Extracting More Entropy for TRNGs Based on Coherent Sampling

Jing Yang[1,2,3], Yuan Ma[1,2](✉), Tianyu Chen[1,2,3], Jingqiang Lin[1,2], and Jiwu Jing[1,2]

[1] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
{yangjing,yma,tychen,linjq,jing}@is.ac.cn

[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

[3] University of Chinese Academy of Sciences, Beijing, China

Abstract. True Random Number Generators (TRNGs) are essential for cryptographic systems and communication security. According to the published standards, sufficient entropy derived from the stochastic model is required for TRNGs. Compared with the directly sampling jittery oscillating signal, the coherent sampling is a more efficient entropy extraction technique. In this paper, under the premise that the entropy per bit is sufficient, we focus on how to extract the entropy as much as possible from the coherent sampling in order to enhance the throughput of TRNGs. We provide a parameter adjustment method to maximize the generated entropy rate, and this method is based on our proposed stochastic model. According to the method, we design a TRNG architecture and implement it in Field Programmable Gate Arrays (FPGAs). In the experiment, the improved generation speed is up to 4 Mbps, and the output sequence is able to pass NIST SP 800-22 statistical tests without postprocessing. Compared to the basic coherent sampling, the bit generation rate is improved to 12 times.

Keywords: True Random Number Generators · Coherent sampling · FPGA · Stochastic model · Entropy extraction

1 Introduction

Random Number Generators (RNGs) play an important role in many cryptographic applications, such as the session key generation in communications, digital signature generation and key exchange. The property of generated random numbers determines the security of cryptographic systems. Generally speaking, RNGs are separated into two categories: Pseudo Random Number Generators (PRNGs) and True Random Number Generators (TRNGs). PRNGs extend the seed to extremely long sequence by using deterministic algorithms, so the PRNG security is based on the unpredictability of the seed. TRNGs collect randomness from physical phenomena such as temperature, noises, radiation, which are

R. Deng et al. (Eds.): SecureComm 2016, LNICST 198, pp. 694–709, 2017.
DOI: 10.1007/978-3-319-59608-2_38

assumed to contain unpredictable random components. In addition, the TRNG output usually serves for the seeds of PRNGs, so it is important to design security TRNGs with sufficient entropy.

Entropy is used as the measurement of the unpredictability, and also quantifies the true randomness of a TRNG output. The standards ISO 18031 [6] and AIS 31 [7] recommend to use the entropy derived from stochastic model to assess the security of a TRNG. Several works provided different modeling and entropy calculation methods for different types of TRNGs. For example, the entropy of oscillator-based TRNGs was calculated in [1,8,10], and Cherkaoui et al. [3] analyzed the behavior of self-timed ring (STR) and estimated the entropy of a STR based TRNG.

In addition to the entropy, the speed (i.e., the generation rate) is another important factor for a TRNG. Although the traditional method of sampling jittery oscillating signals has been well studied in the aspect of entropy estimation [1,10], the amount and the utilization rate of the randomness are both very low, yielding that the bit generation speed is very slow. Hence, the improvements either on refining the oscillator structure (such as [3,17]) or on improving the probability of capturing jitter (such as [12,14]) have been presented in literature.

Coherent sampling is one of the improvement techniques, where an oscillating signal is sampled by another with a similar frequency. The principle of this method utilizes the tiny difference between the two close frequencies of the signals to distinguish the jitter accumulation. In the traditional sampling, the accumulation of jitter within one sampling interval is required to be large than half or even one period of the sampled signal, thus the sampling interval has to be significantly large to guarantee the sufficiency of entropy. While, in coherent sampling, the required jitter accumulation is approximated to be the period difference between the two signals, thus the accumulation time can be much shortened to acquire a much higher generation speed. In general, the sampling result is called *beat* signal, and its period is equal to an integer times of the period of sampling signal. Actually, this integer times is random due to the accumulation of jitter. Hence, an intuitive method is counting the number edges of sampling signal within the period of *beat* signal, and using the Least Significant Bit (LSB) as the outputted random bit.

The TRNG based on coherent sampling was first presented in [9], and the random bit sequence was generated at a speed of up to 0.5 Mbps with good statistical properties in Field Programmable Gate Arrays (FPGAs). For the model of a Phase Locked Loop (PLL) based TRNG structure [4], Bernard et al. [2] proposed a mathematical model using two oscillating signals with rationally related frequencies, and then estimated the entropy per bit. An enhancement of this type of a TRNG was presented in [16] which employed the mutual sampling principle, and the improved speed up to 4 times compared to the basic coherent sampling.

In this paper, on the premise that the entropy per bit is sufficient, we focus on how to extract more entropy from the coherent sampling to enhance the speed of TRNGs. Our key insight is that the counting edge number in the *beat* signal contains more entropy which is more than 1 bit in the basic [9] or 2 bits in the enhanced [16]. Therefore, we provide a parameter adjustment method to maximize the generated entropy rate, and this method is based on our proposed stochastic model. According to the method, we design a TRNG architecture and implement it in FPGAs. In the experiment, the improved generation speed is up to 4 Mbps, and the output sequence is able to pass NIST SP 800-22 statistical test suite [13]. Compared to the basic coherent sampling, the bit generation rate is improved to 12 times.

In summary, we make the following contributions.

