International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.3 (2014), pp.235-248
http://dx.doi.org/10.14257/ijmue.2014.9.3.22

Metastability-based Feedback Method for Enhancing FPGA-based TRNG

Donggeon Lee, Hwajeong Seo and Howon Kim

Dept. of Computer Engineering, Pusan National University, South Korea
guneez@pusan.ac.kr, hwajeong84@pusan.ac.kr, howonkim@pusan.ac.kr

Abstract

This paper presents a novel and efficient method to enhance the randomness of a Programmable Delay Line (PDL)-based True Random Number Generator (TRNG) by introducing Metastability-based Feedback scheme. As a principal tool for the security of sensor network, random number generator is one of important security primitives. At the CHES2011 conference, a new method of generating random numbers by inducing metastability using precise PDL and PI (proportional-Integral) control has been proposed. As the proposed TRNG could not achieve sufficient randomness on its own, a filtering scheme was used for higher entropy. Unfortunately, the intrinsic characteristics of filters made the throughput of the TRNG decrease by approximately 50\%. To preserve the original throughput and attain a high level of randomness, we present a simple solution that analyzes the probability of outputs and assign long time to metastable state. The proposed scheme shows high randomness in the NIST randomness test suite without any of the throughput loss caused by filtering.

Keywords: *Metastability-based Feedback; Metastability; FPGA; True Random Number Generator; Programmable Delay Lines*

1. Introduction

The need for random and pseudorandom numbers is high in many security schemes for sensor networks including the generation of secrets, public keys, digital signatures, or challenges in authentication protocols. Traditional True Random Number Generators (TRNGs) are based on an analogue design that amplifies white noise in circuits, but these are not cost effective due to a heavy custom redesign and an increased budget. Thus, the popularity of digital TRNG designs has increased because of their flexibility, fast time to market, and reduced needs for custom redesigns.

A number of digital TRNGs have been developed. Most designs contain ring oscillators, which consist of an odd number of logic inverters connected cyclically to form a ring, to generate randomness from the associated jitters [8, 1-2]. In other approaches, TRNGs based on metastability have been introduced. These use multiple single inverter rings, which consists of an inverter and a multiplexer [9-10]. In such designs, the single inverter ring is used as the ring oscillator and for inducing metastability. In 2010, lightweight TRNGs using either a gate-level or transistor-level design was proposed. This method is based on metastability whereby, when a circuit switches from a metastable state to a bi-stable state, the output exhibits high entropy [3].

At CHES2011, TRNGs based on Programmable Delay Lines (PDLs), proportional-integral (PI) control, and metastable state were introduced [11]. Compared with previous methods, the TRNG provides precise control using an at-speed monitor-and-control mechanism. When the

ISSN: 1975-0080 IJMUE

output leaves the metastable state, a closed-loop feedback system adjusts the delay of the PDL. Finally, TRNG generates randomness while inducing metastability.

Nevertheless, the intrinsic nature of PI control skews the output at the boundary of the error range. For this reason, the output generates a predictable pattern that is skewed towards the values of 1 or 0. The paper introduced a filtering to remove this drawback, but this system but lowered the bitrate almost half of the original bit-rate. For an accurate and complete TRNG, we should attain desired criteria including throughput and randomness.

Contributions

In this work, we propose a novel technique, named the Metastability-based Feedback (MF) scheme that refers to the past probability of 1s and 0s, and then allocates more clock cycles for the metastable state. Thus, the technique helps to maintain metastability state for a long period and to avoid the deterministic state. As a result, we achieve high throughput together with high randomness. Our contributions are as follows.

- We provide the PDL delay characteristics when it is implemented with LUTs on a Spartan6 FPGA.
- For the precise delay tuning, we introduce an optimal delay configurations method using the propagation delay report.
- We introduce a Metastability-based Feedback scheme that has no losses of throughput from the filtering system.
- The proposed TRNG is implemented on a Spartan6 FPGA, and shows high performance in NIST random number tests [14].

This paper contains the following sections: Section 2 describes TRNG-related works, before we introduce our experimental method for measuring precise delay in Section 3. Section 4 describes the proposing Metastability-based Feedback scheme. Our Experimental results are given in Section 5, and the strengths of the proposed method are discussed in Section 6. Finally we draw conclusions in Section 7.

2. Related Works

In this section, we present related works in order to give a better understanding of the proposed TRNG design.

2.1. Metastability

Metastability is the ability of electronic circuits to persist for an unbounded time in a metastable state [4]. The cause of metastability is the violation of the flip-flop's setup and hold times. During the period from the setup time to the hold time, the input signal of the flip-flop should not be changed. If the input signal is fluctuates, it can possibly trigger the metastable state.

Traditional chip designers considered metastability to be a problem because it acts unpredictably and generates system errors. However, metastability is currently used as a good source of randomness, because uncertainty is the main priority of TRNGs. To force the flip-flop into metastability, the PI-based controller can be used to adjust the propagation delay [11]. We use this basic concept of a metastable-based TRNG in this paper.

