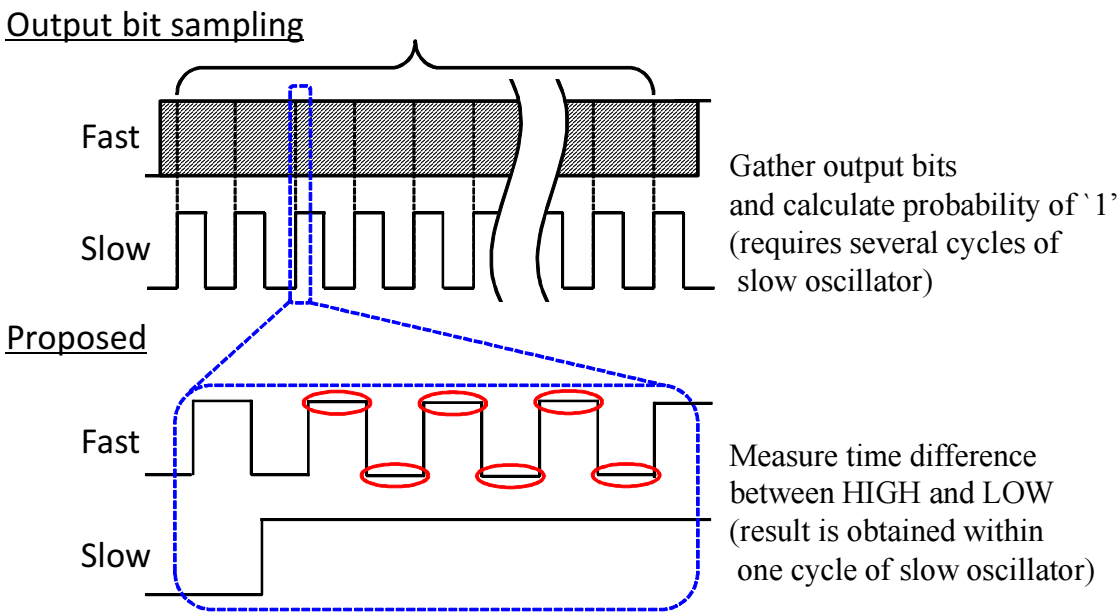

Figure 4.2: Concept of the duty cycle monitor. Output bit sampling is also shown for comparison.

of fast increases.

4.4 Duty cycle corrector

The duty cycle corrector this section presents is a programmable delay cell. The propagation delay for rising input is controllable while the delay for falling input is constant. Figure 4.5 shows the behavior of the delay cell. An oscillating signal is input to in and is output to out being delayed through the cell. The time of HIGH and LOW of in are T_{HIGH} and T_{LOW}, and those of out are T'_{HIGH} and T'_{LOW}, respectively. The propagation delay for fall edge is D_{fall} and that for rise edge is D_{rise}, which is controlled with ctrl. Here, $T'_{HIGH} = T_{HIGH} - D_{rise} + D_{fall}$ and $T'_{LOW} = T_{LOW} + D_{rise} - D_{fall}$ as illustrated in Fig. 4.5. Then, the duty cycle of in, d_{in}, and that of out, d_{out}, are expressed as follows:

$$d_{in} = \frac{T_{HIGH}}{T_{HIGH} + T_{LOW}}, \tag{4.1}$$

$$\begin{aligned} d_{out} &= \frac{T'_{HIGH}}{T'_{HIGH} + T'_{LOW}} \\ &= \left(d_{in} + \frac{D_{fall}}{T_{HIGH} + T_{LOW}}\right) + \frac{D_{rise}}{T_{HIGH} + T_{LOW}}. \end{aligned} \tag{4.2}$$

The programmable delay cell tunes d_{out} by changing D_{rise}.

Figure 4.6 shows a schematic of the programmable delay cell, namely the duty cycle corrector. An oscillating signal is input to in and is output to out. A programmable inverter [64] and a normal inverter whose input capacitance is C_{inv} compose the corrector. The programmable inverter consists of inverters connected in parallel, one of which includes gating transistors to control NMOS drive strength. A rising signal of in propagates through the

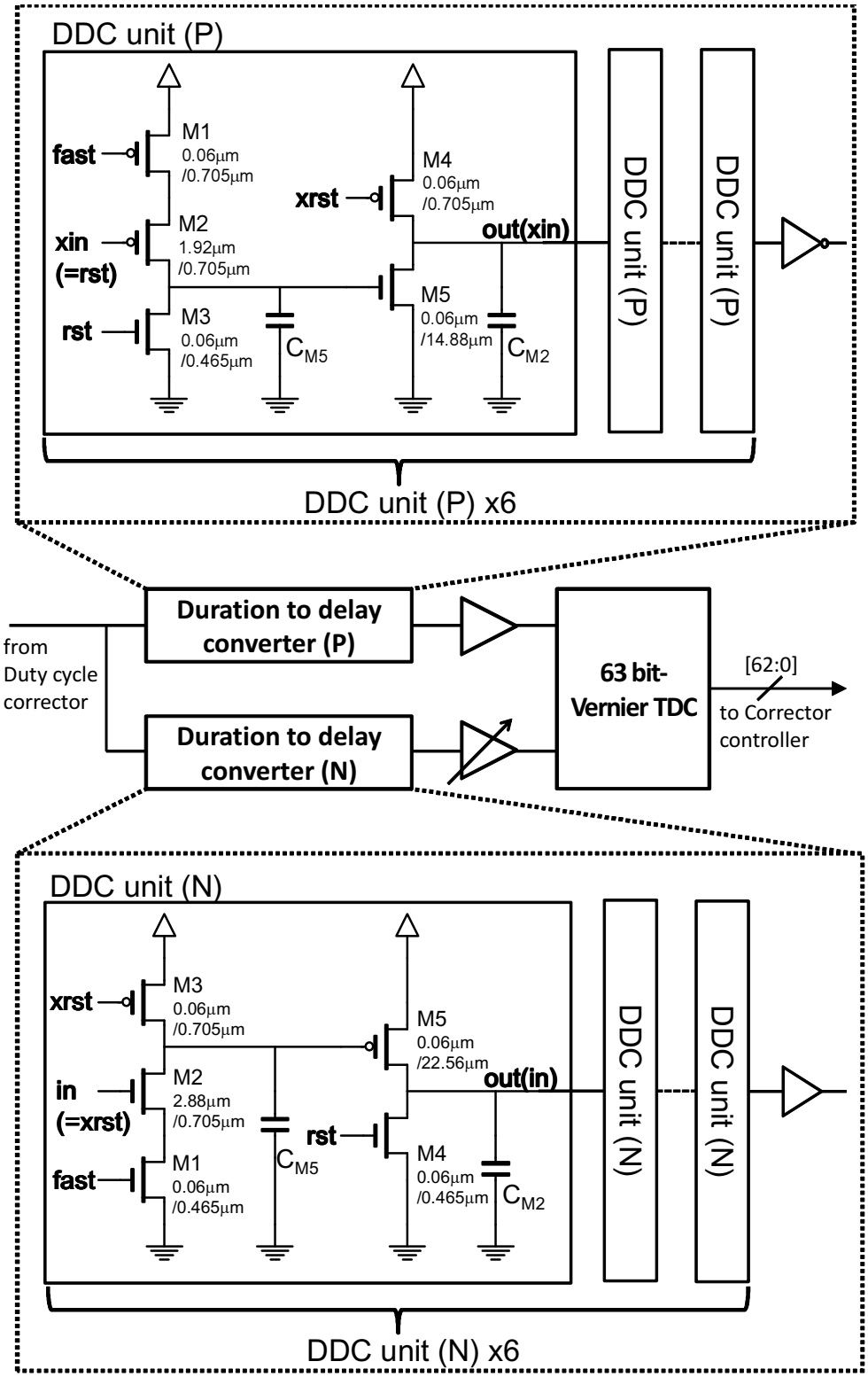

Figure 4.3: Circuit diagram of the duty cycle monitor. (Length/Width)

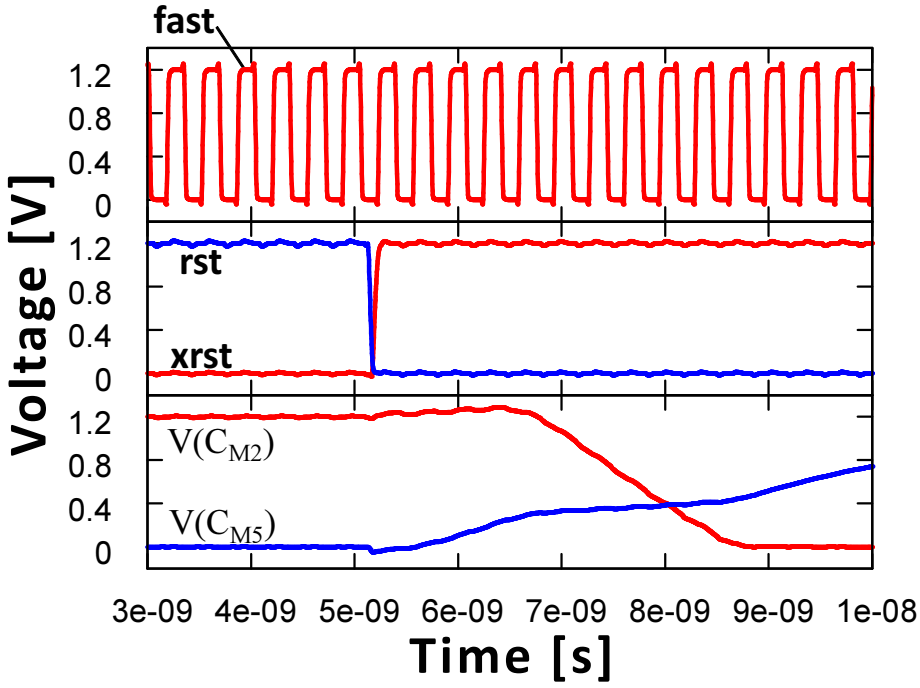

Figure 4.4: Waveforms inside a P-type DDC unit.

