Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2009, Article ID 501672, 8 pages
doi:10.1155/2009/501672

Research Article

Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings

Knut Wold and Chik How Tan

NISlab, Department of Computer Science and Media Technology, Gjøvik University College, 2802 Gjøvik, Norway

Correspondence should be addressed to Knut Wold, knutw@hig.no

Received 26 February 2009; Accepted 19 June 2009

Recommended by Lionel Torres

A true random number generator (TRNG) is an important component in cryptographic systems. Designing a fast and secure TRNG in an FPGA is a challenging task. In this paper, we analyze the TRNG designed by Sunar et al. (2007) based on XOR of the outputs of several oscillator rings. We propose an enhanced TRNG with better randomness characteristics that does not require postprocessing and passes the statistical tests. We have shown by experiment that the frequencies of the equal length oscillator rings in the TRNG are not identical. The difference is due to the placement of the inverters in the FPGA and the resulting routing between the inverters. We have implemented our proposed TRNG in an Altera Cyclone II FPGA. Our implementation has passed the NIST and DIEHARD statistical tests with a throughput of 100 Mbps and with a usage of less than 100 logic elements in the FPGA. The restart experiments have shown that the output from our TRNG behaves truly random and not pseudorandom.

Copyright © 2009 K. Wold and C. H. Tan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Traditionally, a high assurance implementation of cryptographic algorithms has been done in application specific integrated circuit (ASIC). During the recent years, more and more of these implementations are done in field programmable gate array (FPGA). There are several reasons for this development. The FPGA can be reprogrammed, leading to more flexibility for modification of algorithms, changing algorithms, and fixing bugs. The development of an algorithm in an FPGA is easier and faster as compared to an ASIC design, resulting in a shorter time-to-market. In addition, the latest FPGA devices are manufactured with the state-of-the-art technology.

It is well known that a true random number generator (TRNG) is an important component of today's cryptographic systems. Typically a TRNG can be used for generating keys, initialization vectors, random sequences for cryptographic challenges-responses, and so forth. In a cryptographic system, a private or secret parameter is normally generated by a TRNG and is an interesting property to an attacker. Therefore, the generation of a random bit sequence is important and should be unpredictable to an attacker. One common method for generating a truly random sequence is to amplify the thermal noise in a diode [1]. The disadvantage of this method is the use of external components. This approach enables an attacker to manipulate and read the random bit sequence from the device and consequently violate the security of the entire cryptographic system. If the TRNG is implemented entirely inside the FPGA, an attacker will have difficulties in retrieving and manipulating the random bit sequence. The challenge is to design a TRNG in an FPGA passing all statistical tests and at the same time using as few resources as possible and achieving a high throughput of random bits.

In this paper, we examine more closely the TRNG based on oscillator rings proposed by Sunar et al. [2]. We show that the TRNG described in [2] is not random without postprocessing. We propose an enhancement of the proposal from [2] and experimentally show improved performance with respect to FPGA resource usage and throughput. We also show that our TRNG has no bias and, therefore, no need for complicated postprocessing. We experimentally demonstrate that the frequencies of the oscillator rings are different due to the placement and routing of the inverters inside the FPGA.

We have implemented our proposal in an Altera Cyclone II FPGA [3]. Our implementation of the TRNG based on oscillator rings passes the NIST and DIEHARD statistical tests with a throughput of 100 Mbps and the usage of less than 100 logic elements in the FPGA. Repeated restarts of the TRNG from the same reset state have shown that the output of our random generator behaves truly random and not pseudorandom. The standard deviation has been calculated from 1000 traces recorded after reset. A short startup period should be omitted in order to obtain good quality of the randomness, but after the TRNG output stabilizes, the standard deviation becomes constant and in accordance with the theoretical values.

The rest of this paper is organized as follows: in Section 2, we briefly examine the previous work on TRNG in FPGA. In Section 3, we analyse the TRNG of [2]. In Section 4, we propose an enhancement of the TRNG to achieve better randomness on the output sequence. The analysis of the randomness of our proposed TRNG and the investigation of distribution of frequencies on oscillator rings are discussed in Sections 5 and 6, respectively. In Section 7, we describe in detail an implementation of our proposed TRNG. In Section 8, we investigate the behavior of our TRNG after repeated restarts from known reset state, and finally we make a conclusion in Section 9.

2. Related Work

Several implementations of TRNG in FPGA have been proposed during the recent years. The common entropy source used is jitter on clock signals. Jitter can be viewed as timing deviation from the theoretically correct position due to electronic or thermal noise [4]. The random jitter will typically follow a Gaussian distribution characterized by a certain standard deviation (σ). Usually, jitter is an unwanted property in a system, but this behavior is useful when generating random signals in a TRNG.

