IEICE TRANS. ELECTRON., VOL.E104–C, NO.7 JULY 2021

PAPER Special Section on Solid-State Circuit Design — Architecture, Circuit, Device and Design Methodology

Energy-Efficient Post-Processing Technique Having High Extraction Efficiency for True Random Number Generators

Ruilin ZHANG[†a)], ***Nonmember***, **Xingyu WANG**[†], ***Student Member***, ***and*** **Hirofumi SHINOHARA**[†], ***Member***

SUMMARY In this paper, we describe a post-processing technique having high extraction efficiency (ExE) for de-biasing and de-correlating a random bitstream generated by true random number generators (TRNGs). This research is based on the N-bit von Neumann (VN_N) post-processing method. It improves the ExE of the original von Neumann method close to the Shannon entropy bound by a large N value. However, as the N value increases, the mapping table complexity increases exponentially (2^N), which makes VN_N unsuitable for low-power TRNGs. To overcome this problem, at the algorithm level, we propose a waiting strategy to achieve high ExE with a small N value. At the architectural level, a Hamming weight mapping-based hierarchical structure is used to reconstruct the large mapping table using smaller tables. The hierarchical structure also decreases the correlation factor in the raw bitstream. To develop a technique with high ExE and low cost, we designed and fabricated an 8-bit von Neumann with waiting strategy (VN_8W) in a 130-nm CMOS. The maximum ExE of VN_8W is 62.21%, which is 2.49 times larger than the ExE of the original von Neumann. NIST SP 800-22 randomness test results proved the de-biasing and de-correlation abilities of VN_8W. As compared with the state-of-the-art optimized 7-element iterated von Neumann, VN_8W achieved more than 20% energy reduction with higher ExE. At 0.45 V and 1 MHz, VN_8W achieved the minimum energy of 0.18 pJ/bit, which was suitable for sub-pJ low energy TRNGs.

key words: *post-processing techniques, von Neumann entropy extractor, true random number generator, low-power*

1. Introduction

True random number generator (TRNG) is a basic element in hardware security. TRNGs use random physical phenomena (e.g., thermal noise) to generate random unpredictable bitstreams, which are used in various areas such as cryptography algorithms and session identifications. However, the raw bitstream may not be used directly because of statistical defects, such as biases or correlations introduced by intentional or unintentional variations [1]–[3]. To improve the statistical quality of raw bitstreams and achieve a good tradeoff between the total power and area, additional post-processing techniques are incorporated in many designs [4]–[11].

Post-processing techniques can be classified into two categories. The first category is the fixed extraction efficiency (hereinafter ExE) technique. ExE is defined as the ratio of the bitstream length after using the post-processing technique and before using the technique. In this category, ExE is fixed regardless of the quality variations of the raw bitstream. Among them, the XOR extractor [4] is the simplest technique, but it cannot achieve high bias reduction. Some linear and non-linear codes were investigated in [12]–[14] to improve the de-biasing levels and ExE, but they could not achieve zero bias. Markov chains [15] and linear feedback shift registers (LFSRs) [5] function efficiently for the correlation problem without discarding the raw bit. Nevertheless, the entropy per bit can only be improved by compressing the raw bitstream [16]. Strong blenders in [16], decorrelators combined with BIW in [6], and PRESENT in [7] can address both the bias and correlation problems. However, the entropy source must satisfy the minimum entropy level. This is the same requirement as that in very complex hashing algorithm based SHA and block cipher based AES post-processing techniques. Furthermore, these techniques have long latency and tend to be power-hungry.

Another category is the variable-ExE post-processing technique. The von Neumann method [4], [17] is a well-known lightweight technique. It can remove all the biases in the independent and identically distributed (*i.i.d.*) raw bitstream by processing two bits each time and saving the first bit if the values are not identical, as shown in Fig. 1 (a). However, its ExE is only a maximum of 25% and it decreases with the input bias.

To improve the ExE of the original von Neumann, two types of solutions have been proposed. One solution is the iterated von Neumann (IVN) method [18], as shown in Fig. 1 (b). ExE increases by reusing the discarded information (XOR, R) as the input of the next stage VN_2 post-processing modules. It approaches the Shannon entropy

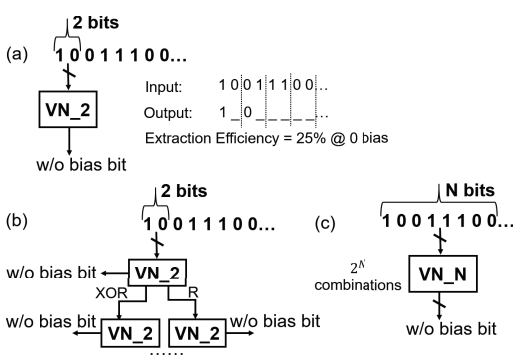

Fig. 1 Basic structure of von Neumann post-processing. (a) Original von Neumann. (b) Iterated von Neumann. (c) N-bit von Neumann.

Manuscript received July 30, 2020.
Manuscript revised November 1, 2020.
Manuscript publicized January 28, 2021.

[†]The authors are with the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu-shi, 808–0135 Japan.
a) E-mail: zrlsmile@asagi.waseda.jp

DOI: 10.1587/transele.2020CDP0006

bound by N iterations. An incomplete tree-based structure is presented in [19] under hardware constraints. Then, in [9], a Markov chain is combined with the IVN structure to address the correlation and bias problem. However, a 16-bit LFSR is also needed for removing the residual correlations. A hierarchical IVN structure in [10] is applied to the high throughput TRNG with multi-entropy sources.

In another solution, the N-bit von Neumann (VN_N) is proposed in theory [20]. It improves the ExE by gathering N bits (N ≥ 2) and processing them simultaneously, as shown in Fig. 1 (c). VN_N can reduce the sequential data transition in every cycle in the processing modules in IVN, and it can potentially reduce the dynamic power. However, the mapping table complexity grows exponentially as the N value increases, which makes VN_N unsuitable for hardware implementation.

To tackle the complexity problem, we proposed a waiting strategy [21] to achieve high ExE with a small N value and prototype concept of a two-step logic structure in [22]. This work is an extended version of [21], [22]. We further propose a Hamming weight mapping-based hierarchical structure that solves the 2^N mapping table complexity by using three small tables. We addressed a lightweight solution to the correlated input data by using the hierarchical structure. We implemented a hierarchical 8-bit von Neumann with waiting strategy (VN_8W) with 62.21% ExE in a 130-nm CMOS. VN_8W addresses the bias and correlation in the raw bitstream. VN_8W achieved a minimum energy consumption of 0.18 pJ/bit at 0.45 V, which was suitable for low-power TRNGs.

The rest of the paper is organized as follows. The general idea of VN_N is explained in Sect. 2. Our proposed VN_N with waiting strategy is shown in Sect. 3. Section 4 describes the implementation of the hierarchical VN_8W. The experiment results are explained in Sect. 5, and Sect. 6 presents the conclusions.