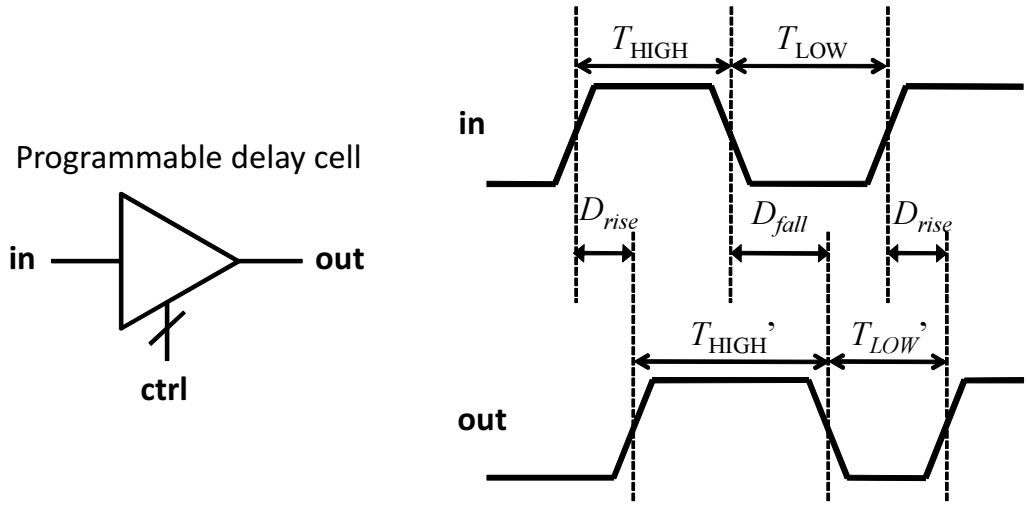

Figure 4.5: Behavior of programmable delay cell for duty cycle correction.

programmable inverter by discharging C_{inv} with the NMOSs, and the NMOS drive strength depends on the number of gating transistors which are turned on. The control signal, ctrl, thus changes the delay for a rise signal in the programmable inverter. On the other hand, the PMOS drive strength depends on the sizes of the PMOSs, M1 and M3, and it is constant. Consequently, the duty cycle corrector tunes the duty cycles by changing the propagation delay for a rise input.

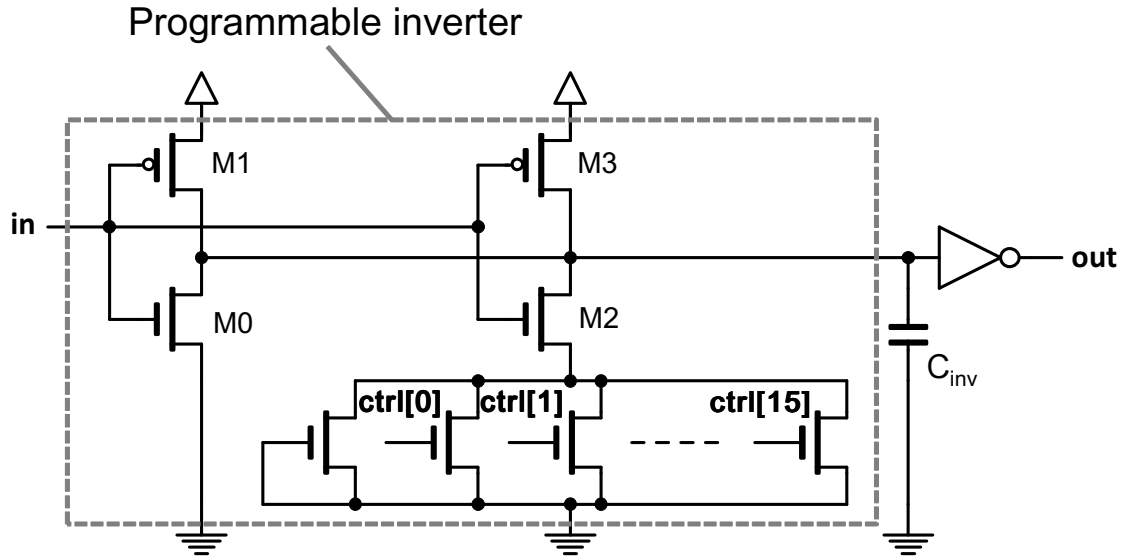

Figure 4.6: Schematic of the duty cycle corrector.

4.5 Experimental results

This section describes a test chip which implements the proposed self-calibration system and experimental results.

4.5.1 Test structure

An oscillator-based TRNG with the proposed self-calibration system was fabricated with a 65 nm CMOS process (Fig. 4.7). The area for the duty cycle corrector is 190 μm^2 and that for the duty cycle monitor is 2,790 μm^2. The fast oscillator is a 7-stage ring oscillator whose average period and frequency are 350 ps and 2.9 GHz from circuit simulation. The proposed corrector and the monitor of duty cycle are also implemented. A 293-stage ring oscillator with 16-frequency-dividers are the slow oscillator. Average period and frequency of the slow oscillator are 150 ns and 6.7 MHz from measurement with a real time oscilloscope.

The probability of '1' occurrence was roughly estimated as 42 % from measurement results, and then, the target probability of '1' was set as 42.00 % in this section though an ideal probability was 50.00 %. Careful layout in designing TRNG and the body biasing technique discussed in Sec. 2.3.1 deal with the offset of the probability. This section focuses on dynamic variations such as fluctuation of temperature rather than static variations, and thus, the section does not discuss in detail the issue of the offset compensation.

4.5.2 Resolution of duty monitor

Figure 4.8 shows the relationships between the measured monitor outputs and the probabilities of '1' occurrences at temperatures of 5, 20, 40 and 60 °C. The VDD for DDC was 0.9 V and the others were 1.2 V. The number of the on-transistors in the duty cycle corrector

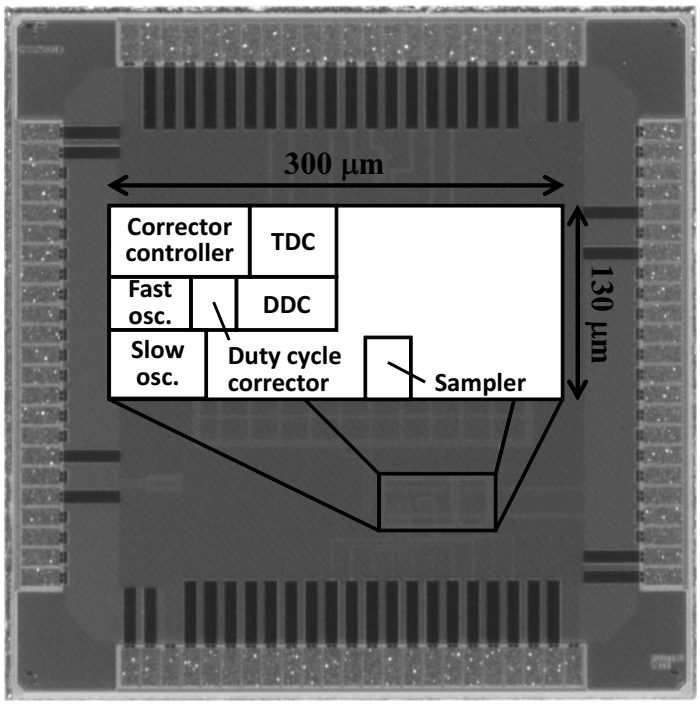

Figure 4.7: Chip photo of a TRNG with self-calibration system.

was varied from 1 to 17, and only the measurement at 5 °C changes the number of transistors by twos. This experiment measured the outputs from the duty cycle monitor through the corrector controller, which converts the bit length of the measured value from 63 bits to 6 bits by encoding. One hundred of outputs from the duty cycle monitor were obtained with a logic analyzer and their average was calculated. One hundred sequences of 100 kbits of random bits were used to estimate the true probability of '1' as a reference. Fig. 4.8 shows that the measured output value of the monitor is linear to the probability of '1', which validates the functionality of the proposed duty cycle monitor. This section also derived first order approximations, $y\,[\%] = ax + b$, from the measurement results and displayed them in Fig. 4.8. Here, the coefficient a represents the average resolution of the duty cycle monitor. The monitor at 20 °C, for example, increments the output value when the input duty cycle increases by 0.16 %. The average resolutions of the monitor were from 0.14 to 0.20 % and they were small enough, since the frequency test of NIST tests requires the range of probability to be within 0.25 %.

4.5.3 Quickness of duty cycle measurement

It is important to estimate the duty cycle of fast oscillator quickly with high accuracy, when considering environmental fluctuation with high frequency such as power supply noise. Employing the average of the several duty cycles enhances the accuracy, but it decreases the speed of the estimation. This subsection evaluates the measurement quickness of the proposed duty cycle monitor. For comparison, the output bit sampling method is considered here, which gathers a bit sequence and calculates the proportion of '1' occurrence as the duty

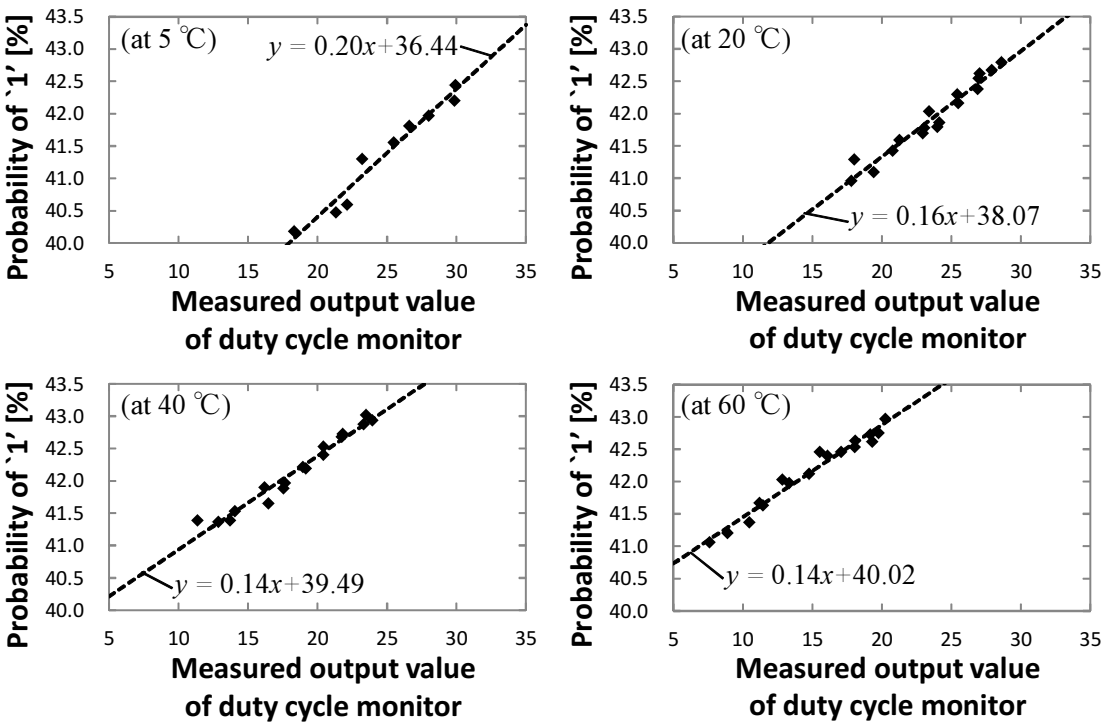

Figure 4.8: Probabilities of '1' occurrences vs. measured values of monitor outputs. Broken lines represent the first order approximations.

cycle.