In 2002, Fischer et al. [5] used the jitter in analogue phase-locked loop (PLL) in FPGAs from Altera as entropy source in a TRNG. The strategy was to create different clock signals with jitter from the PLL and sample one of the clock signals with the other. This method is restricted to FPGAs containing such analogue components. Later, Kohlbrenner and Gaj [6] used a similar technique, but the clocks are generated by oscillator rings containing two transparent latches, a buffer and an inverter. Since the frequencies of the two oscillator rings have to be almost equal, the oscillator rings have to be correctly matched. Tkacik [7] proposed a TRNG using a linear feedback shift register (LFSR) and a cellular automaton shift register (CASR) clocked by two independent oscillator rings. Selected outputs from the LFSR and CASR are combined by an XOR generating the final random signal. The disadvantage of this scheme is that the TRNG has memory and is, therefore, not stateless as pointed out in [8]. In 2006, Golić [9] proposed a TRNG using a Galois ring oscillator (GARO) and a Fibonacci ring oscillator (FIRO). These LFSR-like structures use inverters as delay elements instead of register elements. The outputs from one GARO and one FIRO are combined by means of an XOR and the random sequence is generated by sampling with a D flip-flop. This design was further investigated by Dichtl and Golić [10]. The output signal from these FIRO/GARO structures has a noisy analogue behavior, making them more susceptible to cross-talk from other signals inside the FPGA than ordinary digital signals. In 2007, Sunar et al. [2] gave a theoretical proposal of a TRNG based on several equal length oscillator rings made up of an odd number of inverters (see Figure 1). The outputs from the oscillator rings are XORed together and sampled with a D flip-flop. To compensate for the imbalance between the number of zeros and ones in the random signal, a postprocessing stage is present on the output of the D flip-flop. Schellekens et al. [11] implemented this scheme in a Xilinx FPGA, but with a large number of rings in order to make the random sequence output pass the statistical tests. In 2008, Vasyltsov et al. [12] proposed a TRNG based on a 5-stage metastable ring oscillator, where each stage contains only one inverter. The result is a fast and small implementation of a TRNG in an FPGA or ASIC, but optimization in the synthesis process causes difficulties in the FPGA implementation. Recently, Danger et al. [13] proposed a fast TRNG based on creating metastability in open loop structures in FPGAs.

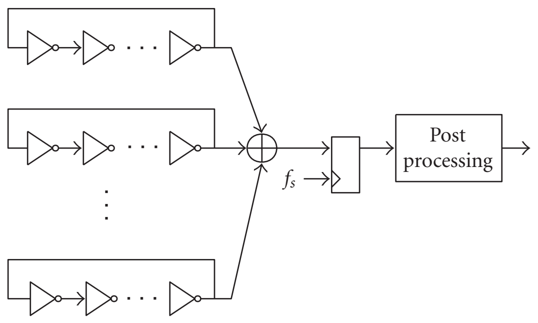


FIGURE 1: TRNG based on oscillator rings [2].

3. TRNG Based on Oscillator Rings

Since our proposed enhancement is based on the TRNG of Sunar et al. [2], we take a closer look on the design from [2], see Figure 1. The TRNG consists of several equal length oscillator rings connected to an XOR tree. The output from the XOR tree is sampled by a D flip-flop, and the output signal of the D flip-flop is then postprocessed in order to increase the entropy and remove bias from the random signal. The proposed postprocessing in [2] is a resilient function implemented as a BCH-code. The suggested design of the TRNG consists of 114 oscillator rings where each ring consists of 13 inverters. The suggested sampling frequency is 40 MHz and the postprocessing is a [256, 16, 113] extended BCH code. The resulting throughput of the TRNG in [2] is 2.5 Mbps.

The entropy source of the TRNG is the jitter created by each oscillator ring. The jitter has a Gaussian distribution around each clock transition between logic low and logic high level. This jitter will create an accumulated phase drift in each ring so that the transitions will be at different times

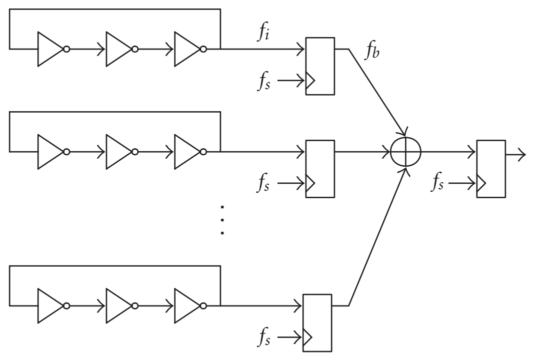


Figure 2: Our proposal.

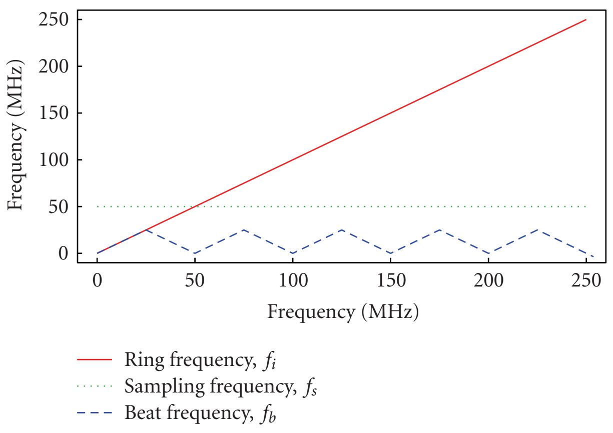


Figure 3: Beat frequency.

in the sampling period. Due to the jitter, the unpredictable transition region is assumed to be uniformly distributed in the sampling period. The number of rings needed can then be calculated based on the coupon collector's problem, that is, the number of uniform random selections of N urns such that all urns are selected at least once. The number of urns is determined by the proportion of the jitter size compared to the frequency of the oscillator ring. Because the number of rings grows exponentially when filling up the last urns, a lower fill rate than 100% is selected. To compensate for this, a BCH-code is used for postprocessing. The resulting random number throughput is reduced by a factor of 16 due to this postprocessing scheme.