2.2. Programmable Delay Logic

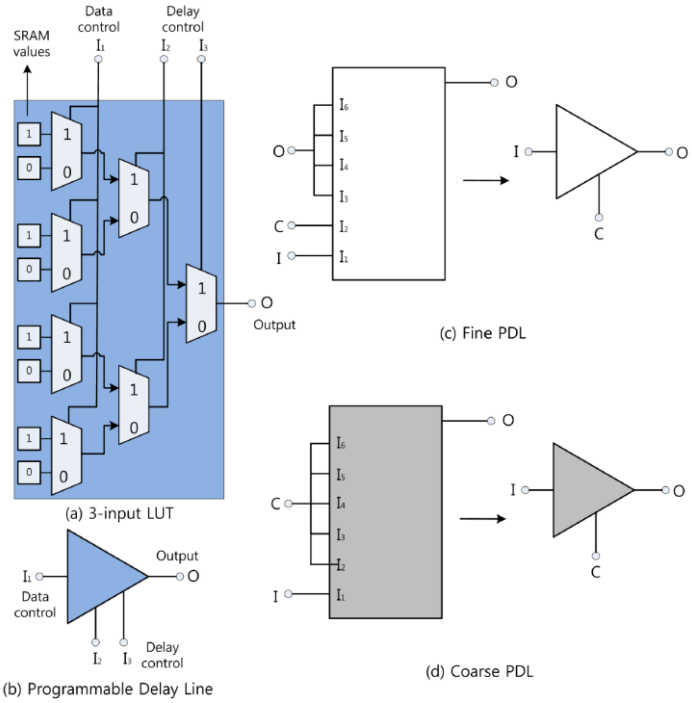

Figure 1. (a) Expected structure of 3-input LUT, (b) Symbol of PDL, (c,d) Coarse and fine PDL using a single 6-input LUT

Majzoobi *et al.* proposed Programmable Delay Line to control the propagation delay of LUT on FPGA chip. Figure 1 shows the PDL proposed in [11]. The propagation delay of PDL can be changed by choosing the signal propagation path. Basically, the LUT is consist of a set of SRAM cells for storing intended values and a tree-like structure of multiplexers(MUXes) that enables the output of stored values to be selected depending on the input signals. By choosing the routing path, each stored values are come out in different ways.

Using above structure, they defined fine PDL (Figure 1 (c)) and coarse PDL (Figure 1(d)). By fixing the 4 input signals to 0 and using only 1 signal for delay control, the PDL is used to finely control the propagation delay. Coarse PDL is made by tying the 5 input signals. However, the proposed method is not suited to all FPGA chip, because the routing path of the LUT design, which is determined by FPGA designers and manufacturers, is unpredictable. Therefore, a straightforward implementation of the proposed method is not desirable.

In this paper, we measure the all of the delay differences for all possible cases. In the case of LUT6, the five inputs alter the propagation path length and provide $2^5 = 32$ discrete levels for precise delay control. To determine the input signals for the fine and coarse tuning, we find the smallest and biggest differences for each discrete level. The details will be introduced in Section 3.

2.3. Proportional-Integral Control

A PI control is a loop feedback mechanism that calculates an "error" value as the difference between a measured output and a desired set point [5]. The PI aims to minimize the

error by adjusting the process control inputs. The Final form of the PI algorithm is as follows:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \quad (1)$$

where,

- K_p : Proportional gain, a tuning parameter
- K_i : Integral gain, a tuning parameter
- e : Error = SP(Known ideal value)- PV(Measurable output)
- SP : Known ideal value
- PV : Measurable output
- t : Time or instantaneous time (the present)
- MV : Manipulated variable
- $u(t)$: Controller output

In [11], knowledge of PI control is applied to a monitor-and-control mechanism that constructs a closed-loop feedback system. A control based on probability information tracks the output within the metastable state using a counter, which is increased when the flip-flop generates a `1' output, and decreased when the output is a `0'. The method seemed to offer complete and flawless PI control. However, we found that to get into the metastable state, the counter roams out of the bounds of the metastable state. We introduce an efficient countermeasure in Section 4.

2.4. Counter-based Filtering System

In [11], to remove the bias in the output sequence in a systematic way and eliminate predictable patterns, a counter-based filtering mechanism was proposed. This method evaluated the proportion and occurrence of outputs that generate a biased and skewed result. After analyzing the probability of 1 and 0 at the specific counter value, the counters found to cause the non-metastable state are filtered. As a result, the randomness increases but the bit-rate is reduced to approximately half. Therefore, we seek a solution to replace the filtering method.

2.5. Post-processing method

Random number generators (RNGs) are designed to produce random numbers with high entropy. However, due to variations in environmental conditions, RNG often give rise to biased outputs. To remove the skewed and low quality outputs, various post-processing techniques are conducted. A Simple Von Neumann corrector and a Linear Feedback Shift Register (LFSR) are often used for post-processing due to their simple but effective functionality [6-7]. Other methods include the use of an extractor function or one-way hash function, such as SHA-1 [12, 15]. Considering the efficiency of implementation, we chose von Neumann correction to balance the bit distribution and remove biased outputs.