Figure 4.9 shows the accuracies of duty cycle estimation with the duty cycle monitor and with the output bit sampling. The y-axis represents the standard deviation of measured 100 duty cycles, which corresponds to the statistical accuracy of the duty cycle estimation. The x-axis is the time spent to measure a duty cycle. In other words, it is the required time to attain the corresponding accuracy. The VDD for the DDC of the duty cycle monitor was 0.9 V and the others were 1.2 V. Note that the measured output value of the monitor was digital data whose bit length was 6 bits, and the value was converted into the duty cycle. The first order approximation of the probability of '1' at 25 °C was $y = 0.15x + 38.59$, and then, the measured output value was multiplied by 0.15. For example, when a standard deviation of measured output values is 3.0, the converted accuracy of duty cycle is $3.0 \times 0.15 = 0.45$ %. Fig. 4.9 indicates that the required time with the proposed monitor is 3,500 times as small as the output bit sampling when setting a pass mark of accuracy to 0.25 % as an example.

4.5.4 Resolution of duty cycle corrector

Figure 4.10 shows probabilities of '1' occurrences when the number of on-transistors of the duty cycle corrector is varied. Temperatures were 5, 20, 40 and 60 °C. This evaluation measured 100 sequences of 100 kbits to obtain the probability of '1'. This subsection also derived first order approximations of the measurement results, which are found in Fig. 4.10. Fig. 4.10 shows that the proposed corrector has good linearity, since the measured probabil-

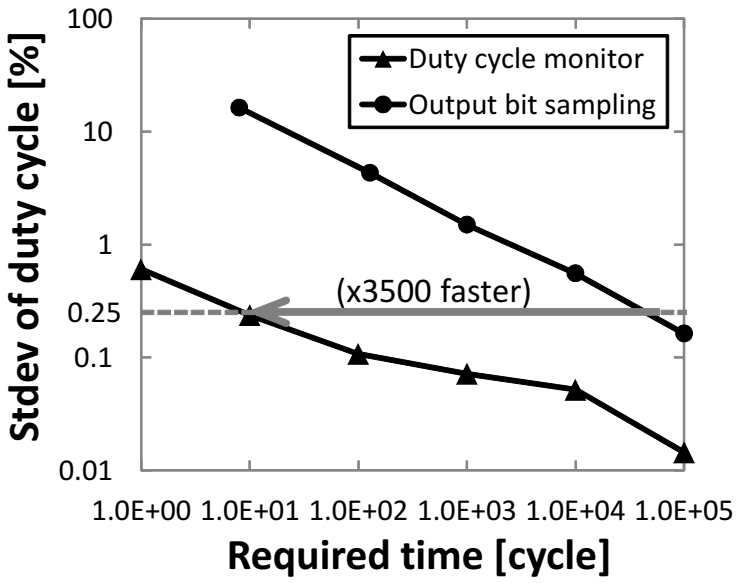

Figure 4.9: Required number of cycles with duty cycle monitor vs. output bit sampling. Target standard deviation of probability is 0.25 %. Temperature is 25 °C.

ities are well correlated with the approximated linear expression. The coefficient of x in the linear expression represents the average resolution of the duty cycle corrector. The corrector tunes the duty cycle with average resolution of from 0.11 to 0.15 %, which is sufficiently small compared to the acceptable range of the frequency test of 0.25 %.

4.5.5 Tolerance to temperature

This subsection explains stabilization of probability of '1' by self-calibration system. As a preparation, target output values of the duty cycle monitor were set for each temperature from Fig. 4.8. For example, the first order approximation of the duty cycle monitor at 40 °C is $y = 0.14x + 39.49$, and then, x is set to 18 when y is 42.00. Table 4.1 lists the target output values of the duty cycle monitor. This section manually set the target output values for each temperature while they would be stored in the non-volatile memory and automatically according to the thermal sensor output in Fig. 4.1.

Figure 4.11 plots the probabilities of '1' occurrences under temperature fluctuation with and without the self-calibration. The VDD for the DDC was 0.9 V and the others were 1.2 V. The probability without the calibration spans from 41.55 to 42.46 %, and the range is 0.90 %. On the other hand, the probability with the calibration is within 41.96 to 42.01 %, and the range is 0.05 %, which is small enough to pass the frequency test of NIST tests explained in Sec. 1.2.4. Thus, the proposed calibration system makes the quality of output bit stream robust to temperature fluctuation.

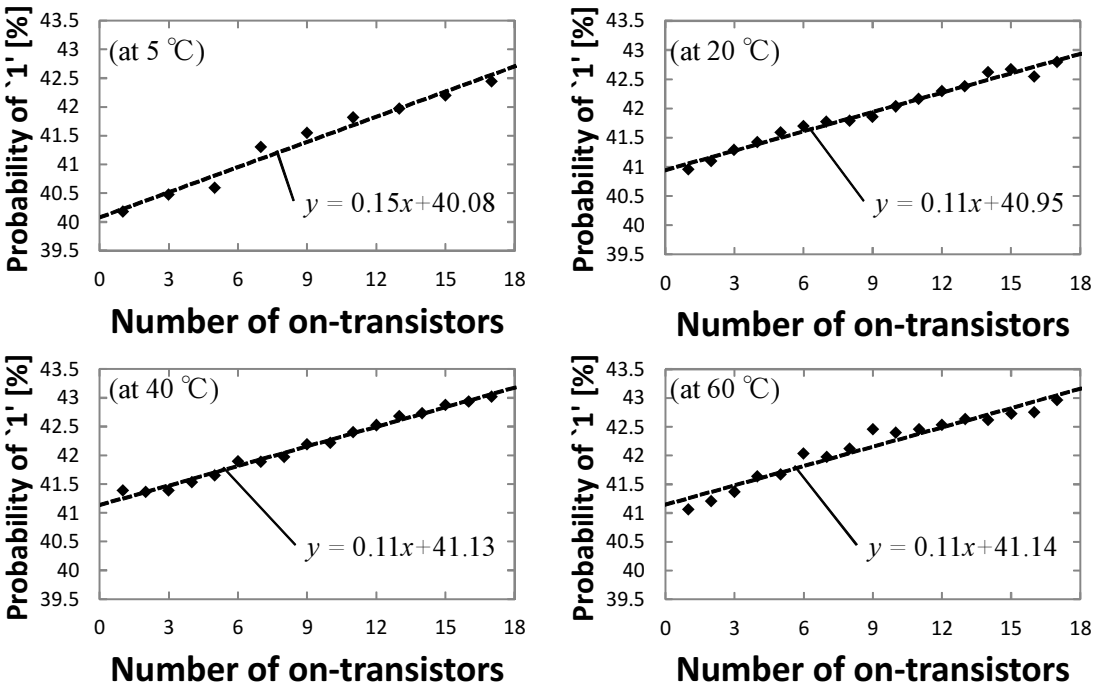

Figure 4.10: Probabilities of '1' occurrences vs. the number of on-transistors in duty cycle corrector. Broken lines represent the first order approximations.

Table 4.1: Target output values of duty cycle monitor.

Temperature [°C]	Target value
5	6'd29
20	6'd23
40	6'd18
60	6'd14

4.6 Conclusion

This chapter proposed a system to self-calibrate the duty cycle of fast oscillator, which includes a duty cycle corrector and a duty cycle monitor. A oscillator-based TRNG with the proposed system was implemented with a 65 nm CMOS process. The proposed monitor and the corrector attained sufficient performance to pass the frequency test of the NIST test set. Estimation of duty cycle with the proposed duty cycle monitor is 3,500 times faster compared to the output bit sampling with counting '1' occurrences. The proposed system reduced the variation of the probability of '1' due to temperature fluctuation to be within 0.05 %.

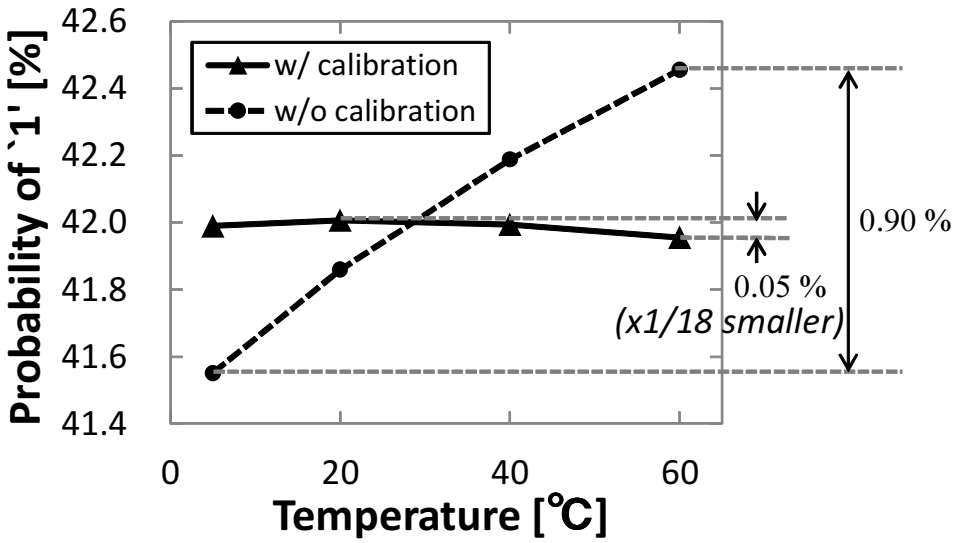

Figure 4.11: Stabilization of probability of '1' occurrence with self calibration system. Target duty cycle is 42.00 %.

Chapter 5

Conclusion

Many security applications need unpredictable random numbers, which are generated from physical random sources. This thesis focuses on an oscillator-based TRNG among various types of TRNGs because of its compatibility with on-chip integration and inherent robustness to deterministic noise. An computationally efficient model for randomness evaluation is required to design an oscillator-based TRNG which produces sufficiently random numbers even under deterministic noises. The amount of jitter of an oscillator, which is the random source of the TRNG, is not enough to attain enough randomness, and then an oscillator with large jitter is demanded. Environmental variability, such as temperature change, biases a probability of '1' occurrence of output bits. The output biasing of an oscillator-based TRNG depends on the distortion of oscillator duty cycle, and therefore, a self-calibration of duty cycle is indispensable.

Chapter 2 proposes a stochastic behavior model of the oscillator-based TRNG and a worst-case-aware design methodology using the model. The design methodology evaluates the worst χ value of a poker test under deterministic noise not requiring bit generation. The proposed model is validated with hardware measurement of an oscillator-based TRNG which is fabricated in a 65 nm CMOS process. It is experimentally verified that the worst-case-aware design methodology efficiently estimates the lower bound of randomness. In addition, the chapter exemplifies a design space exploration with the proposed methodology.

Chapter 3 describes a jitter amplifier, which increases the amount of jitter and enhances the quality of output. A core element of the proposed circuit is a LC buffer controlled by a timing generator. The chapter presents two types of LC buffer, and implements an oscillator-based TRNG for each LC buffer with a 65 nm CMOS process. The measurement results confirms that the proposed circuit amplifies the jitter and improves randomness. The chapter also derives an analytical expression to estimate the jitter gain, and the estimated gains are well correlated to the measurements. The jitter amplifier attains higher throughput per area than a simple method of jitter accumulation with frequency dividers.

