PAPER *Special Section on Reconfigurable Systems*

FPGA Implementation of Metastability-Based True Random Number Generator

Hisashi HATA[†*], ***Nonmember and*** **Shuichi ICHIKAWA**[†**a)], ***Senior Member***

SUMMARY True random number generators (TRNGs) are important as a basis for computer security. Though there are some TRNGs composed of analog circuit, the use of digital circuits is desired for the application of TRNGs to logic LSIs. Some of the digital TRNGs utilize jitter in free-running ring oscillators as a source of entropy, which consume large power. Another type of TRNG exploits the metastability of a latch to generate entropy. Although this kind of TRNG has been mostly implemented with full-custom LSI technology, this study presents an implementation based on common FPGA technology. Our TRNG is comprised of logic gates only, and can be integrated in any kind of logic LSI. The RS latch in our TRNG is implemented as a hard-macro to guarantee the quality of randomness by minimizing the signal skew and load imbalance of internal nodes. To improve the quality and throughput, the output of 64–256 latches are XOR'ed. The derived design was verified on a Xilinx Virtex-4 FPGA (XC4VFX20), and passed NIST statistical test suite without post-processing. Our TRNG with 256 latches occupies 580 slices, while achieving 12.5 Mbps throughput.

key words: *TRNG, synchronous digital circuit, FPGA, entropy*

1. Introduction

Many algorithms and systems depend on random numbers; particularly, true random numbers are indispensable for some applications including cryptographic applications. Although pseudorandom number generators (PRNGs) are widely adopted, they produce a sequence of numbers by a deterministic algorithm; therefore, a PRNG has to be seeded by a true random number to generate a unique and unpredictable result for each occasion. In actual fact, a true random number generator (TRNG) is a critical component in many practical applications.

A TRNG is a hardware device to generate true random numbers. It is typically based on a physical process, such as thermal noise or quantum phenomenon. Though many TRNGs are implemented with analog circuit technology, it is difficult or expensive to integrate it with digital circuits. It is thus desirable to compose a TRNG with digital circuits, which can be easily integrated into a logic LSI.

This study presents a design and implementation of a TRNG, which exploits the metastability of RS latch. This TRNG is fully composed of digital circuits, and thus can be integrated into any kind of logic LSIs. Though Field Programmable Gate Array (FPGA) technology was adopted in this study, semi- or full-custom LSI technologies are also eligible. FPGA implementation is worthwhile by itself, since FPGA devices are widely accepted.

The rest of this paper is organized as follows. Section 2 gives an overview of the preceding studies on TRNGs comprised of digital circuit. Particularly, the metastability-based TRNGs are described in Sect. 3. Then, Sect. 4 summarizes the tools and measures of evaluation. Section 5 details the design trade-offs and the implementation, and then Sect. 6 reports the quality of randomness, the logic scale, and the throughput of our TRNG. The conclusions are given in Sect. 7.

2. Related Studies

The TRNGs made of digital circuits are categorized by their sources of entropy.

Fairfield et al. [1] presented an oscillator-sampling TRNG, which generates a random bit stream by sampling a high-frequency oscillating signal with a low-frequency clock signal using a D-type flip-flop (DFF). Tsoi et al. [2] evaluated an oscillator-sampling TRNG with a Xilinx Virtex FPGA device, and reported that the TRNG successfully passed NIST test [3], [4] at 5–29 kbps of generation rate. Since an oscillator-sampling TRNG requires two independent clock signals, Tsoi et al. [2] connected external parts (two resistors and one capacitor) to the FPGA device. Meanwhile, this study presents a TRNG of fully synchronous digital circuit, which does not use any external parts.

No external parts are necessary if two clock signals are generated internally. Fischer et al. [5], [6] proposed a TRNG design that utilizes an on-chip PLL to generate the second clock signal from the system clock, where the entropy is derived from jitter of a PLL. This circuit passed NIST test with Altera APEX 20K FPGA and Stratix FPGA. Likewise, Kwok and Lam [7] proposed a TRNG, which utilizes a DCM (digital clock manager) of Xilinx Virtex II Pro FPGA to generate the second clock signal. It is obvious that an on-chip PLL (or its equivalent) is indispensable for this kind of TRNG; meanwhile, this work presents a TRNG that consists of fundamental logic gates only.

Another digital device to generate an independent clock signal is a ring oscillator (RO), which consists of an odd number of inverters. Sunar et al. [8] discussed a TRNG (Sunar-type TRNG) that generates a random bit stream by

Manuscript received April 18, 2011.
Manuscript revised September 2, 2011.

[†]The authors are with the Department of Knowledge-based Information Engineering, Toyohashi University of Technology, Toyohashi-shi, 441–8580 Japan.
[*]Presently, with Renesas Electronics Corporation.
[**]Presently, with Numazu National College of Technology.
a) E-mail: ichikawa@ieee.org

DOI: 10.1587/transinf.E95.D.426

XOR'ing the outputs of a number of free-running ROs. Schellekens et al. [9] evaluated a Sunar-type TRNG with a Xilinx Virtex II Pro FPGA, and reported that a TRNG with 110 ROs of three inverters generated a random bit stream of 2 Mbps. Other designs of TRNG that utilize free-running ROs were also presented by Kohlbrenner and Gaj [10], Golić [11], and Dichtl and Golić [12].

Wold and Tan [13] proposed to enhance a Sunar-type TRNG by adding an extra D flip-flop after each free-running RO. Their TRNG was implemented with 50 ROs, which passed the NIST and Diehard statistical tests at a throughput of 100 Mbps using 167 LEs of the Altera Cyclone II FPGA. The advantages and weak points of Wold-type TRNG have been investigated by recent studies (e.g., [14], [15]).

A serious problem of these types of TRNGs is the power dissipation of free-running ROs. For example, the implementation of Schellekens et al. [9] includes 110 ROs, each of which operates at approximately 330 MHz. The energy consumption of these ROs is far from negligible. Particularly for embedded systems, it is considered to be prohibitive.

This study presents a TRNG that derives entropy from the metastability of RS latches. Since the design of this TRNG is fully synchronous, it is possible to stop the clock of TRNG to suppress power consumption. The operation principles of this TRNG are detailed in Sect. 3.

3. TRNG Based on Metastability

3.1 Principles of Operation

Generally, a latch (or flip-flop) malfunctions if the setup and hold time constraints are violated. In this case, the latch enters a metastable state, where the output lingers or oscillates between logic levels [16]. It is possible to design a TRNG (metastability-based TRNG) that exploits this phenomenon for random number generation.