2. N-Bit von Neumann Post-Processing Fundamentals

This section gives an overview of the VN_N theory proposed in [20]. Given an *i.i.d.* random binary bitstream X, p is defined as the probability of ones, and q is defined as the probability of zeros. For any consecutive N bits of X, it has 2^N combinations, which can be divided into $N+1$ groups g_k ($0 \le k \le N$), according to the Hamming weight k. Each group has $\#g_k$ ($= C_N^k$) members with a group probability of $Prob.(g_k) = p^k q^{N-k} \#g_k$. In one group (e.g., g_k), the members have identical probability. Thus, the entropy in the group is as follows:

$$H_{g_k} = -\sum \frac{1}{\#g_k} \log_2\left(\frac{1}{\#g_k}\right) = \log_2(\#g_k) = \log_2(C_N^k). \tag{1}$$

This is also the theoretical number of the random bits extracted from each group member, as summarized in Table 1. Thus, the theoretical ExE for N input bits is

Table 1 Theoretical VN_N.

Group	1s probability	Number of members	Entropy
g_0	q^N	1	0
g_1	$p^1 q^{N-1}$	N	$\log_2 N$
⋮	⋮	⋮	⋮
g_k	$p^k q^{N-k}$	C_N^k	$\log_2(C_N^k)$
⋮	⋮	⋮	⋮
g_N	p^N	1	0
total	1	2^N	$\sum_{k=0}^{N} p^k q^{N-k} C_N^k \log_2(C_N^k)$

Output		X1,X0			
		0,0	0,1	1,0	1,1
X3,X2	0,0	–,–	0,0	0,1	1,1
	0,1	1,0	1,–	1,0	0,0
	1,0	1,1	0,1	0,–	0,1
	1,1	0,0	1,1	1,0	–,–

g_0, g_4 g_1 g_3 g_2

Note: '–' is invalid.

Fig. 2 VN_4 bit assignments.

$$ExE\_Theoretical = \frac{\sum_{k=0}^{N} Prob.(g_k) H_{g_k}}{N} = \frac{\sum_{k=0}^{N} p^k q^{N-k} C_N^k \log_2(C_N^k)}{N}. \tag{2}$$

If H_{g_k} is an integer, H_{g_k} bits can be extracted. However, H_{g_k} is not always an integer. In this case, as a realistic solution, $\#g_k$ is decomposed into the power of 2s.

$$\#g_k = a_{kn} 2^n + a_{kn-1} 2^{n-1} + \cdots + a_0 2^0 = \sum_{j=0}^{n} a_{kj} 2^j, \tag{3}$$

where $a_{kn} = 1$, and $a_{kj} \in \{0, 1\}$ ($0 \le j < n$). For 2^j, j bits are assigned for each member. Therefore, the realistic ExE is given as

$$ExE\_Realistic = \frac{\sum_{k=0}^{N} \sum_{j=0}^{n} p^k q^{N-k} a_{kj} 2^j j}{N}. \tag{4}$$

For example, the 4-bit von Neumann (VN_4) bit assignment is illustrated in Fig. 2. The four consecutive input bits (X3, X2, X1, and X0) can be divided into five groups g_0, g_1, g_2, g_3, and g_4. Group members in g_0 and g_4 are assigned 0 ($= \log_2(1)$) bit; group members in g_1 and g_3 are assigned 2 ($= \log_2(4)$) bits, and group members in g_2 are assigned 2.58 ($\approx \log_2(6)$) bits in theory although we can realistically assign only an integer number of bits. Therefore, 6 is decomposed into 2^2+2^1; four members are assigned 2 bits, and two members are assigned 1 bit. Consequently, there is an ExE drop in the group g_2. Finally, the bias problem in the raw bitstream is solved by evenly assigning "0" and "1" across the output mapping table. Figure 3 summarizes the ExE for different N values without any input bias ($= |p - q|/2$). When N values increase, VN_N has a higher ExE. However,

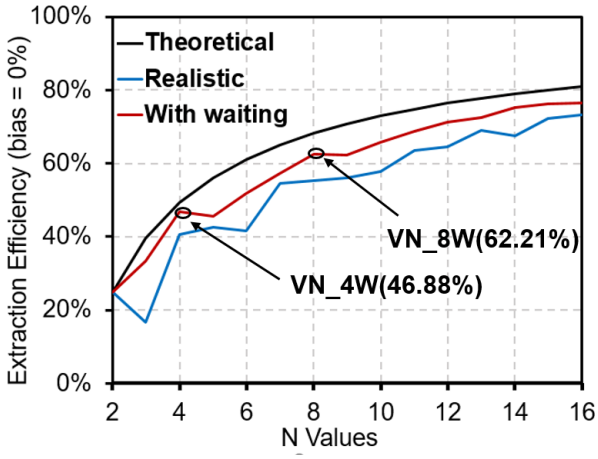

Fig. 3 Extraction efficiency versus N values.

there is a gap between the theoretical value (see black line in Fig. 3) and the realistic value (see blue line in Fig. 3).

3. N-Bit von Neumann with Waiting Strategy

To fill the ExE gap mentioned above and achieve high ExE with a small N value, we propose a waiting strategy [21]. The flow is shown in Fig. 4. For the group g_k $(0 \le k \le N)$, if $\log_2(\#g_k)$ is an integer, $\log_2(\#g_k)$ bits are assigned to each group member for the direct output. If not, $\#g_k$ is factorized into the power of 2, 2^m, and another integer b, where m is the number of direct output bits, and b is the base of the waiting flag. The waiting flag takes values ranging from 0 to $b-1$. When two waiting flags are gathered, the waiting output bits are obtained by decomposing b^2. The equations are expressed as follows:

$$\#g_k = 2^m \times b,\ b^2 = \sum_{j=0}^{n} A_{kj} 2^j, \tag{5}$$

where $A_{kn} = 1$, and $A_{kj} \in \{0,1\}$ $(0 \le j < n)$. Figure 5 presents an example of the code assignment in 4-bit von Neumann with waiting strategy (VN_4W). As compared with the no-waiting solution, the difference is the group g_2. Also, $\log_2(\#g_2) \approx 2.58$ is not an integer; therefore, $\#g_2$ is factorized as $\#g_2 = 6 = 2 \times 3$. Thus, each member is assigned one direct output bit ("1" or "0") and one waiting flag ("*0*" or "*1*" or "*2*"). Then, the two waiting flag bits generate three or zero ($3^2 = 2^3 + 2^0$) output bits. As a result, the average assigned bit count is 2.33 $[1+(3\times8)/(9\times2)]$, which is close to the ideal value of 2.58 and 1.4 times higher than the realistic assignment.