Chapter 4 explains a self-calibration system of duty cycle to remove the biasing. The proposed system includes a duty cycle monitoring circuit and a duty cycle adjustment circuit. The evaluation of duty cycle with the proposed monitor is quite faster than output bit sampling. The proposed method is suitable for quickly catching up with environmental variation. An oscillator-based TRNG accompanied by the self-calibration system is fabricated with a 65 nm CMOS process. Measurement results show that the duty cycle monitor and the duty cycle corrector achieve sufficiently high resolutions. The self-calibration system effectively reduces the biasing of output bits under temperature variation.

Table 5.1: Comparison with other TRNGs.

	Bucci2003 [42]	Bucci2008 [38]	Pareschi2010 [57]	Srinivasan2010 [71]	This work
Principal	Direct amp.	Oscillator-based	Chaos-based	Metastable-based	Oscillator-based
Technology	180 nm	90 nm	180 nm	45 nm	65 nm
Area	25,000 μm^2	13,000 μm^2	126,000 μm^2	4,004 μm^2	7,500 μm^2
Normalized area	1,563 μm^2	3,250 μm^2	7,875 μm^2	4,004 μm^2	3,595 μm^2
Throughput	40 Mbps	1.74 Mbps	80 Mbps	2.4 Gbps	2 Mbps
Randomness Assessment	FIPS140-1 Knuth	AIS31 Entropy eval.	NIST SP800-22	NIST SP800-22 Entropy eval. Autocorrelation eval. Run length eval.	NIST SP800-22 FIPS140-2 Entropy eval. Worst χ eval.
Post processing	XOR	LSFR	-	-	-

For a comparison, let us assume a TRNG designed with the proposed circuits so that it generates highly random numbers. The fast oscillator is a 7-stage ring oscillator whose frequency is 2.9 GHz from circuit simulation, and its duty cycle is adjusted by body biasing technique. The slow oscillator is a 251-stage ring oscillator with a 64-frequency divider and its output frequency, which is the throughput of the TRNG, is 2 MHz. The jitter amplifier with the two-voltage LC buffer and the self-calibration system are implemented. The sampler is similar to that in Fig. 4.7. Total area of the TRNG is 7,500 μm^2. With this configuration, the TRNG is expected to attain enough entropy and pass NIST test suite and FIPS140-2 randomness tests, because the slow oscillator and the jitter amplifier are similar to those of the chip A in Chapter 3 and the frequency of the fast oscillator is higher than that of the chip A. In addition, the worst χ value of the TRNG is calculated as 0.1 when duty cycle is 50.1 %, variance constant is 1.77×10^{-14}, and jitter gain is 8.4, which means the TRNG guarantees enough randomness under deterministic noises.

Table 5.1 shows the performance comparison to other recent TRNGs. The normalized area is the scaled area when the TRNGs are assumed to be fabricated in 45 nm CMOS process. For example, the area of this work is 7,500 μm^2 and the technology is 65 nm, and then, the normalized area becomes $7,500 \times (45/65)^2$ =3,595 μm^2. From Table 5.1, this work attains the smallest normalized area in the TRNGs which pass NIST test suite without post processing. Additionally, this work has a significant advantage that the TRNG can guarantee sufficient randomness even when deterministic noises are imposed.

Consequently, this thesis attains an oscillator-based TRNG which generates highly random numbers even under deterministic noises and environmental fluctuation. The proposed TRNG is process-portable and process-scalable since it can be implemented with digital circuits. In addition, the designers do not need to care the deterministic noises in designing the TRNG, since the proposed circuit guarantees enough randomness even under deterministic noises. Though deterministic noises and environmental fluctuation degrade the randomness of a bit stream and lower the quality of security system, the robust TRNG can decrease such risks. Thus, the TRNG enhances the reliability of security system such as cryptosystem and authentication.

Finally, the future direction and the remaining works are remarked in the following. The worst-case-aware design methodology presented in Chapter 2 employs χ value of a poker test as a metric of randomness. Evaluation of χ value, however, is a simple randomness test, and then, more profound evaluation is required to enhance the reliability of the design methodology. For example, entropy of a bit stream is a widely accepted as a metric of randomness. Then, the direct worst entropy estimation with the proposed stochastic model is one of the

future works.

The quickness of duty cycle estimation with the proposed monitor is compared to the output bit sampling in Sec. 4.5.3. On the other hand, recent TRNGs [38, 71] update their operating configurations every cycle to remove the bias. The speed of compensation is higher than the output bit sampling, while the accuracy of estimation of biasing is not high since it utilizes only one output bit for the estimation. Accordingly, to compare the proposed system with the recent studies is a future work of this thesis.

In addition, a self-calibration system of jitter amount is another future work. Although injection locking attack is discussed in Sec. 2.6.3, the reduction in the amount of jitter is still a concern for the oscillator-based TRNG. A TRNG with tolerance to the jitter reduction will include a jitter monitoring circuit and frequency dividers whose number of division is controlled according to the monitored amount of jitter. Estimation of jitter amount from output bits will be another choice, though its algorithm should be carefully selected for the circuit not to consume large area and power.

Appendix A

Mathematical proof of the worst case under deterministic noise

In Section 2.2, the representative phases of cycles are fixed to the center of the HIGH period (duty cycle > 0.5) or the LOW period (duty cycle < 0.5), to evaluate the worst randomness. This appendix here clarifies that such condition results in the worst entropy, which is a popular metric of randomness.

In this appendix, t is defined as a time interval from a rising edge of fast oscillator to the next rising edge of slow signal ($0 \leq t < t_{\mathrm{fast}}$). Assuming a Gaussian distribution, the probability density function of t is

$$f(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t-t_0)^2}{2\sigma^2}\right). \tag{A.1}$$

where t_0 is the representative phase.

Due to the definition of t, the fast signal is HIGH for $0 \leq t < dt_{\mathrm{fast}}$ and is LOW for $dt_{\mathrm{fast}} \leq t < t_{\mathrm{fast}}$, where d is the duty cycle of the fast oscillator. Thus, p_n, which is the probability of '1' occurrence at the n-th bit in successive bits, can be calculated by integrating $f(t)$ as follows:

$$p_n = \sum_{l=-\infty}^{\infty} \int_{lt_{\mathrm{fast}}}^{lt_{\mathrm{fast}}+dt_{\mathrm{fast}}} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t-t_{0n})^2}{2\sigma^2}\right) dt, (l \in \mathbb{Z}), \tag{A.2}$$

where t_{0n} is the representative phase for the n-th bit. In the following discussion, duty cycle d is $0.5 \leq d < 1$, which means that the HIGH period is longer. The same discussion is doable for $0 \leq d < 0.5$ with a substitution of $d' = 1 - d$. t_{fast} and d are assumed to be independent from n, because their fluctuations are considered as the random jitter or the variation of t_{0n}.

The derivative of p_n with respect to t_{0n} is derived.

$$\begin{aligned}\frac{\partial}{\partial t_{0n}} p_n &= \frac{1}{\sqrt{2\pi}\sigma} \sum_{l=-\infty}^{\infty} \int_{lt_{\text{fast}}}^{lt_{\text{fast}}+dt_{\text{fast}}} \frac{t - t_{0n}}{\sigma^2} \mathrm{e}^{-\frac{(t-t_{0n})^2}{2\sigma^2}} dt. \\ &= \frac{1}{\sqrt{2\pi}\sigma} \sum_{l=-\infty}^{\infty} \left[\exp\left(-\frac{(t - t_{0n})^2}{2\sigma^2} \right) \right]_{lt_{\text{fast}}}^{lt_{\text{fast}}+dt_{\text{fast}}} \\ &= \frac{1}{\sqrt{2\pi}\sigma} \sum_{l=-\infty}^{\infty} \left\{ -\exp\left(-\frac{(lt_{\text{fast}} + dt_{\text{fast}} - t_{0n})^2}{2\sigma^2} \right) + \exp\left(-\frac{(lt_{\text{fast}} - t_{0n})^2}{2\sigma^2} \right) \right\}. \end{aligned} \tag{A.3}$$

$(lt_{\text{fast}} + dt_{\text{fast}} - t_{0n})^2$ is compared with $(lt_{\text{fast}} - t_{0n})^2$.

$$(lt_{\text{fast}} + dt_{\text{fast}} - t_{0n})^2 - (lt_{\text{fast}} - t_{0n})^2 = -2dt_{\text{fast}}t_{0n} + (2ldt_{\text{fast}}^2 + d^2t_{\text{fast}}^2). \tag{A.4}$$

When Eq. (A.4) = 0, t_{0n} becomes

$$t_{0n} = t_{\text{fast}}(l + \frac{d}{2}). \tag{A.5}$$

Here, l is 0 because of the definition $0 < t_{0n} \leq t_{\text{fast}}$, and therefore, $t_{0n} = 0.5dt_{\text{fast}}$. Under the condition of $t_{0n} < 0.5dt_{\text{fast}}$,

$$(lt_{\text{fast}} + dt_{\text{fast}} - t_{0n})^2 > (lt_{\text{fast}} - t_{0n})^2, \tag{A.6}$$

$$\int_{lt_{\text{fast}}}^{lt_{\text{fast}}+dt_{\text{fast}}} \frac{t - t_{0n}}{\sigma^2} \exp\left(-\frac{(t - t_{0n})^2}{2\sigma^2} \right) dt > 0, \tag{A.7}$$

$$\frac{\partial}{\partial t_{0n}} p_n > 0. \tag{A.8}$$

On the other hand, under the condition of $t_{0n} > 0.5dt_{\text{fast}}$,

$$(lt_{\text{fast}} + dt_{\text{fast}} - t_{0n})^2 < (lt_{\text{fast}} - t_{0n})^2, \tag{A.9}$$

$$\int_{lt_{\text{fast}}}^{lt_{\text{fast}}+dt_{\text{fast}}} \frac{t - t_{0n}}{\sigma^2} \exp\left(-\frac{(t - t_{0n})^2}{2\sigma^2} \right) dt < 0, \tag{A.10}$$

$$\frac{\partial}{\partial t_{0n}} p_n < 0. \tag{A.11}$$

Thus, p_n attains the maximum value with $t_{0n} = 0.5dt_{\text{fast}}$. When t_{0n} is $0.5dt_{\text{fast}}$ for every n, p_n becomes a constant p' irrelevant to n, and obviously $0.5 \leq p' < 1$.

On the other hand, entropy for a bit stream H is defined as

$$H = -p \log p - (1 - p) \log(1 - p), \tag{A.12}$$

where p is a probability of '1' occurrence across the bit stream. Assuming the condition of $0.5 \leq p < 1$, H becomes the minimum in case that p is the maximum [124]. From these discussion, it is proved that the randomness gets the worst when t_{0n} is always $0.5dt_{\text{fast}}$, namely, the representative phases of cycles are fixed to the middle point of HIGH period.

Appendix B

Calculation of output jitter

In Sec. 3.2.2, the gain of the proposed jitter amplifier is discussed. This appendix will detail the calculation of the amount of the output jitter. Note that the definition of the notations are found in Sec. 3.2.2.

