High-Throughput Von Neumann Post-Processing for Random Number Generator

Ruilin Zhang[1], Sijia Chen, Chao Wan, Hirofumi Shinohara
Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Fukuoka, Japan
zrlsmile@asagi.waseda.jp[1]

***Abstract*—This paper presents the improvement and implementation of *N* bits Von Neumann (VN_N) post-processing technique, which is used to produce unbiased random bits sequence from biased one. Algorithm to realize general *N* bits VN_N and circuit level implementation of 4 bits VN_4 are shown. VN_4 achieved 40.6% output rate. A waiting strategy is further proposed to improve the output rate. VN_4+waiting and VN_8+waiting reached to 46.9% and 62.5% output rate, respectively. They are 1.88x and 2.50x improvements compared with original Von Neumann (VN_2) with 25.0%, respectively.**

I. INTRODUCTION

In IoT (Internet of Things) era, information security and cryptography system have become critically important. High randomness random numbers are the foundation of cryptographic system, which are often used in secure key generation, digital signature schemes and session IDs. The random number generated by algorithm is called pseudo random number. Pseudo random number has good statistical properties but has periodic problem. Another way is so-called true random number generator, where physical entropy source is used to generate the unpredictable random numbers. However, as mentioned in [1,2], due to the instability of the circuits post-processing is needed to attain high statistical characteristic.

Von Neumann post processing is first proposed in [3]. By discarding some information in the biased input bits, it can output unbiased random bits. However, the highest output rate is limited to 25.0% of the input sequence length. Iterating Von Neumann (IVN) method is proposed in [5]. Through infinity times of iterations of IVN, one can reach output rate close to the Shannon entropy bound. Under the limitation of circuit resource, paper [7] proposes the uncompleted binary tree method to attain optimal output efficiency. Paper [4] first proposes *N* bits Von Neumann (VN_N) method and gives the output efficiency. It should be mentioned that with the increase of *N* values, one can extract all entropy contained in the input sequence with only 1 time VN_N theoretical post processing. However, there is an output rate gap between theoretical value and realistic value.

As shown in Fig. 1(a) and (b), the original Von Neumann process 2 input bits each time, while *N* bits Von Neumann process *N* input bits each time. Both of them output unbiased bits. *N* bits Von Neumann just enlarges the processing input bits. From this perspective, we merge the original 2 bits Von Neumann into *N* bits Von Neumann (VN_N) and call it VN_2. The contributions of this work are listed in the following:

1. We implemented VN_4 both in algorithm and logic gate level. A waiting strategy is further proposed. Experiment results show that VN_4+waiting reached to 46.9% output rate. VN_8+waiting reached to 62.5% output rate.

2. For any *N* value in VN_N, we present a mapping algorithm which can be used to generate the lookup table. The merits of this work exits in large *N* value.

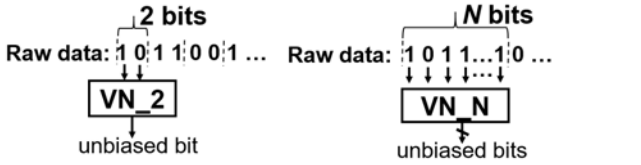

Fig.1. Von Neumann Post Processing

The rest of the paper is organized in the following order. Section II shows the overview of Iterating Von Neumann and *N* bits Von Neumann. Section III introduces the implementation on *N* bits Von Neumann. Section IV lists the experiment results. Section V sums up the conclusions.

II. ITERATING VON NEUMANN AND *N* BITS VON NEUMANN

We assume a raw bit sequence *x1,x2,x3,...xi,...,xm* with bit length *m*, which follows *i.i.d.* (independent and identically distributed). Each *xi* is the value of 0 or 1. For all *m* bits, the probability of ones is *p*, the probability of zeros is *q*, *q* = 1- *p*. We define a bias *e* in the raw sequence as follows:

$$e = \frac{|p-q|}{2}, \quad 0 \le e \le 0.5 \tag{1}$$

In the subsection, we will show the overview of Iterating Von Neumann and *N* bits Von Neumann.

A. Iterating Von Neumann (IVN)

According to [5] IVN (Iterating Von Neumann) is defined as follows: after the first stage VN_2 post processing, IVN collects the discarded information (XOR and R in Fig. 2 and Fig. 3) and reuses them by sending them to the next stage VN_2. Fig. 2 shows the example and Fig. 3 shows the structure. Each IVN module contains three process blocks: VN_2 (if input pair is '01' or '10' then output '0' or '1', others no output), XOR (if '00' or '11' pairs occur then output '0', '01' or '10' pairs occur then output '1'), R (if '00' then output '0', if '11' then output '1', others no output). VN_2 has output rate at $25.0\% - e^2$ without bias, it can be directly output. XOR has 50.0% output rate with $2e^2$ bias. R has $25.0\% + e^2$ output rate with $2e/(1+4e^2)$ bias.

