International Journal of Trend in Research and Development, Volume 5(2), ISSN: 2394-9333
www.ijtrd.com

Implementation of True Random Number Generator Using FPGA Based System

[1]Dr. N. G. Narole, [2]Kshanada B. Choudhary,
[1]HOD, [2]Research Scholar,
[1]Department of Electronics & Telecommunication Engineering, [2]Department of Electronics Engineering,
[12]Rajiv Gandhi College of Engineering & Research, Nagpur, India

***Abstract*—** Cryptographic systems have become an integral part of our daily life through the need of security activities such as communication, electronic money systems, disc encryptions. Random numbers is a key component for strengthening and securing the confidentiality of electronic communications and used in many cryptographic applications like key generation, encryption, masking protocols, internet gambling. Unpredictable random numbers are essential for the security of cryptographic algorithms for generating the underlying secret keys. TRUE random number generators (TRNGs) have become an vital component in many cryptographic systems, including PIN/password generation, authentication protocols, key generation, random padding, and nonce generation. The circuit utilizes undetermined random process, usually in the form of electrical noise, as a basic source. Field programmable gate arrays (FPGAs) form an ideal platform for hardware implementations of many of these security algorithms. Proposed TRNG is based on the principle of beat frequency detection for Xilinx-FPGA-based applications.

***Keywords*—**True random number generator (TRNG), Cryptography, Field programmable gate arrays (FPGA), Bit frequency detection (BFD), Dynamic reconfiguration port (DRP).

I. INTRODUCTION

In today‘s world security is of highest importance and hence cryptography plays an important role in computer and networking security. Cryptography is a set of techniques for hiding information. It is employed in several fields as part of security protocols to secure classified information and data. Communication, being an integral part of life, including the internet and other means of communication has given rise to security threats. Cryptography thus provides the necessary protection from the threats by protecting the data, i.e. providing different means and methods of converting data into an unreadable form. The basic aim of cryptography is that the unauthorized user can not accessed data. The content of the data frames should be encrypted with definite pattern. Another application is to ensure that the data must always be acknowledged by the originator of the message.

Random numbers are essential to security because cryptographic systems depend on the existence of some secret data known to authorized users and unpredictable by others and most often random strings are employed to warrant its unpredictability (e.g., in keys, salts, nounces, challenges, initialization vectors, and other one-time quantities)[1].

Cryptographically protected random number generators are important for this purpose. A random number generator is a computational device designed to generate a string of numbers. Methods for generating random has been used from ancient times, including dice, coin flipping, the shuffling of playing cards, the use of yarrow stalks, and many other techniques. There are number of random number generation schemes and Random Number Generators actively used in IT security products. The random numbers generated should be truly random, else they can significantly weaken the security system. They should unpredictable. It has to be designed with a good cryptographic quality. It should be uniformly distributed on a given range and should not be dependent on each other. Thus there is a need for an ideal RNG that satisfies all these requirements [3].Cryptographic quality is achieved by random numbers that satisfy the requirements of cryptographic algorithms.

Random number generators can be classified as either pseudo random number generators or true random number generators. A pseudo random number generator produces a stream of numbers that appears to be random but actually follow predefined sequence. A true random number generator produces a stream of unpredictable numbers that have no defined pattern[3].

1. True Random Number Generators:

There are three commonly used techniques, namely (i)oscillator sampling, (ii)direct amplification and (iii)discrete time chaos. In the oscillator sampling approach, period conversions (i.e. oscillator jitter) in a low frequency clock of low quality factor (Q) is developed by using it to sample a high frequency clock. The direct amplification technique digitizes thermal or shot noise, using a amplifier and comparator. Finally, chaotic systems can be used to produce TRNGs[11].

It is well known that starting with a good mathematical concept (like a LFSR), someone can build a random number generator, called a Pseudo Random Number Generator (PRNG), obtaining the same randomness test results as a good True Random Number Generator (TRNG). In this generator output is a function of the previous one. In most of the cases it can become the greatest vulnerability of the whole cryptographic system. This is why a TRNG consists of three main components as described in Fig. 1.

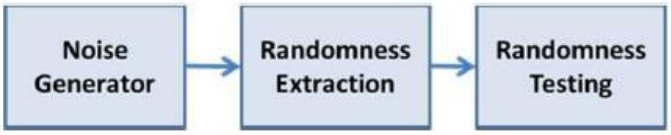

Fig 1. Main components of TRNG

The noise generator is the black box used for generating random sequences. It is based on different kind of physical unpredictable phenomenon, like cosmic radiations, oscillators jitter, sound and light propagation throughout different environments, etc. The randomness extraction box is used to aid the generator to uniformly distribute the 0 and 1 bits along

the output[2].The Randomness Testing block consists of a set of statistical tests, used for testing the randomness of the output. The last two blocks are based on purely mathematical concepts.

2. Pseudorandom Number Generators:

There are many methods to generate pseudorandom sequences, and the classical software based methods, all of which can be implemented in hardware. A common method of producing a PRNG is to use the output of a linear feedback shift register (LFSR)[11].The linear feedback shift register (LFSR) is a common building block for implementing a pseudo-random number generator (PRNG) since it can be compactly constructed from a series of cascaded flip-flops and a few XOR gates. However, the LFSR is generally inadequate by itself for generating high quality random number sequences. Its linear behavior allows an encryption key to be easily recovered if it is used as keystream generator[8]. Although this technique has good statistical properties and leads to very efficient hardware implementations, the Berlekamp–Massey algorithm can be used to efficiently deduce the connection polynomial from the LFSR's output sequence, making it unsuitable for cryptographic applications[11].