In [10], some weaknesses of this implementation were mentioned. The main concern of the authors of [10] is that the XOR-tree and the sampling D flip-flop cannot handle the high number of transitions from the oscillator rings. The frequency of an oscillator ring is approximately the same or higher than the sampling frequency. With many oscillator rings in parallel, the number of transitions during a sampling period will be so high that the setup- and hold-times for the lookup table (LUT) and the internal register element in the FPGA will be shorter than specified for the device.

The analysis of the TRNG from [2] is shown in Sections 5 and 6 as it is better to present a comparison with the proposed enhanced TRNG.

4. Our Proposed Enhancement

To cope with the problem with many transitions in the sampling period, we suggest an enhancement of the TRNG based on the oscillator rings in [2] by adding an extra D flip-flop after each ring (Figure 2). As we will show, this configuration will improve the randomness of the TRNG. The randomness of the configuration relies on the jitter variations of the oscillator rings. Adding these extra flip-flops will not alter the collection of the randomness of each ring, but improve the overall randomness at the output.

The frequency of the oscillator ring (f_i) is dependent on the odd number of inverters in the ring. The frequency will increase with the decreasing number of inverters. In order to have a fast and small TRNG, the number of inverters should be as low as possible making the frequency of the rings become high as compared to the sampling frequency (f_s). The advantage of our enhancement is that the signals on the input of the XOR will now be synchronous with the sampling clock and only updated once in the sampling period. Due to this reduction in transitions on the input to the XOR tree, the setup- and hold-times for the internal logic in the FPGA will now be within acceptable limits.

The frequency of the beat signal (f_b) after the extra flip-flop will always be less than half of the sampling frequency and lie in the interval $[0, f_s/2]$ (Figure 3). The sampling frequency f_s should be chosen such that the beat frequency f_b at the input of the XOR is as high as possible, and avoid the frequency of the oscillator rings be a multiple of the sampling frequency resulting in a beat frequency near zero or no transitions (in the worts case) in the beat signal.

The result of adding the extra D flip-flop, is that the switching activity on the input to the XOR-tree is significantly reduced. The XOR calculation becomes deterministic while the randomness is collected by the sampling of the free running oscillator rings. The sampling of a free running oscillator ring could lead to metastability in the flip-flop causing the output of the flip-flop neither to be logic low or logic high for a short time period. This phenomenon can arise when a transition occurs during the setup and hold-time of the flip-flop, which is the case for the extra D flip-flops in our proposed TRNG. To avoid the metastable state to propagate into the XOR tree, the output of the oscillator ring could be sampled by one additional D flip-flop.

If a large number of rings is needed, the logic of the XOR tree will be deep and contain many logic levels. The result could be violating the timing inside the FPGA because the time delay through the XOR tree is longer than the sampling period. In this case, one or more register levels can be inserted into the XOR tree. This will not affect the throughput, but it will increase the latency of the TRNG output and increase the resources used in the FPGA.

5. Bias in TRNG

One of the basic statistical tests of random number generators is the frequency test of ones and zeros. For a good random bit sequence the probability of a zero or a one should be equal to 1/2. In other words, there should be no bias in the random bit sequence.

Let X and Y be two random bit sources with expected values $E(X) = E(Y) = \mu$, respectively, and let ρ be their correlation. Then the expected value of the XOR of the two sequences $(X \oplus Y)$ is given by (see e.g., [14]).

$$E(X \oplus Y) = \frac{1}{2} - 2\left(\mu - \frac{1}{2}\right)^2 - 2\rho\mu(1-\mu). \tag{1}$$

If μ is close to 1/2, (1) can be written as

$$E(X \oplus Y) \approx \frac{1}{2}(1-\rho). \tag{2}$$

It can be seen that correlation between the two sequences will generate bias in the output from the XOR of two random bit sequences. If X and Y are linearly independent, then $\rho = 0$ and $E(X \oplus Y) \approx 1/2$.

If there are n independent bits, each with expected value μ, then the expected value of XOR of all these bits will be given by

$$\frac{1}{2} + (-2)^{n-1}\left(\mu - \frac{1}{2}\right)^n = \frac{1}{2}(1 + (-2\varepsilon)^n), \tag{3}$$

where $\varepsilon = \mu - 1/2$. Since $\mu \in (0,1) \Rightarrow |2\varepsilon| < 1$, the expected value in (3) will converge to 1/2 for increasing number of sequences, n. In other words, adding more oscillator rings in the TRNG design should improve the bias if the rings are independent.

