Łukasz MATUSZEWSKI, Mieczysław JESSA
POLITECHNIKA POZNAŃSKA, WYDZIAŁ ELEKTRONIKI I TELEKOMUNIKACJI,
ul. Polanka 3, 60-965 Poznań

A digital true random number generator implemented in different Xilinx FPGAs

Mgr inż. Łukasz MATUSZEWSKI

He received the M.S. degree from Poznan University of technology in 2010. Now he prepares the Ph.D. thesis. He works in the Chair of Telecommunication Systems and Optoelectronics of the Poznan University of Technology as teaching assistant. His research interest include randomness in digital logic, reconfigurable systems and Field Programmable Gate Arrays.

e-mail: lukasz.matuszewski@et.put.poznan.pl

Dr hab. inż. Mieczysław JESSA

He works in the Chair of Telecommunication Systems and Optoelectronics at Poznan University of Technology. Initially, his research interest included phase-locked loops and PDH/SDH network synchronization. His current research concerns randomness and pseudo-randomness, the applications of the chaos phenomenon, and mathematical models of systems evolution. He is the author or co-author of over one hundred journal and conference papers and fifteen patents.

e-mail: mjessa@et.put.poznan.pl

Abstract

In cryptography we often require sequences of numbers with unpredictable elements. Such sequences cannot be produced by purely deterministic systems. A novel method for producing true randomness and increasing the randomness of a combined TRNG using ring oscillators is described. In this paper we show that the proposed method provides similar results for generators implemented using different technologies offered by Xilinx. Thus, the proposed generator can be implemented in different FPGAs with other elements of a cryptographic system.

Keywords: true random number generator, ring oscillator, auxiliary source of randomness, cryptography, field programmable gate array.

Cyfrowy generator ciągów losowych zaimplementowany w układach FPGA firmy Xilinx

Streszczenie

W kryptografii często wymaga się ciągów liczb złożonych z nieprzewidywalnych elementów. Takie sekwencje nie mogą być wytwarzane w systemach czysto deterministycznych. Inżynierowie muszą opracować źródła losowości, których właściwości muszą być ocenione i potwierdzone przez niezależne badania, przynajmniej doświadczalnie. W artykule pokazano, że proponowana metoda wytwarzania losowości jest stabilna pod względem technologicznym. Uzyskano bardzo zbliżone rezultaty dla generatorów losowych zrealizowanych w strukturach FPGA (Field Programmable Gate Array) wykonanych w różnych technologiach jakie oferuje firma Xilinx. W żadnym przypadku nie korzystano z manualnego rozmieszczania elementów w matrycy FPGA, aby uzyskać lepsze rezultaty. Położenie poszczególnych składników zależało tylko od oprogramowania dostarczanego przez producenta. Zatem proponowany generator może być implementowany w różnych układach FPGA razem z innymi elementami systemu kryptograficznego.

Słowa kluczowe: układy FPGA, generator losowy, oscylator pierścieniowy, kryptografia.

1. Introduction

Combining XOR bits produced at the same time by many independent random number generators is an efficient method for producing random sequences that pass every statistical test [1-8]. This method requires relatively large resources, and excellent statistical properties can be observed for both deterministic and nondeterministic systems. From a cryptography perspective, the amount of resources devoted to random bit generation using FPGAs is not critical for most of the cryptographic systems but the possibility of producing random bits that can be predictable is not acceptable. In [9] it was proved in simulation experiments that combining XOR bits produced by many independent random number generators using ring oscillators (ROs) can provide sequences that pass all statistical tests when ring oscillators do not exhibit any jitter. Through experiments, it was also proved that a very small amount of randomness, present in a single-ring oscillator, accumulates as the function of the number of combined XOR bits [5-8]. Thus, this type of generator can provide a bit sequence suitable for cryptographic applications, but every j-th bit from the original sequence has to be chosen as the output, where j must be greater than a certain minimum value m_{min} [5-8]. It reduces the efficient bit rate from 300 Mbit/s suggested in [3] for RO-based TRNG to approximately 7.14 Mbit/s for 50 ring oscillators [5, 6]. The value of m_{min} is determined with the restart mechanism and the chi-square test. The method for computing m_{min} is described in detailed in [8]. To increase m_{min}, an auxiliary source of randomness (ASR) that perturbs the XOR signals produced by the ROs is used [8]. A smaller m_{min} value increases the output bit rate compared to the traditional TRNG using many independent RO-based TRNGs [8]. The Galois ring oscillator with a 31-degree polynomial was used as the ASR. It was shown experimentally that for comparable resources, the TRNG with an ASR provides sequences that pass statistical tests with significantly smaller m_{min} than the combined TRNG described in [2, 3]. The same conclusion was drawn when comparing the TRNG with an ASR with the combined TRNG composed of Galois ROs [8].