In case of an RS latch, it is generally prohibited to activate both of R and S inputs simultaneously; if this happens, an RS latch may become metastable and generate an indefinite output. Figure 1 illustrates the TRNG based on the metastability of RS latch. When $Clk = 0$, this latch is stable with $(Q, \bar{Q}) = (1, 1)$. Meanwhile, when $Clk = 1$, it becomes stable at either $(Q, \bar{Q}) = (1, 0)$ or $(0, 1)$. More precisely, the latch enters a metastable state at the rising edge of Clk, and eventually transitions to one of the stable states. This process corresponds to a coin toss, which generates 1 bit of

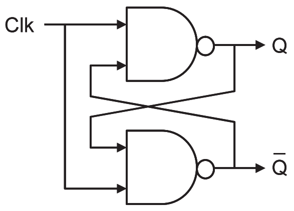


Fig. 1 Random number generation with an RS latch.

information.

The transition is accomplished by amplifying the deviation from equilibrium point (e.g., thermal noise), which results in randomness of output. The transition time naturally deviates with a probabilistic distribution. The probability $P(t > t_c)$ of the metastable behavior lasting longer than t_c is estimated by the following equation, where τ is a time constant of the RS latch [16].

$$P(t > t_c) = e^{-t_c/\tau} = e^{-(A-1)t_c/RC} \quad (1)$$

The time constant τ is estimated by $RC/(A-1)$, where A is the gain of NAND gates of RS latch. The throughput of a metastability-based TRNG is bounded by this probability distribution.

Although the principle of a metastability-based TRNG is simple, it is not easy to achieve high quality of randomness. Any kind of unbalance in circuit will lead to biasing of output; e.g., the output Q might be biased if there is a skew in Clk signal. Dispersion of circuit elements may also lead to the bias of output, if it involves the difference in drive strengths of two NAND gates. Even if the circuit is perfectly balanced, the output might be affected by the previous output, which may remain as a small voltage difference of internal nodes after initialization (i.e., the period of $Clk = 0$).

For these reasons, metastability-based TRNGs have been regarded as unreliable and difficult for practical applications [11].

3.2 Preceding Studies on Metastability-based TRNGs

Most of the metastability-based TRNGs have been implemented and evaluated with full-custom LSI technology.

Bellido et al. [17] presented a method to generate true random numbers from the metastability of an RS latch. The circuit was designed and implemented with close attention; e.g., a dedicated initialization clock was adopted to precharge and discharge internal nodes. The operation was verified with a 2 μm CMOS process.

Kinniment and Chester [18] proposed a TRNG based on R-flop, which has a differential preamplifier input followed by a bistable latch. Their TRNG avoids the bias in random numbers, using a negative feedback loop that adjusts the bias voltage of R-flop. Their design was verified with a 0.6 μm CMOS process.

Epstein et al. [19] presented a TRNG, which was implemented with 0.18 μm CMOS. The source of entropy is metastability that is initiated by the transition from the oscillator mode to the latch mode. This circuit is large in logic scale, and consumes much power in the oscillator mode.

Holleman et al. [20], [21] presented two types of metastability-based TRNGs: DC-nulling type and FIR type. Their TRNGs were implemented with 0.35 μm CMOS, and passed NIST test.

Tokunaga et al. [22], [23] designed a TRNG circuit, which is based on the metastability of an RS latch. This circuit has a feedback loop to maximize the resolution time of metastability, which improves the quality of randomness.

All these TRNGs were implemented with full-custom LSI technology. Meanwhile, our TRNG is composed of logic gates, which can be implemented and evaluated with an FPGA device.

There is a preceding study on the FPGA implementation of metastability-based TRNG. Danger et al. [24] presented a TRNG, which is based on the metastability caused by setup/hold time violation of D latch. Since the D latches embedded in an FPGA are resistant to metastability [25]–[27], Danger implemented D latches with look-up tables (LUTs) of FPGA. Their TRNG was manually placed and routed, adjusting the timing of signals with wire delays.

Danger's TRNG depends on delicate adjustment of timing, and has problems in reproducibility, portability, and stability. For example, their TRNG might be easily affected by operational environment, since wire delays are very sensitive to supply voltage and operating temperature. On the other hand, the design of our TRNG is symmetrical, and consequently robust against the change of operational environment.

After we first presented the preliminary results of our TRNG in January 2009 [28], a new study on a metastability-based TRNG was presented by Varchola and Drutarovsky [29] in August 2010. They proposed a bi-stable structure named TERO (Transition Effect Ring Oscillator), where oscillatory phase can be forced. Randomness is harvested as a variance of the number of oscillations counted after each excitation. The circuit structure of TERO is much similar to our TRNG, while the source of randomness is totally different from ours (the final state of the bistable circuit). An experimental TRNG was implemented by xor-ing two TEROs, which produced a random sequence at 250 kbps which passed NIST statistical test without postprocessing. Although the quantitative evaluations of TERO and our TRNG are anticipated, we leave it as a theme of future works.

3.3 Random Number Generation with FPGA devices

As stated above, careful consideration is required to implement metastability-based TRNGs. Custom LSI technologies have been adopted in many preceding studies, while there is no evidence that the FPGA implementation is unreasonable or impossible. In this section, a simple experiment is conducted to examine the feasibility of FPGA implementation.

Using a Xilinx Virtex-4 FX FPGA device [30], we implemented dozens of latches shown in Fig. 1. Each output of an TRNG was multiplexed and then connected to an IO buffer, whose output voltage was set to LVCMOS 3.3 V. Each NAND gate was implemented with a 4-input LUT.

Figure 2 displays a typical waveform of a latch. Five seconds of three signals (Clk, Q, and $\bar{Q}$) are shown with 25 ns/div sweep rate, which were observed with a Tektronix TDS 2024B digital storage oscilloscope. It is readily seen that the latch generates a random output for each clock, where the transition time has a certain probability distribution. The observed transition time has reached 100 ns (max),

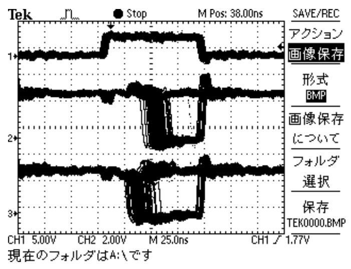


Fig. 2 Observed metastability of an RS latch, which was implemented with a Virtex-4 FPGA device.

which becomes a guideline of the sampling interval in the following experiments.