We have carried out some experiments on the randomness of the TRNG in [2] (without any postprocessing) and our proposal in Figure 2. The experiments are carried out on a Starter Development Board from Altera containing a Cyclone II FPGA. This device has a core voltage of 1.2 V and is fabricated in 90 nm technology. Quartus II WebEdition 6.1 is used for synthesis and Place and Route (P&R). The sequences of random bits generated inside the FPGA are stored in an external SRAM and transmitted to a PC for analysis through an asynchronous serial connection. The result is a number of blocks of subsequent random number bits from the TRNG where each block has a maximum size of 4 Mbit. The sampling frequency used in this experiment is 50 MHz. No constrains have been put on the P&R tool regarding the placement of the inverters in the FPGA.

We have implemented the two configurations of TRNG (Figures 1 and 2), recorded 10 blocks of 1 Mbit of random data from each configuration, and determined the frequency of ones in all blocks. We have performed the experiment with oscillator rings of lengths 3 and 13, and with varying number of rings. The results are shown in Figure 4. They indicate that the design in [2] has a bias after the XOR of the oscillator rings. The tendency is that the bias increases with the increasing number of rings and there is a majority of zeros in the output. Comparing these observations against (1)–(3) shows that there is some dependency or correlation in the random sequences creating a bias. It seems that this bias is due to the problem with the high number of transitions at the input of the XOR tree and the sampling flip-flop. For our configuration (Figure 2), it is seen that

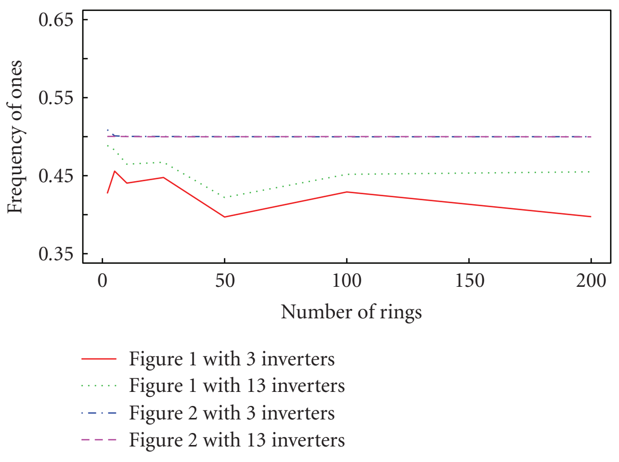


Figure 4: Bias of the TRNG on Figures 1 and 2.

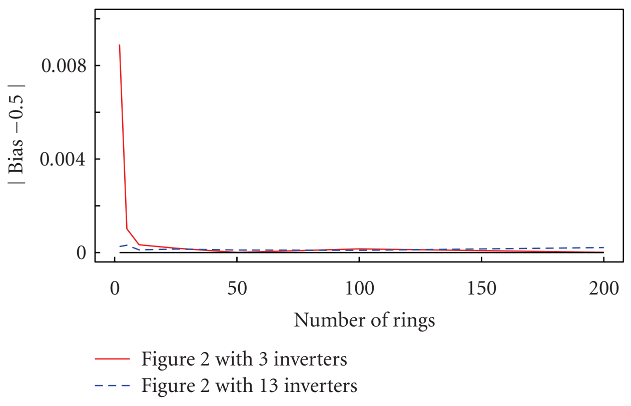


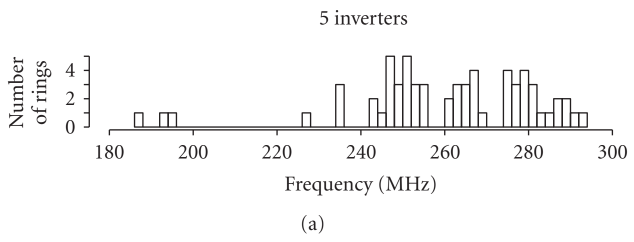
Figure 5: |Bias − 1/2| of the TRNG on Figure 2.

the expected value is close to 1/2 for increasing number of oscillator rings. Figure 5 shows a closer view of the curves, where the absolute value of the bias from the ideal 0.5 level is shown for our configuration with 3 and 13 inverters in the oscillator rings. It is seen that the bias converges to 0, and that our enhanced TRNG behaves according to the theory of XOR of independent random sequences.

6. Distribution of Ring Frequencies

According to [2], the assumption of randomness is that the equal length oscillator rings will have the same frequency while the phase drift related to the jitter causes the drifting of the transition regions. We believe that the frequencies of the oscillator rings will be different from each other. We have carried out an experiment where we have implemented 64 oscillator rings in the Altera Cyclone II FPGA and tapped out the signal from each of these rings to I/O-pins on the FPGA. These frequencies were measured with an oscilloscope.

Figure 6 shows the histograms of the frequencies of oscillator rings with 5 and 31 inverters, respectively. (For oscillator rings with 3 inverters, the measured signals are outside the specification of the I/O-pins for our Cyclone II FPGA (maximum frequency of 300 MHz). However, the



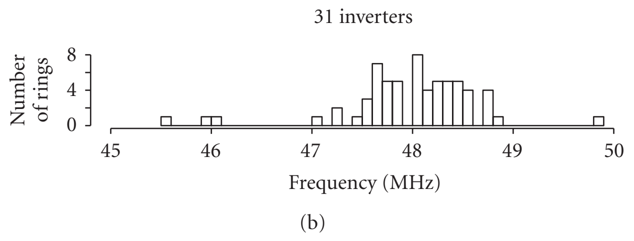


Figure 6: Histogram of ring frequencies.