– We establish an equivalent model for coherent sampling from the aspect of the bias of two frequencies rather than the ratio, thus the model has a wider applicability.
– Based on the model, we propose a parameter adjustment method to maximize the generated entropy rate, and design the TRNG architecture to acquire a higher bit generation speed.
– We provide the simulation results to validate the correctness of the equivalent model, and implement the TRNG architecture in Xilinx Virtex-5 FPGA. In the experimental results, the generated bit sequence passes NIST SP800-22 statistical tests without postprocessing at a speed of 4 Mbps. The improvement factor is 12 compared to the speed of the basic coherent sampling.

The rest of paper is organized as follows. In Sect. 2, we mainly establish an equivalent model to evaluate entropy per bit. Next, we propose an architecture of TRNGs, which is based on an improved method to extract more entropy in Sect. 3. In Sect. 4, we give the simulation and implementation results to verify the effectiveness of the architecture, and compare with other related work. We conclude the paper in Sect. 5.

2 Equivalent Stochastic Model

In this section, we first introduce the principle of the traditional sampling and the coherent sampling. Then, we propose an equivalent model to transfer the coherent sampling process to the traditional sampling process. Finally, based on the equivalent model, we evaluate the bit-rate entropy and give the required condition to acquire sufficient entropy.

2.1 Principle of Traditional and Coherent Sampling Methods

The traditional sampling is defined that a stable slow clock signal (such as crystal clock signal) samples an unstable fast oscillating signal to generate bit sequences [1,10]. Relatively, the coherent sampling is defined that an oscillating signal S_{ro_1} is sampled using a D flip-flop by another oscillating signal S_{ro_2} with a similar period of S_{ro_1} [9]. The basic components of the coherent sampling are shown as Fig. 1. The signal on the output of the D flip-flop is called a *beat* signal S_{beat} and it is a low-frequency signal depending on the period difference between S_{ro_1}

and S_{ro_2}. Figure 2 shows the principle of the basic coherent sampling. The period of *beat* signal is always equal to an integer period number of S_{ro_2}. Since the S_{ro_1} and S_{ro_2} are unstable due to the jitter, the number is random. Therefore, the period number of S_{ro_2} during the period of *beat* signal can be counted as the random output.

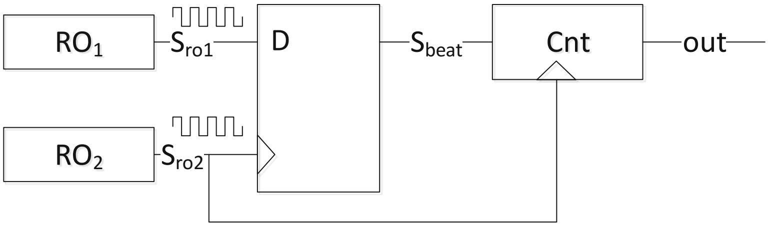

Fig. 1. Basic components of the coherent sampling

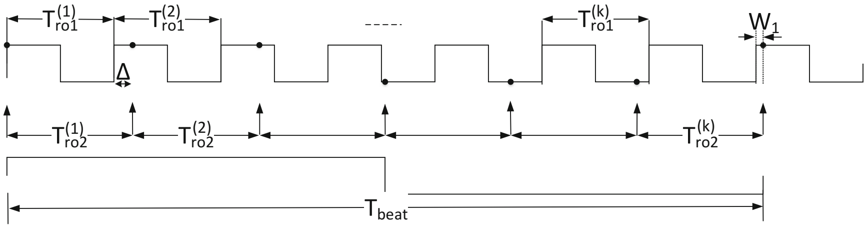

Fig. 2. Principle of the coherent sampling

2.2 Proposed Equivalent Model

Bernard et al. [2] proposed a mathematical model for the case of two oscillating signals with rationally related frequencies. Their model is efficient for the signals with known relationship (i.e., integer ratio), e.g., for the signals generated from two PLLs [4]. However, for two free-oscillating signals, the ratio could not be exactly the ratio of two (small) integers, thus the model is not applicable for this case. Therefore, we provide a more general model from the aspect of the bias of two frequencies rather than the ratio, and we succeed in transferring the coherent sampling process to the traditional sampling process, whose model and entropy have been well studied in literature [1,8,10].

Definition. The important notations are shown in Fig. 2, where the periods $T_{ro_1}^{(k)}$ and $T_{ro_2}^{(k)}$ are the time intervals between two adjacent rising edges of signal S_{ro_1} and S_{ro_2}, respectively. In this paper, we assume that $T_{ro_1}^{(k)}$ and $T_{ro_2}^{(k)}$ are independent and identically distributed (i.i.d.), and T_{ro_1} and T_{ro_2} are independent of each other. The time span between the rising edge of the signal S_{beat} and the previous rising edge of the signal S_{ro_1} is denoted as W_i. The rising edge number of signal S_{ro_2} from time zero to ith T_{beat} is denoted as N_i. Hence, N_i is represented as $N_i = min\{k|Y_k > X_{k+i}\}$, where $X_k = T_{ro_1}^{(1)} + T_{ro_1}^{(2)} + \cdots + T_{ro_1}^{(k)}$, $Y_k = T_{ro_2}^{(1)} + T_{ro_2}^{(2)} + \cdots + T_{ro_2}^{(k)}$, meaning N_i is the first increasing k ensuring that the signal S_{ro_1} has more i rising edges than the signal S_{ro_2}.