In this paper we show that the method proposed in [8] is technologically stable, i.e., it provides similar results for the generators implemented in different technologies offered by Xilinx. In all of the cases considered, we do not manipulate the placement of the elements in the FPGA structure to achieve "better" results. The placement of the elements depends only on the software provided by a manufacturer.

2. Combined TRNG with an ASR

The scheme of combining XOR of K, $K > 1$ bit stream produced by RO-based TRNGs with outputs of ROs that are perturbed by the signal produced by an ASR is shown in Figure 1. Because the number of inputs of LUTs used in our designs is limited, the XOR bits are combined in several steps. The combined TRNG produces the first bit with a delay equal to three periods of the quartz oscillator. To assess the quality of the generated sequence, the bits are sent to a personal computer (PC) via a buffer and a USB 2.0 interface. No post-processing is performed in the PC. We assume that the source generators have frequencies $f_{H,k}$, $k = 1, 2, \ldots, K$, which are not lower than f_L. The delay τ was realized by one latch. We did not use inverters due to the greater resources required to implement the combined TRNG (see [8], Table I for details). The auxiliary source of randomness can be any source of entropy that is easily implementable in an FPGA. The ASR should provide random bit sequences with small m_{min} values.

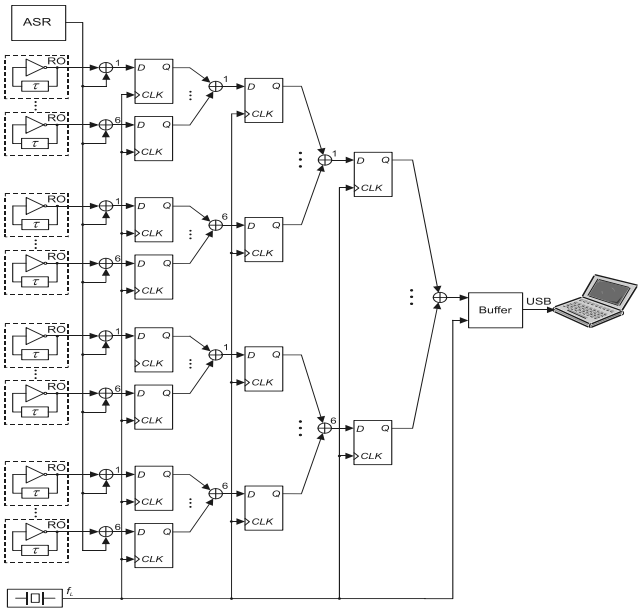

Fig. 1. Combined TRNG with an ASR [8]
Rys. 1. Łączony generator TRNG z ASR [8]

The statistical properties are secondary because excellent statistical properties are guaranteed by combining XOR bits produced by many RO-based TRNGs. The natural candidates for ASR are the Galois and Fibonacci ring oscillators. GARO and FIRO are defined by a feedback polynomial, as in the classical LFSR Galois and Fibonacci types, but the memory elements are replaced with inverters [10, 11]. For both GARO and FIRO configurations, the output of the oscillating signal comprises both pseudo-randomness and true randomness. The restart mechanism and the chi-square test were used to assess the amount of true randomness [8]. We examined the GAROs and FIROs with the following feedback polynomials [10]:

$$f(x) = x^7 + x^5 + x + 1 \tag{1}$$

$$f(x) = x^7 + x^6 + x^2 + 1 \tag{2}$$

$$f(x) = x^{15} + x^{14} + x^7 + x^6 + x^5 + x^4 + x^2 + 1 \tag{3}$$

$$f(x) = x^{20} + x^{18} + x^{16} + x^{15} + x^{13} + x^{12} + x^5 + x^4 + x^2 + 1 \tag{4}$$

$$f(x) = x^{21} + x^{19} + x^{17} + x^{16} + x^7 + x^3 + x^2 + 1 \tag{5}$$

$$f(x) = x^{31} + x^{27} + x^{23} + x^{21} + x^{20} + x^{17} + x^{16} + x^{15} + x^{13} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1 \tag{6}$$