3. Experimental Method

In this section, we describe the experimental method used for the implementation of our proposed TRNG. In [13], a delay measuring scheme for FPGA which utilizes ring-oscillator was proposed and it motivated our experimental method. The development environments are given in Appendix A. When we implemented Majzoobi *et al.*'s design [11], we could not achieve successful operation. Through a series of experiments, we found that the delay characteristics of the LUTs in our target FPGA had some minor differences with those presented in their paper. Therefore, we needed to measure the exact LUT delays according to various input signals. After measuring the delay, we chose the optimal control signal values for the coarse and fine tuning, and then applied those values to the TRNG implementation.

3.1. Modeling the Delay Characteristics of LUTs

To measure the exact delay values for the input signals of the LUTs, we implemented the ring oscillator (RO)-based delay measurement circuit shown in Figure 2.

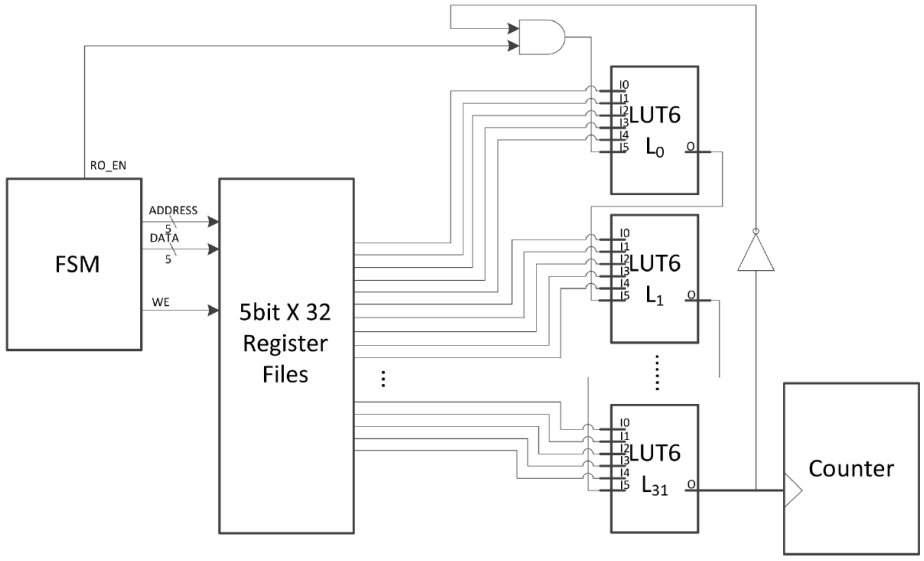

Figure 2. Ring-oscillator-based delay measurement circuit. The circuit under test consists of 32 LUTs

The circuit is composed of a Finite State Machine (FSM) that controls the entire circuit, a register file that stores values for controlling the input signals of each LUT, a frequency counter that counts the oscillation from the LUTs, and 32 serially connected LUTs. The short delay path has a feature whereby the RO oscillates more frequently and makes the counter increase. Though it is impossible to measure the precise delay in ps with our circuit, the delay difference for each control signal can be estimated through iterative measuring processes. Using the following process, we can measure the delay differences according to the position of the LUTs and their input signals:

Step 1. Initialize all the memories in the register file to 0 (mod 32). Set "ADDRESS" as 0 (mod 32).

Step 2. Enable the "RO_EN" signal.

Step 3. Count the number of pulses from the RO.

Step 4. Disable the "RO_EN" signal and trigger the "DONE" signal.

Step 5. Increase "DATA" by one and write to memory by enabling the "WE" signal.

Step 6. Check whether "DATA" is 0 (mod 32); if so, go to Step 7, else reset the frequency counter and go to Step 2.

Step 7. Increase "ADDRESS" by one.

Step 8. Check whether "ADDRESS" is 0 (mod 32); if so, end this procedure, otherwise reset the frequency counter and go to Step 2.

When the "DONE" signal is triggered, the logic analyzer captures a counter value. In Step 2, the "RO_EN" signal is activated during the thousands of clocks. If the measurement time is too short, the delay differences are difficult to recognize. For this reason, the counter needs to be activated for a sufficiently long time. After collecting 250 experimental results, the average of the counters for all of the input signals corresponding to each address is computed, and the result is depicted in Figure 3. For readability, the order of Y-axis is sorted in ascending order of average count for each address. The figure shows a similar pattern for each input signal from the LUTs.

We repeated the same experiment for other positions of the LUTs and acquired similar results. In experiments on another FPGA chip with same model, the resulting pattern was again similar.