In general, the ExE of the VN_N with waiting is

$$ExE\_With\ waiting = \frac{\sum_{k=0}^{N} p^k q^{N-k} C_N^k \left(m + \frac{\sum_{j=0}^{n} \tilde{A}_{kj} 2^j j}{2b^2}\right)}{N}. \tag{6}$$

Compared with the realistic assignment, the waiting solution (see red line in Fig. 3) achieves a higher ExE for each N value (N ≥ 2). For example, VN_4W achieves 46.88% ExE, which is much higher than the conventional 6-bit von Neumann with 41.67% ExE; and the ExE of VN_8W (62.21%) approaches the conventional 12-bit von Neumann (64.63%).

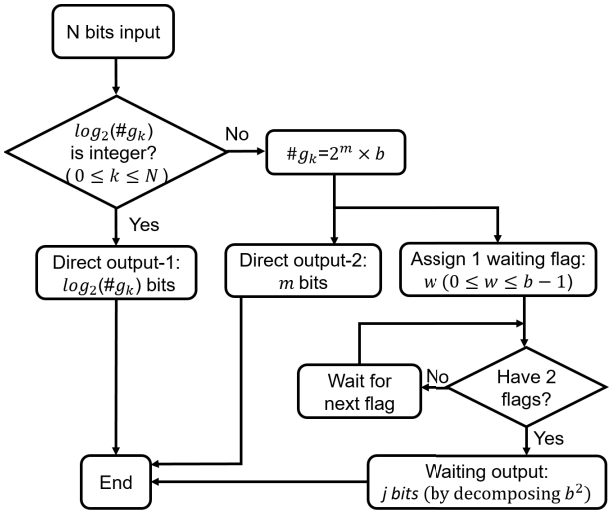

Fig. 4 Waiting strategy flowchart.

(a)

Output		X1,X0			
		0,0	0,1	1,0	1,1
X3,X2	0,0	–,–	0,0	0,1	1,*0*
	0,1	1,0	1,*2*	1,*1*	0,0
	1,0	1,1	0,*1*	0,*0*	0,1
	1,1	0,*2*	1,1	1,0	–,–

(b)

Output		w2		
		0	*1*	*2*
w1	*0*	0,0,0	0,0,1	0,1,0
	1	0,1,1	1,0,0	1,0,1
	2	1,1,0	1,1,1	–,–,–

$3^2 = 2^3 + 2^0$

g_0 , g_4 g_1 g_3 g_2

Note: '–' is invalid; the **black-colored** digit is valid output bit; the **red-colored** digit is waiting flag bit;

Fig. 5 VN_4W bit assignments: (a) first-time mapping table and (b) waiting mapping table.

4. Hierarchical 8-Bit von Neumann with Waiting Strategy

To cope with the exponentially increasing complexity in the VN_N hardware, we propose a Hamming weight mapping-based hierarchical structure. Considering a high ExE, VN_8W was implemented based on this structure.

4.1 Bit Assignment Strategy

Table 2 summarizes the overall bit assignment strategy. All the 2^8 combinations of the 8-bit input are divided into nine groups: g_0 to g_8 based on the Hamming weight. When the input belongs to the group g_1 or g_7, it is assigned three direct output bits. If the input belongs to the groups g_2, g_3, g_5, or g_6, it is assigned two or three direct output bits and one waiting flag (base-7). If the input belongs to the group g_4, it has four methods of bit assignment: a) assigning only direct output bits; b) assigning one direct output bit and one large

Table 2 VN_8W bit assignments strategy.

Group	Total members	Direct output	Waiting flag*
g_0, g_8	$1 = 2^0$	0 bit	no
g_1, g_7	$8 = 2^3$	3 bits	no
g_2, g_6	$28 = 2^2 \times 7$	2 bits	yes: base-7
g_3, g_5	$56 = 2^3 \times 7$	3 bits	yes: base-7
g_4	a) $70 = 2^6 + 2^2 + 2^1$; Adopted	6, 2, 1 bits	no
	b) $70 = 2^1 \times 35$; Rejected	1 bit	yes: base-35
	c) $70 = 2^1 \times 5 \times 7$; Rejected	1 bit	yes: base-5, base-7
	d) $70 = (2^5 + 2^1) \times 7$; Rejected	3, 1 bits	yes: base-7

* Two waiting flags construct a waiting logic: $7^2 = 49 = 2^5 + 2^4 + 2^0$.

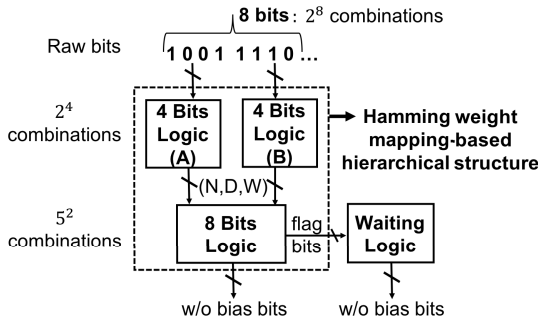

Fig. 6 Hierarchical 8-bit von Neumann with waiting strategy.

waiting flag (base-35); c) assigning one direct output bit and two waiting flags (base-5 and base-7); and d) assigning three or one direct output bits and one waiting flag (base-7) [22]. The average assigned bit counts are 5.63, 5.71, 5.05, and 4.89 for the four methods a), b), c), and d), respectively. Method b) has slightly higher ExE than the method a), but the waiting logic for base-35 is too complex. Method c) has the second smallest ExE, and it needs two kinds of waiting logics for base-5 and base-7. Method d) has the smallest ExE. Considering the tradeoff between the hardware cost and ExE, method a) is adopted. As a result, VN_8W yields a 62.21% ExE.

4.2 Hierarchical Structure

Instead of using a flat mapping with 2^8 complexity, we propose a hierarchical structure. As shown in Fig. 6, the 8-bit input is divided into two 4-bit parts and each part is fed to a 4 Bits Logic (A or B). The 4 Bits Logic has 2^4 complexity and it is an extended version of VN_4W. It generates the intermediate variables: N (Hamming weight of 4-bit), D (output bits), and W (waiting flag bits). An 8 Bits Logic block is constructed based on the Hamming weight N. NA, NB stands for the Hamming weight of the 4 Bits Logic A, B, respectively. Each has five kinds of values (0~4); therefore, the 8 Bits Logic has 5^2 complexity. Finally, the hierarchical structure enables the construction of a mapping table (2^8 combinations) by using two 4 Bits Logics (2^4 combinations) and one 8 Bits Logic (5^2 combinations). The following subsections specify the code assignments in each block.