Then we denote $R_i = N_i - N_{i-1}$ as the rising edge number of signal S_{ro_2} within the ith T_{beat}, which is employed as the random output. Then we have

$$\begin{aligned} R_i &= min\{k|Y_k > X_{k+i}\} - min\{k|Y_k > X_{k+i-1}\} \\ &= min\{k|\sum_{j=N_{i-1}+1}^{N_{i-1}+k} (T_{ro_2}^{(j)} - T_{ro_1}^{(j+i-1)}) + W_{i-1} > T_{ro_1}^{(N_{i-1}+k+i)}\} \end{aligned} \tag{1}$$

Let $\{\varDelta_n\} = \{T_{ro_2}^{(1)} - T_{ro_1}^{(1)}, T_{ro_2}^{(2)} - T_{ro_1}^{(2)}, \cdots T_{ro_2}^{(N_{i-1}+1)} - T_{ro_1}^{(N_{i-1}+i)}, \cdots T_{ro_2}^{(N_i)} - T_{ro_1}^{(N_i+i-1)}, T_{ro_2}^{(N_i+1)} - T_{ro_1}^{(N_i+i+1)}, \cdots\}$, where $\{\varDelta_n\}$ is a sequence of random variable $\varDelta$. The mean and variance of $\varDelta$ are denoted as $\mu_\varDelta$ and $\sigma_\varDelta^2$, respectively. Let $\{S_n\} = \{T_{ro_1}^{(N_1+1)}, T_{ro_1}^{(N_2+2)}, \cdots T_{ro_1}^{(N_i+i)}, \cdots\}$, where $\{S_n\}$ is a sequence of random variable S. The mean and variance of S are denoted as μ_S and σ_S^2, respectively. Under the above assumptions about the two oscillating signals, we conclude

(1) $\varDelta_n$ are i.i.d. and $\varDelta$ is subject to the same distribution with $T_{ro_2} - T_{ro_1}$;
(2) S_n are i.i.d. and S is subject to the same distribution with T_{ro_1};
(3) $\varDelta$ and S are mutually independent.

According to Eq. (1), R_i also means the number of $\varDelta$ within the interval S. We ignore the jitter of S because jitter accumulation rate of which is much slower than $\varDelta$ (i.e., $\frac{\sigma_\varDelta^2}{\mu_\varDelta} \gg \frac{\sigma_S^2}{\mu_S}$). The time span W_i corresponds to the waiting time in paper [10]. Therefore, we can declare that the coherent sampling process (called the coherent sampling model) is approximated to the following sampling process (called the traditional sampling model) as Fig. 3.

- The half-periods of the unstable fast oscillating signal is $\varDelta$;
- The sampling period of the stable slow oscillating signal is $\mu_S (= \mu_{T_{ro_1}})$.

Next, we only consider the case of injecting independent Gaussian jitter to both oscillating signals in order to obtain the distribution of various random variables. Let us assume the two oscillating signals are derived from two Ring Oscillators (ROs), and let $\mu_{T_{ro_1}}$ and $\mu_{T_{ro_2}}$ be the two ideal jitter-free periods. Hence, the periods of two oscillating signals T_{ro_1} and T_{ro_2} are assumed to be Gaussian distributions

$$T_{ro_1} \sim N(\mu_{T_{ro_1}}, \sigma_{T_{ro_1}}^2), \tag{2}$$

$$T_{ro_2} \sim N(\mu_{T_{ro_2}}, \sigma_{T_{ro_2}}^2), \tag{3}$$

where $N(0, \sigma^2)$ denotes a zero-mean normal distribution with standard variance σ. The values $\sigma_{T_{ro_1}}^2$ and $\sigma_{T_{ro_2}}^2$ denote the variances of T_{ro_1} and T_{ro_2}, respectively.

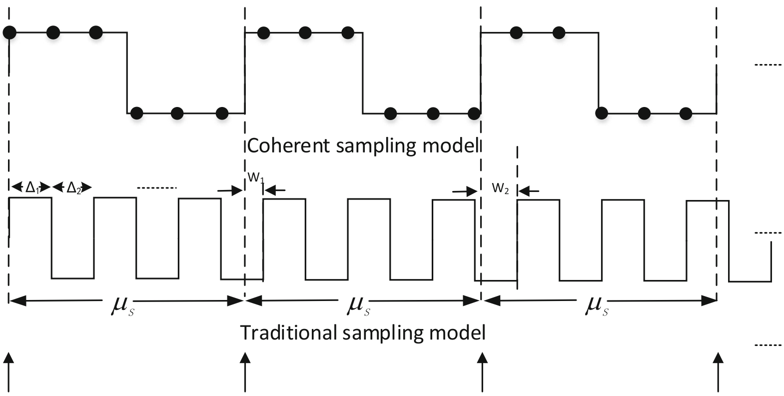

Fig. 3. The description of equivalence between two models

Assuming $\mu_{T_{ro_2}} > \mu_{T_{ro_1}}$ without loss of generality, we express the distribution of the variable Δ as

$$\Delta \sim N(\mu_\Delta, \sigma_\Delta^2), \tag{4}$$

where $\mu_\Delta = \mu_{T_{ro_2}} - \mu_{T_{ro_1}}$, $\sigma_\Delta = \sqrt{\sigma_{T_{ro_1}}^2 + \sigma_{T_{ro_2}}^2}$.

Remark. In order to simplify the model, we assume only independent random jitter exists in oscillating signals. Just as [5,10], the correlated noise also exists in oscillating signals. However, research and analysis based on correlated noise behavior are too complex to model. It is noticed as long as the amount of independent random jitter is enough, the generated bits entropy is sufficient. Therefore, we do not consider the influence of correlated noise in our model.