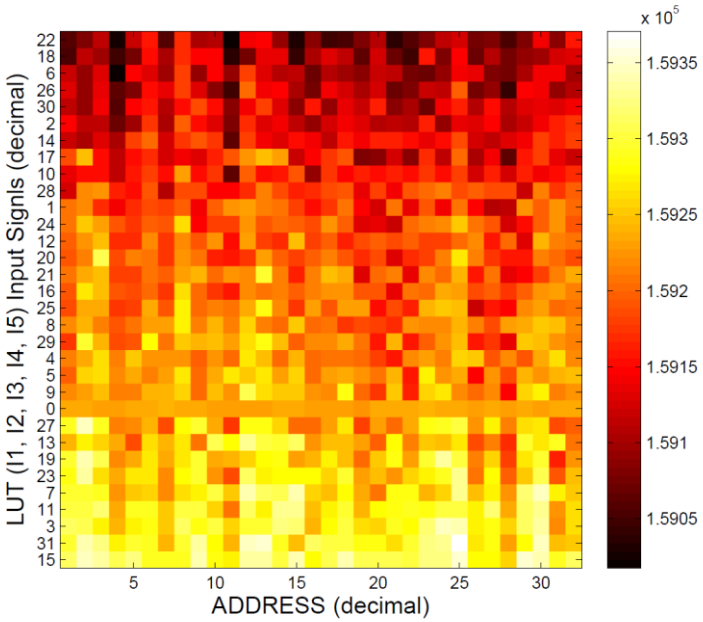

Figure 3. Delay measurement results for each input signals of each LUTs

3.2. Finding Optimal Signals for Coarse and Fine Tuning

After identifying the delay characteristics of the LUTs for each input signal, we need to find out the optimal input signal values for coarse and fine tuning. The input signals for coarse control should have bigger delay differences than the signals for fine tuning. Even though we observed the delay pattern in the previous experiment, we could not determine

which input values should be used for coarse and fine control. Thus, we performed another experiment, using the same circuit as in Figure 2 with a different procedure. At first, all the values in the register file were initialized to 15(01111_2), as this gives the biggest counter value (*i.e.*, the shortest delay) on average. Moving from the lowest address to the highest, the values were changed to specific values for each measurement, and the counter was observed each time. The detailed procedures is as follows:

Step 1. Initialize all the memories in each address of the register file to 15 mod 32 (01111_2). Set "ADDRESS" as 0 (mod 32).

Step 2. Set "DATA" as 0 (mod 32).

Step 3. Enable the "RO_EN" signal.

Step 4. Count the number of pulses from RO.

Step 5. Disable the "RO_EN" signal and trigger the "DONE" signal.

Step 6. Write "DATA" to the memory by enabling the "WE" signal.

Step 7. Increase "ADDRESS" by one.

Step 8, Check whether "ADDRESS" is 0 (mod 32); if so, go to Step 9, else reset the frequency counter and go to Step 3.

Step 9. Increase "DATA" by one.

Step 10. Check whether "DATA" is 0 (mod 32); if so, end this procedure, else set all the memories in the register file to 15 (mod 32), reset the frequency counter, and go to Step 3.

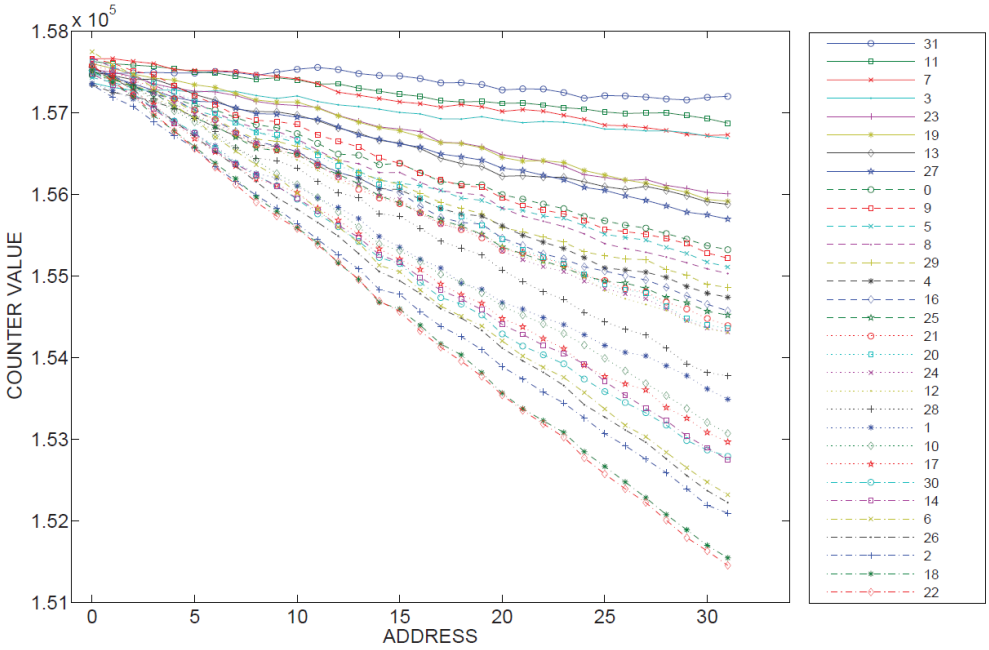

Figure 4. Delay measurement results of varying the values of each input signal from the LUTs

As in previous experiments, we gathered the counter values in Step 5 using the logic analyzer. Figure 4 represents the delay of the 32 LUTs during the experiments. From the result, we selected 15(01111_2) vs 22(10110_2) as input signal values for the coarse control, and 15(01111_2) vs 11(01011_2) for the fine control. Although 31 and 15 have the smallest difference in counter values, these values decreased nonlinearly as the address increased. For this reason, we chose 11 as the fine control value.