(a)

D	X1,X0 = 0,0	0,1	1,0	1,1
X3,X2 = 0,0	–,–	0,1	1,0	–,0
0,1	1,1	–,1	–,0	1,1
1,0	0,0	–,1	–,0	0,0
1,1	–,1	0,1	1,0	–,–

g_0, g_4 g_1 g_3 g_2

(b)

D	X1,X0 = 0,0	0,1	1,0	1,1
X3,X2 = 0,0	X2,X2	X1,X0	X1,X0	X2,X2
0,1	X2,X2	X1,X0	X1,X0	X2,X2
1,0	X2,X2	X1,X0	X1,X0	X2,X2
1,1	X2,X2	X1,X0	X1,X0	X2,X2

D = (X1 == X0) ? {X2,X2} : {X1,X0};

Note: '–' is invalid, it can be replaced by the blue-colored symbol

Fig. 7 D mapping table in the 4 Bits Logic: (a) actual binary codes; (b) input-symbol-based codes.

(a)

W	X1,X0 = 0,0	0,1	1,0	1,1
X3,X2 = 0,0	–,–	–,–	–,–	0,0
0,1	–,–	0,1	0,1	–,–
1,0	–,–	1,0	1,0	–,–
1,1	0,0	–,–	–,–	–,–

g_0, g_4 g_1 g_3 g_2

(b)

W	X1,X0 = 0,0	0,1	1,0	1,1
X3,X2 = 0,0	0,0	0,0	0,0	0,0
0,1	X3,X2	X3,X2	X3,X2	X3,X2
1,0	X3,X2	X3,X2	X3,X2	X3,X2
1,1	0,0	0,0	0,0	0,0

W = (X3 == X2) ? {0,0} : {X3,X2};

Note: '–' is invalid, it can be replaced by the blue-colored symbol

Fig. 8 W mapping table in the 4 Bits Logic: (a) actual binary codes; (b) input-symbol-based codes.

4.2.1 4 Bits Logic

The 4 Bits Logic processes a 4-bit input and generates the Hamming weight N, the output bits D of VN_4W, and the waiting flag bits W of VN_4W. N is generated by a 4-bit adder. D and W are generated according to the mapping tables. Instead of using the actual binary code assignments shown in Fig. 7 (a) and Fig. 8 (a), in which we need to assign bits for 2^4 mapping cases one by one, we propose an input-symbol-based code assignment method, as shown in Fig. 7 (b) and Fig. 8 (b). The blue bits are invalid and do not affect the output. The final output values are the same in the above two methods, but the input-symbol-based code assignment can simplify the mapping cases and reduce the logic cost. There are only two output patterns in Fig. 7 (b) and Fig. 8 (b), where cell colors are coded according to the output patterns. They can be easily described in the Verilog HDL codes using the conditional operator:

```
D = (X1 == X0) ? {X2,X2} : {X1,X0};
W = (X3 == X2) ? {0,0} : {X3,X2};
```

Considering the invalid bits, the judging process is only needed in the 8 Bits Logic for the final valid output.

4.2.2 8 Bits Logic

Figure 9 shows the overall bit assignment strategy in the 8 Bits Logic. The input vectors are 4-bit Hamming weight information generated by the two 4 Bits Logics (A & B); the upper value in each cell shows the binary form of the Hamming weight NA/NB, and the lower value shows the number of group members, for example, when NA ="0,0,1", this group has 4 ($= C_4^1$) members, which is denoted by <4>. In the output cell, the upper value is the number of direct output bits with or without a waiting flag (as denoted by '+w'); the lower value indicates the number of cell members. The members in one group have the same color. This

Bit Assignments		NA (NA2,NA1,NA0) 0,0,0 <1>	0,0,1 <4>	0,1,0 <6>	0,1,1 <4>	1,0,0 <1>	
NB (NB2, NB1, NB0)	0,0,0 <1>	0 bit <1>	3 bits <4>	2 bits + w <6>	3 bits + w <4>	1 bit <1>	g_4
	0,0,1 <4>	3 bits <4>	2 bits + w <16>	3 bits + w <24>	6 bits <16>	3 bits + w <4>	g_5
	0,1,0 <6>	2 bits +w <6>	3 bits + w <24>	6 or 2 bits <32+4>	3 bits + w <24>	2 bits + w <6>	g_6
	0,1,1 <4>	3 bits + w <4>	6 bits <16>	3 bits + w <24>	2 bits + w <16>	3 bits <4>	g_7
	1,0,0 <1>	1 bit <1>	3 bits + w <4>	2 bits + w <6>	3 bits <4>	0 bit <1>	g_8

Note: +w means assigning the waiting flag.

$\#g_0 = \#g_8 = C_8^0 = 1$

$\#g_1 = \#g_7 = C_8^1 = 8 = 4 + 4$

$\#g_2 = \#g_6 = C_8^2 = 28 = 6 + 16 + 6$

$\#g_3 = \#g_5 = C_8^3 = 56 = 4 + 24 + 24 + 4$

$\#g_4 = C_8^4 = 70 = 1 + 16 + 36 + 16 + 1$

Fig. 9 Bit assignments strategy in the 8 Bits Logic.

DOUT		NA (NA2,NA1,NA0) 0,0,0 <1>	0,0,1 <4>	0,1,0 <6>	0,1,1 <4>	1,0,0 <1>
NB (NB2, NB1, NB0)	0,0,0 <1>	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB2
	0,0,1 <4>	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1
	0,1,0 <6>	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1		NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1
	0,1,1 <4>	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1
	1,0,0 <1>	NB0,DA1,DA0, DB1,DB0,NB2	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1	NB0,DA1,DA0, DB1,DB0,NB1

Note: blue-colored bit is invalid.

		WA (WA1,WA0) 0,0	0,1	1,0
WB (WB1,WB0)	0,0	NB0,WA0,DA0 WB0,DB0,NB1	NB0,WA0,DA0 WB0,DB0,NB1	NB0,WA0,DA0 WB0,DB0,NB0
	0,1	NB0,WA0,DA0 WB0,DB0,NB1	NB0,WA0,DA0 WB0,DB0,NB1	NB0,WA0,DA0 WB0,DB0,NB0
	1,0	NB0,WB1,DA0 WA0,DB0,NB0	NB0,WB1,DA0 WA0,DB0,NB0	NB0,WA0,DA0 WB0,DB0,NB0

Fig. 10 DOUT mapping table in the 8 Bits Logic.

mapping covers all 2^8 group cases in Table 2 and has only 5^2 complexity. The output of the 8 Bits Logic can be realized by using the intermediate variables: DA/DB, WA/WB, and NA/NB.

The input-symbol-based code assignment of the output bits, DOUT, is shown in Fig. 10. To indicate whether the output bit is valid or not, DVALID mapping table is needed, which is shown in Fig. 11. Because of the input-symbol-based code assignment, each symbol output bit may represent one of the two kinds of values "1" or "0" with equal probability. For example, for the column NA = "0,0,1", "DA1,DA0" indicates the values from "0,0" to "1,1" for all four combinations. Cells of the same color are assigned the same codes. Thus, DOUT can be described using only five patterns with the help of DVALID mapping table. Likewise, the waiting flag bits, DWAIT also can be described using four patterns, as shown in Fig. 12. To indicate the valid DWAIT, it only needs to judge the value of NA + NB. If the result is equal to 2, 3, 5, 6 in decimal, i.e., the 8-bit input belongs to g_2, g_3, g_5, g_6, then DWAIT is valid, and vice versa.