The data after XOR and R post processing are used as the raw data for the next iteration stage of VN_2. By adding more iteration stages the output rate increases. However, due to the R processing bias further increased in the next iteration module, the efficiency decreased further. For example, if the initial bias in raw data is 10%, after 2nd iteration, the generated R sequence bias in M3 (Module 3 in Fig. 3) is more than 30.0%, so the output rate of the further VN_2 is about 0%.

978-1-5386-4260-3/18/$31.00 ©2018 IEEE

Raw data	00	01	10	00	11	10	01	10	11	01	…
VN_2		0	1			1	0	1		0	…
XOR	0	1	1	0	0	1	1	1	0	1	…
R	0			0	1				1		…

Fig. 2. IVN Mapping Example

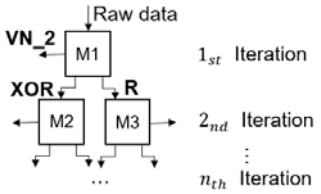

Fig. 3. IVN Structure

B. N bits Von Neumann (VN_N)

VN_N is defined as follows[4]: Given N bits sequence, it has 2^N possible input permutations. This method further divides all the permutations into N+1 groups: S_k , k from 0 to N . k is the number of ones in the permutation group. Since probability of permutations ($p^k q^{N-k}$) are same within a S_k group, we can assign bits without bias. The number of the assigned bits for each member within the S_k group is expressed as $\log_2 C_N^k$, C_N^k is the number of members $\#S_k$. It must be noted that this is an ideal maximum number. Table. 1 summarizes VN_N. Theoretical output rate is given in the following formula:

$$\text{Theoretical Rate: } \sum_{k=0}^{N} C_N^k \log_2(C_N^k)\ p^k q^{N-k} / N \tag{2}$$

For the realistic bit mapping, C_N^k is decomposed as follows:

$$C_N^k = a_{kn}2^n + a_{kn-1}2^{n-1} + \ldots + a_{k0}2^0 = \sum_{j=0}^{n} a_{kj}2^j \tag{3}$$

The binary expansion coefficient $a_{kn} = 1$, a_{ki} is 1 or 0, $0 \le i < n$. For all $a_{kj} \ne 0, 0 \le j \le n$, the mapping rule of VN_N is assigning j bits different codes to each permutation member. When k = 0 or N, C_N^k = 1, there are no assign bits. VN_N realistic output rate is given in the following formula:

$$\text{Realistic Rate: } \sum_{k=0}^{N} \sum_{j=0}^{n} a_{kj}2^j\, jp^k q^{N-k} / N \tag{4}$$

By increasing the value of N, VN_N can get high output rate close to the Shannon entropy bound with only 1 times processing. That is to say, for any raw sequence generated by TRNG, we can output as much entropy as we can through designing the N value. However, large N needs large circuit resource. Fig. 4 shows the theoretical output rate and realistic output rate as the function of N value. Theoretical output rate is 49.2%, 68.2%, 81.0% when N = 4, 8, 16, respectively. While the realistic output rate is 40.6%, 55.2%, 73.1%, respectively. Meanwhile, beyond N = 17 the efficiency becomes saturated. So we need to select the appropriate N values.

Table. 1. Summary of VN_N

Group: S_k	**#of members:** $\#S_k$	**Probability of member**	**#of assigned bits for each member (ideal)**
S_0	1	q^N	0
S_1	N	$p^1 q^{N-1}$	$\log_2(N)$
⋮	⋮	⋮	⋮
S_k	C_N^k	$p^k q^{N-k}$	$\log_2 C_N^k$
⋮	⋮	⋮	⋮
S_N	1	p^N	0
Total	2^N	1	**Expected Value:** $\sum_{k=0}^{N} C_N^k \log_2(C_N^k)\ p^k q^{N-k}$