The generators were implemented in Xilinx Virtex-5 FPGA (XC5VLX50T). The outputs were sampled, and the obtained bits were sent via USB 2.0 to a personal computer. During a single restart, 20000 bits were produced. There were 2048 restarts, i.e., it was the same as in [8]. If among of the 20000 sequences there are one or two successive sequences that fail the chi-square test, these failures may be produced by a perfect random source because single sequence fail the chi-square test statistically every 103rd sequence and a pair of sequences do not pass the same test statistically every 10604 sequences. Three successive sequences fail the test statistically every $\approx 10^6$ sequences (see [8] for the computation method; significance level of the test equal to 0.01). Thus, among 20000 chi-square test results there can be at most one such case when three successive sequences fail the test for a perfectly random source. If the sequences are produced by a nonrandom source, 2048-bit sequences fail the test for three successive values of m, m = 1, 2, …, 20000 numerous times. A computer program searches the results of 20000 chi-square tests for the greatest m for which the 2048-bit sequences fail the chi-square test for m, m - 1 and m - 2. The smallest j is equal to m + 1 and denoted by m_{min}. If we select every j-th bit from the original sequence as the output, where $j < m_{min}$, we can obtain a sequence with initial bits similar to those previously generated by repeating the generation. To avoid this result, we should select every j-th bit with j satisfying $j \geq m_{min}$. We emphasize that the repetition of the generation is typical in cryptography because random numbers are produced when they are needed. The continuous generation of random numbers leads to an undesirable waste of power and simplifies attacks against the TRNG.

The research performed with the described methodology concluded that the GARO with a feedback polynomial (6) is the best candidate for the ASR because it offers the smallest m_{min} (m_{min} = 3). In this case, for every 3rd bit of the generated sequence, the probabilities of obtaining zero and one are equal, although the original sequences fail most of the statistical tests described in the NIST 800-22 statistical test suite [12]. The basic reason are correlations observed between the bits.

3. Implementation of the combined TRNG with ASR in different FPGAs

The TRNG from Figure 1 was implemented using different Xilinx FPGA devices, such as Spartan-3, Spartan-6, Virtex-4 and Virtex-5 [13]. It is important to note that in the experiment these devices are constructed using different technologies. To assess the properties of the generated sequences, the experiment is repeated many times with the same initial conditions [8]. To ensure identical initial conditions, the inverters from Figure 1 are replaced with NAND gates that are triggered by an external signal. The ASR is a GARO with a polynomial (6). All of the inverters in the ASR are replaced in the same manner as for the ROs. Each signal produced by the ring oscillator is perturbed XOR with the signal produced by the ASR. The output is then sampled with a D-type flip-flop. The outputs of the D-type flip-flops are divided into groups and combined with the XOR bits in the group. The number of groups depends on the number of inputs in a single look-up table (LUT). Next, the outputs are sampled by the same clock and combined with the XOR bits to obtain new bits, and so on. The process ends when one bit is produced. The placement of each element depends on the software provided by the manufacturer of the FPGA. Table 1 compares technologies in different Xilinx devices. Table 2 lists the resources of FPGAs used by the proposed combined TRNG with GARO as the ASR (K = 20).

Tab. 1. Comparison of different FPGAs technologies
Tab. 1. Porównanie różnych układów FPGA pod względem technologii

Device	Technology
Spartan 3A (XC3S700A)	90 nm, eight-layer metal process
Spartan 6 (XC6SLX16)	45 nm, nine-metal-layer, dual-oxide process
Virtex 4 (XC4VLX25)	90 nm, eleven-metal-layer, triple-oxide process
Virtex 5 (XC5VLX50T)	65 nm, twelve-metal-layer, second-generation triple-oxide process

Tab. 2. Resources used by the combined RO-based TRNG with a GARO as the ASR
Tab. 2. Wykorzystane zasoby dla generatora TRNG z oscylatorem typu GARO jako ASR dla K = 20