2.3 Entropy Evaluation

Ma et al. [10] presented a stochastic model to evaluate the entropy of oscillator-based TRNGs, and used the typical example that a stable slow clock signal samples an unstable fast oscillating signal to generate random bits which is the same as proposed equivalent model (traditional sampling model). Hence, the traditional sampling model can be employed to calculate the bit-rate entropy. We use the conclusion in this paper that in the worst case, when the standard variance of the counting results σ_R is larger than 1, the bit-rate entropy is sufficient. According to the conclusion from [15], we can express σ_R by

$$\sigma_R = \sqrt{\frac{\mu_{T_{ro1}}}{\mu_\Delta}} \cdot \frac{\sigma_\Delta}{\mu_\Delta}. \tag{5}$$

3 Proposed Architecture

In this section, based on the analysis in previous section, we first propose an improved method for extracting more entropy. Then, we propose an achievable circuit architecture for the implementation.

3.1 Improved Method for Extracting More Entropy

Key insight. Through the results of [15] and our experimental results, we have noticed that the standard variances of the counting result σ_R are significantly larger than 1. While, the condition of sufficient entropy derived from the proposed equivalent model is just $\sigma_R \geq 1$, which suggests that more entropy is contained in individual counting process, not only the LSB of the counting result R. Hence, our method is designed to maximize the extracted entropy from the counting process.

According to the principle of coherent sampling, the bit generation speed F_s is expressed as

$$F_s = 1/(\frac{\mu_{T_{ro_1}}}{\mu_\Delta} \cdot \mu_{T_{ro_2}}). \tag{6}$$

In order to enhance throughput under the status of sufficient entropy, our aim is to increase F_s and meanwhile guarantee $\sigma_R^2 \geq 1$. If $\sigma_R^2 > 1$, we can reduce the sampling period μ_S in the above equivalent model. According to Eqs. (5) and (6), when the value of σ_R^2 drops to 1, the value of F_s would be increased to σ_R^2 times. Therefore, the bit generation speed can be increased up to σ_R^2 times in theory. If we can further adjust the period difference μ_Δ to improve the sensitivity to jitter accumulation, the bit generation speed would be improved to more than σ_R^2 times.

Our approach for maximizing the extracted entropy is listed as the following Steps.

1. Minimize the period difference between two oscillating signals for increasing the sensitivity to jitter accumulation (i.e., reduce μ_Δ);
2. Use the signal S_{beat} to generate the m-multiple-frequency signal S'_{beat}, where m is the largest value to guarantee the variance of the counting numbers of T_{ro_2} is greater than 1.
3. Count the number of periods T_{ro_2} during the half-period of S'_{beat}, and use the LSB as the random bit.

It is observed that the approach also agrees with the proposed equivalent model. In the approach, we reduce μ_Δ (i.e., the half-periods of the unstable fast oscillating signal in equivalent model) and reduce μ_S (i.e., the sampling period in equivalent model), so the efficiency of extracting entropy is improved. When the period difference μ_Δ has been adjusted to an expected value in Step 1, we obtain

$$\sigma_{R'}^2 = \frac{1}{2m} \cdot \sigma_R^2 \quad (F'_s = 2m \cdot F_s), \tag{7}$$

where $\sigma_{R'}^2$ and F'_s denote the variance of counting results and bit generation speed based on the improved method, respectively. The values σ_R^2 and F_s denote the variance of counting results and bit generation speed based on the basic coherent sampling, respectively. It means that the bit generation speed is increased to 2 m times when the variance of counting results is decreased to 2 m times.

3.2 Circuit Architecture

Challenges. We have described the improved approach, but it do not involve the implementation methods. In practice, there are two challenges.

- For Step 1, how to perform a fine-grained adjustment to minimize the period difference between two oscillating signals.
- In Step 2, employing a PLL is common to generate multiple-frequency signal, but such an analog device is too heavy for a lightweight TRNG design. How to use the existing digital components to complete the same function of Step 2 is a challenging task, especially to dynamically adjust the frequency multiple.

Carry-Chain Primitive. In FPGAs, we employ the carry-chain primitives to address the above implementation problems. In Xilinx FPGAs, the circuit as shown in Fig. 4 represents the fast carry logic in a Slice. The carry chain consists of a series of four MUXes and four XORs that connect to the other logic in the Slice via dedicated routes to form more complex function [18]. If we set the port "CI" or "CYINIT" as the input port and the port "CO" as the output port, the signal is just propagated through the four MUXes (called single delay elements). It is found that the delay of a single delay element in a carry chain is much smaller than a Look Up Table (LUT).

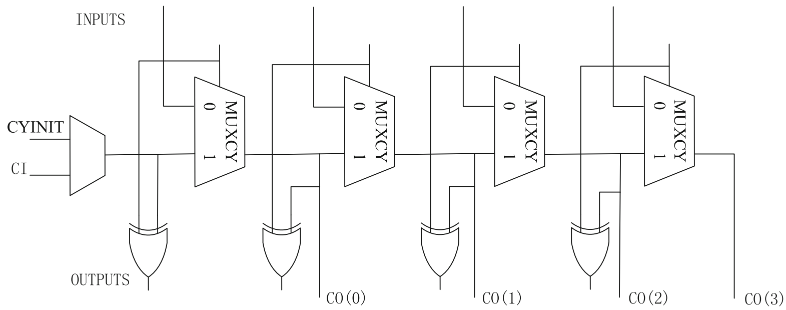

Fig. 4. Carry-chain primitives

Due to the much smaller delay and the property of cascade connection, carry chains in FPGAs have two primary uses to implement our approach:

- Finely adjusting to the period difference μ_Δ between the two oscillating signals;
- Leading out more delayed sampled signals with the smaller delay Δt of adjacent delayed sampled signals.