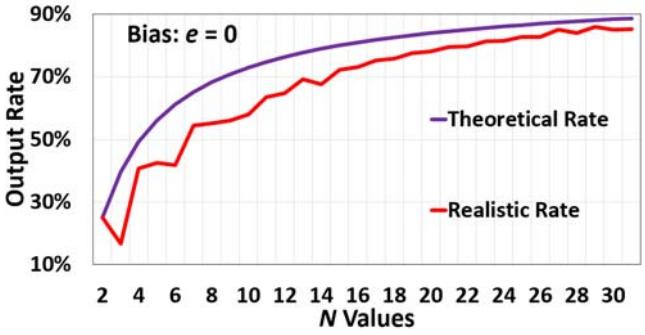

Fig. 4. VN_N Output Rate Cure

III. Implementation and Improvement on VN_N

A. Mapping Algorithm

For any value of N ($N \ge 2$), we have designed a mapping algorithm for VN_N realistic implementation. Algorithm. 1 shows the pseudocode. The merit of this algorithm exits in large N value. It can be used to generate a lookup table in ROM implementation.

To demonstrate the mapping algorithm, here we show an example when N = 4, k =2 case: first calculate $C_4^2 = 6$, then turn the decimal 6 into binary [0, 1 , 1] and assign to m. Update m to [0, 2, 4], delete 0 element and further extend to [2, 2 ,4, 4, 4, 4]. Now travel all N = 4 combinations, find k = 2 cases and use two pointer p_value, p_bit to aid the assign bits process. p_value changed with 1,2,3,4,5,6. p_bit changed with 0,1,00,01,10,11. Repeat all of the above process for all k and finish the bit assignment. Next subsection we will talk VN_4' implementation in details.

B. Implement of VN_4

For a simple but efficient logic implementation VN_4 is adopted. Fig. 5 shows the mapping rules for VN_4. In Fig. 5(a), number of 1s in 4 bits sequence are shown. All the input sequences are divided

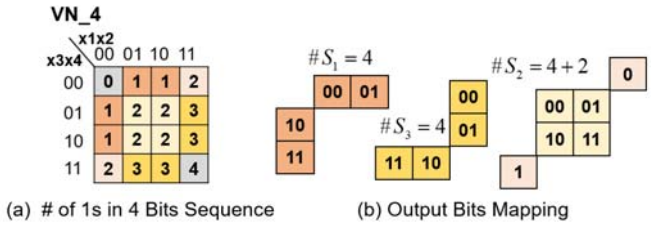

Fig. 5. VN_4 Mapping Rule

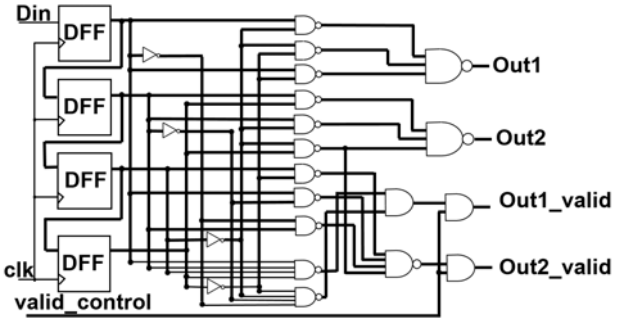

Fig. 6. Circuit Implement of VN_4

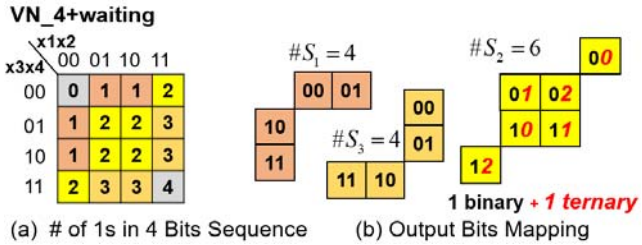

Fig. 7. VN_4+waiting Mapping Rule

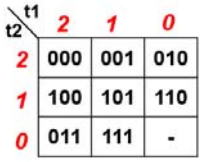

Fig. 8. Mapping from Ternary Bits to Binary Bits

into 5 groups: S_0 to S_4. For example, $\#S_1 = 4$ means, for all input sequence only with one 1s have 4 possible combinations (‘0001’, ‘0010’, ‘0100’, ’1000’), each combination has the same probability with p^1q^3. We assign 2 bits (‘00’,’01’,’10’,’11’) for each input case with equal probability as shown in Fig. 5(b). $S_3(\#S_3 = 4)$ is just same as S_1. For S_2 group, since $\#S_2 = 6$, $\log_2 6$ is not an integer. We decompose 6 into 4+2, assign 2 bits for each 4 input cases and 1 bit for each 2 input cases. In this way VN_4 outputs random bit sequence without bias. The total output rate is shown in the following equation:

$$R_{VN\_4} = \frac{13}{32} - \frac{5}{4}e^2 - \frac{3}{2}e^4 \quad (5)$$

Without bias, VN_4 reached 40.6% output rate, while VN_2 only has 25.0% output rate. VN_4 post processing has been implemented in circuit level. Fig. 6 shows the logic implementation of VN_4. It has 3 input (Din, clk, valid_control) ports and 4 output ports (2 data signal ports Out1 and Out2, 2 valid signal ports Out1_valid and Out2_vaild) The valid signals are used to indicate whether the output data can be used. Table. 2 shows the resource utilization analysis using gate equivalent(GE). GE’s calculation is shown in [8]. In total, VN_4 consumes 46.02 GE, which is 23.3% less than IVN3.

```
function Mapping(N)
  map[2^N] ← null                          //e.g. N=4
  if 0<k<N then                            //e.g. k=2
    m ← dec2binvec(C_N^k-C_N^k%2)          //m=[0,1,1]
    for any m(i) in m
      m(i) ← m(i)*2^(length(m)-1)          //m=[0,2,4]
    m ← m(2:length(m))                     //m =[2,4]
    m ← [m[1],m[1],m[2],m[2],...]          //m=[2,2,4,4,4,4]
    p_value ← 0                            //pointer for the value in m
    p_bit ← 0                              //pointer for the assign bits
    while 0<=j and j<=2^N do
      if j contains k 1's then
        map(j) ← asign log2(m(p_value)) length of  p_bits
        p_value ← p_value +1
        if m(p_value) == m(p_value - 1)
          p_bit ← p_bit +1
        else p_bit ← 0
End function
```

Algorithm. 1. VN_N Mapping Function

Table. 2. Gate Equivalents(GE) Analysis of Three Logic Implementations

Cells(GE)	VN_2[3]	VN_4	IVN3*
INV(0.67)	1	4	6
AND2(1.33)	1	3	12
XOR(2.00)	1		3
NAND2(1.00)		9	
NAND3(1.33)		2	
NAND4(1.67)		3	
DFF(5.67)	2	4	6
TOTAL(GE)	15.34	46.02	60.00

(*: IVN3 is implemented in this work according to [7])

C. Improvement of VN_N (VN_N+waiting)

In this subsection we propose a waiting strategy. With adding the waiting time, realistic output rate increases closely to the theoretical rate. It is called VN_N+waiting. In Fig. 7 VN_4+waiting is shown as an example. The difference is the mapping rule for S_2 group, where $\#S_2 = \log_2 C_4^2$ is not an integer. We assign one binary bit (‘0’ or ‘1’) and one ternary bit (‘2’,’1’,’0’). The binary bits can be output directly, while the ternary bit needs to wait to meet next ternary bit, then output 3 bits or no output bit according to the ternary bits mapping rules as shown in Fig. 8. VN_4+waiting also outputs sequence without bias. The total output rate of VN_4+waiting is shown in the equation (6).

$$R_{VN\_4+waiting} = \frac{15}{32} - \frac{7}{4}e^2 - \frac{1}{2}e^4 \quad (6)$$

VN_4+waiting reached to 46.9% output rate, when e = 0%. That is very closely to the theoretical output rate bound 49.2%. By increasing the waiting number, one can extract the entropy up to the