DVALID		NA 0,0,0 <1>	0,0,1 <4>	0,1,0 <6>	0,1,1 <4>	1,0,0 <1>
NB (NB2, NB1, NB0)	0,0,0 <1>	0,0,0, 0,0,0	1,1,1, 0,0,0	0,0,1, 0,0,1	1,1,1, 0,0,0	0,0,0, 0,0,1
	0,0,1 <4>	1,0,0, 1,1,0	0,1,1, 0,0,0	1,0,0, 1,1,0	1,1,1, 1,1,1	1,0,0, 1,1,0
	0,1,0 <6>	0,0,0, 0,1,1	1,1,1, 0,0,0		1,1,1, 0,0,0	0,0,0, 0,1,1
	0,1,1 <4>	1,0,0, 1,1,0	1,1,1, 1,1,1	1,0,0, 1,1,0	0,1,1, 0,0,0	1,0,0, 1,1,0
	1,0,0 <1>	0,0,0, 0,0,1	1,1,1, 0,0,0	0,0,1, 0,0,1	1,1,1, 0,0,0	0,0,0, 0,0,0

		WA 0,0	0,1	1,0
WB (WB1,WB0)	0,0	1,1,1, 1,1,1	1,1,1, 1,1,1	1,1,1, 1,1,1
	0,1	1,1,1, 1,1,1	1,1,1, 1,1,1	1,1,1, 1,1,1
	1,0	1,1,1, 1,1,1	1,1,1, 1,1,1	0,0,1, 0,1,0

Fig. 11 DVALID mapping table in the 8 Bits Logic.

DWAIT		NA (NA2,NA1,NA0) 0,0,0 <1>	0,0,1 <4>	0,1,0 <6>	0,1,1 <4>	1,0,0 <1>
NB (NB2, NB1, NB0)	0,0,0 <1>	0,DA1,DA0	0,DA1,DA0	1,WA1,WA0	0,DA1,DA0	0,DA1,DA0
	0,0,1 <4>	0,DB1,DB0	0,DB1,DB0	0,DB1,DB0	0,DB1,DB0	0,DB1,DB0
	0,1,0 <6>	1,WB1,WB0	1,WB1,WB0	1,WB1,WB0	1,WB1,WB0	1,WB1,WB0
	0,1,1 <4>	0,DB1,DB0	0,DB1,DB0	0,DB1,DB0	0,DB1,DB0	0,DB1,DB0
	1,0,0 <1>	0,DA1,DA0	0,DA1,DA0	1,WA1,WA0	0,DA1,DA0	0,DA1,DA0

Note: blue-colored bit is invalid.

Fig. 12 DWAIT mapping table in the 8 Bits Logic.

DOUT_WAIT		DWAIT_2 (W2,W1,W0) 0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0
DWAIT_1 (W5, W4, W3)	0,0,0	W1,W0,W5, W4,W3 <28>				W1,W0,W5, W4,W3 <14>		W4,W3,W2, W2,W2 <4>
	0,0,1							
	0,1,0							
	0,1,1							
	1,0,0							W4,W3,W2, W2,W2 <2>
	1,0,1							
	1,1,0							W4,W3,W2, W2,W2 <1>

$7^2 = 49 = 2^5 + 2^4 + 2^0 = (28 + 4) + (14 + 2) + 1$

Note: blue-colored bit is invalid.

Fig. 13 DOUT_WAIT mapping table in the Waiting Logic.

4.2.3 Waiting Logic

The Waiting Logic is shared for all base-7 waiting flags in g_2, g_3, g_5, and g_6. Based on the decomposition of 7^2, 32 input cases are assigned five nonoverlapping bits, 16 input cases are assigned four nonoverlapping bits, and 1 input case is assigned zero bit. During the process "0" and "1" are evenly assigned; therefore, the waiting output can be used as a no-bias bitstream. Because of the input-symbol-based code assignment, 49 cases of bit assignments in DOUT_WAIT are simplified to two patterns, as shown in Fig. 13. The related valid mapping table, DVALID_WAIT, is shown in Fig. 14.

DVALID_WAIT		DWAIT_2									
		0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0			
DWAIT_1 (W5, W4, W3)	0,0,0	1,1,1, 1,1				0,1,1, 1,1		1,1,1, 1,1			
	0,0,1										
	0,1,0										
	0,1,1										
	1,0,0							0,1,1, 1,1			
	1,0,1										
	1,1,0							0,0,0, 0,0			

Fig. 14 DVALID_WAIT mapping table in the Waiting Logic.

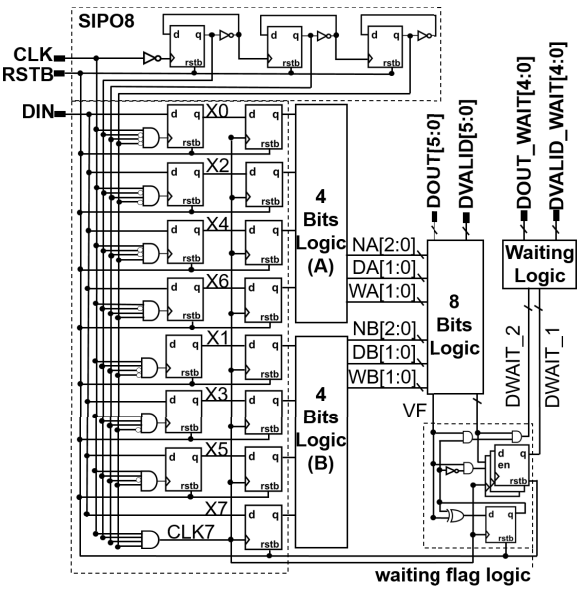

Fig. 15 Logic structure of VN_8W.

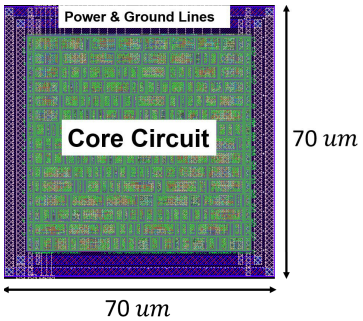

Fig. 16 Layout image of VN_8W.

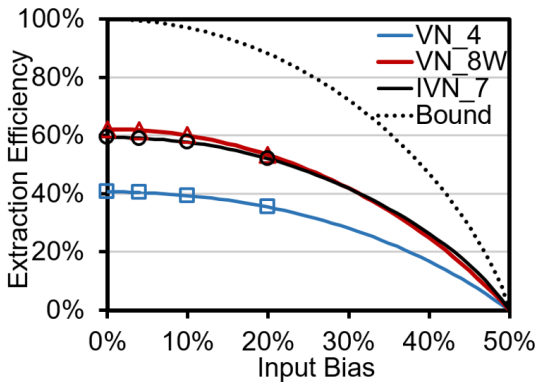

Fig. 17 Extraction efficiency versus input bias: the solid lines show the expected values; the hollow points show the measured values.