Architecture. By employing the carry chains, we propose the circuit architecture to implement the improved method, as shown in Fig. 5, which consists of an entropy source, a sampler circuit, a counter circuit and a bit generation circuit.

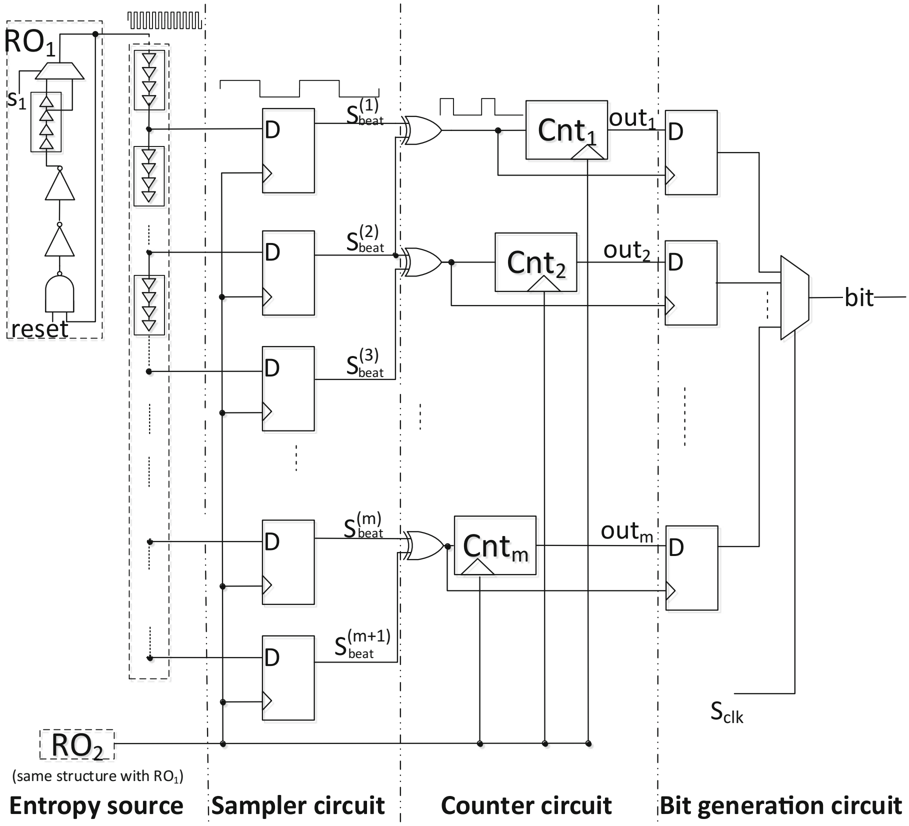

Fig. 5. Proposed circuit architecture based on the improved method

The entropy source is composed of two independent and identically configured ROs and a fast, tapped delay line. The frequency of two oscillating signals is selected to be closest but not identical. One of the oscillating signals as the sampled signal is propagated through the fast, tapped delay line to produce $m + 1$ delayed sampled signals. The sampler unit uses another oscillating signal to sample all the delayed sampled signals and produces $m + 1$ beat signals with low-frequency. XORing the adjacent beat signals produces m XORed signals and the lengths of these XORed signals lasting in high level are counted in counter circuit. The bit generation circuit uses the XORed signals produced by counter circuit to sample the LSB of the counting results, and then uses the random bit clock signal S_{clk} which should has 2m periods during $T_{beat}^{(i)}$ as the clock signal to combine multiple-channel random bits. Next, we introduce various components in details.

Entropy Source. Our ROs consist of a NAND gate, even inverters, some faster delay elements and a multiplexer. The faster delay elements and the multiplexer are used to slightly alternate propagation delay of RO to adjust the period difference μ_Δ, in which we choose the smallest period difference μ_Δ for improving the sensitivity to jitter accumulation. Then, the sampled signal in our architecture is propagated through a fast, tapped line to generate more delayed sampled signals.

Sampler Circuit. The sampler unit in our design uses the sampling signal to sample all delayed sampled signals respectively and produces m *beat* signals $S_{beat}^{(i)}$ with period $T_{beat}^{(i)}$. The signal after XORing these signals can be treated as the multiple-frequency signal, i.e., S_{clk} in the bit generation circuit.

Counter Circuit. In order to acquire the length of the delay between two adjacent *beat* signals, the adjacent *beat* signals are XORed (i.e., $S_{xor}^{(i)} = S_{beat}^{(i)} \oplus S_{beat}^{(i+1)}$) as enable terminal of respective counter and the lengths of these XORed signals lasting in high level are counted in counter unit. Only the two adjacent *beat* signals rather than all *beat* signals are XORed because it can be easier to ensure smaller impact caused by the difference of placement and routing.

An example of the counting process (without jitter) in the counter circuit is illustrated in Fig. 6, when $m = 3$. The shaded part is counting process, and the blank part denotes halting process. We can see that the counting results are all sampled at the halting process where these results are stable.