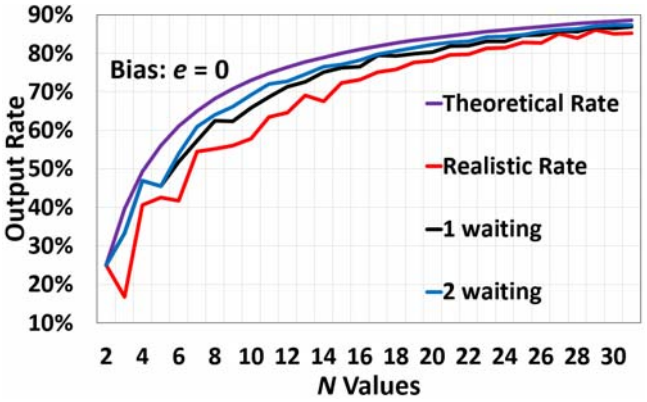

Fig. 9. VN_N+waiting Output Rate

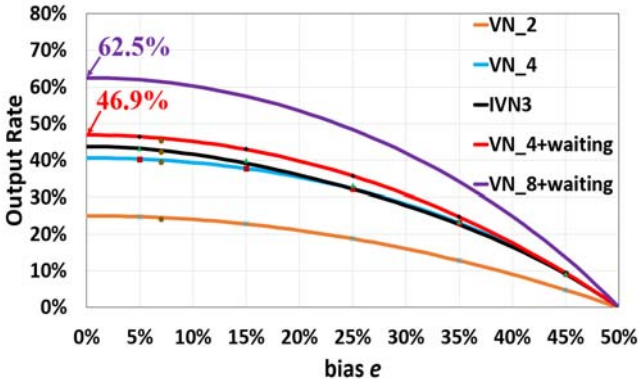

Fig. 10. Output Rate with Different Bias

Table. 3. Output Rate and NIST Test Result Under Different Techniques

Data Source	Bias *e*	VN_2[3]		IVN3[7]		VN_4(this work)		VN_4+waiting(this work)	
		Output Rate	P-value (Average)	Output Rate	P-value (Average)	Output Rate	P-value (Average)	Output Rate	P-value (Average)
True	6.95%	24.09%	0.4794	42.30%	0.6820	39.49%	0.5976	45.35%	0.5595
Pseudo	5.03%	24.68%	0.5382	43.24%	0.3164	40.26%	0.5346	46.45%	0.5731
	14.98%	22.82%	0.3709	39.95%	0.4092	37.80%	0.5407	42.98%	0.4162
	25.01%	18.76%	0.4550	33.27%	0.5357	32.22%	0.4693	35.75%	0.5047
	34.97%	12.77%	0.6731	23.38%	0.5608	23.09%	0.5189	24.71%	0.4483
	44.97%	4.77%	0.5229	9.21%	0.5515	9.20%	0.6004	9.43%	0.6256

theory bound. Fig. 9 shows the VN_N output rate with 1 waiting and 2 waiting. VN_N with 1 waiting (VN_N+waiting) curve lies in the middle of the theoretical curve and realistic curve. It has an average 6.1% improvements compared with realistic curve when $N \leq 16$.

IV. Experiment Results

Five biased pseudo random bit sequences generated from algorithm and one biased true random bit sequence are used as test data. Each sequence has 10^6 bits length. Table. 3 shows the experiment results of output rate and average P-value in NIST test [6]. We compared VN_2, VN_4, IVN3 (IVN with 3 modules) and VN_4+waiting (with 1 waiting). All test data after post processing techniques pass the NIST test. Fig. 10 shows the output rate curves for each technique under different bias. The sold line is the theoretical line, while the discrete points shows the experiment results listed in Table. 3. VN_8+waiting theoretical curve is also drawn. When bias equal to 0%, VN_8+waiting reaches to 62.5% output rate. VN_4+waiting has higher output rate than IVN3. IVN3 has better performance than VN_4 when bias is less than 20%. However, as the increasing of bias, especially when bias is large than 25.0%, they meet and have almost the same output rate.

V. Conclusions

In this research, we focus on *N* bits Von Neumann(VN_N) improvement. VN_4 is implemented in the form of algorithm and logic gate. Output rate reached to 40.6% without bias. Waiting strategy is further proposed as an improvement scheme. VN_4+waiting (with 1 time waiting) achieved 46.9% output rate, VN_8+waiting achieved 62.5% output rate. Meanwhile, a mapping algorithm is presented in our research, which can be used for any *N* bits VN_N design, especially target for the lookup table in ROM implementation.

References

[1] Barker, Elaine, and John Kelsey. "Recommendation for the entropy sources used for random bit generation." Draft NIST Special Publication, 2012: 800-900.

[2] Fischer, Viktor. "A closer look at security in random number generators design." International Workshop on Constructive Side-Channel Analysis and Secure Design. Springer, Berlin, Heidelberg, 2012.

[3] J. von Neumann, "Various Techniques Used in Connection With Random Digits," National Bureau of Standards Applied Math Series 12, 1951, pp. 36–38.

[4] P. Elias, "The efficient construction of an unbiased random sequence," Ann. Math. Statist., 1972, pp. 865–870.

[5] Y. Peres, "Iterating Von Neumann's Procedure for Extracting Random Bits," Ann. Statist., 1992, pp. 590–597.

[6] A. Rukhin et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." Special-Pub:800- 22 NIST, 2008.

[7] V. Rožić et al., "Iterating Von Neumann's post-processing under hardware constraints" Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on. IEEE, 2016, pp. 37-42.

[8] Nangate open cell library. http://www.nangate.com/.

Acknowledgement

This research is supported by ROHM Co., Ltd. and Kitakyushu Foundation for the Advancement of Industry, Science and Technology (FAIS).