4. Platform and Measures for Evaluation

4.1 Evaluation Platform

In this study, Xilinx ML405 board was adopted as an evaluation platform, which includes a Virtex-4 FX FPGA (XC4VFX20), 128 MB DDR SDRAM, and peripheral I/O devices. For design and implementation, ISE Foundation 10.1.03 software was used with default parameters.

A Virtex-4 FX FPGA is targeted for high-performance full-featured embedded applications, and includes one (or more) embedded PowerPC 405 (PPC405) RISC core(s). TRNG circuit is attached to the peripheral bus of PPC405 in the XC4VFX20 device. The overall evaluation system consists of a PPC405 processor, SDRAM, Ethernet MAC, System ACE (Compact Flash configuration controller), UART, and TRNG. The respective clock frequencies are 300 MHz for PPC405 and 100 MHz for the peripheral devices.

TRNG has a buffer of 1 MB SRAM, and generates random data until the buffer is filled. The PPC405 processor polls the status of TRNG, reads the data out when the buffer is full, and then restarts TRNG for more data. Since a large quantity of data is required for statistical tests of randomness, embedded Linux operating system was ported for this system. The random data are stored in a filesystem of a Compact Flash memory card.

The Clk signal for TRNG is generated by dividing the peripheral bus clock by a factor $2m$, where m can be set via a control register. The duty ratio of Clk is fixed to 50% in this study, and the consequent clock period of Clk is $20m$ ns.

4.2 Measures of the Quality of Randomness

This study adopts two renowned tests for randomness for two different purposes. The Diehard test is convenient for screening various design alternatives, since it works with a small set of data in a short time. On the other hand, NIST test gives comprehensive tests on a large set of data; it is thus suited for the final verification purpose.

4.2.1 Diehard Test

The Diehard test [31] is a battery of statistical tests for measuring the quality of random numbers. This study adopts 14 tests from the Diehard test v0.2 beta [32]. Though the Diehard v0.2 consists of 17 tests, three of them (GCD, Gorilla, and Overlapping permutation) were excluded since they require large amount of data.

Each test outputs a *p*-value. *P*-values should be distributed uniformly in the interval [0,1], if the *p*-values are derived from true random numbers. In this study, we adopt the number of *p*-values of $0.01 < p < 0.99$ as the measure of TRNG quality. Since the Diehard test outputs 229 values, approximately 225 values are expected to appear in the interval (0.01, 0.99), if the TRNG is good enough. In the Diehard test, some of the tests may quit without calculating *p*-values, whenever the test aborted by any reasons (e.g., too little entropy). It should be noted that such tests are automatically excluded from the evaluation result by our measure.

The above method is neither sufficient nor exhaustive; however, it is useful enough to reject inferior options in design process.

4.2.2 NIST Test

NIST test [3], [4] is a statistical test suite for random numbers, which is provided by National Institute of Standards and Technology. NIST test consists of 15 tests, and explicitly defines the recommendations and guidelines. In this study, NIST test version 1.8 is adopted with default parameters. NIST test uses 10^9 bit of random data to examine the distribution of the p values, which are generated by repeating a test of 10^6 bit 1000 times. The input data are regarded to be truly random if and only if they passed all tests of NIST test suite.

5. Implementation

5.1 Design Overview

In an FPGA device, logic functions are implemented with look-up tables (LUTs). Two NAND gates of an RS latch (Fig. 1) are thus replaced by two LUTs in our FPGA implementation (Fig. 3), which is designated as **LUT latch** in the following discussion.

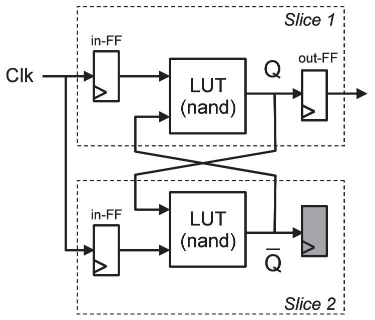


Fig. 3 LUT latch: an RS latch which consists of look-up tables (LUTs) and embedded flipflops (FFs).

Although it is possible to implement an RS latch in an FPGA as a digital component, it is impossible to adjust the analog imbalance caused by individual difference or process variation. Therefore, little or no entropy might be generated by an individual LUT latch. To harvest sufficient entropy, it is necessary (1) to control the quality of LUT latches and (2) to collect entropy from many LUT latches.

It is practically important to generate a qualified random sequence from biased random numbers, which are often derived from physical random number generators. Thus, there have been many studies on post-processing functions for physical random number generators. In this study, we adopt an XOR corrector [33], which is one of the oldest and the most widely accepted methods. Figure 4 illustrates the block diagram of our TRNG, which generates a single output by XOR'ing the output of *N* latches. This circuit is designated by **LUT latch** *N* in the following discussion.

It should be noted that a Sunar-type TRNG is derived by replacing LUT latches with free-running ring oscillators. To compare our TRNG and a Sunar-type TRNG with the same technology, we implemented and evaluated a Sunar-type TRNG that was presented by Schellekens et al. [9] This design is designated as **Ring Osc 110**, since it includes 110 ROs consisting of 3 inverters.

5.2 Implementation Details of an LUT latch

Though the internal structure of an FPGA is specific to its family and manufacturer, the architectures of recent FPGAs are fundamentally analogous. Logic functions are realized by SRAM-based logic resources, which are connected by interconnect resources. An FPGA chip consists of a hierarchical array of logic resources, interconnect resources (wires and switches), and other specialized resources.

In case of the Xilinx Virtex-4 FPGA [30], a configurable logic block (CLB) consists of four slices (two SLICELs and two SLICEMs) as illustrated in Fig. 5. SLICELs are located at the odd columns, while SLICEMs resides at the even columns of a two-dimensional array of slices. Both SLICEL and SLICEM include two LUTs and two DFFs. A SLICEM additionally provides a RAM function called *distributed RAM* mode.

An LUT latch is implemented with two slices, as shown in Fig. 3. These two slices cannot be placed in the same CLB, since it is impossible to allocate the necessary interconnection. Therefore, an LUT latch is composed by two slices of the same kind from two different CLBs. The

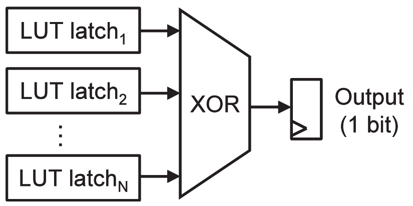


Fig. 4 Block diagram of LUT latch *N*.