In order to obtain the output jitter, $t_{out(n+2)} - t_{out(n+1)}$ is calculated;

$$\begin{aligned} t_{out(n+1)} &= t_{crise(n+1)} + \left(l - \frac{t_{crise(n+1)} - t_{in(n+1)}}{\Delta D_{bf(n)}}\Delta l\right)\frac{\Delta D_{bs(n)}}{\Delta l} \\ &= D_{crise(n)} + \left(l - \frac{D_{crise(n)} - D_{in(n)}}{\Delta D_{bf(n)}}\Delta l\right)\frac{\Delta D_{bs(n)}}{\Delta l}, \end{aligned} \tag{B.1}$$

$$\begin{aligned} t_{out(n+2)} &= t_{crise(n+2)} + \left(l - \frac{t_{crise(n+2)} - t_{in(n+2)}}{\Delta D_{bf(n+1)}}\Delta l\right)\frac{\Delta D_{bs(n+1)}}{\Delta l} \\ &= D_{in(n)} + D_{crise(n+1)} + \left(l - \frac{D_{crise(n+1)} - D_{in(n+1)}}{\Delta D_{bf(n+1)}}\Delta l\right)\frac{\Delta D_{bs(n+1)}}{\Delta l}, \end{aligned} \tag{B.2}$$

$$\begin{aligned} t_{out(n+2)} - t_{out(n+1)} &= (D_{in(n)} - D_{crise(n)} + D_{crise(n+1)}) - (D_{in(n)} - D_{crise(n)})\frac{\Delta D_{bs(n)}}{\Delta D_{bf(n)}} \\ &+ (D_{in(n+1)} - D_{crise(n+1)})\frac{\Delta D_{bs(n+1)}}{\Delta D_{bf(n+1)}} - \frac{l}{\Delta l}(\Delta D_{bs(n)} - \Delta D_{bs(n+1)}). \end{aligned} \tag{B.3}$$

From Eq (3.2), $1/\Delta D_{bf(n)}$ is approximated as follows;

$$\begin{aligned} \frac{1}{\Delta D_{bf(n)}} &\approx \frac{1}{\mu_{bf}^2}\left(\mu_{bf} - D_{bfr(n)}\right) \\ &= \frac{\Delta D'_{bf(n)}}{\mu_{bf}^2}, \end{aligned} \tag{B.4}$$

where $\Delta D'_{bf(n)} = \mu_{bf} - D_{bfr(n)}$ follows normal distribution whose mean and variance are μ_{bf} and σ_{bf}^2. Here, $\Delta D'_{bf(n_1)}$ is independent of $D_{in(n_2)}$, $D_{crise(n_3)}$, $\Delta D_{bf(n_4)}$ and $\Delta D_{bs(n_5)}$ for arbitrary

n_1, n_2, n_3, n_4 and n_5 ($n_1, n_2, n_3, n_4, n_5 \in \mathbb{N}$). From Eq. (B.4), Eq. (B.3) is approximated as;

$$\begin{aligned} t_{out(n+2)} - t_{out(n+1)} \approx & \left(D_{in(n)} - D_{crise(n)} + D_{crise(n+1)}\right) \\ & - \Delta D_{bs(n)} \left\{ \frac{1}{\mu_{bf}^2} (D_{in(n)} - D_{crise(n)}) \Delta D'_{bf(n)} + \frac{l}{\Delta l} \right\} \\ & + \Delta D_{bs(n+1)} \left\{ \frac{1}{\mu_{bf}^2} (D_{in(n+1)} - D_{crise(n+1)}) \Delta D'_{bf(n+1)} + \frac{l}{\Delta l} \right\}, \end{aligned} \tag{B.5}$$

where the first, second and third terms are called hereafter as (A), (B), and (C).

(A) and $(B + C)$ are not independent of each other since they share the same random variables. Therefore, $\mathrm{Var}[A + B + C] = \mathrm{Var}[A] + \mathrm{Var}[B + C] + 2\mathrm{Cov}[A, B + C]$. The mean and the variance of (A) are;

$$\mathrm{E}[A] = \mu_{in} - \mu_{crise} + \mu_{crise} = \mu_{in}, \tag{B.6}$$

$$\mathrm{Var}[A] = \sigma_{in}^2 + \sigma_{crise}^2 + \sigma_{crise}^2 = \sigma_{in}^2 + 2\sigma_{crise}^2. \tag{B.7}$$

The mean of $(B + C)$ is;

$$\begin{aligned} \mathrm{E}[B + C] &= -\mu_{bs} \left\{ \frac{1}{\mu_{bf}^2} (\mu_{in} - \mu_{crise}) \mu_{bf} + \frac{l}{\Delta l} \right\} + \mu_{bs} \left\{ \frac{1}{\mu_{bf}^2} (\mu_{in} - \mu_{crise}) \mu_{bf} + \frac{l}{\Delta l} \right\} \\ &= 0. \end{aligned} \tag{B.8}$$

(B) is independent of (C) since they do not share the same random variables, and therefore, $\mathrm{Var}[B + C] = \mathrm{Var}[B] + \mathrm{Var}[C]$. From $\mathrm{Var}[XY] = \mathrm{Var}[X]\mathrm{Var}[Y] + \mathrm{E}[X]^2\mathrm{Var}[Y] + \mathrm{E}[Y]^2\mathrm{Var}[X]$ where X and Y are independent of each other, the variance of (B) is calculated as follows;

$$\begin{aligned} \mathrm{Var}[B] &= \mathrm{Var}\left[\Delta D_{bs(n)} \left\{ \frac{1}{\mu_{bf}^2} (D_{in(n)} - D_{crise(n)}) \Delta D'_{bf(n)} + \frac{l}{\Delta l} \right\} \right] \\ &= \mathrm{Var}[\Delta D_{bs(n)}] \mathrm{Var}\left[\frac{1}{\mu_{bf}^2} (D_{in(n)} - D_{crise(n)}) \Delta D'_{bf(n)} + \frac{l}{\Delta l} \right] \\ &+ \mathrm{E}[\Delta D_{bs(n)}]^2 \mathrm{Var}\left[\frac{1}{\mu_{bf}^2} (D_{in(n)} - D_{crise(n)}) \Delta D'_{bf(n)} + \frac{l}{\Delta l} \right] \\ &+ \mathrm{E}\left[\frac{1}{\mu_{bf}^2} (D_{in(n)} - D_{crise(n)}) \Delta D'_{bf(n)} + \frac{l}{\Delta l} \right]^2 \mathrm{Var}[\Delta D_{bs(n)}]. \end{aligned} \tag{B.9}$$

Here,

$$\mathrm{E}\left[\frac{1}{\mu_{bf}^2}(D_{in(n)}-D_{crise(n)})\Delta D'_{bf(n)}+\frac{l}{\Delta l}\right]=\frac{\mu_{in}-\mu_{crise}}{\mu_{bf}}+\frac{l}{\Delta l}, \tag{B.10}$$

$$\begin{aligned}&\mathrm{Var}\left[\frac{1}{\mu_{bf}^2}(D_{in(n)}-D_{crise(n)})\Delta D'_{bf(n)}+\frac{l}{\Delta l}\right]\\ &=\frac{1}{\mu_{bf}^4}\mathrm{Var}[(D_{in(n)}-D_{crise(n)})\Delta D'_{bf(n)}]\\ &=\frac{1}{\mu_{bf}^4}\left\{(\sigma_{in}^2+\sigma_{crise}^2)\sigma_{bf}^2+(\mu_{in}-\mu_{crise})^2\sigma_{bf}^2+\mu_{bf}^2(\sigma_{in}^2+\sigma_{crise}^2)\right\}.\end{aligned} \tag{B.11}$$

From Eqs. (B.10)(B.11), Eq. (B.9) is calculated as;

$$\begin{aligned}\mathrm{Var}[B]=&\frac{\mu_{bs}^2+\sigma_{bs}^2}{\mu_{bf}^4}\left\{(\sigma_{in}^2+\sigma_{crise}^2)\sigma_{bf}^2+(\mu_{in}-\mu_{crise})^2\sigma_{bf}^2\right\}\\ &+\frac{(\mu_{bs}^2+\sigma_{bs}^2)(\sigma_{in}^2+\sigma_{crise}^2)}{\mu_{bf}^2}+\sigma_{bs}^2\left\{\frac{\mu_{in}-\mu_{crise}}{\mu_{bf}}+\frac{l}{\Delta l}\right\}.\end{aligned} \tag{B.12}$$

Var[C] is also calculated in the same manner;

$$\begin{aligned}\mathrm{Var}[C]=&\frac{\mu_{bs}^2+\sigma_{bs}^2}{\mu_{bf}^4}\left\{(\sigma_{in}^2+\sigma_{crise}^2)\sigma_{bf}^2+(\mu_{in}-\mu_{crise})^2\sigma_{bf}^2\right\}\\ &+\frac{(\mu_{bs}^2+\sigma_{bs}^2)(\sigma_{in}^2+\sigma_{crise}^2)}{\mu_{bf}^2}+\sigma_{bs}^2\left(\frac{\mu_{in}-\mu_{crise}}{\mu_{bf}}+\frac{l}{\Delta l}\right).\end{aligned} \tag{B.13}$$

Therefore, Var[$B+C$] becomes;

$$\begin{aligned}\mathrm{Var}[B+C]=&\frac{2(\mu_{bs}^2+\sigma_{bs}^2)}{\mu_{bf}^4}\left\{(\sigma_{in}^2+\sigma_{crise}^2)\sigma_{bf}^2+(\mu_{in}-\mu_{crise})^2\sigma_{bf}^2\right\}\\ &+\frac{2(\mu_{bs}^2+\sigma_{bs}^2)(\sigma_{in}^2+\sigma_{crise}^2)}{\mu_{bf}^2}+2\sigma_{bs}^2\left\{\frac{\mu_{in}-\mu_{crise}}{\mu_{bf}}+\frac{l}{\Delta l}\right\}.\end{aligned} \tag{B.14}$$

In order to obtain the variance of $(A)+(B+C)$, the covariance of (A) and $(B+C)$ is calculated.

$$\begin{aligned}\mathrm{Cov}[A,B+C]&=\mathrm{E}[A(B+C)]-\mathrm{E}[A]\mathrm{E}[B+C]\\ &=\mathrm{E}[A(B+C)].\end{aligned} \tag{B.15}$$

$(A) + (B + C)$ is calculated as follows;

$$\begin{aligned}
A(B+C) = &-\frac{\Delta D_{bs(n)}\Delta D'_{bf(n)}}{\mu_{bf}^2}(D_{in(n)}-D_{crise(n)})^2 - \frac{\Delta D_{bs(n)}\Delta D'_{bf(n)}}{\mu_{bf}^2}(D_{in(n)}-D_{crise(n)})D_{crise(n+1)} \\
&+ \frac{\Delta D_{bs(n+1)}\Delta D'_{bf(n+1)}}{\mu_{bf}^2}D_{in(n+1)}(D_{in(n)}-D_{crise(n)}+D_{crise(n+1)}) \\
&- \frac{\Delta D_{bs(n+1)}\Delta D'_{bf(n+1)}}{\mu_{bf}^2}D_{crise(n+1)}(D_{in(n)}-D_{crise(n)}) \\
&- \frac{\Delta D_{bs(n+1)}\Delta D'_{bf(n+1)}}{\mu_{bf}^2}D_{crise(n+1)}^2 \\
&- \frac{l}{\Delta l}\Delta D_{bs(n)}(D_{in(n)}-D_{crise(n)}+D_{crise(n+1)}) \\
&+ \frac{l}{\Delta l}\Delta D_{bs(n+1)}(D_{in(n)}-D_{crise(n)}+D_{crise(n+1)}).
\end{aligned} \tag{B.16}$$