4.3 CMOS Logic Implementation

Figure 15 shows the logic structure of VN_8W. The input signals are DIN (for raw random data), CLK (for clock), and RSTB (for resetting registers when RSTB = 0), and the output signals include DOUT[5:0], DVALID[5:0], DOUT_WAIT[4:0], and DVALID_WAIT[4:0]. In addition to the 4 Bits Logic, 8 Bits Logic, and Waiting Logic, VN_8W also includes a logic to store the valid waiting flag bits (VF signal represents whether the waiting flag is valid) and serial-in parallel-out logic SIPO8. By using the gated clock CLK7, the mapping logic is triggered every 8 clock cycles. Thus, the dynamic power caused by the rate of logic transitions can be reduced.

This hierarchical structure enables the scrambling of the input bits without any hardware overhead. The consecutive 8-bit input is separated into parts with odd and even order and fed into two 4 Bits Logics. This can reduce the most significant correlation, which often occurs at lag 1 in the autocorrelation check. The correlations are further masked by the Hamming weight information-based 8 Bits Logic. The decorrelation ability will be discussed in Sect. 5.

For the functional verification and power analysis, VN_8W was designed and fabricated using a 130-nm standard cell through ASIC design flow. Figure 16 shows the layout image of VN_8W. The core circuit costs 381 gate equivalents (GEs). The total area including core circuit, the power and ground lines is 4900 um^2. This is around 1.25 times larger than IVN_7 with 59.38% ExE designed using the same ASIC design flow.

5. Experiment Results

For comparison, VN_4 with 40.63% ExE, and IVN_7 with 59.38% ExE [19] were also designed and fabricated in 130-nm CMOS. VN_4 and IVN_7 cost 66 GEs and 204 GEs, respectively.

5.1 Extraction Efficiency and Randomness Check for Biased Data

As the test data, one 3M-long bitstream generated by the inverter-based TRNG in [11] with 4% bias and three 3M-long bitstreams generated by a MATLAB algorithm with 0%, 10%, and 20% biases, respectively, were used. The graph for ExE versus input bias is shown in Fig. 17. All the measured points follow the expected values well. The randomness of the post-processed bitstreams was tested by the National Institute of Standards and Technology (NIST) SP 800-22 statistical test suite [23]. Each test bitstream is 1M-long. All the post-processed bitstreams passed all 15 NIST tests.

5.2 Autocorrelation and Randomness Check for Correlated Data

To verify the de-correlation ability, a 3M-long correlated bitstream generated by the ARIMA [24] model was used as the test data. Figure 18 shows the autocorrelation results of 1M bits with lags of 1 to 10. The autocorrelation factor in the raw bitstream is 0.033 at lag 1 (i.e., 15 times the confidence boundary) and 0.007 at lag 2, and the factor is around the boundary after lag 2. The correlated data is post-processed through VN_4, IVN_7, LFSR_16, and VN_8W. Each 1M of the post-processed data was also applied to the autocorrelation check. VN_8W and LFSR_16 reduce the factor in the raw bitstream at lag 1 to around the boundary. VN_4 and IVN_7 can reduce the factor, but the factor remains beyond the boundary. The NIST test results are listed in Table 3; only the VN_8W post-processed bitstream passed all the terms, because of the de-correlation ability by the hierarchical design.

5.3 Power and Energy

Four test chips were measured at room temperature at the frequencies of 1 MHz, 5 MHz, 10 MHz, and 50 MHz. Figure 19 illustrates the average value of the minimum operating voltage under each frequency condition. The mapping logic in VN_8W is triggered by a gated clock CLK7 every 8 clock cycles. Therefore, not only the dynamic power was reduced but also the delay issue was not significant even at low voltage. At 1 MHz, the minimum operating voltage of VN_4 and VN_8W was 0.45 V, whereas, for IVN_7, the minimum operating voltage was 0.7 V. With the increase in frequency, the operating voltage of VN_4 becomes slightly higher than that of VN_8W, and IVN_7 always need the largest operating voltage.

Figure 20 shows the energy per unbiased bit E_{VN} for three circuits with operational voltages ranging from 0.45 V to 1.2 V. The E_{VN} equation is as follows:

$$E_{VN} = \frac{Power}{Frequency \times ExE}. \tag{7}$$

For bias = 3% of the input bitstream, the ExE values for VN_4, VN_8W, and IVN_7 are 40.51%, 62.01%, and 59.23%, respectively. The results show that the E_{VN} of each circuit increases with the operational voltage. At 0.45 V

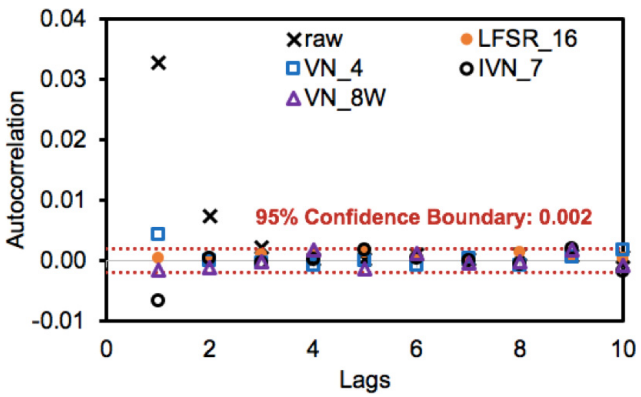

Fig. 18 Autocorrelation of 1M bits with 10 lags.

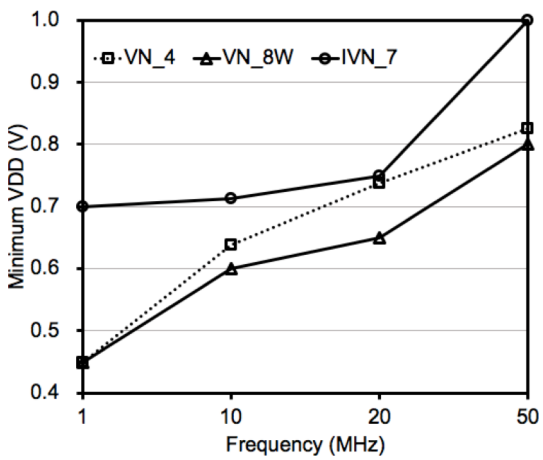

Fig. 19 Average minimum operating voltage versus frequency.

Table 3 NIST SP 800-22 test results of correlated raw data and post-processed data.