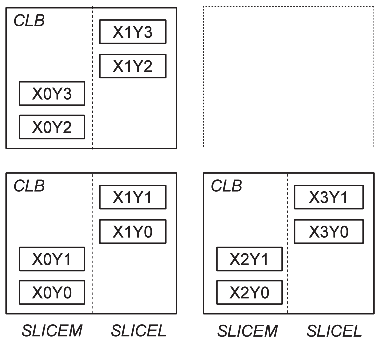


Fig. 5 Virtex-4 CLB array [30].

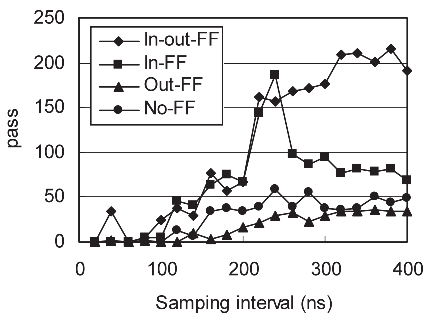


Fig. 6 The effect of in-FFs and out-FFs on the quality of randomness.

other slices in these CLBs are not wasted; they are automatically allocated to other purposes by CAD software.

Each LUT latch is implemented as a hard macro; i.e., two slices of an LUT latch are placed with a predefined collocation and connections, which are manually designed to minimize the skew of internal signals. For the present, we construct an LUT latch with two SLICELs that are adjacent in X-coordinate. That is, if Slice 1 of Fig. 3 is placed at X1Y0 of Fig. 5, Slice 2 is placed at X3Y0. Other possibilities are examined in Sect. 5.3.

The use of hard macro promotes the reproducibility of results as well as the quality of randomness. Meanwhile, the placement and routing of the whole TRNG is performed by automatic place-and-route software. In other words, our TRNG (LUT latch N) is a soft macro, which includes N instances of a hard macro (LUT latch).

The in-FFs and out-FF of Fig. 3 are implemented with embedded FF elements, whose clock is fed by the peripheral clock (100 MHz). Although two in-FFs look logically unnecessary, they are actually important to reduce the skew of *Clk* signal. Each in-FF is embedded in the same slice as the corresponding LUT to minimize the skew from the rising edge of the peripheral clock. The out-FF is required to decouple the capacitive load of output wire from the internal node (Q). Without the out-FF, the capacitive load of Q becomes much larger than that of $\bar{Q}$, which severely degrades the randomness of the output. An out-FF for $\bar{Q}$ physically exists in Slice2, though it is not used.

To examine the effect of in-FFs and out-FFs, the following four designs of LUT latch were implemented and evaluated.

In-out-FF is a design that includes both in-FFs and out-FFs, as illustrated in Fig. 3.
In-FF is a design that includes in-FFs only, without implementing out-FFs.
Out-FF is a design that includes out-FFs only, without implementing in-FFs.
No-FF is a design that includes neither in-FFs nor out-FFs.

Figure 6 summarizes the results of the Diehard test, where 32 latches were implemented with the above four designs of the LUT latch. Overall, In-out-FF is the best choice. As readily seen in Fig. 6, No-FF and Out-FF are much inferior to the other two, which suggests that in-FFs are important for entropy generation. At the same time, it is also obvious that out-FF is necessary for the best results.

In the following discussion, the design In-out-FF is adopted for our TRNG.

5.3 Distance between two LUTs

The distance between two slices might affect the quality and throughput of an LUT latch. There is a possibility that longer wires may collect larger noise and consequently generate larger entropy. Meanwhile, there are many drawbacks to a large distance between slices. Long wires should be avoided because they consume more interconnect resources. A long wire also incurs heavier capacitive load, which naturally leads to a larger time constant and a smaller throughput. The entropy of a latch may decrease, since the skew tends to increase according to the length of wire. Such pros and cons have to be examined quantitatively with experiments.

This section examines the effect of the distance between two SLICELs of an LUT latch. Setting the origin to the first SLICEL (Slice 1 in Fig. 3), various layouts were examined with various locations for the second SLICEL (Slice 2). Each layout is designated by dXiYj, where i and j are the relative coordinates of Slice 2 in X and Y axes. As stated in Sect. 5.2, i must be even, since two slices are the same type. If the origin is set at X1Y0, the layout dX2Y0 designates the latch that consists of two SLICELs at X1Y0 and X3Y0 (see Fig. 5 for these coordinates).

Table 1 lists the results of the Diehard test for various collocation of slices, where the number of latches was 64 and the sampling interval was 320 ns. The layout dX0Y1 was rejected by CAD, because it was impossible to connect two SLICELs of the same CLB. As readily seen, the quality of randomness was not improved by keeping larger distance.

It may seem strange that the results of dX4Y0 and dX8Y0 are much worse than that of dX2Y0 and dX6Y0. The routing resources of Virtex-4 consist of three classes of segments: double (covering 2 logic blocks), hex (covering 6 blocks), and long (covering all blocks in a row or column) [34]. This naturally means that the number of routing segments does not increase monotonically as the distance between blocks increases. The discontinuous behavior in

Table 1 The quality of randomness for various collocation of LUTs in LUT latch 64.

	dX2Y0	dX4Y0	dX6Y0	dX8Y0	dX0Y2	dX0Y3	dX0Y4
pass	226	0	224	0	215	224	131

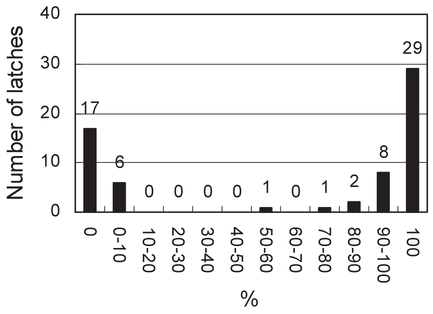


Fig. 7 Histogram of the probability for a latch to output "1", where 64 LUT latches were examined.

Table 1 might suggest that the number of routing segments affects the quality of TRNG. Although more detailed investigations are anticipated, further investigation on this matter is left for future studies.

Generally speaking, it is desirable to keep a hard-macro small, since a large hard-macro consumes more routing resources and negatively affects the placement and routing of other components. After all, we could not find any reasons to adopt a long-distance implementation. The layout dX2Y0 is adopted in this study, since dX2Y0 is slightly better than dX0Y2.