Here, because $(D_{in(n)} - D_{crise(n)})$ and $D_{crise(n+1)}$ are normally distributed, they can be changed into standard normal distribution by standardization. Also, the square of the standard normal random variable has chi-squared distribution whose degrees of freedom is one. Therefore, the means of $(D_{in(n)} - D_{crise(n)})^2$ and $D_{crise(n+1)}^2$ can be obtained from the means of chi-squared distribution. Because of $(D_{in(n)} - D_{crise(n)}) \sim N(\mu_{in} - \mu_{crise}, \sigma_{in}^2 + \sigma_{crise}^2)$ and $D_{crise(n+1)} \sim N(\mu_{crise}, \sigma_{crise}^2)$, the means of their squares are;

$$\mathrm{E}[(D_{in(n)} - D_{crise(n)})^2] = \sigma_{in}^2 + \sigma_{crise}^2 + (\mu_{in} - \mu_{crise})^2, \tag{B.17}$$

$$\mathrm{E}[D_{crise(n+1)}^2] = \sigma_{crise}^2 + \mu_{crise}^2. \tag{B.18}$$

Then, $\mathrm{Cov}[A, B + C]$ is calculated;

$$\mathrm{Cov}[A, B + C] = -\frac{\mu_{bs}}{\mu_{bf}}(\sigma_{in}^2 + 2\sigma_{crise}^2). \tag{B.19}$$

From Eqs. (B.7)(B.14)(B.19), $\mathrm{Var}[A + B + C]$, is calculated as;

$$\begin{aligned}
\mathrm{Var}[A+B+C] = &\left(1 - 2\frac{\mu_{bs}}{\mu_{bf}}\right)(\sigma_{in}^2 + 2\sigma_{crise}^2) \\
&+ \frac{2(\mu_{bs}^2 + \sigma_{bs}^2)}{\mu_{bf}^4}(\sigma_{in}^2 + \sigma_{crise}^2)\sigma_{bf}^2 + \frac{2(\mu_{bs}^2 + \sigma_{bs}^2)}{\mu_{bf}^4}(\mu_{in} - \mu_{crise})^2\sigma_{bf}^2 \\
&+ \frac{2(\mu_{bs}^2 + \sigma_{bs}^2)}{\mu_{bf}^4}\mu_{bf}^2(\sigma_{in}^2 + \sigma_{crise}^2) + 2\sigma_{bs}^2\left(\frac{\mu_{in} - \mu_{crise}}{\mu_{bf}} + \frac{l}{\Delta l}\right).
\end{aligned} \tag{B.20}$$

σ_{bf}^k/μ_{bf}^k and σ_{bs}^k/μ_{bf}^k $(k \geq 2)$ are approximated as zeros since σ_{bf} and σ_{bs} are much smaller than μ_{bf}. With $a = \sigma_{crise}^2/\sigma_{in}^2$ and $x = \mu_{bs}/\mu_{bf}$, the output jitter $\mathrm{Var}[A+B+C]$ is approximated;

$$\begin{aligned}
\mathrm{Var}[A+B+C] \approx &\left\{2(a+1)x^2 - 2(2a+1)x + (2a+1)\right\}\sigma_{in}^2 \\
&+ 2\sigma_{bs}^2\left(\frac{l}{\Delta l} - \frac{\mu_{crise} - \mu_{in}}{\mu_{bf}}\right).
\end{aligned} \tag{B.21}$$

Appendix C

Derivation of constraints

In Sec. 3.2.3, the constraints on LC buffer is discussed. Here, this appendix will present the process in which Eqs. (3.6)(3.7) are derived.

The sufficient condition for the jitter amplifier to work properly can be expressed as two conditions;

$$t_{in(n+1)} < t_{crise(n+1)} < t_{out(n+1)}, \tag{C.1}$$

$$t_{out(n+1)} < t_{cfall(n+1)} < t_{in(n+2)}. \tag{C.2}$$

The constraint on $t_{crise(n+1)}$, Eq. (C.1), is rewritten as follows;

$$0 < t_{crise(n+1)} - t_{in(n+1)} < t_{out(n+1)} - t_{in(n+1)}. \tag{C.3}$$

The right side of Eq. (C.3) is the delay in the LC buffer, and then it is not less than the latency in the LC buffer in fast mode. Then, the sufficient condition of $t_{crise(n+1)}$ is;

$$\begin{aligned} 0 &< t_{crise(n+1)} - t_{in(n+1)} < \frac{l}{\Delta l}\Delta D_{bf(n+1)}, \\ 0 &< D_{rise(n)} - D_{in(n)} < \frac{l}{\Delta l}\Delta D_{bf(n+1)}. \end{aligned} \tag{C.4}$$

Introducing the coefficient m, Eq. (C.4) is expressed as the following two conditions;

$$0 < (\mu_{rise} - \mu_{in}) - m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2}, \tag{C.5}$$

$$(\mu_{rise} - \mu_{in}) + m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} < \frac{l}{\Delta l}\mu_{bf} - m\sigma_{bf}\sqrt{\frac{l}{\Delta l}}. \tag{C.6}$$

They can be rewritten as conditions of μ_{rise}.

$$\mu_{in} + m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} < \mu_{rise}, \tag{C.7}$$

$$\mu_{rise} < \mu_{in} - m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} + \frac{l}{\Delta l}\mu_{bf} - m\sigma_{bf}\sqrt{\frac{l}{\Delta l}}. \tag{C.8}$$

Here, μ_{rise} is expressed as $\mu_{rise} = \mu_{rise\_offset} + s\Delta\mu_{rise}(s \in \mathbb{Z}, s \geq 0)$. Also, μ_{rise_offset} assumed

to be less than the right side of Eq. (C.8). Then, the sufficient condition is;

$$\Delta\mu_{rise} < \mu_{in} - m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} + \frac{l}{\Delta l}\mu_{bf} - m\sigma_{bf}\sqrt{\frac{l}{\Delta l}} - \left(\mu_{in} + m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2}\right),$$
$$\Delta\mu_{rise} + 2m\sqrt{\sigma_{rise}^2 + \sigma_{in}^2} < \frac{l}{\Delta l}\mu_{bf} - m\sigma_{bf}\sqrt{\frac{l}{\Delta l}}. \tag{C.9}$$

Next, the condition of $t_{cfall(n+1)}$, Eq. (C.2), is;

$$t_{out(n+1)} - t_{in(n+1)} < t_{cfall(n+1)} - t_{in(n+1)} < t_{in(n+2)} - t_{in(n+1)}. \tag{C.10}$$

The left side of Eq. (C.10) is the delay in the LC buffer, and then it is not more than the latency in the LC buffer in slow mode. Then, the sufficient condition of $t_{cfall(n+1)}$ is;

$$\frac{l}{\Delta l}\Delta D_{bs(n)} < t_{cfall(n+1)} - t_{in(n+1)} < t_{in(n+2)} - t_{in(n+1)},$$
$$\frac{l}{\Delta l}\Delta D_{bs(n)} < D_{fall(n)} - D_{in(n)} < D_{in(n+1)}. \tag{C.11}$$

Eq. (C.10) is expressed as the two equations with the coefficient m;

$$\frac{l}{\Delta l}\mu_{bs} + m\sigma_{bs}\sqrt{\frac{l}{\Delta l}} < (\mu_{fall} - \mu_{in}) - m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2}, \tag{C.12}$$
$$(\mu_{fall} - \mu_{in}) + m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} < \mu_{in} - m\sigma_{in}. \tag{C.13}$$

They can be rewritten as conditions of μ_{fall}.

$$\frac{l}{\Delta l}\mu_{bs} + m\sigma_{bs}\sqrt{\frac{l}{\Delta l}} + \mu_{in} + m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} < \mu_{fall}, \tag{C.14}$$
$$\mu_{fall} < \mu_{in} - m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} + (\mu_{in} - m\sigma_{in}). \tag{C.15}$$

Here, μ_{fall} is expressed as $\mu_{fall} = \mu_{fall\_offset} + t\Delta\mu_{fall}$ ($t \in \mathbb{Z}, t \geq 0$). In addition, μ_{fall_offset} is assumed to be less than the right side of Eq. (C.15). Then, the sufficient condition is;

$$\Delta\mu_{fall} < \mu_{in} - m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} + (\mu_{in} - m\sigma_{in}) - \left(\frac{l}{\Delta l}\mu_{bs} + m\sigma_{bs}\sqrt{\frac{l}{\Delta l}}\right) - \mu_{in} - m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2},$$
$$\Delta\mu_{fall} + 2m\sqrt{\sigma_{fall}^2 + \sigma_{in}^2} < (\mu_{in} - m\sigma_{in}) - \left(\frac{l}{\Delta l}\mu_{bs} + m\sigma_{bs}\sqrt{\frac{l}{\Delta l}}\right). \tag{C.16}$$

Bibliography

[1] D. Eastlake, J. Schiller, and S. Crocker, "Randomness requirements for security," RFC4086, 2005. [Online]. Available: http://tools.ietf.org/pdf/rfc4086.pdf

[2] H. Tanaka, "Physical random number generation technology supporting security network [I]: physical random number generator and information security," *The journal of the institute of electronics, information, and communication engineers*, vol. 94, no. 10, pp. 915–919, Oct. 2011 (in Japanease).

[3] T. Dierks and C. Allen, "The TLS Protocol," RFC2246, Jan. 1999. [Online]. Available: http://tools.ietf.org/html/rfc2246/

[4] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on information theory*, vol. 22, no. 6, pp. 644–654, 1976.

[5] W. Simpson, "PPP Challenge Handshake Authentication Protocol (CHAP)," RFC1334, Aug. 1996. [Online]. Available: http://tools.ietf.org/html/rfc1994/

[6] I. Goldberg and D. Wagner, "Randomness and the Netscape browser," *Dr. Dobb's Journal*, pp. 66–70, Jan. 1996. [Online]. Available: http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html

[7] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 317–331, 2009.

[8] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM transaction on modeling and computer simulation*, vol. 8, no. 1, pp. 3–30, Jan. 1998.

[9] E. Trichina, M. Bucci, D. D. Seta, R. Luzzi, "Supplemental cryptographic hardware for smart cards," *IEEE Micro*, vol. 21, no. 6, pp. 26–35, 2001.

[10] G. Taylor and G. Cox, "Digital randomness," *IEEE spectrum*, vol. 48, No, 9, pp. 32–58, Sep. 2011.