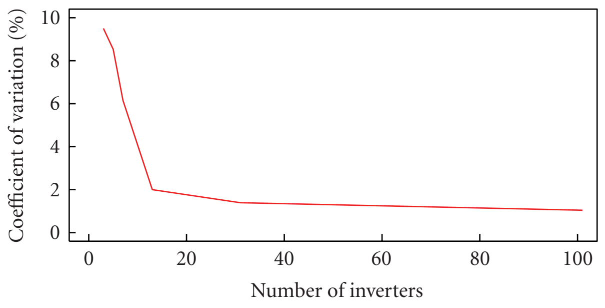


Figure 7: Dispersion of frequencies.

frequencies are measurable and the measurements with 3 inverters gave a similar histogram as shown in Figure 6 with 5 inverters.) From this experiment, it is observed that the distribution of the frequencies for short rings does not follow a Gaussian distribution and the frequencies are clustered in groups. For longer rings, the clustering is not so obvious and the distribution is approaching Gaussian with only some values far from the mean.

When examining similar histograms for other lengths, it is observed that the dispersion is decreasing with increasing number of inverters. In Figure 7, the dispersion is measured by the coefficient of variation defined as the percentage of σ/μ, where σ is the standard deviation and μ is the mean of the measured frequencies. It can be seen that the dispersion is high for short rings and decreasing with longer oscillator rings. Based on this observation, using oscillator rings with only 3 inverters will give the highest dispersion in the frequencies.

To explain the behavior of the frequency distribution, the architecture of the Altera Cyclone II FPGA [3] has to be examined. This FPGA consists of a matrix with logic elements (LEs), each containing a programmable register and an LUT for implementing any logic function of four inputs. 16 of these LEs are then grouped into a logic array block (LAB). All the LEs and LABs are connected together via different routing resources depending on the distance between them inside the FPGA. When running P&R for the design in an FPGA, the inverters in the oscillator rings are located at physical LEs. Depending on the placement, the routing delay between the LEs will differ. If all the inverters are placed in LEs inside one LAB, the routing delay will be short. If the inverters are placed in LEs in different LABs, the routing delay will be increased resulting in a lower frequency of the oscillator ring. In addition, there will also be a variation in the delay of each LUT, typically following a Gaussian distribution. All these variations in the routing delays cause the distribution of the oscillator ring frequencies and the clustering for short rings. For short oscillator rings, the inverters of some of the rings are placed in the same LAB, but for some of the other rings, the inverters are placed in two or more LABs, resulting in routing delays with large variations. For long oscillator rings, the difference between the routing delays of each ring will be smaller due to the fact that the inverters have to be placed in more than one LAB. From Figure 7, this can be seen indirectly. For small number of inverters, the dispersion is high, and decreasing until the rings contains more than 16 inverters and therefore filling up more than one LAB. For more than 16 inverters, the dispersion is constant, indicating that the variation is only due to the natural timing variation between the difference logic elements in the FPGA. For other FPGAs with similar architecture, the oscillator ring frequencies will result in similar distributions.

Due to the observed distribution of frequencies of equal length oscillator rings, the transition regions will quickly be spread out over the sampling time period, much faster than if only the accumulation of the oscillator ring jitter was contributing. In order to examine the effect of this frequency distribution on the randomness, we carried out an experiment where a model of the TRNG was made in MATLAB without involving the jitter in the generation of the output sequence. 100 blocks of 1 Mbit each were recorded, and for each block a new set of oscillator ring frequencies was generated from a Gaussian distribution (same as generating one block of data from 100 different TRNGs). The resulting output sequence was tested by the NIST randomness test suite [15], and it showed that with 50 or more oscillator rings in the MATLAB TRNG model, the quality of the generated random sequences was good enough to pass the NIST test suite. This experiment shows that a TRNG combining several equal length oscillator rings outputs where there is a dispersion between the frequencies, generates bit sequences that have good qualities even though they are deterministic. The jitter introduced in the oscillator rings contributes with the unpredictable behavior that is necessary to have a true random source.

7. TRNG Implementation

We have implemented our proposed TRNG from Figure 2. In order to have a fast and small TRNG, the number of inverters in the oscillator ring is selected to be 3. A sampling frequency of 100 MHz is selected, resulting in a throughput of 100 Mbps since our TRNG does not use any postprocessing. In most of real TRNG designs in cryptographic systems,

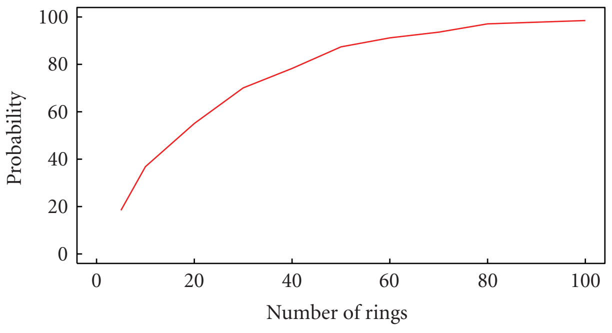


Figure 8: Simulated probability of hitting a transition region.