5.4 Individual Difference of LUT latch

Even using a hard macro, each instance of LUT latch has different characteristics, which vary depending on individual chips or systems. Such differences occur for various reasons: e.g., characteristic dispersion of transistor, process variation, and operating conditions.

In this section, the individual difference in LUT latches is examined. 64 latches were implemented on an XC4VFX20 FPGA device, and 1 MB of output was collected from each latch. The sampling interval was set to 320 ns.

Figure 7 displays the histogram of the probability of the appearance of "1" in each output sequence. If all latches are ideal, the histogram should show a binomial distribution whose peak is at 50%. However, the reality is far from ideal; 17 latches are stuck at zero, while 29 latches are stuck at one. That is, 46 latches out of 64 generated no entropy in this experiment. A certain amount of entropy was generated by the remaining 18 latches, though most of them exhibit apparent biases.

Hence, it is obvious that a certain number of latches are required for our TRNG to generate random numbers of acceptable quality. This topic is quantitatively discussed in Sect. 5.6.

Table 2 The quality of randomness, which was derived from LUT latch 32 with three hard macros.

	Lxxx	LLxx	LLMM
pass	209	1	28

5.5 Placement of LUT latches

The hard macro of LUT latch is implemented by two SLICELs of two adjacent CLBs. Since each CLB includes two SLICELs, two adjacent CLBs can accommodate two LUT latches. Moreover, it is also possible to compose an LUT latch with two SLICEMs. In consequence, maximally four LUT latches can be placed in two adjacent CLBs, using all SLICELs and SLICEMs.

The question is whether we should place two or more latches in a pair of CLBs. Implementing more latches in a pair of CLBs, the whole TRNG can be laid out more compactly. Meanwhile, packing many latches tightly, the difficulties in routing might result in the degradation of randomness. If two latches are laid closely in line, the output of these latches might show a correlation, which will degrade the quality of randomness immediately.

To assess such situation quantitatively, we designed the following three hard macros and committed an experiment.

Lxxx implements a single LUT latch with two SLICELs from two adjacent CLBs. It should be noted that it is still possible for two latches to reside in the same pair of CLBs. Since these macros are automatically placed by CAD software, two latches might be unexpectedly allocated to the same pair of CLBs.

LLxx implements two LUT latches with two adjacent pairs of SLICELs. Though two latches are laid adjacently, it does not always mean that they reside in the same pair of CLBs. In case of Fig. 5, the origins of two latches may be placed at either (X1Y0, X1Y1) or (X1Y1, X1Y2). In the former case, two latches reside in the same pair of CLBs; in the latter case, two latches reside in the adjacent (but different) pairs of CLBs.

LLMM implements four LUT latches with two adjacent CLBs, where two latches are composed of four SLICELs and the other two are composed of four SLICEMs. For example, four pairs of slices correspond to (X0Y0, X2Y0), (X0Y1, X2Y1), (X1Y0, X3Y0), and (X1Y1, X3Y1) in Fig. 5.

Figure 8 displays three layouts of LUT latch 32 with each of three macros (Lxxx, LLxx, and LLMM), where the placement of hard macros is displayed. As expected, latches are more tightly grouped with LLxx and LLMM macros.

These three implementations were evaluated with the Diehard test at the sampling interval of 320 ns. Table 2 summarizes the evaluation results, which clearly show that Lxxx

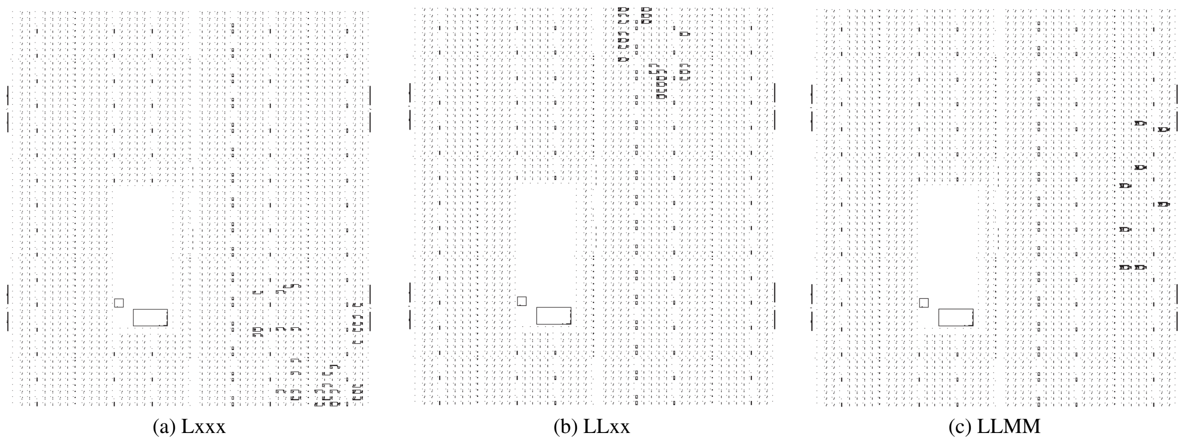


Fig. 8 The layouts of TRNG with three kinds of hard macros: Lxxx, LLxx, and LLMM.

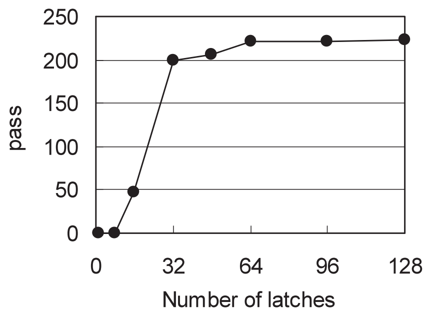


Fig. 9 The effect of the number of latches upon the quality of randomness.

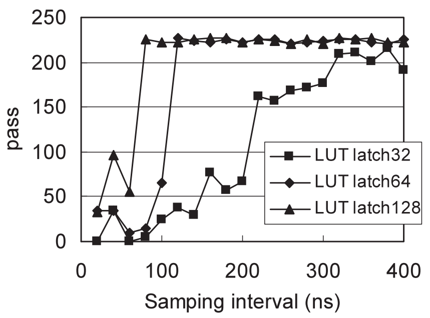


Fig. 10 The effect of sampling interval upon the quality of randomness.

is superior to LLxx and LLMM. Therefore, Lxxx is adopted in the following discussion.