	Raw		VN_4		IVN_7		LFSR_16		VN_8W	
	P Val.	Pass?	P Val.	Pass?	P Val.	Pass?	P Val.	Pass?	P Val.	Pass?
Frequency	0.54	Yes	0.94	Yes	0.58	Yes	0.25	Yes	1.00	Yes
Block Frequency	0.00	**No**	0.11	Yes	0.55	Yes	0.33	Yes	0.29	Yes
Runs	0.00	**No**	0.00	**No**	0.00	**No**	0.64	Yes	0.13	Yes
Longest Runs	0.01	Yes	0.11	Yes	0.47	Yes	0.05	Yes	0.05	Yes
Binary Matrix Rank	0.23	Yes	0.55	Yes	0.31	Yes	0.06	Yes	0.79	Yes
FFT	0.51	Yes	0.80	Yes	0.99	Yes	0.59	Yes	0.96	Yes
Non-Overlapping Template	0.15	Yes	0.48	Yes	0.46	Yes	0.50	Yes	0.43	Yes
Overlapping Template	0.00	**No**	0.85	Yes	0.76	Yes	0.85	Yes	0.20	Yes
Universal Statistical	0.01	Yes	0.51	Yes	0.82	Yes	0.30	Yes	0.72	Yes
Linear Complexity	0.23	Yes	0.42	Yes	0.55	Yes	0.79	Yes	0.92	Yes
Serial	0.08	Yes	0.88	Yes	0.44	Yes	0.60	Yes	0.38	Yes
Approximate Entropy	0.00	**No**	0.54	Yes	0.95	Yes	0.36	Yes	0.73	Yes
Cumulative Sums	0.81	Yes	0.46	Yes	0.81	Yes	0.40	Yes	0.71	Yes
Random Excursions	0.34	Yes	0.59	Yes	0.52	Yes	0.00	**No**	0.66	Yes
Random Excursions Variant	0.73	Yes	0.51	Yes	0.21	Yes	0.00	**No**	0.70	Yes

Table 4 Comparison with prior post-processing techniques.

	N-bit von Neumann		Iterated von Neumann			Finite Field Theory-based & Block Cipher		
	This work		[19] HOST 2016	[9] SSCL 2018	[10] JSSC 2019	[6] JSSC 2016	[16] TCAS-II 2019	[7] TCAS-I 2015
	VN_4	**VN_8W**	IVN_7	Markov + IVN_16 + LFSR	Hierarchical VN	Decorrelators + BIW	Strong Blenders	PRESENT
Process Technology	**130-nm CMOS**	**130-nm CMOS**	130-nm[a] CMOS	65-nm CMOS	ARRIA FPGA	14-nm CMOS	45-nm[b] NanGate	32-nm[b] CMOS PTM
Gate Equivalent (GE)	**66**	**381**	204[a]	N/A	750[c]	586	166.3 ~13K[d]	1171
Max. Extraction Efficiency	**40.63%**	**62.21%**	59.38%	≈ 78%[e]	43.75%	12.5%	4%~20%[d]	50%/80%[d]
De-Biasing	**Yes**	**Yes**	Yes	Yes	Yes	Yes	Yes	Yes
De-Correlation	**No**	**Yes**	No	Yes	No	Yes	Yes	Yes
Energy per full-entropy bit (pJ/bit)	**0.18 @0.45 V**	**0.18 @0.45 V**	0.73 @0.70 V[a]	2.58 @0.53 V[f]	N/A	9 @0.75 V	N/A	1.0~2.5[f] @0.9 V

[a] Implemented in this work. The original work does not show this data. The GE count is based on the final layout circuit.
[b] Only for Synthesis/Simulation.
[c] Gate counts.
[d] Including several versions.
[e] Read from Fig. 6 (a) in [9].
[f] Including TRNG cores.

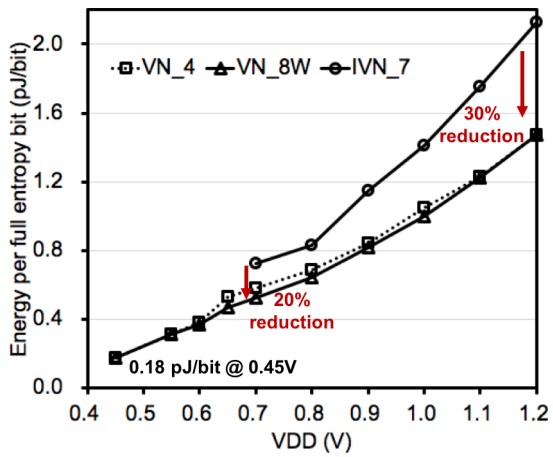

Fig. 20 Energy per full entropy bit versus operating voltage.

and 1 MHz, VN_4 and VN_8W have the minimum energy of 0.18 pJ/bit. The minimum energy for IVN_7 is 0.73 pJ/bit at 0.7 V, 1 MHz. VN_8W has nearly the same or slightly less energy than VN_4. When compared with IVN_7, even though VN_8W costs larger GEs and area, it achieves 20% to 30% energy reduction at the same operating voltage thanks to the clock gating with CLK7.

Let us consider a TRNG system that consists of a TRNG core and a von Neumann post-processing circuit. The energy consumption of the TRNG core for its raw output bitstream is denoted by E_{TRNG}. For unbiased random bitstream after post-processing, the system needs 1/ExE times larger energy. Thus, the total energy of the system can be expressed as follows:

$$E_{Total} = \frac{E_{TRNG}}{ExE} + E_{VN}. \quad (8)$$

VN_8W has the largest ExE and smallest E_{VN} and is the best choice for a TRNG system.

5.4 Comparisons

Table 4 shows the comparison with the previous post-processing techniques. VN_4 costs the smallest number of GEs but has a moderate ExE. Compared with IVN based methods [9], [10], [19], VN_8W achieves a comparable ExE and consumes the least energy. Compared with finite field theory-based and block cipher methods in [6], [7], [16], VN_8W has the smallest GEs, and it can also address the bias and correlation problems.

6. Conclusions

We proposed a waiting strategy and hierarchical structure to improve the VN_N post-processing technique. A hierarchical VN_8W with 62.21% ExE was prototyped in 130-nm CMOS and achieved the minimum energy of 0.18 pJ/bit at 0.45 V. The hierarchical structure design enables the decorrelation ability in VN_8W and solves the limitation of the VN based techniques to eliminate only the bias.

Acknowledgments

This work is supported by ROHM Co., Ltd. The authors would like to thank Mr. Koichi Takanashi from ROHM Co., Ltd. for technical supports during the back-end ASIC design process.