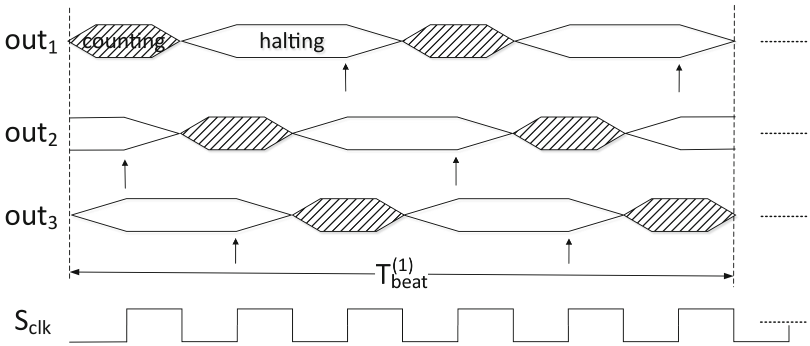

Fig. 6. Wave diagrams for the counter circuit ($m = 3$)

Bit Generation Circuit. There are m channels counting results from counter circuit ($out_1, out_2, ...out_m$). In order to acquire the random bit, the following measures are taken. At first, the bit generation unit uses all signals $S_{xor}^{(i)}$ as clock signal to sample corresponding LSB of out_i to obtain m channels random bit. The constant counting results are sampled through this way for acquiring more accurate counting values. Then, we use the random bit clock signal S_{clk} as the clock signal to combine the multiple-channel random bits.

4 Simulation and Implementation

In this section, we simulate these processes using Matlab to verify the proposed equivalent model and the improved method. Furthermore, we implement our proposed method in FPGAs and use statistical tests to test the output quality of the generator. Finally, we evaluate the speed of the implementation, and provide a comparison with related work.

4.1 Simulation Results in Matlab

We first use Matlab simulation to validate that the coherent sampling model is approximated to the traditional sampling model, where the environment is assumed to be ideal as the above mentioned. In the simulation, the period of sampled signal T_{ro_1} is set to be a normal distribution $N(5 \times 10^{-9},\ 5 \times 10^{-12})$, i.e., $\mu_{T_{ro_1}} = 5000\,ps$ (200 MHz), $\sigma_{T_{ro_1}} = 5\,ps$. And the period of sampling signal T_{ro_2} is set to be $N(5.04 \times 10^{-9},\ 5 \times 10^{-12})$, i.e., $\mu_{T_{ro_2}} = 5040\,ps,\ \sigma_{T_{ro2}} = 5\,ps$. Then the period difference Δ is set to be $N(40 \times 10^{-12},\ 5\sqrt{2} \times 10^{-12})$, i.e., $\mu_\Delta = 40\ ps,\ \sigma_\Delta = \sqrt{5^2 + 5^2}\,ps$. Then, we simulate the following two sampling processes to verify the correctness of the equivalent model.

- **Process 1:** Coherent sampling the sampled signal S_{ro_1} using the sampling signal S_{ro_2};
- **Process 2:** Traditional sampling the period difference Δ with the interval of $\mu_{T_{ro_1}}$.

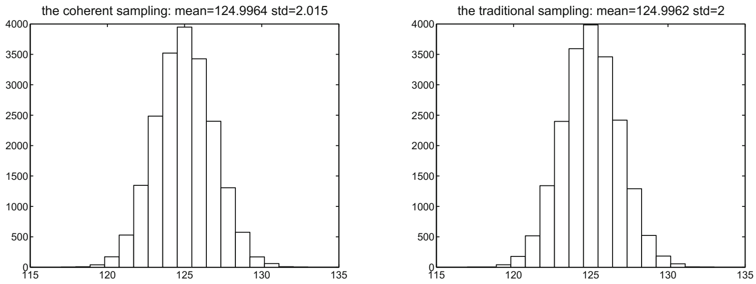

Fig. 7. Histogram of the simulated R_{coh} vs. R_{tra}

Figure 7 presents the results of counter based on the coherent sampling R_{coh} (the left), and which of the traditional sampling R_{tra} (the right). Obviously, both of the distributions are normal, and the deviation of corresponding statistics (including the expectation and variance) for these two distributions is negligible, i.e., satisfying the same distribution, which agrees with our theoretical proof mentioned above.

In order to verify the relationship predicted by the theory (Eq. (7)), we calculate the variances of counting results in term of the adjustable parameter m using Matlab numerical calculation and plot the shape of $\sigma^2_{R'}$ as a function of m with simulation data (shown in Fig. 8). The variances of counting results and the bit generation speeds for different m from 2 to 7 are listed in Table 1. The $\mu_{T_{ro_1}}$ and $\mu_{T_{ro_2}}$ are set to be 5000 ps (200 MHz) and 5040 ps respectively. The variables T_{ro_1} and T_{ro_2} are injected the same random jitter $10\sqrt{2}$ ps, i.e., $\sigma_{T_{ro_1}} = \sigma_{T_{ro_1}} \approx 14.1$ ps. We can see that the expression of fitting curve is $\sigma^2_{R'} = 16.8355/m \approx \sigma^2_R/2\,\text{m}$, and the results indicate that the change of fitting curve is coordinated with the change of Eq. (7).

Table 1. The variances and bit generation speeds for different m.