There are many other possibilities to lay out latches. For example, according to the above results, it might be worth placing latches sparsely. Although it is possible to control the placement manually, we did not conduct further experiments for the following reasons. Generally speaking, manual placement interferes in the global optimization of CAD software; it may deteriorate the overall quality of the system, and should be avoided whenever possible. Manual placement is inevitably dependent on a specific device, while this study aims at a portable design of TRNG. Since sufficient quality was derived with automatic placement of our hard macro, we could not find any good reasons for manual placement in this study.

5.6 Sampling Interval and the Number of Latches

As stated in Sect. 5.1, the quality of randomness is expected to depend on the number of latches. Figure 9 summarizes the results of the Diehard test, where the sampling interval was set to 320 ns. As expected, the quality improves with the number of latches, and reaches a point of diminishing returns with 64 latches (or more).

The sampling interval is another important factor in the quality of randomness. The resolution time of metastability has a probabilistic distribution, which depends on the gain and the time constant of each LUT latch. More entropy is thus generated with a longer sampling interval, while the throughput of TRNG decreases in inverse proportion to the sampling interval. The optimal sampling interval has to be determined experimentally, because it is dependent on such factors as design, implementation, and target device.

Figure 10 summarizes the results of the Diehard test for LUT latch 32, 64, and 128. As readily seen, 32 latches are not enough to pass the Diehard test, even with long sampling intervals. LUT latch 64 seems to pass the Diehard test with the sampling interval larger than 120 ns; meanwhile, LUT latch 128 passes when the sampling interval is larger than 80 ns. This difference simply suggests that more latches generate more entropy in the same period.

5.7 Initialization Time and Transition Time

The previous section confirmed that there is a minimal sampling interval to maintain the quality of randomness. Here, it should be reminded that the duty ratio was fixed to 50% in the above experiments (c.f. Sect. 4.1). Actually, the respective periods of $Clk = 0$ and $Clk = 1$ correspond to different physical processes: the initialization of internal nodes ($Clk = 0$) and the resolution of metastability ($Clk = 1$). Therefore, these two periods may be independently changed to minimize the overall cycle time of Clk.

Two experiments were conducted to elucidate the bot-

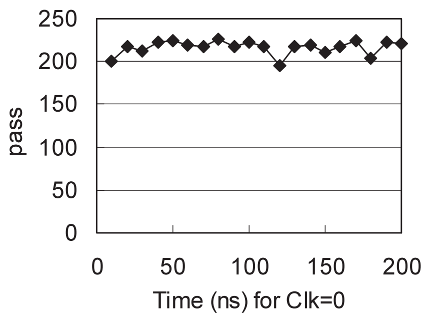


Fig. 11 The effect of initialization time on the quality of randomness.

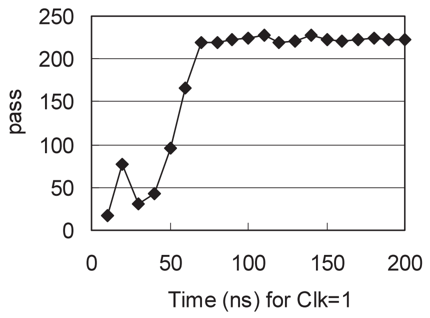


Fig. 12 The effect of resolution time (or transition time) on the quality of randomness.

tleneck in the minimal sampling interval of Sect. 5.6. The number of latches is set to 64, and the TRNG was evaluated with the Diehard test.

The first experiment examines the effect of the initialization periods (*Clk* = 0). The circuit was evaluated for various initialization period (*Clk* = 0) with a fixed resolution period of 200 ns. As readily seen in Fig. 11, the initialization period had little effect upon the quality of randomness. No definite degradation was observed even for short periods.

In the second experiment, the effect of the resolution period (*Clk* = 1) was examined. The circuit was evaluated with a fixed initialization time of 200 ns. In Fig. 12, a definite degradation is observed at 70 ns, which is the physical limit of this TRNG.

The results of this section suggest that the throughput of the TRNG might be improved by changing the periods of *Clk* = 0 and *Clk* = 1 independently. However, such situation depends on the design and implementation of the TRNG. This kind of optimization should be adopted optionally as a part of device-dependent adjustment at the final stage of implementation.

In the following evaluation, we adopt 50% as the duty ratio of the *Clk* signal for a baseline implementation of our TRNG.

6. Evaluation

This section summarizes the logic scale, the throughput (generation rate), and the quality of randomness of our TRNG.

Table 3 Logic scale (slice).

Design	TRNG	System
LUT latch 64	145	7013
LUT latch 128	290	7159
LUT latch 256	580	7447
Ring osc 110	359	7219

Table 3 lists the logic scales of various designs. The column "TRNG" designates the logic scale of TRNG itself, while the column "System" designates the whole Linux system including TRNG. The target device is XC4VFX20, which contains 8544 slices. In summary, the TRNG of LUT latch 64 occupies no more than 1.7% of the device, while the whole system occupies about 82%. The logic scale of Ring Osc 110 is listed as a contrast (c.f. Sect. 5.1); LUT latch 64 is as small as 40% of Ring Osc 110.

Table 4 summarizes the results of NIST test for LUT latch 64 and Ring Osc 110, where the sampling interval was set to 320 ns. Version 1.8 of NIST test was used with default parameters to examine 1000 sets of 10^6 bit. In Table 4, the column "Proportion" designates the ratio of passes in 1000 data sets. The tests with many items are indicated in the form of passes/items. The failed items are emphasized in bold style, while the aborted items are indicated by N/A (not available).

In short, LUT latch 64 successfully passes NIST test without post-processing, while Ring Osc 110 fails in many items. This does not immediately mean that a Sunar-type TRNG is inferior to our TRNG. Schellekens et al. [9] proposed a post-processing circuit as a prerequisite for Sunar-type TRNG, while the sole TRNG was evaluated in this study. It should be also noted that the parameters of Ring Osc 110 were not optimized for this evaluation platform. Better results might have been derived, if a Sunar-type TRNG was designed with optimal parameters: e.g., the number of inverters for each RO, the number of ROs, and the sampling interval.

Nevertheless, in our evaluation, our TRNG is superior to Ring Osc 110 in both the logic scale and the quality of randomness. If the post-processing circuit is added to Ring Osc 110, the difference in logic scale becomes even larger.