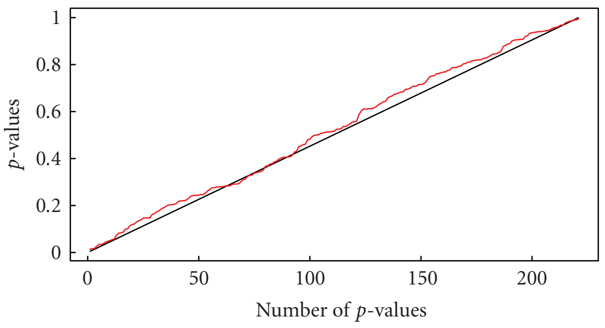


Figure 10: Results from DIEHARD.

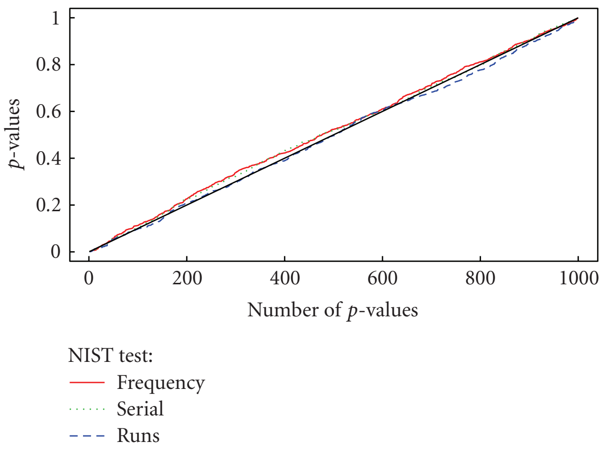


Figure 9: Selected results from the NIST suite.

Table 1: Resources used in the Altera FPGA.

Oscillator rings	LUT only LEs	LUT/Register LEs	Total LEs
25	57	26	83
50	116	51	167

using postprocessing is recommended in order to improve the randomness by increasing the entropy and removing bias. But, for our TRNG, a postprocessing is not needed to pass the statistical tests, and, therefore, an additional post-processor can be of a simple type like an XOR of two subsequent bits or a von Neumann corrector.

The required number of rings is estimated based on the probability to hit the transition region with the sampling. Sunar et al. [2] computed this by using a combinatorial approach (coupon collector's problem). An alternative way is to make a statistical model of the TRNG and perform simulations in order to decide how many oscillator rings are needed to achieve a high probability such that at least one ring is sampled in the transition region. When the size of the jitter is small compared to the sampling period, the simulations show that the number of rings in the transition region follows a Poisson distribution with a parameter $\lambda = k \cdot r$ where r is the number of rings and k is a constant depending on the size of the jitter compared to the sampling period. The probability of sampling in at least one of the transition regions versus the number of rings is shown in Figure 8. It shows that the probability increases rapidly for small number of rings, but many oscillator rings are needed to get a 100% certainty.

We have carried out an experiment where we have used 50 oscillator rings with 3 inverters, a sampling frequency of 100 MHz and no postprocessing. A total of 1000 blocks of 1 Mbit (a total of 1 Gbit) of random data have been captured from the TRNG. The data was tested by using the statistical tests of NIST (SP 800-22) [15] and DIEHARD [16]. The random data passed both tests. We also performed the same experiment with only 25 oscillator rings. The random data also passed both the NIST and the DIEHARD tests (Figures 9 and 10). From these figures it can be observed that the sorted P-values from the tests follow the ideal diagonal line. These experiments indicate that it is probably not necessary to have almost 100% certainty to hit at least one transition region in order to pass the NIST and DIEHARD statistical tests for this kind of TRNG.

Table 1 shows the amount of resources used for our TRNG in the Altera Cyclone II FPGA. For 25 oscillator rings, the number of LEs is less than 100 (<1% of the total number of LEs in our medium size FPGA). For comparison, the original design in [2] occupies more than 1800 LEs.

A TRNG design based on several oscillator rings is robust because the placement of the inverters inside the FPGA is not critical and no constraints on the P&R tool are necessary. As the experiments show, having different delays of the inverter chains contributes to the quality of randomness. However, if there are interactions between the oscillator rings making the rings oscillate with the same frequency and phase, the output bit sequence will naturally not be random. We have not seen any sign of interaction between the oscillator rings in our experiments.

All the tests were performed at the room temperature. The effect of varying the temperature is beyond the scope of this study, but in general, changing the temperature will influence the oscillator ring frequencies. An increase in temperature will decrease the oscillator ring frequencies and vice versa. But since all the rings will be influenced in the same manner, there will be a shift in all the frequencies and the dispersion between the frequencies will remain approximately unchanged.

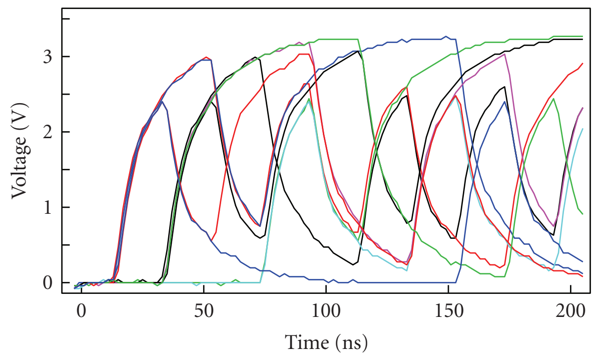


Figure 11: 10 restarts with 25 rings.