	Basic	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$
$\sigma^2_{R'}$	31.6131	8.0996	5.5265	4.1949	3.3225	2.8601	2.4413
$F_{s'}[Mbps]$	1.587	6.347	9.520	12.695	15.870	19.041	22.216

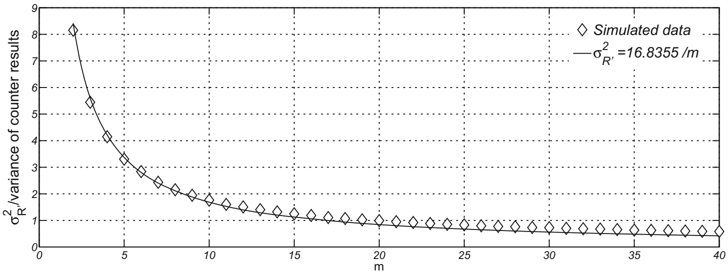

Fig. 8. The shape of $\sigma^2_{R'}$ as a function of m

4.2 Implementation Results in FPGA

We implement the circuit on Xilinx Virtex-5 FPGA. The two ROs producing oscillating signals consist of a single NAND gate, 8 inverters, 4 faster delay elements and a multiplexer, where the single NAND gate, these inverters and the multiplexer are implemented by LUTs, the faster delay elements are implemented by a stage carry chain. In order to guarantee the period difference between the two oscillating signals as small as possible, we should handle the placement and routing manually and further adjust the two multiplexers. The frequency of one RO producing sampled signal is about 146.22 MHz, The other RO producing sampling signal is about 145.88 MHz. A fast, tapped delay line is implemented by 54-stages carry chains ($54 \cdot 4 = 216$ single delay elements). We obtain $\mu_\Delta \simeq 16$ ps and $T^{(1)}_{beat} \simeq T^{(2)}_{beat} \cdots \simeq 0.34$ MHz.

An example of some key signals captured on oscilloscope is given in Fig. 9 with the case of $m = 3$. The upper signal is $S_{xor}^{(1)}$, the out_1 is counting process in high level of which. The middle signal is $S_{xor}^{(2)}$, similarly, the out_2 is counting process in high level of which. the bottom signal is the random bit clock signal S_{clk} in bit generation unit.

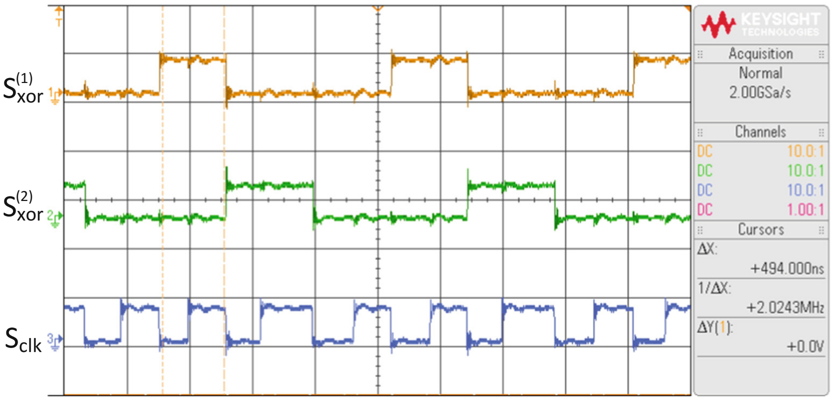

Fig. 9. Experimental $S_{xor}^{(1)}$, $S_{xor}^{(2)}$ and S_{clk} signals example with $m = 3$.

We implement a TRNG that can manually select the number of delayed sampled signal m. The parameter m can be set as 2, 3, 6 and 9 respectively. For all cases, we test the quality of generator output with different m using both the FIPS 140-2 [11] and NIST [13] statistical tests. For the NIST statistical test suite, we use the software (version 2.1) with default significance level $\alpha = 0.01$ and collect a set of 1000 consecutive sequences of 10^6 random bits for each case of m.

Table 2. Statistical tests and output bit-rate results for different m.

Throughput	Basic	$m = 2$	$m = 3$	$m = 6$	$m = 9$
	0.34 Mbps	1.36 Mbps	2.04 Mbps	4.08 Mbps	6.12 Mbps
FIPS 140-2	Pass	Pass	Pass	Pass	Pass
NIST	Pass	Pass	Pass	Pass	Fail

Table 2 shows the results of both statistical tests and output bit-rate results for different parameters m. We can see that all the cases successfully pass the FIPS tests. However, the case for $m = 9$ does not pass the NIST test. Hence, we draw the conclusion that a larger m implies a higher throughput, but also a lower quality of the random bits due to the fact that the jitter accumulation

Table 3. Results of the NIST test suite with $m = 6$ and $m = 9$.

Statistical test	$m = 6$		$m = 9$	
	P-value	Passing Rate	P-value	Passing Rate
Frequency	0.366918	990/1000	0.452173	994/1000
BlockFrequency	0.000136	983/1000	0.000000	941/1000
CumulativeSums	0.266235	990/1000	0.967382	991/1000
Runs	0.777265	991/1000	0.729870	989/1000
LongestRun	0.851383	986/1000	0.325206	9085/1000
Rank	0.858002	985/1000	0.368587	994/1000
FFT	0.861264	990/1000	0.426272	987/1000
NonOverlappingTemplate	0.329850	997/1000	0.522100	995/1000
OverlappingTemplate	0.534146	989/1000	0.969588	990/1000
Universal	0.699313	987/1000	0.189625	987/1000
ApproximateEntropy	0.000126	981/1000	0.000000	976/1000
RandomExcursions	0.739918	638/642	0.620056	612/615
RandomExcursionsVariant	0.785760	639/642	0.979761	610/615
Serial	0.380407	986/1000	0.695200	988/1000
LinearComplexity	0.363593	992/1000	0.645448	987/1000

Table 4. Comparison with related work.