4. Metastability-based Feedback Scheme

In this section, we propose the Metastability-based Feedback Scheme to realize higher randomness and throughput. Our approach is ensures a long time period for counter values that give good metastable outputs, and a short time period for deterministic values.

Figure 5 shows the detail of the Metastability-based Feedback TRNG implementation. The circuit is comprised of a PDL-based TRNG and a probability analyzer.

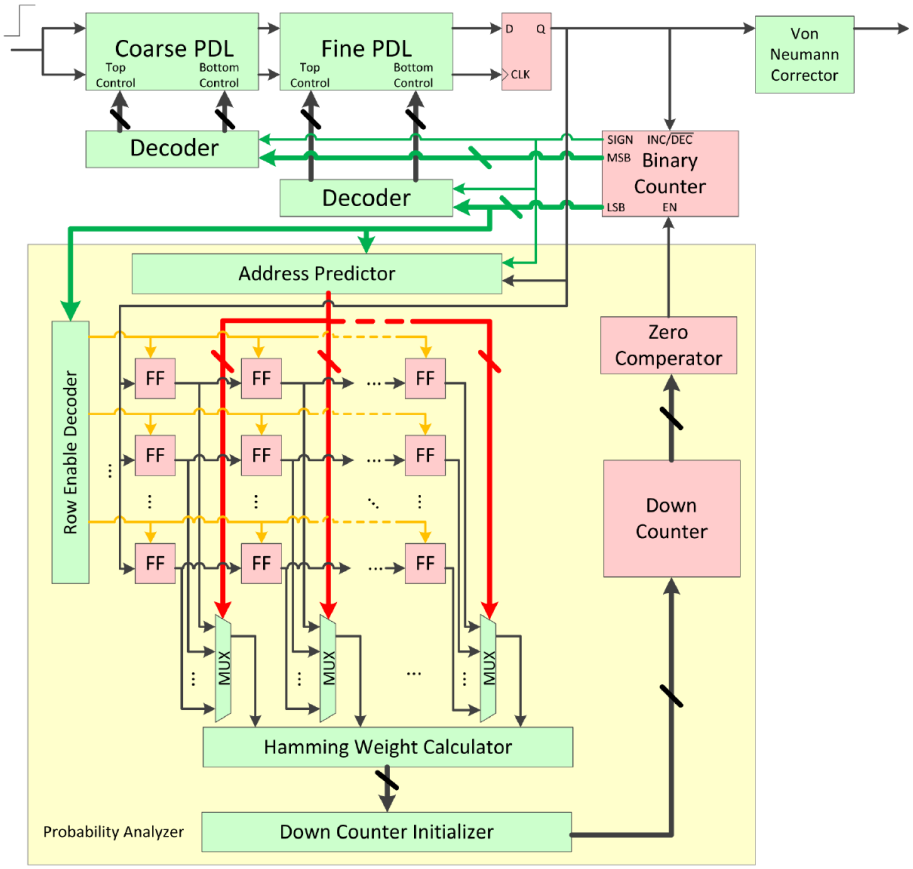

Figure 5. Circuit Layout of the TRNG using the Metastability-based Feedback scheme

The probability analyzer incorporates a matrix of flip-flops (memory), row enable decoder, next address predictor, Hamming weight calculator, down counter initializer, down counter, and a zero comparator. In the probability analyzer, we introduced a cache-like structure to store the history of outputs from the TRNG according to the counter values. To analyze the probability of 1s and 0s for each counter value, we need to store their output values in memory. However, it is impossible to maintain memories for all counter values because of the

cost of chip size. By analyzing the patterns of counter values in the steady state, we noticed that the counter values rise and fall in a small range (the difference between maximum and minimum values was 6~12), and we considered the structure of the cache memory. Part of the lower bits of the counter value is used for the memory address. The output values from the TRNG are fed into the Most Significant Bit (MSB) of the memory, and all bits in the entry are right-shifted (the Least Significant Bit (LSB) will be lost).

The down counter determines the period for the next counter value. It is initialized by the down counter initializer, and its value decreases with every clock cycle. When the down counter reaches zero, the binary counter is enabled by the zero comparator and the next counter value calculated by PI control is written to the binary counter.

The initial value of the down counter is determined as follows: First, the address predictor calculates the next counter value by analyzing the output bit of the TRNG, as well as the current binary counter value and its sign (this process is included in the binary counter, but we have extracted it in the diagram for a clearer understanding). Second, using the next counter value as a memory address, the output history of the next counter value is fed into the Hamming weight calculator. Third, the Hamming weight calculator determines how many bits in the memory are set. Finally, the weight calculator analyzes the probability of 1s and 0s from the Hamming weight, and gives a large initial value for middle Hamming weight (in the case of equally distributed 1s and 0s) and a small initial value for the minimum and maximum Hamming weight (in the case of biased 1s and 0s).