To determine the maximal generation rate, the random sequences were generated at various sampling intervals, and were verified with NIST test. Table 5 lists the maximal generation rates of LUT latch 64, 128 and 256. The results of preceding studies, which implemented TRNGs with FPGA devices, are also listed in Table 5. In terms of the published generation rate, our metastability-based TRNGs compare well with the other types of TRNGs. Here, it should be noted that the throughput of TRNG depends on various factors (e.g., evaluation platform). The throughput in Table 5 should be interpreted as a rough measure, since the target devices and other factors are not standardized.

It is also misleading to compare the generation rate separately, since it is very easy to double the generation rate by

Table 4 Detailed results of NIST test, where boldface indicates failure.

	LUT latch 64		Ring osc 110	
	P-value	Proportion	P-value	Proportion
frequency	0.907419	0.9930	**0.000000**	**0.0000**
block-frequency	0.576961	0.9870	**0.000000**	**0.0030**
cumulative sums-up	0.532132	0.9930	**0.000000**	**0.0000**
cumulative sums-down	0.175691	0.9930	**0.000000**	**0.0000**
runs	0.779188	0.9900	**0.000000**	**0.0000**
longest-run	0.670396	0.9870	**0.000000**	**0.5260**
rank	0.701366	0.9890	0.739918	0.9890
fft	0.128874	0.9880	0.006107	0.9840
nonperiodic-templates	148/148	148/148	**38/148**	**41/148**
overlapping-templates	0.846338	0.9870	**0.000000**	**0.0000**
universal	0.632955	0.9900	**0.000000**	**0.9140**
apen	0.906069	0.9860	**0.000000**	**0.0000**
random-excursions	8/8	8/8	N/A	N/A
random-excursions variant	18/18	18/18	N/A	N/A
serial1	0.680755	0.9890	**0.000000**	**0.4760**
serial2	0.972382	0.9950	0.038565	0.9900
linear-complexity	0.516113	0.9880	0.484646	0.9920

Table 5 The throughputs (Mbps) of various TRNGs implemented with FPGA technology.

Design category	Reference	Device	[Mbps]
Oscillator sampling	Tsoi et al. [2]	Xilinx XCV300E	0.03
PLL/DLL	Fischer, Drutarovský [5]	Altera EP20K200E	0.07
	Fischer et al. [6]	Altera EP1S25	1.
	Kwok, Lam [7]	Xilinx XC2VP20	6.
Free-running oscillator	Kohlbrenner, Gaj [10]	Xilinx XCV1000	0.5
	Schellekens et al. [9]	Xilinx XC2VP30	2.5
	Dichtl, Golić [12]	Xilinx XC3S200	12.5
	Wold, Tan [13]	Altera Cyclone II	100.
	Hada, Abe [15]	Xilinx XC5VLX50	50.
D latch metastability	Danger et al. [24]	Altera EP1S25	20.
Transition Effect Ring Oscillator	Varchola, Drutarovsky [29]	Xilinx XC3S500E	0.25
RS latch metastability	(LUT latch 64)	Xilinx XC4VFX20	3.85
	(LUT latch 128)	Xilinx XC4VFX20	8.33
	(LUT latch 256)	Xilinx XC4VFX20	12.5

implementing two equivalent TRNGs in parallel.[†] Since the generation rate and the logic scale are equally important for TRNG, the throughput per slice should be also examined. In case of our TRNGs, two instances of LUT latch 128 may generate random numbers at 16.7 Mbps, while a single LUT latch 256 achieves no more than 12.5 Mbps. According to Tables 3 and 5, LUT latch 128 was the most area-efficient in our three designs (LUT latch 64, 128, and 256).

7. Conclusion

This study presented the evaluation results of a metastability-based TRNG that was implemented with a common FPGA device. The proposed TRNG consists of digital circuits only, and passes NIST test without any post-processing. Our TRNG with 256 latches generates random numbers at 12.5 Mbps with 580 slices of a Xilinx Virtex4 FPGA (XC4VFX20) device.

The power consumption of our TRNG is expected to be modest for the following reasons; (1) the logic scale is relatively small, and (2) the clock signal can be arbitrarily stopped. Thus, our TRNG is particularly suited to the embedded applications that are obliged to suppress power consumption. Although many previous studies claimed that metastability-based TRNGs are unreliable, this study demonstrated that our TRNG can generate high-quality random sequences with various design choices including the number of latches, sampling interval, and clock duty ratio.

Although the fundamental functionality of our TRNG was verified in this study, more studies are required for practical applications of this TRNG.

First of all, our TRNG should be examined against different power supply voltages and operating temperatures. Since the time constant and the gain are generally dependent on the voltage and temperature, the generation rate of our TRNG may be affected by these factors to a certain extent. Meanwhile, the quality of randomness is expected to be insensitive to these factors, because the circuit of our TRNG is symmetric. If the operating temperature and the supply voltage are equally applied to each element of the circuit, the circuit should remain well-balanced to generate entropy.

The experiments with other technologies are also desired. Since our TRNG simply consists of LUTs and FFs,

[†]Practically, close attention should be paid to avoid the correlation between two TRNGs.

the use of other technologies (e.g., Altera FPGA or semi-custom LSI) is possible and straightforward. Although it is important to optimize the design parameters for the adopted technology, it may be performed systematically as detailed in this study.

Further analyses and improvements of LUT latch are anticipated. The designs to generate more entropy should be explored. There are large individual differences in LUT latches as described in Sect. 5.4, and many latches generate no entropy (Fig. 7). This phenomenon should be examined under various conditions to improve the implementation of LUT latches. The latches that generate no entropy in one situation may generate some entropy in other situations.

The investigations of statistical properties are very important for the reproducibility and the reliability of our TRNG. The statistics of latches should be firstly examined. The quality, or the entropy generation rate, of a latch is supposed to be destined by the physical properties of its location. It is thus necessary to examine the qualities of latches on various locations. Secondly, the individual differences among FPGA chips should be examined.

The quantitative modeling of our design is also essential for generalization and theoretical assurance of security. All these important items are left for future studies.

Acknowledgments

This work was partially supported by a Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science (JSPS).

References

[1] R. Fairfield, R. Mortenson, and K. Coulthart, "An LSI random number generator (RNG)," Advances in Cryptology: Proc. CRYPTO 84, LNCS 196, pp.203–230, Springer, 1985.

[2] K. Tsoi, K. Leung, and P. Leong, "Compact FPGA-based true and pseudo random number generators," Proc. IEEE Annual Symp. Field-Programmable Custom Computing Machines (FCCM 2003), pp.51–61, IEEE Computer Society, 2003.