Work	Platform	Resources	Throughput
This work	Virtex 5	109 Slices	4.08 Mbps
[9]	SLAAC-1 V	Not reported	0.5 Mbps
[16]	Actel	14 tiles,1 PLL	2 Mbps
[3]	Cyclone 3	> 511 LUTs	133 Mbps
	Virtex 5	> 511 LUTs	100 Mbps
[17]	Spartan 3E	Not reported	0.25 Mbps
[14]	Not reported	Not reported	2.5 Mbps
[12]	Spartan 6	67 Slices	14.3 Mbps

time is shortened. In addition, Table 3 shows the results of running the NIST suite for cases $m = 6$ and $m = 9$, respectively. As the trade-off between the security and speed, the output with the case $m = 6$ passes all of the tests, while the BlockFrequency and the ApproximateEntropy are failed for $m = 9$.

The comparison with related work is summarized in Table 4. Our design achieves higher throughput than all TRNGs based on coherent sampling [9,16]. As for other implementation, Our design achieves higher throughput than [14, 17]. However, the TRNG in [3] uses more than 511 LUTs. The generated data of TRNG in [12] are compressed using XOR postprocessing. Our entropy source are a dual ROs which consumes 109 Slices. In addition, the circuit design can adjust the period difference of two ROs and select various bit generation speeds to serve different cryptographic applications.

5 Conclusion and Future Work

Under this premise of sufficient entropy, the throughput is an indispensable factor for TRNG designs, such as for the application of session key generation in high-speed communication systems. In this paper, we design and implement a coherent sampling-based TRNG which can extract entropy as much as possible to enhance the bit generation speed. We first provide a parameter adjustment method to maximize the generated entropy rate, and this method is based on our proposed stochastic model. According to the method, we design a TRNG architecture and implement it in FPGAs. In the experiment, the improved generation speed is up to 4 Mbps, and the output sequences pass NIST SP 800-22 statistical tests successfully without postprocessing. Compared to the basic coherent sampling, the bit generation rate is improved to 12 times. In future work, we will further design the embedded module for the health test or online test of the TRNG.

Acknowledgments. This work was partially supported by National Basic Research Program of China (973 Program No. 2013CB338001), Strategy Pilot Project of Chinese Academy of Sciences (No. XDA06010702) and National Natural Science Foundation of China (No. 61602476, No. 61402470).

References

1. Baudet, M., Lubicz, D., Micolod, J., Tassiaux, A.: On the security of oscillator-based random number generators. J. Cryptology **24**(2), 398–425 (2011)
2. Bernard, F., Fischer, V., Valtchanov, B.: Mathematical model of physical RNGs based on coherent sampling. Tatra Mountains Math. Publ. **45**(1), 1–14 (2010)
3. Cherkaoui, A., Fischer, V., Fesquet, L., Aubert, A.: A very high speed true random number generator with entropy assessment. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 179–196. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40349-1_11
4. Fischer, V., Drutarovský, M.: True random number generator embedded in reconfigurable hardware. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 415–430. Springer, Heidelberg (2003). doi:10.1007/3-540-36400-5_30
5. Haddad, P., Teglia, Y., Bernard, F., Fischer, V.: On the assumption of mutual independence of jitter realizations in P-TRNG stochastic models. In: Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, 24–28 March 2014, pp. 1–6 (2014)
6. ISO/IEC JTC 1/SC 27, Berlin, Germany: ISO/IEC 18031: Information technology - Security techniques - Random bit generation (2011)
7. Killmann, W., Schindler, W.: AIS 31: Functionality Classes and Evaluation Methodology for True (Physical) Random Number Generators. Version 3.1. T-Systems GEI GmbH and Bundesamt fr Sicherheit in der Informationstechnik (BSI), Bonn, Germany (2001)

8. Killmann, W., Schindler, W.: A design for a physical RNG with robust entropy estimators. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 146–163. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85053-3_10
9. Kohlbrenner, P., Gaj, K.: An embedded true random number generator for FPGAs. In: Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, FPGA 2004, Monterey, California, USA, 22–24 February 2004, pp. 71–78 (2004)
10. Ma, Y., Lin, J., Chen, T., Xu, C., Liu, Z., Jing, J.: Entropy evaluation for oscillator-based true random number generators. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 544–561. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44709-3_30
11. PUB, N.F.: 140–2: Security Requirements for Cryptographic Modules, Washington, DC, USA, May 2001
12. Rozic, V., Yang, B., Dehaene, W., Verbauwhede, I.: Highly efficient entropy extraction for true random number generators on FPGAs. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 7–11 June 2015, pp. 116: 1–116: 6 (2015)
13. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., et al.: A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications, pp. 800–822. NIST special publication, USA (2001)
14. Sunar, B., Martin, W.J., Stinson, D.R.: A provably secure true random number generator with built-in tolerance to active attacks. IEEE Trans. Comput. **56**(1), 109–119 (2007)
15. Valtchanov, B., Fischer, V., Aubert, A.: A coherent sampling-based method for estimating the jitter used as entropy source for true random number generators. In: SAMPTA 2009, pp. Special-session (2009)
16. Valtchanov, B., Fischer, V., Aubert, A.: Enhanced TRNG based on the coherent sampling. In: International Conference on Signals, Circuits and Systems (SCS) (2009)
17. Varchola, M., Drutarovsky, M.: New high entropy element for FPGA based true random number generators. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 351–365. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15031-9_24
18. Xilinx: Virtex-5 Libraries Guide for HDL Designs (2012). http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/virtex5_hdl.pdf