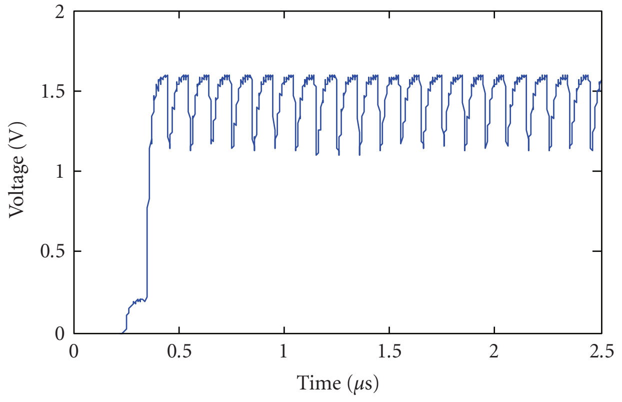


Figure 12: Standard deviation of 1000 traces, sampling frequency 10 MHz and 25 rings.

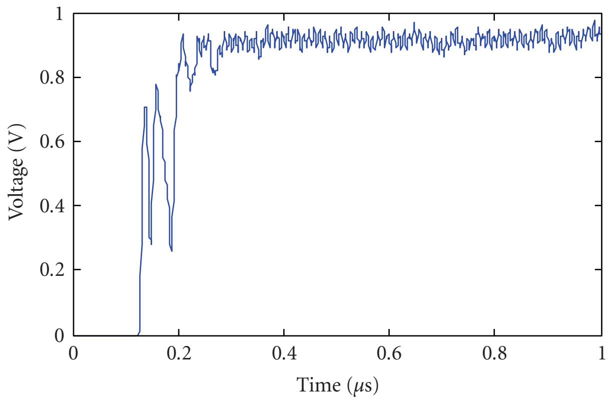


Figure 13: Standard deviation of 1000 traces, sampling frequency 100 MHz and 25 rings.

8. Restart Experiment

In order to examine the randomness of our TRNG after startup, an oscilloscope was used to capture the random output when restarting the TRNG several times from the same reset state. While the reset is active, the oscillator ring outputs are kept at zero or low level. When the reset is deactivated, the oscillator rings start to oscillate. In Figures 11 and 10 restart sequences from the output of the TRNG are captured where the oscilloscope is triggered on a clocked version of the reset signal at the origin of the graph. The sampling frequency is 50 MHz. Because of the bandwidth limitation in the oscilloscope, the measured outputs are not square signals. It can be seen that all the outputs start at zero, but there is a deviation after the first clock period of 20 ns. This experiment shows that our TRNG outputs randomness quickly after a restart. The experiment also shows that since the traces are deviating from each other, the output contains true randomness and not pseudorandomness. If the random signal had been only pseudorandom, the restart experiment should have given equal traces when repeatedly starting the TRNG from the same reset state.

The restart experiment was expanded to capture 1000 restarts. The standard deviation for all these traces was calculated and is shown in Figure 12 for a sampling frequency of 10 MHz. The form of the curve is very regular because all the traces are aligned with the sampling frequency, and because the random signal is digital with voltage level of either +3.3 V or 0 V. Theoretically, the standard deviation can be calculated by looking at the probability of a voltage level of logic zero and logic one, and the probability of a transition between the two logic levels. A good random signal should have equal probability of zeros and ones, and also equal probability of a transition or no transition. The mean value of the voltage with these conditions, is then $\mu = 1.65$ V. The standard deviation can then be calculated for the two cases: (1) the signal is in the middle of the sampling period, and (2) the signal is at the sampling point. For case (1), the standard deviation is $\sigma = 1.65$ V, and for case (2) the standard deviation is $\sigma = 1.17$ V. From Figure 12 we can see that the theoretical values match the measured data when the starting period is omitted. It is also observed that the standard deviation is stable after a short startup period.

Figure 13 shows the standard deviation with a sampling frequency of 100 MHz. Due to the band limitation of the oscilloscope, the values of the standard deviation differ from the theoretical values. It is observed that in a short startup period, the quality of the randomness is not optimal because the standard deviation is not stabilized. From Figures 12 and 13, it is seen that this startup period is constant regarding the sampling frequency or the throughput bit-rate. For the case of a TRNG with 25 oscillator rings with 3 inverters in each ring, this startup time is about 300ns. This indicates that the first data bits should be omitted in order to have good quality of the random sequence.

9. Conclusion

We have analyzed the TRNG in [2] and have proposed an enhancement of a TRNG based on oscillator rings. By adding an extra flip-flop after each inverter ring before the XOR tree, we have shown that the performance is much better than [2] regarding the random signal. We have also shown that the frequencies of each ring are not equal but have some kind of distribution. Smaller rings will have higher dispersion in the distribution and therefore also better potential for fast generation of randomness after restart.

We have implemented the TRNG from Figure 2 and carried out statistical tests on the resulting random bit sequences. We have shown that our TRNG passes both the NIST and DIEHARD tests without postprocessing. The throughput of the TRNG is 100 Mbps and the resources used in the FPGA are less than 100 logic elements in an Altera Cyclone II FPGA.