[11] "Intel digital random number generator(DRNG) software implementation guide," Aug. 2012. [Online]. Available: http://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide/

[12] Y. Hu, X. Liao, K. Wong and Q. Zhou, "A true random number generator based on mouse movement and chaotic cryptography," *Chaos, Solitons & Fractals*, 2007.

[13] M. Rohe, "A true-random generator based on radioactive decay," security and cryptography research group, saarland university, 2003.

[14] Y. Yoshizawa, H. Kimura, H. Inoue, K. Fujita, M. Toyama, and O. Miyatake, "Physical random numbers generated by radioactivity," *Journal of the japanese society of computational statistics*, vol. 12, no. 1, pp. 67–81, 1999.

[15] A. Uchida, "Physical random number generation technology supporting security network [III]: review on fast physical random number generators with chaotic lasers," *The journal of the institute of electronics, information, and communication engineers*, vol.

95, no. 1, pp. 74–80, Jan. 2012 (in Japanease).

[16] A. Uchida, K. Amano, M. Inoue, K. Hirano, S. Naito, H. Someya, I. Oowada, T. Kurashige, M. Shiki, S. Yoshimori, K. Yoshimura, and P. Davis, "Fast physical random bit generation with chaotic semiconductor lasers," *Nature Photonics*, vol. 2, no. 12, pp. 728–732, 2008.

[17] I. Reidler, Y. Aviad, M. Rosenbluh, and I. Kanter, "Fast random bit generation with bandwidth-enhanced chaos in semiconductor lasers," *Optics express*, vol. 18, no. 6, pp. 5512–5524, 2010.

[18] I. Kanter, Y. Aviad, I. Reidler, E. Cohen, and M. Rosenbluh, "An optical ultrafast random bit generator," *Nature Photonics*, vol. 4, no. 1, pp. 58–61, 2009.

[19] T. Harayama, S. Sunada, K. Yoshimura, P. Davis, K. Tsuzuki, and A. Uchida, "Fast nondeterministic random-bit generation using on-chip chaos lasers," *Physical Review A*, vol. 83, no. 3, p. 031803, 2011.

[20] A. Argyris, S. Deligiannidis, E. Pikasis, A. Bogris, and D. Syvridis, "Implementation of 140 Gb/s true random bit generator based on a chaotic photonic integrated circuit," *Optics express*, vol. 18, no. 18, pp. 18763–18768, 2010.

[21] K, Hirano, K. Amano, A. Uchida, S. Naito, M. Inoue, S. Yoshimori, K. Yoshimura, and P. Davis, "Characteristics of fast physical random bit generation using chaotic semiconductor lasers," *IEEE journal of quantum electronics*, vol. 45, no. 11, pp. 1367–1379, 2009.

[22] T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger, "A fast and compact quantum random number generator," *Review of scientific instruments*, vol. 71, no. 4, pp. 1675–1680, 2000.

[23] J. F. Dynes, Z. L. Yuan, A. W. Sharpe, and A. J. Shields, "A high speed, postprocessing free, quantum random number generator," *Applied physics letters*, vol. 93, no. 3, p. 031109, 2008.

[24] C. Gabriel, C. Wittmann, D. Sych, R. Dong, W. Mauerer, U. L. Andersen, C. Marquardt, and G. Leuchs, "A generator for unique quantum random numbers based on vacuum states," *Nature photonics*, vol. 4, no. 10, pp. 711–715, 2010.

[25] Y. Yamanashi, "Physical random number generation technology supporting security network [V]: development of superconductive physical random number generator," *The journal of the institute of electronics, information, and communication engineers*, vol. 95, no. 4, pp. 352–356, Apr. 2012 (in Japanease).

[26] Y. Yamanashi and N. Yoshikawa, "Superconductive random number generator using thermal noises in SFQ circuits," *IEEE transactions on applied superconductivity*, vol. 19, no. 3, pp. 630–633, 2009.

[27] S. Ergün and S. Özoğuz, "Truly random number generators based on a non-autonomous chaotic oscillator," *AEU international journal of electronics and communications*, vol. 61, no. 4, pp. 235–242, 2007.

[28] S. Ergün, "Modeling and analysis of chaos-modulated dual oscillator-based random number generators," in *Proc. European signal processing conference*, pp. 1-5, Aug. 2008.

[29] M. Dichtl and N. Janssen, "A high quality physical random number generator," in *Proc. Sophia antipolis microelectronics forum*, pp. 48–53, 2000.

[30] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE transactions on computers*,

vol. 56, no. 1, pp. 109–119, 2007.

[31] K. Wold, C. H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," *International journal of reconfigurable computing*, vol. 2009, pp. 4, 2009.

[32] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, "Physical unclonable function and true random number generator: a compact and scalable implementation," in *Proc. ACM great lakes symposium on VLSI*, pp. 425–428, 2009.

[33] V. Fischer and M. Drutarovskỳ, "True random number generator embedded in reconfigurable hardware," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 103–125, 2003.

[34] H. Bock, M. Bucci, and R. Luzzi, "An offset-compensated oscillator-based random bit source for security applications," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 268–281, 2004.

[35] M. Bucci and R. Luzzi, "Design of testable random bit generators," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 147–156, 2005.

[36] B. Jun and P. Kocher, "The Intel random number generator," cryptography research inc., white paper prepared for Intel corporation, Apr. 1999.

[37] G. K. Balachandran and R. E. Barnett, "A 440-nA true random number generator for passive RFID tags," *IEEE transactions on circuits and systems*, vol. 55, no. 11, Dec. 2008.

[38] M. Bucci and R. Luzzi, "Fully digital random bit generators for cryptographic applications," *IEEE transactions on circuits and systems I: regular papers*, vol. 55, no. 3, pp. 861–875, Apr. 2008.

[39] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo, "A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC," *IEEE transactions on computers*, vol. 52, no. 4, pp. 403–409, Apr. 2003.

[40] K. H. Tsoi, K. H. Leung, and P. H. W. Leong, "High performance physical random number generator," *IET computers & digital techniques*, vol. 1, no. 4, pp. 349–352, July 2007.

[41] W. Killmann and W. Schindler, "A design for a physical rng with robust entropy estimators," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 146–163, 2008.

[42] M. Bucci, L. Germani, R. Luzzi, P. Tommasino, A. Trifiletti, and M. Varanonuovo, "A high-speed IC random-number source for smartcard microcontrollers," *IEEE transactions on circuits and systems I: fundamental theory and applications*, vol. 50, no. 11, pp. 1373–1380, 2003.

[43] W. T. Holman, J. A. Connelly, and A. B. Dowlatabadi, "An Integrated analog/digital random noise source," *IEEE transactions on circuits and systems I: fundamental theory and applications*, vol. 44, no. 6, pp. 521–528, 1997.

[44] C. S. Petrie and J. A. Connelly, "Modeling and simulation of oscillator-based random number generators," in *Proc. IEEE international symposium on circuits and systems*, vol. 4, pp. 324–327, May 1996.

[45] V. Bagini and M. Bucci, "A design of reliable true random number generator for cryptographic applications," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 204–218, 1999.

[46] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and design of analog*

integrated circuits, 4th Edition, John Wiley & Sons, 2001.

[47] M. Matsumoto, S. Yasuda, R. Ohba, K. Ikegami, T. Tanamoto, and S. Fujita, "1200μm^2 physical random-number generators based on SiN mosfet for secure smart-card application," in *IEEE international solid-state circuits conference, digest of technical papers*, pp. 414-624, 2008.

[48] F. Pareschi, G. Scotti, L. Giancane, R. Rovatti, G. Setti, and A. Trifiletti, "Power analysis of a chaos-based random number generator for cryptographic security," in *Proc. IEEE international symposium on circuits and systems*, pp. 2858–2861, 2009.

[49] 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 transactions on circuits and systems I: regular papers*, vol. 53, no. 2, pp. 326–337, 2006.

[50] S. Callegari, R. Rovatti, and G. Setti, "Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos," *IEEE transactions on signal processing*, vol. 53, no. 2, pp. 793–805, 2005.

[51] V. V. Kaenel and T. Takayanagi, "Dual true random number generators for cryptographic applications embedded on a 200 million device dual cpu soc," in *Proc. IEEE custom integrated circuits conference*, pp. 269–272, 2007.

[52] F. Pareschi, G. Setti, and R. Rovatti, "A fast chaos-based true random number generator for cryptographic applications," in *Proc. IEEE european solid-state circuits conference*, pp. 130–133, 2006.

[53] M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle, "True random bit generation from a double-scroll attractor," in *IEEE transactions on circuits and systems I: regular papers*, vol. 51, no. 7, pp. 1395–1404, 2004.

[54] D. Maher and R. Rance, "Random number generators founded on signal and information theory," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 219–230, 1999.

[55] T. Stojanovski and L. Kocarev, "Chaos-based random number generators-part I: analysis," *IEEE transactions on circuits and systems I: fundamental theory and applications*, vol. 48, no. 3, pp. 281–288, 2001.

[56] T. Stojanovski and L. Kocarev, "Chaos-based random number generators-part II: practical realization," *IEEE transactions on circuits and systems I: fundamental theory and applications*, vol. 48, no. 3, pp. 382–385, 2001.

[57] F. Pareschi, G. Setti, and R. Rovatti, "Implementation and testing of high-speed CMOS true random number generators based on chaotic systems," *IEEE transactions on circuits and systems I: regular papers*, vol. 57, no. 12, pp. 3124–3137, Dec. 2010.

[58] C. S. Petrie and J. A. Connelly, "A noise-based IC random number generator for applications in cryptography," *IEEE transactions on circuits and systems I: regular papers*, vol. 47, no. 5, pp. 615–621, May 2000.

[59] M. J. Bellido, A. J. Acosta, M. Valencia, A. Barriga, and J. L. Huertas, "A simple binary random number generator: new approaches for cmos vlsi," *Proc. IEEE international midwest symposium on circuits and systems*, pp. 127–129, 1992.

[60] B. Fechner and A. Osterloh, "A true random number generator with built-in attack detection," in *Proc. international conference on dependability of computer systems*, pp. 111–118, 2008.

[61] J. Holleman, S. Bridges, B. P. Otis, and C. Diorio, "A 3 μW cmos true random number generator with adaptive floating-gate offset cancellation," *IEEE journal of solid-state*

circuits, vol. 43, no. 5, pp. 1324–1336, 2008.

[62] J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio, "A 2.92 μw hardware random number generator," in *Proc. IEEE european solid-state circuits conference*, pp. 134–137, 2006.

[63] D. J. Kinniment and E. G. Chester, "Design of an on–chip random number generator using metastability," in *Proc. IEEE european solid-state circuits conference*, pp. 595–598, 2002.

[64] S. Srinivasan, S. Mathew, V. Erraguntla, R. Krishnamurthy, "A 4Gbps 0.57 pj/bit process-voltage-temperature variation tolerant all-digital true random number generator in 45nm cmos," in *Proc. IEEE conference on VLSI design*, pp. 301–306, 2009.