5. Experimental Results

In our experiments on the Metastability-based Feedback method, the following parameters are used:

- Number of LUT6 : 64 LUTs for the top path, 64 for the bottom path. Half of each path is used for coarse tuning, and the other half is for fine tuning.
- Size of Binary Counter : The counter consists of 11 bits, including 1 bit for the sign.
- Memory size for Storing the Output History : The least significant 4 bits of the counter value are used for the memory address, and there are 2^4 entries in the memory. Each entry has 16 bits per block.
- Clock Assignment according to the Probability of 1s : Please refer to Table. 1

Table 1. Clock assignment according to the probability of 1s

Hamming Weight		Clock Cycles
0, 16		1 clock
1, 15		8 clocks
2, 3, 13, 14		16 clocks
4, 5, 11, 13		32 clocks
6, 7, 9, 10		64 clocks
8		128 clocks

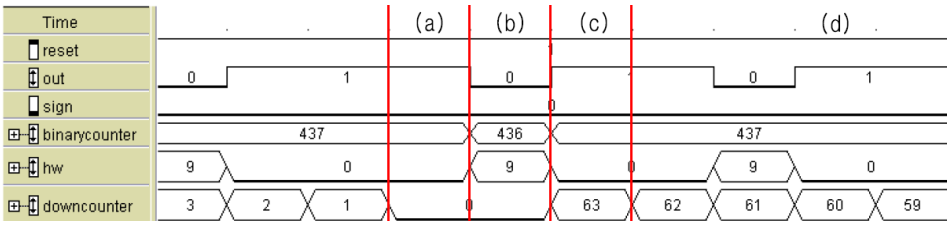

Figure 6. Waveform of the implemented TRNG

Figure 6 shows a waveform captured by the logic analyzer. The signal "binarycounter" is the output value from the binary counter, and the signal "sign" is its sign. "hw" is the hamming weight of the data in the memory addressed by the next counter value of the address predictor. The signal "downcounter" is the current down counter value, and "out" contains random bits generated by the TRNG. In Period (a), the memory content was 1101001010011010_2 at address -437, and 0000000000000000_2 at -436. After period (a), the output 1 updates the value of address -437 to 1110100101001101_2. During period (a), when the "downcounter" reaches 0, the probability of 1s for the next counter value is calculated from the Hamming weight of the history. The current value of the binary counter is -437 and the output of the TRNG is 1, so the next counter value should be -436 and the Hamming weight of the next counter value in (a) is 0. Therefore, there was a sequence of 16 zeros when the binary counter was -436, and as the system is in the non-metastable state, the time allocated for the value -436 is assigned for one clock cycle. In period (b), the time for the next counter value -437 is determined within one clock cycle. Because the output value is 0, the next binary counter value should be -437, and the figure shows that the Hamming weight of the historic values for the binary counter -437 is 9. According to Table 1, a Hamming weight of 9 gives 64 clock cycles for the next counter value. Thus, the next "downcounter" value can be 63 in period (c). Finally, during period (d), the metastable state is maintained for the 64 clock cycles and the TRNG output keeps switching.

Table 2. NIST Randomness tests. The minimum pass rate for each statistical test is approximately 96

Statistical Test	Result	Prop (%)	P-Value	Bit Length	Block Length
Frequency	PASS	97	0.00136	4,096	-
Frequency within blocks	PASS	100	0.0519	131,072	128
Cumulative sums	PASS	96.5	0.429	4,096	-
Runs	PASS	98	0.7197	1,023	-
Longest run within blocks	PASS	99	0.3669	16,384	-
Binary Rank	PASS	98	0.3191	1,000,000	-
FFT	PASS	98	0.6579	1,000,000	-
Maurer's universal test	PASS	98	0.0062	1,000,000	7
Random excursions	PASS	100	0.1540	1,000,000	-
Random excursions Variant	PASS	100	0.1397	1,000,000	-
Linear complexity	PASS	100	0.3191	1,000,000	500

Table 2 shows the result of NIST randomness tests [14]. In the NIST randomness test, the original sequence from the TRNG does not achieve a high level of randomness. Thus, we

applied the von Neumann corrector. Although the random outputs caused by the metastable state are unpredictable and seem to be a truly random sequence, some tests failed. For a performance comparison, we also implemented the design in [11] on a Spartan6 FPGA. The test results are presented in Appendix B, and show that our scheme passes more randomness tests than the previous design.

6. Discussion

Figure 7 presents counter values and associated bit probabilities measured in the range of fluctuation. The upper graphs in the figure represent counters associated with an output probability of 1. After accurately measuring the delay and tuning the PDL module, the delay is well-controlled and PI controller efficiently forces the circuit to the metastable state. However, due to the variability of environmental conditions, some counters provides opposite output probability. The lower graphs show how our method induces the metastable state. Majzoobi et al.'s result gives a normal distribution graph, which means that the highest value doesn't have a huge gap to adjacent values. However, our method's counter value is highly concentrated in the middle, metastable state, so that the non-deterministic state occurs more often than the deterministic state.