The restart experiments show that the output of the TRNG behaves truly random and not pseudorandom since the traces differ when restarted from the same reset state. These experiments also show that the standard deviation of the traces is in accordance with the theory, and that it is stable after a short startup period. Due to this startup period, the first bits should be omitted in order to have good quality of the randomness.

Acknowledgment

This paper is an extended version of the conference paper [17] presented at ReConFig08. This work was carried out while Chik How Tan was at Gjøvik University College.

References

[1] B. Jun and P. Kocher, "The Intel Random Number Generator," White paper prepared for Intel Corporation, April 1999.

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

[3] Altera Corporation, "Cyclone II Device Handbook," June 2006, http://www.altera.com.

[4] Tektronix Inc, "A Guide to Understanding and Characterizing Timing jitter," 2003, http://www.tektronix.com/jitter.

[5] V. Fischer and M. Drutarovský, "True random number generator embedded in reconfigurable hardware," in *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '03)*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 415–430, Springer, 2003.

[6] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for FPGAs," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '04)*, pp. 71–78, ACM, 2004.

[7] T. E. Tkacik, "A hardware random number generator," in *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '03)*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 450–453, Springer, 2003.

[8] M. Bucci and R. Luzzi, "Design of testable random bit generators," in *Proceedings of the 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '05)*, vol. 3659 of *Lecture Notes in Computer Science*, pp. 147–156, Springer, 2005.

[9] J. D. Golić, "New methods for digital generation and postprocessing of random data," *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1217–1229, 2006.

[10] M. Dichtl and J. D. Golić, "High-speed true random number generation with logic gates only," in *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '07)*, vol. 4727 of *Lecture Notes in Computer Science*, pp. 45–62, Springer, 2007.

[11] D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '06)*, pp. 1–6, IEEE, 2006.

[12] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy, "Fast digital TRNG based on metastable ring oscillator," in *Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '08)*, vol. 5154 of *Lecture Notes in Computer Science*, pp. 164–180, Springer, 2008.

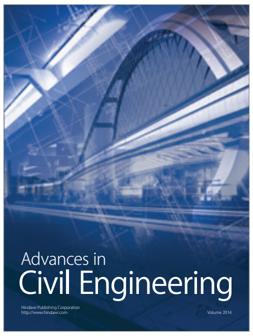
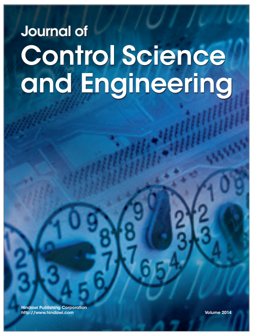
[13] J.-L. Danger, S. Guilley, and P. Hoogvorts, "High speed true random number generator based on loop structures in FPGAs," *Microelectronics Journal*. In press.

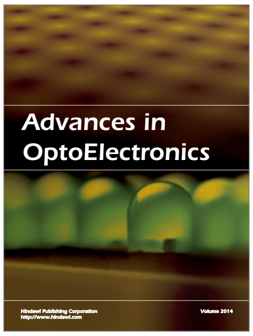
[14] R. Davies, "Exclusive Or (XOR) and Hardware Random Number Generators," February 2002, http://www.robertnz.net.

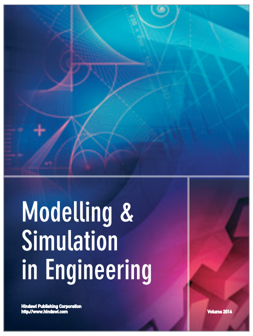
[15] NIST Special Publication 800-22, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," October 2000.

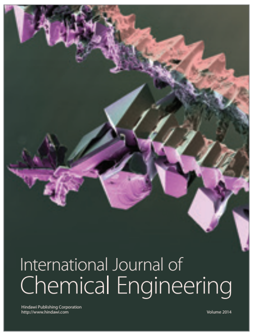
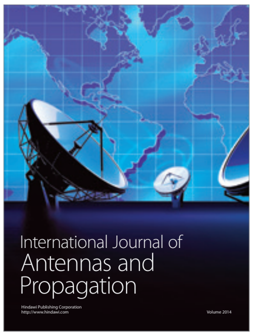
[16] G. Marsaglia, "DIEHARD: A Battery of Tests of Randomness," 1996, http://stat.fsu.edu/pub/diehard.

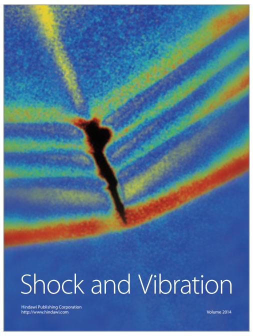
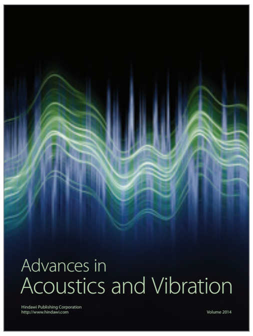
[17] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '08)*, pp. 385–390, 2008.





Hindawi

Submit your manuscripts at
http://www.hindawi.com