References

[1] B. Sunar, "True random number generators for cryptography," Cryptographic Engineering, pp.55–73, Springer, Boston, MA, 2009.
[2] V. Fischer, "A closer look at security in random number generators design," Proc. Int. Workshop on Constructive Side-Channel Analysis and Secure Design, pp.167–182, April 2012.
[3] K. Yang, D. Blaauw, and D. Sylvester, "Hardware design for security

in ultra-low-power IoT systems: An overview and survey," IEEE Micro, vol.37, no.6, pp.72–89, Nov. 2017.
[4] M. Kim, U. Ha, K.J. Lee, Y. Lee, and H.-J. Yoo, "A 82-nW chaotic map true random number generator based on a sub-ranging SAR ADC," IEEE J. Solid-State Circuits, vol.52, no.7, pp.1953–1965, July 2017.
[5] A.T. Do and X. Liu, "25 fJ/bit, 5Mb/s, 0.3 V true random number generator with capacitively-coupled chaos system and dual-edge sampling scheme," Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC), pp.61–64, Nov. 2017.
[6] S.K. Mathew, D. Johnston, S. Satpathy, V. Suresh, P. Newman, M.A. Anders, H. Kaul, A. Agarwal, S.K. Hsu, G. Chen, and R.K. Krishnamurthy, "μRNG: A 300–950 mV, 323 Gbps/W all-digital full-entropy true random number generator in 14 nm FinFET CMOS," IEEE J. Solid-State Circuits, vol.51, no.7, pp.1695–1704, July 2016.
[7] V.B. Suresh and W.P. Burleson, "Entropy and energy bounds for metastability based TRNG with lightweight post-processing," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.62, no.7, pp.1785–1793, July 2015.
[8] J. Kim, H. Nili, N.D. Truong, T. Ahmed, J. Yang, D.S. Jeong, S. Sriram, D.C. Ranasinghe, S. Ippolito, H. Chun, and O. Kavehei, "Nano-intrinsic true random number generation: A device to data study," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.66, no.7, pp.2615–2626, July 2019.
[9] V.R. Pamula, X. Sun, S.M. Kim, F. ur Rahman, B. Zhang, and V.S. Sathe, "A 65-nm CMOS 3.2-to-86 Mb/s 2.58 pJ/bit highly digital true-random-number generator with integrated de-correlation and bias correction," IEEE Solid-State Circuits Lett., vol.1, no.12, pp.237–240, Dec. 2018.
[10] S.K. Satpathy, S.K. Mathew, R. Kumar, V. Suresh, M.A. Anders, H. Kaul, A. Agarwal, S. Hsu, R.K. Krishnamurthy, and V. De, "An all-digital unified physically unclonable function and true random number generator featuring self-calibrating hierarchical von Neumann extraction in 14-nm tri-gate CMOS," IEEE J. Solid-State Circuits, vol.54, no.4, pp.1074–1085, April 2019.
[11] X. Wang, H. Liu, R. Zhang, K. Liu, and H. Shinohara, "An inverter-based true random number generator with 4-bit von-Neumann post-processing circuit," Proc. IEEE 63rd Int. Midwest Symp. on Circuits & Systems (MWSCAS), pp.285–288, Aug. 2020.
[12] M. Dichtl, "Bad and good ways of post-processing biased physical random numbers," Proc. Int. Workshop on Fast Software Encryption (FSE), pp.137–152, 2007.
[13] P. Lacharme, "Post-processing functions for a biased physical random number generator," Proc. Int. Workshop on Fast Software Encryption (FSE), pp.334–342, 2008.
[14] S.-H. Kwok, Y.-L. Ee, G. Chew, K. Zheng, K. Khoo, and C.-H. Tan, "A comparison of post-processing techniques for biased random number generators," Proc. IFIP Int. Workshop on Information Security Theory and Practices, pp.175–190, 2011.
[15] P.A. Samuelson, "Constructing an unbiased random sequence," J. American Statistical Association, vol.63, no.324, pp.1526–1527, 1968.
[16] V. Rožić and I. Verbauwhede, "Hardware-efficient post-processing architectures for true random number generators," IEEE Trans. Circuits Syst. II, Express Briefs, vol.66, no.7, pp.1242–1246, July 2019.
[17] J.V. Neumann, "Various techniques used in connection with random digits," Monte Carlo Method, National Bureau of Standards Applied Mathematics Series, vol.12, pp.36–38, 1951.
[18] Y. Peres, "Iterating von Neumann's procedure for extracting random bits," Ann. Statist., vol.20, no.1, pp.590–597, March 1992.
[19] V. Rožić, B. Yang, W. Dehaene, and I. Verbauwhede, "Iterating von Neumann's post-processing under hardware constraints," IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST), pp.37–42, May 2016.
[20] P. Elias, "The efficient construction of an unbiased random sequence," Ann. Math. Statist., vol.43, no.3, pp.865–870, June 1972.
[21] R. Zhang, S. Chen, C. Wan, and H. Shinohara, "High-throughput von Neumann post-processing for random number generator," Proc. Int. Symp. on VLSI Design, Automat. and Test (VLSI-DAT), pp.1–4, April 2018.
[22] R. Zhang et al., "High-throughput & power efficiency 8 bits von Neumann post-processing with waiting strategy for true random number generators," Proc. Taiwan and Japan Conf. on Circuits and Systems (TJCAS), Aug. 2019.
[23] "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications," Nat. Inst. Standards Technol. (NIST), Pub 800–22, 2010.
[24] MathWorks Help Center, "Arima," https://www.mathworks.com/help/econ/arima.html, accessed Oct. 14. 2020.

Ruilin Zhang received the B.S. degree from Beijing University of Chemical Technology, Beijing, China, in 2015, and the M.S. degree from Waseda University, Fukuoka, Japan, in 2017, where she is currently pursuing the Ph.D. degree. Her current research interests include true random number generators (TRNGs) design, TRNG post-processing, and hardware security.

Xingyu Wang received the B.S. degree in advanced energy material and devices from Southeast University, Nanjing, China, in 2018, and the M.S. degree in electrical engineering from Waseda University, Fukuoka, Japan, in 2019. He is currently a Ph.D. candidate at Waseda University. His research interests include hardware security and mixed-signal VLSI design.

Hirofumi Shinohara received B.S. and M.S. degrees in electrical engineering and Ph.D. degree in informatics from Kyoto University, in 1976, 1978, and 2008, respectively. In 1978, he joined the LSI Laboratory of Mitsubishi Electric Corporation, where he was involved in research and development of MOS SRAMs, memory compilers, logic building blocks and neurochips. From 2003 to 2009 he was engaged in development of basic logic circuits, memory macros and design methodology for advanced CMOS technologies in Renesas Technology Corporation. He moved to Semiconductor Technology Research Academic Center (STARC) in 2009, and directed a joint research project on extremely low power circuits and systems that operate near/sub-threshold region with universities in Japan. Since 2015 he has been a professor of Graduate School of Information, Production and Systems, Waseda University, Kitakyushu Japan. His current research interests include energy-efficient random circuits for security such as physical unclonable functions, true random number generator as well as low power analog circuits. He has received an invention award for semiconductor memory in 1989 and a best paper award in ICCD 1993 for fast multiplier. Dr. Shinohara has been serving on the International Technical Program Committee of the IEEE International Solid-State Circuits (ISSCC) and the IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT) since 2017 and 2016, respectively.