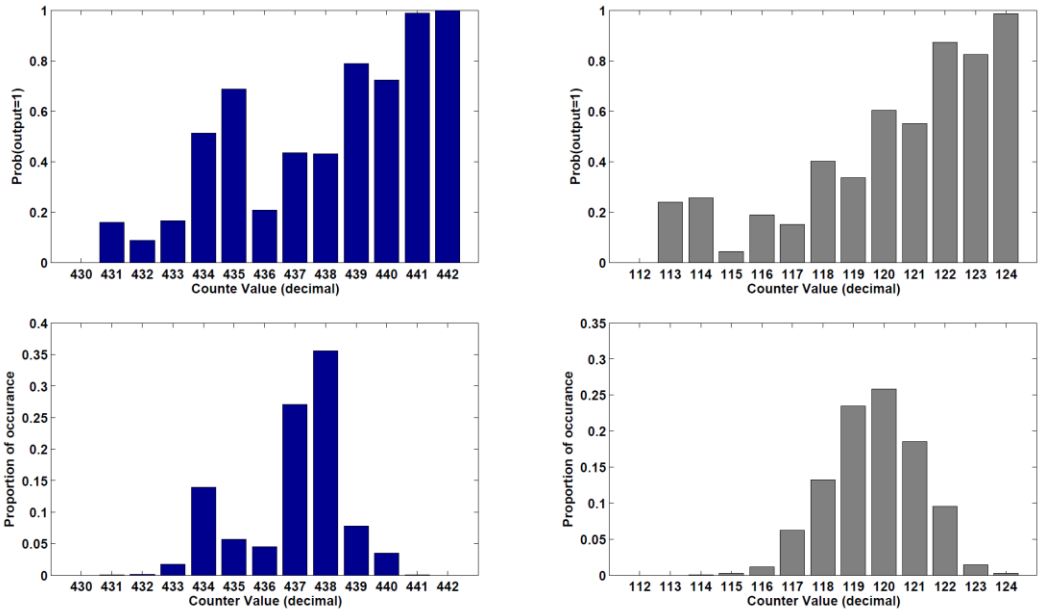

Figure 7. Distribution of the counter values and associated output probabilities. Left-hand graphs show results using out method, right-hand graphs show results of Majzoobi *et al.*

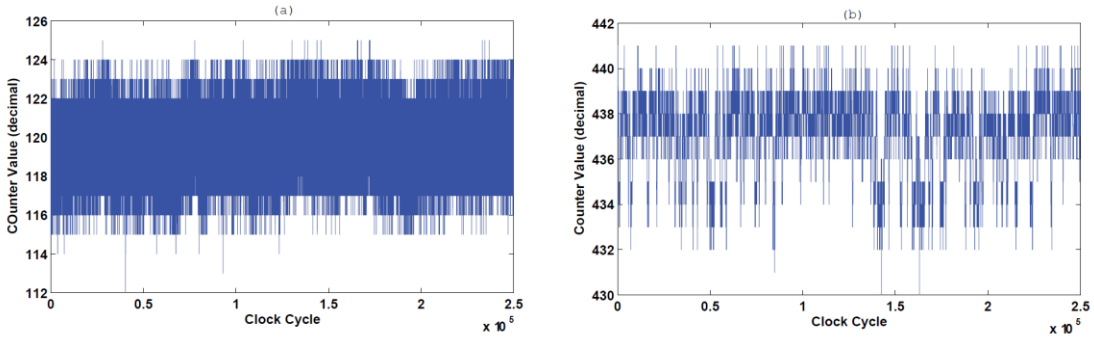

Figure 8. Range of counter fluctuation; (a) previous method, (b) this paper

Figure 8 describe the distribution of counter fluctuations. In the case of (a), the metastable state is constructed at 120, so the counter values vibrate between 116 and 123. Our method exhibits a shorter width of fluctuation. The value 438 is estimated to be the metastable state, and the counter values between 437 and 439 are observed. Therefore, our method has a higher possibility of entering the metastable state.

Table 3 shows the efficiency of throughput. Our method's throughput is reduced to 21.15% after applying von Neumann corrector. In the case of Majzoobi *et al.*, the original output is reduced to 19.17% which means their method generates more biased, skewed, or constant pattern than our proposed method. However, the original output does not provide a high level of randomness, so a filtering mechanism should be applied. After conducting a filtering method, the size of the original bit-rate decreases to 9~18%. These results show that our method guarantees a higher throughput than other methods.

Table 3. Comparison of throughput efficiency. The table presents the proportion of random bits remaining after post-processing. Letters in parentheses represents the condition of the filtering system. Filter (a) includes outputs where the counter is 120; Filter (b) includes 119, 120, 121 and Filter (c) includes 118, 119, 120, 121, and 122 when counter 120 is estimated to be the metastable state

Methods		Original(%)	Filtering(%)	Von-Neumann corrector
This paper		100	-	21.15
[11]	Original	100	-	19.17
	Filter(a)	100	23.62	9.22
	Filter(b)	100	62.82	16,65
	Filter(c)	100	86.17	18.14

7. Conclusion and Future Works

In this paper, we proposed a novel method for enhancing the randomness and throughput of the TRNG. By assigning a higher weight to counter values that emit metastable outputs, our system achieved a high level of randomness without losses from filtering.