[3] NIST, "Statistical test suite," Aug. 2008. http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html.

[4] A. Rukhin et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications," tech. rep., NIST, 2001. NIST Special Publication 800-22 (with revisions dated May 15, 2001).

[5] V. Fischer and M. Drutarovský, "True random number generator embedded in reconfigurable hardware," Proc. Cryptographic Hardware and Embedded Systems - CHES 2002, LNCS 2523, pp.415–430, Springer, 2002.

[6] V. Fischer, M. Drutarovský, M. Šimka, and N. Bochard, "High performance true random number generator in Altera Stratix FPLDs," Proc. Intl. Conf. Field Programmable Logic and Application (FPL 2004), LNCS 3203, pp.555–564, Springer, 2004.

[7] S.H.M. Kwok and E.Y. Lam, "FPGA-based high-speed true random number generator for cryptographic applications," Proc. IEEE TENCON 2006, pp.1–4, IEEE, 2006.

[8] B. Sunar, W.J. Martin, and D.R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," IEEE Trans. Comput., vol.56, no.1, pp.109–119, 2007.

[9] D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," Proc. Intl. Conf. Field Programmable Logic and Applications (FPL 2006), pp.1–6, IEEE, Aug. 2006.

[10] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for FPGAs," Proc. ACM/SIGDA 12th Intl. Symp. Field Programmable Gate Arrays (FPGA 2004), pp.71–78, ACM, 2004.

[11] J.D. Golić, "New methods for digital generation and postprocessing of random data," IEEE Trans. Comput., vol.55, no.10, pp.1217–1229, 2006.

[12] M. Dichtl and J.D. Golić, "High-speed true random number generation with logic gates only," Proc. Cryptographic Hardware and Embedded Systems - CHES 2007, LNCS 4727, pp.45–62, Springer, 2007.

[13] K. Wold and C.H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," International Journal of Reconfigurable Computing, vol.2009, pp.1–8, 2009. Article ID 501672.

[14] N. Bochard, F. Bernard, and V. Fischer, "Observing the randomness in RO-based TRNG," Proc. International Conference on Reconfigurable Computing and FPGAs, pp.237–242, IEEE Computer Society, 2009.

[15] K. Hada and K. Abe, "Performance enhancement of the ring oscillator type true random number generator on FPGA," IEICE Technical Report, CSEC2011-53, 2011. (in Japanese).

[16] L. Kleeman and A. Cantoni, "Metastable behavior in digital systems," IEEE Design and Test of Computers, vol.4, no.6, pp.4–19, 1987.

[17] M. Bellido, A. Acosta, M. Valencia, A. Barriga, and J. Huertas, "Simple binary random number generator," Electron. Lett., vol.28, no.7, pp.617–618, March 1992.

[18] D. Kinniment and E. Chester, "Design of an on-chip random number generator using metastability," Proc. 28th European Solid-State Circuits Conference (ESSCIRC 2002), pp.595–598, IEEE SSCS, Sept. 2002.

[19] 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," Proc. Cryptographic Hardware and Embedded Systems - CHES 2003, LNCS 2779, pp.152–165, Springer, 2003.

[20] J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio, "A $2.92\,\mu$w hardware random number generator," Proc. 32nd European Solid-State Circuits Conference (ESSCIRC 2006), pp.134–137, IEEE, Sept. 2006.

[21] J. Holleman, S. Bridges, B.P. Otis, and C. Diorio, "A $3\,\mu$W CMOS true random number generator with adaptive floating-gate offset cancellation," IEEE J. Solid-State Circuits, vol.43, no.5, pp.1324–1336, May 2008.

[22] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," IEEE International Solid-State Circuits Conference (ISSCC 2007), pp.404–405, IEEE, Feb. 2007.

[23] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," IEEE J. Solid-State Circuits, vol.43, no.1, pp.78–85, Jan. 2008.

[24] J.L. Danger, S. Guilley, and P. Hoogvorst, "Fast true random generator in FPGAs," Proc. IEEE Northeast Workshop on Circuits and Systems (NEWCAS 2007), pp.506–509, IEEE, Aug. 2007.

[25] Altera, Metastability in Altera Devices, May 1999. Application Note 42.

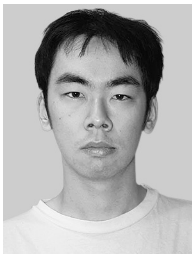
[26] Xilinx, Metastability Considerations, XAPP077. Jan. 1997.

[27] P. Alfke, Metastable Recovery in Virtex-II Pro FPGAs. Xilinx, XAPP094. Feb. 2005.

[28] H. Hata and S. Ichikawa, "FPGA implementation of metastability-based true random number generator," IEICE Technical Report, CPSY2008-57, 2009. (in Japanese).

[29] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generators," Proc. Cryptographic

Hardware and Embedded Systems - CHES 2010, LNCS 6225, pp.351–365, Springer, 2010.
[30] Xilinx, Virtex-4 FPGA User Guide, June 2008. UG070 (v2.5).
[31] G. Marsaglia, "The Marsaglia random number CDROM including the Diehard battery of tests of randomness," Oct. 2008. http://www.stat.fsu.edu/pub/diehard/.
[32] CISC and HKU, "Diehard battery of tests of randomness v0.2 beta," Dec. 2008. http://i.cs.hku.hk/˜diehard/.
[33] R. Davies, "Exclusive OR (XOR) and hardware random number generators," Feb. 2002. http://www.robertnz.net/pdf/xor2.pdf.
[34] P.B. Minev and V.S. Kukenska, "The Virtex-5 routing and logic architecture," Annual Journal of Electronics, Technical University of Sofia, vol.3, pp.107–110, 2009.



Hisashi Hata received his B.E. and M.E. degrees in 2007 and 2009 from the Department of Knowledge-based Information Engineering of Toyohashi University of Technology. Since April 2009, he is affiliated with Renesas Technology Corporation.

Shuichi Ichikawa received his D.S. degree in Information Science from the University of Tokyo in 1991. He was affiliated with Mitsubishi Electric Corporation, Nagoya University, and Toyohashi University of Technology. Since April 2011, he is a professor of Numazu National College of Technology. His research interests include parallel processing, high-performance computing, and custom computing systems, and computer security. He is a member of IEEE, ACM, and IPSJ.