	Spartan 3	Spartan 6	Virtex 4	Virtex 5
Slice Registers	24	46	24	46
Slice LUTS	49	33	50	30
Slice LUT-Flip Flop pairs	95	92	96	93

4. Research results

A comparison of the results for different FPGA devices producing random bits using an RO-based combined TRNG without an auxiliary source of randomness is shown in Figure 2. In each case, the value of m_{min} falls in an irregular manner as the

number of K source oscillators increases. A similar behavior is observed for the combined TRNG using the ASR, but smaller values of m_{min} are achieved more quickly (Figure 3). Spartan-6 performs slightly worse than the other FPGAs, but this difference is irrelevant for K greater than 15, confirming that the combination of the XOR bits from many independent sources of randomness is independent of the technology used. Despite minor differences, all of the cases exhibit a similar trend.

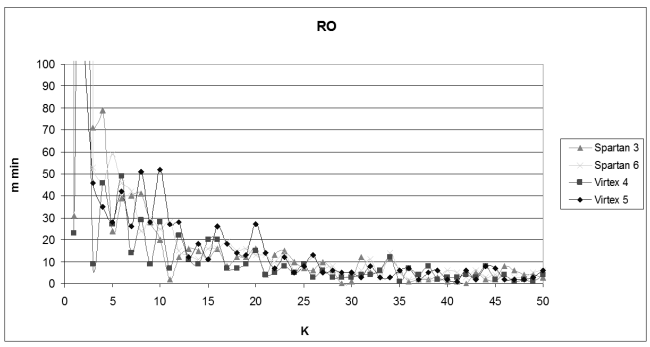

Fig. 2. Values of m_{min} for the TRNGs from Figure 1 without a GARO; f_L = 100 MHz
Rys. 2. Wartość m_{min} dla generatora TRNG z rys. 1 bez GARO; f_L = 100 MHz

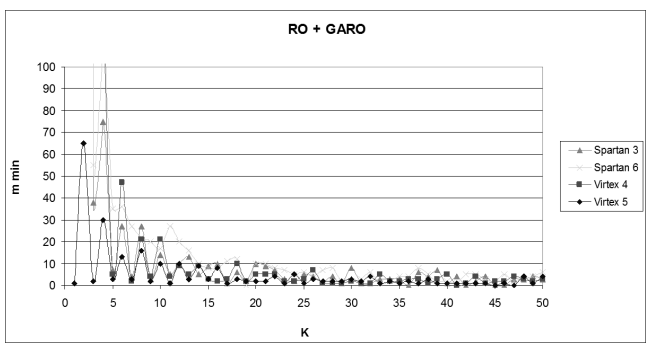

Fig. 3. Values of m_{min} for the TRNGs from Figure 1 with a GARO; f_L = 100 MHz
Rys. 3. Wartość m_{min} dla generator TRNG z rys. 1 z GARO; f_L = 100 MHz

A comparison of Figures 2 and 3 shows that the use of a GARO as an auxiliary source or randomness significantly decreases m_{min} for all of the FPGA devices considered in this paper. The randomness of the output sequences increases with an increasing number of source ring oscillators K. For large numbers of K, the values of m_{min} become similar for the different FPGAs. Large differences in randomness are visible for relatively small values of K. Using an ASR allows the production of random bits that pass all of the statistical tests from the NIST 800-22 test suite at a greater efficient speed. For example, when K = 20, we obtain 10 Mbit/s (m_{min} = 10) for Spartan-3, 10 Mbit/s (m_{min} = 10) for Spartan-6, 20 Mbit/s (m_{min} = 5) for Virtex-4 and 33.33 Mbit/s (m_{min} = 3) for Virtex-5. The results without an ASR for K = 20 are as follows: 5.55 Mbit/s (m_{min} = 18) for Spartan-3, 7.14 Mbit/s (m_{min} = 14) for Spartan-6, 6.25 Mbit/s (m_{min} = 16) for Virtex-4 and 3.56 Mbit/s (m_{min} = 28) for Virtex-5. If the bit rate at the output is fixed, the generator with an ASR passes all statistical tests for smaller K and, consequently, it uses fewer resources (Table 3).