With the TRNG implementation proposed in [11], we could not get the same experimental result, due to the use of a different FPGA target and its LUT delay characteristics. To support that our implementation, we presented experimental results for the delay characteristics of our target FPGA and selected the optimal control signal value for using the LUT as a Programmable Delay Buffer.

Though the TRNG outputs random-like bits, the sequence of output bits does not provide enough randomness for the NIST randomness test. In [11], a filtering method for counter value that gave non-metastable outputs was applied. However, such a method decreases the throughput of the TRNG. Therefore, we proposed the Metastability-based Feedback scheme. In this method, the output history of each counter value is stored in a cache-like memory space, and this is used to predict the probability of 1s and 0s for the next counter value by calculating the Hamming weight of history bits. The Hamming weight determines how many clocks the next counter value can sustain. We have given the lowest priority to the counter value that gives non-metastable outputs. The proposed scheme exhibited better results than the previous scheme in terms of both randomness and throughput.

Unfortunately, all of the schemes in our paper (ours and even Majoobi *et al.*'s) were not passed some test, which were not stated in this paper (*e.g.* non-overlapping templates, overlapping templates, approximate entropy, and serial), in NIST Statistical Test Suite. To pass the tests, some advanced technique is required to be proposed later. Moreover, in our experiments, the TRNG did not pass the test without the von Neumann corrector. The Metastability-based Feedback TRNG outputs truly unexpected random bits, but these patterns may not meet the test requirements. Nevertheless, a new method of enhancing the randomness of TRNGs should be studied.

Acknowledgements

This work was supported by the Industrial Strategic Technology Development Program (No.10043907) funded by the Ministry of Knowledge Economy (MKE, Korea).

References

[1] J. Golić, "New methods for digital generation and postprocessing of random data", IEEE Transactions on Computers, vol. 55, **(2006).**
[2] B. Sunar, W. Martin and D. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks", IEEE Transactions on Computers, vol. 56, **(2007).**
[3] J. Wu and M. O'Neill, "Ultra-lightweight true random number generators", Electronics Letters, vol. 46, **(2010).**
[4] T. Chaney and C. Molnar, "Anomalous behavior of synchronizer and arbiter circuits", IEEE Transactions on Computers, vol. C-22, **(1973).**
[5] S. Bennett, "A History of Control Engineering", P. Peregrinus **(1993)**, pp. 1930-1955**.**
[6] J. von Neumann, "Various Techniques Used in Connection with Random Digits", Nat. Bur. Stand., **(1951)**.
[7] A. J. Menezes, S. A. Vanstone and P. C. V. Oorschot, "Handbook of Applied Cryptography", CRC Press, Inc., Boca Raton, FL, (**1996**).
[8] M. Dichtl and J. Golić, "High-speed true random number generation with logic gates only", Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, **(2007).**
[9] 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", Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, **(2003).**
[10] I. Vasyltsov, E. Hambardzumyan, Y. S. Kim and B. Karpinskyy, "Fast digital TRNG based on metastable ring oscillator", Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, **(2008).**
[11] M. Majzoobi, F. Koushanfar and S. Devadas, "FPGA-based true random number generation using circuit metastability with adaptive feedback control", Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, **(2011).**
[12] B. Barak, R. Shaltiel and E. Tromer, "True random number grnerators secure in a changing environment", Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, **(2003).**
[13] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm FPGAs and beyond", Proceedings of IEEE International Conference on Field Programmable Technology, **(2006).**
[14] NIST, "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications", **(2010).**
[15] N B. Jun and P. Kocher, "The Intel Random Number Generator", **(1999)**.

Authors

Donggeon Lee

He received the BSCE degree from Pusan National University, Busan, and Republic of Korea in 2009, and he received the MS degree in Computer Engineering at Pusan National University in 2011. He is in PhD degree in computer engineering from Pusan National University. His research interests include information security, VLSI implementation of cryptographic algorithms, side channel attacks & countermeasures.

Hwajeong Seo

He received the BSEE degree from Pusan National University, Busan, and Republic of Korea in 2010, and he received the MS degree in Computer Engineering at Pusan National University. He is in PhD degree in computer engineering from Pusan National University. His research interests include sensor networks, information security, Elliptic Curve Cryptography, and RFID security.

Howon Kim

He received the BSEE degree from Kyungpook National University, Daegu, Republic of Korea, in 1993 and the MS and PhD degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied with the COSY group at the Ruhr-University of Bochum, Germany. He was a senior member of the technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea. He is currently working as an associate professor with the Department of Computer Engineering, School of Computer Science and Engineering, Pusan National University, Busan, Republic of Korea. His research interests include RFID technology, sensor networks, information security, and computer architecture. Currently, his main research focus is on mobile RFID technology and sensor networks, public key cryptosystems, and their security issues. He is a member of the IEEE, and the International Association for Cryptologic Research (IACR).