[65] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," *IEEE journal of solid-state circuits*, vol. 43, no. 1, pp. 78–85, 2008.

[66] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," in *IEEE international solid-state circuits conference, digest of technical papers*, pp. 404–611, 2007.

[67] V. B. Suresh and W. P. Burleson, "Entropy extraction in metastability-based TRNG," in *Proc. IEEE international symposium on hardware-oriented security and trust*, pp. 135–140, 2010.

[68] M. Epstein, L. Hars, R. Krasinski, M. Rosner, and H. Zheng, "Design and implementation of a true random number generator based on digital circuit artifacts," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 152–165, 2003.

[69] I. Vasyltsov, E. Hambardzumyan, Y. S. Kim, and B. Karpinskyy, "Fast digital TRNG based on metastable ring oscillator," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 164–180, 2008.

[70] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA-based true random number generation using circuit metastability with adaptive feedback control," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 17–32, 2011.

[71] S. Srinivasan, S. Mathew, R. Ramanarayanan, F. Sheikh, M. Anders, H.Kaul, V. Erraguntla, R. Krishnamurthy, and G. Taylor, "2.4 GHz 7mW all-digital PVT-variation tolerant true random number generator in 45nm CMOS," in *Proc. IEEE symposium on VLSI Circuits*, pp. 203–204, June 2010.

[72] T. Tanamoto, N. Shimomura, M. Matsumoto, R. Ooba, S. Yasuda, S. Ikegawa, S. Fujita, and H. Yoda, "Physical random number generation technology supporting security network [IV]: physical random number generator using advanced semiconductor devices," *The journal of the institute of electronics, information, and communication engineers*, vol. 95, no. 2, pp. 137–141, Feb. 2012 (in Japanease).

[73] T. Tanamoto, N. Shimomura, S. Ikegawa, M. Matsumoto, S. Fujita, and H. Yoda, "High-speed magnetoresistive random-access memory random number generator using error-correcting code," *Japanese journal of applied physics*, vol. 50, no. 4, p. 04DM01, 2011.

[74] R. Brederlow, R. Prakash, C. Paulus, and R. Thewes, "A low-power true random number generator using random telegraph noise of single oxide-traps," In *IEEE international solid-state circuits conference, digest of technical papers*, pp. 1666–1675, 2006.

[75] P. Xu, Y. L. Wong, T. K. Horiuchi, and P. A. Abshire, "Compact floating-gate true random number generator," *IET electronics letters*, vol. 42, no. 23, pp. 1346–1347,

2006.

[76] P. Xu, T. Horiuchi, and P. Abshire, "Stochastic model and simulation of a random number generator circuit," in *Proc. IEEE international symposium on circuits and systems*, pp. 2977–2980, 2008.

[77] S. Fujita, K. Uchida, S. Yasuda, R. Ohba, H. Nozaki, and T. Tanamoto, "Si nanodevices for random number generating circuits for cryptographic security," in *IEEE international solid-state circuits conference, digest of technical papers*, pp. 294–295, 2004.

[78] S. Yasuda, H. Satake, T. Tanamoto, R. Ohba, K. Uchida, and S. Fujita, "Physical random number generator based on MOS structure after soft breakdown," *IEEE journal of solid-state circuits*, vol. 39, no. 8, pp. 1375–1377, 2004.

[79] S. Yasuda, K. Uchida, T. Tanamoto, R. Ohba, and S. Fujita, "Ultra-small physical random number generators based on Si nanodevices for security systems and comparison to other large physical random number generators," in *Proc. IEEE conference on nanotechnology*, vol. 2, pp. 531–534, 2003.

[80] N. Liu, N. Pinckney, S. Hanson, D. Sylvester, and D. Blaauw, "A true random number generator using time-dependent dielectric breakdown," in *Proc. symposium on VLSI circuits*, pp. 216–217, June 2011.

[81] M. Dichtl and J. Golić, "High-speed true random number generation with logic gates only," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 45–62, 2007.

[82] J. D. Golić, "New methods for digital generation and postprocessing of random data," *IEEE transactions on computers*, vol. 55, no. 10, pp. 1217–1229, 2006.

[83] M. Varchola and Y. Drutarovsk, "New FPGA based TRNG principle using transition effect with built-in malfunction detection," in *Proc. International workshop on cryptographic architectures embedded in reconfigurable devices*, pp. 150–155, 2009.

[84] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generators," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 351–365, 2010.

[85] T. Koshiba, "Physical random number generation technology supporting security network [II]: theoretical aspects of physical random bit generation," *The journal of the institute of electronics, information, and communication engineers*, vol. 94, no. 12, pp. 1072–1076, Dec. 2011 (in Japanease).

[86] "Security requirements for cryptographic modules," FIPS, pub. 140-2, May 2001.

[87] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, "A statistical test suite for the validation of random number generators and pseudorandom number generators for cryptographic applications," NIST, pub. 800-22rev1a, Apr. 2010. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf

[88] F. Pareschi, R. Rovatti, and G Setti, "Second-level NIST randomness tests for improving test reliability," in *Proc. IEEE international symposium on circuits and systems*, pp. 1437–1440, 2007.

[89] G. Marsaglia, Diehard battery of tests of randomness, 1995. [Online]. Available: http://stat.fsu.edu/pub/diehard/

[90] W. Killmann and W. Schindler, "A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators," ver. 3.1, 2001.

[91] W. Schindler and W. Killmann, "Evaluation criteria for true (physical) random number generators used in cryptographic applications," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 431–449, 2003.

[92] D. E. Knuth, *The art of computer programming: seminumerical algorithms*, vol. 2, Wesley, 1969.

[93] "Crypt-X," the information security research centre at queensland university of technology. [Online]. Available: http://www.isi.qut.edu.au/resources/cryptx/

[94] P. L'ecuyer and R. Simard, "TestU01: A C library for empirical testing of random number generators," *ACM transactions on mathematical software*, vol. 33, no. 4, 2007.

[95] K. Udawatta, M. Ehsanian, S. Maidanov, and S. Musunuri, "Test and validation of a non-deterministic system-True Random Number Generator," in *Proc. international high level design validation and test workshop*, pp. 77–84, 2008.

[96] J. V. Neumann, "Various techniques used in connection with random digits," National bureau of standards applied mathematics series, vol. 12, pp. 36–38, 1951.

[97] R. B. Davies, "Exclusive OR (XOR) and hardware random number generators," pp. 1–11, Feb. 2002. [Online]. Available: http://www.robertnz.net/pdf/xor2.pdf

[98] B. Barak, R. Impagliazzo, and A. Wigderson, "Extracting randomness using few independent sources," *SIAM Journal on computing*, vol. 36, no. 4, pp. 1095–1118, 2006.

[99] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson, "Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors," *Journal of the ACM*, vol. 57, no. 4, 2010.

[100] J. Kamp and D. Zuckerman, "Deterministic extractors for bit-fixing sources and exposure-resilient cryptography," *SIAM Journal on computing*, vol. 36, no. 5, pp. 1231-1247, 2007.

[101] A. Gabizon, R. Raz, and R. Shaltiel, "Deterministic extractors for bit-fixing sources by obtaining an independent seed," *SIAM Journal on computing*, vol. 36, no. 4, pp. 1072-1094, 2006.

[102] J. Kamp, A. Rao, S. Vadhan, and D. Zuckerman, "Deterministic extractors for small-space sources," *Journal of computer and system sciences*, vol. 77, no. 1, pp. 191-220, Jan. 2011.

[103] R. Shaltiel, "Recent developments in explicit constructions of extractors," in G. Plun, G. Rozenberg, and A. Salomaa, *Current trends in theoretical computer science: the challenge of the new century*, vol. 1, pp. 189–228, World Scientific, 2004.

[104] V. Guruswami, C. Umans, and S. Vadhan, "Unbalanced expanders and randomness extractors from parvaresh-vardy codes," *Journal of the ACM*, vol. 56, no. 4, 2009.

[105] B. Barak, R. Shaltiel, and E. Tromer, "True random number generators secure in a changing environment," in *Proc. Workshop on cryptographic hardware and embedded systems*, pp. 166–180, 2003.

[106] HSPICE and HSPICE RF Documentation, Synopsys online documentation, Z-2007.03, Synopsys.

[107] NanoSim documentation, Synopsys online documentation, B-2008.09, Synopsys.

[108] Verilog-A language reference manual, Open verilog international, 1996. [Online]. Available: http://www.verilog.org/verilog-ams/htmlpages/public-docs/lrm/VerilogA/verilog-a-lrm-1-0.pdf

[109] F. Bernard, V. Fischer, and B. Valtchanov, "Mathematical model of physical RNGs based on coherent sampling," *Tatra mountains mathematical publications*, vol. 45, pp.

1–14, 2010.

[110] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *Journal of cryptology*, pp. 1–28, Springer, 2010.

[111] T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "A design procedure for oscillator-based hardware random number generator with stochastic behavior modeling," in *Proc. International workshop on information security applications*, pp. 107–121, 2010.

[112] V. Fischer, F. Bernard, N. Bochard, and M. Varchola, "Enhancing security of ring oscillator-based TRNG implemented in FPGA," in *Proc. International conference on field programmable logic and applications*, pp. 245–250, 2008.

[113] W. Ledermann, *Handbook of applicable mathematics*, vol. 6, John Wiley & Sons, 1980.

[114] MATLAB, MathWorks. [Online]. Available: http://www.mathworks.com/products/matlab/

[115] DPO/DSA/MSO70000C and D Series Datasheet, Tektronix. [Online]. Available: http://www.tek.com/datasheet/digital-and-mixed-signal-oscilloscopes/

[116] D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," in *Proc. International conference on field programmable logic and applications*, pp. 1–6, 2006.

[117] Y. Ogasahara, M. Hashimoto, and T. Onoye, "All-digital ring-oscillator-based macro for sensing dynamic supply noise waveform," *IEEE journal of solid-state circuits*, vol. 44, no. 6, pp. 1745–1755, 2009.

[118] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE journal of solid-state circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[119] T. Amaki, M. Hashimoto, and T. Onoye, "An oscillator-based true random number generator with jitter amplifier," in *Proc. IEEE international symposium on circuits and systems*, pp. 725–728, May 2011.

[120] T. Amaki, M. Hashimoto, and T. Onoye, "Jitter amplifier for oscillator-based true random number generator," in *Proc. IEEE/ACM asia and south pacific design automation conference*, pp. 81–82, 2011.

[121] R. Z. Bhatti, M. Denneau, and J. Draper, "Duty cycle measurement and correction using a random sampling technique," in *Proc. IEEE midwest symposium on circuits and systems*, pp. 1043–1046, 2005.

[122] S. Henzler, *Time-to-digital converters*, Springer Publishing Company, 2010.

[123] P. Dudek, S. Szczepanski, and J. V. Hatfield, "A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line," *IEEE journal of solid-state circuits*, vol. 35, no. 2, pp. 240–247, 2000.

[124] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.