Tab. 3. VIRTEX-5 resources used by the combined RO-based TRNG with and without a GARO as the ASR
Tab. 3. Zasoby VIRTEX-5 zajmowane przez generator TRNG z GARO jako ASR i bez

	TRNG with a GARO as the ASR (K=9)	TRNG without the ASR (K=25)
Slice Registers	22	79
Slice LUTS	23	30
Slice LUT-Flip Flop pairs	68	79

For example, for 10 Mbit/s (m_{min} = 10), all of the statistical tests from the NIST 800-22 test suite are passed for $K = 25$ source generators without the ASR and for $K = 9$ source generators with a GARO as the ASR. Similar results are observed for the remaining FPGA devices.

5. Conclusions

The use of a GARO as an auxiliary source of randomness in the RO-based combined TRNG offers a greater bit rate at the output compared to the TRNG without an ASR and similar resources. This feature is independent of the FPGA type used for the Xilinx products. For cryptographic applications, every j-th bit from the original sequence should be selected as the output, where $j \geq m_{min}$. The value of m_{min} for the generator from Figure 1 with a GARO as the ASR can be very small, such as 2 or 3 for the implementations described in Section 3. In this case, to pass all of the tests from the NIST 800-22 statistical test suite, K = 35 source generators are required. If we accept m_{min} equal to 10, we need only K = 9 source generators. For a very large K, we do not need an ASR because m_{min} has a small value and is independent of the implementation.

Implementation and testing of this generator in FPGAs produced by other manufacturers, is the work for future research.

6. References

[1] Sunar B., Martin W. J. and Stinson D. R.: A provably secure true random number generator with built-in tolerance to active attacks, IEEE Trans., Comput., vol. 56, pp. 109-119, Jan. 2007.
[2] Wold K. and Tan C. H.: Analysis and enhancement of random number generator in FPGA based on oscillator rings, Proc. of ReConFig 2008, Cancun, 2008, pp. 385-390.
[3] Wold K. and Petrović S.: Optimizing speed of a true random number generator in FPGA by spectral analysis, Proc. of Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT'09, 24-26 Nov. 2009, pp. 1105-1110.
[4] Baudet M., Lubicz D., Micolod J. and Tassiaux A.: On the security of oscillator-based random number generators, J. Cryptology, vol. 24, pp. 398-425, 2011.
[5] Jessa M., Jaworski M.: Generacja binarnych ciągów losowych w układzie Virtex-5, Pomiary, Automatyka Kontrola, vol. 56, No. 7, pp. 681-684, 2010.
[6] Jessa M., Jaworski M.: Randomness of a combined TRNG based on the ring oscillator sampling method, Proc. of International Conference on Signals and Electronic Systems, ICSES'10, Sept. 2010, pp. 323-326.
[7] Jessa M., Matuszewski L.: Losowość generatora TRNG zaimplementowanego w FPGA, Pomiary, Automatyka Kontrola, vol. 57, No. 8, pp. 880-883, 2011.
[8] Jessa M., Matuszewski L.: Enhancing the Randomness of a Combined True Random Number Generator Based on the Ring Oscillator Sampling Method, Proc. of International Conference on ReConFigurable Computing and FPGAs, ReConFig'2011, Nov. 30 – Dec. 2, 2011, pp. 274-279.
[9] Bochard N., Bernard F. and Fischer V.: Observing the randomness in RO-based TRNG, Proc. of ReConFig 2009, 9-11 Dec. 2009, pp. 237-242.
[10] Golić J. D.: New methods for digital generation and postprocessing of random data, IEEE Trans., Comput., vol. 55, pp. 1217-1229, Oct. 2006.
[11] Dichtl M. and Golić J. D.: High speed true random number generation with logic gates only, Proc. Workshop Cryptograph. Hardware Embed. Syst. CHES'2007, LNCS 4727, pp. 45-62, 2007.
[12] Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., Leigh S., Levenson M., Vangel M., Banks D., Heckert A., Dray J., Vo S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST special publication 800-22, National Institute of Standards and Technology, 2001, USA, Available at: http://csrc.nist.gov/rng/.
[13] www.xilinx.com

otrzymano / received: 20.05.2013
przyjęto do druku / accepted: 03.07.2013 *artykuł recenzowany / revised paper*