Classes of TRNG:

There are two main classes of TRNG are (i)thermal noise based and(ii) chaotic circuit based as shown in Fig 2.

(i) The thermal noise generator amplifies the noise made by electrons flowing into a resistor and converts the noise to a random number. The signal level of the thermal noise is below 1mV, making this approach more susceptible to digital switching noise injection (not random, data dependent) in a large-scale SoC. However, the source of noise used in the thermal RNG (white noise caused by electron moving in the resistor) is truly random and adds to the robustness of the TRNG.

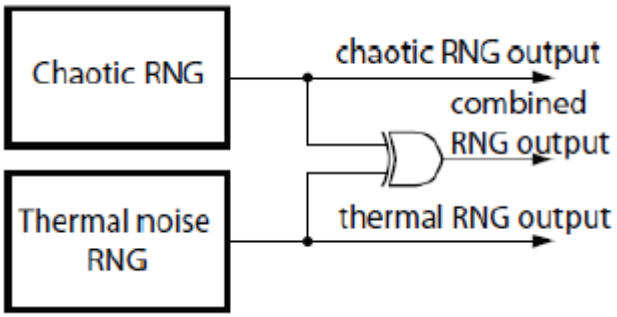

Fig 2. Classes of TRNG

(ii) A chaotic TRNG exploits the unpredictable nature of chaotic oscillators to generate random numbers. Progress in the nonlinear system theory proved that chaotic circuits can be considered truly random. The signal level in a chaotic oscillator can be made much higher that in the thermal TRNG. Hence the Signal to Noise Ratio (SNR) is improved, minimizing data dependency. The system minimizes the effect of supply and substrate noise by combining the 2 non-correlated outputs of both TRNG with an XOR gate (Figure 1) and also by circuit techniques like cascading. Since the transfer function from the supply to the output is different for each circuit and the 2 TRNG use different noise sources to generate the random bits, the 2 bit streams are not correlated[9].

FPGA is used increasingly because of its advantages in performance, design time, power consumption, flexibility, cost or chip area over other systems based on microprocessor, DSP or VLSI. With a FPGA-based random number generation, many cryptographic applications can be effectively implemented using FPGA[10]. Over the decade, many TRNGs have been proposed. For example, Tsoi and Leung[11] proposed a FPGA-based TRNG based on the oscillator phase noise. In their proposal, a high-quality random bit stream can be generated by sampling an accurate high-frequency clock using a ring oscillator formed by gates in the FPGA together with external resistors and capacitors. However, the maximum output date rate of the generator is only 4.7Kbps which is not high enough for many cryptographic applications. Besides, the TRNG can be easily tampered with because it has external components. In Epstein and Hars,[12] a TRNG based on digital circuit metastable event was presented. However, the proposed generator can only be successfully implemented in some lowend digital integrated circuits but not in modern FPGAs because CMOS circuits in modern FPGAs are so fast that the probability of a metastable event occurring in any gate in the FPGAs is very small.

Reconfigurable devices have become an integral part of many embedded digital systems, predicted to become the platform of choice for general computing in the near future. Being primary prototyping devices, reconfigurable systems including FPGAs are widely involved in cryptographic applications, as they can provide acceptable to high processing rate at much lower cost and faster design cycle time [4].

FPGAs being flexible in terms of programming and implementation of several algorithms and functions have been employed for implementing cryptographic algorithms for quite a long time. They are widely used in encryption and R&D applications. FPGAs provide performance flexibility and benefits being compared to applications specific integrated circuit (ASICs). Conventionally, ASICs was used more for the cryptographic implementations [3]. Later, due to greater flexibility and reprogram ability, it has become easier to modify algorithms and program them on FPGAs. The development of an algorithm is faster and allows for a shorter time to exchange on FPGA.

Focus is to design an improved field-programmable gate array (FPGA) based TRNGs, using purely digital components. Using digital building blocks for TRNGs has the advantage that the designs are relatively simple and well suited to the FPGA design flow, as they can suitably leverage the CAD software tools available for FPGA design. However, digital circuits exhibit comparatively limited number of sources of random noise, e.g., metastability of circuit elements, frequency of free running oscillators, and jitters (random phase shifts) in clock signals.

Because of its flexibility and fast time to market, FPGA has become a popular platform for implementing many cryptographic systems that include TRNGs as an essential block. It is essential to develop new FPGA TRNG solutions because: (i) not all the hardware TRNG methods available for ASICs or other platforms are adaptable to FPGA implementation; (ii) the existing FPGA TRNGs have some shortcomings in terms of the throughput-per-unit-area and can be improved; and (iii) active component attacks as well as changes in operational conditions such as variations in temperature and voltage supply may bias and disturb the random property TRNGs output bitstream. Since most of the TRNGs operate in an open-loop fashion, it is important to incorporate a mechanism to constantly provide a feedback signal to adaptively adjust the TRNG system parameters to

increase its output bit randomness[7]. A relatively recent enhancement to FPGA capabilities is Dynamic Partial Reconfiguation (DPR) or Runtime Partial Reconfiguration (RPR). It is the ability to modify (mostly through the addition of functionality) the existing circuit on the FPGA, through "partial reconfiguration" (PR) of the FPGA at run time. DPR allows designer to use smaller devices, reduce power consumption and improve system upgradability. DPR allows modifications to predefined portions of the FPGA logic fabric on-the-fly, without affecting the normal functionality of the FPGA.

TRNG circuit implementation for Xilinx-FPGA-based applications, which has a tunable jitter control capability based on dynamic partial reconfiguration (DPR) available on Xilinx FPGAs. Design techniques exist to prevent any malicious manipulations via DPR which in other ways may affect the security of the system termed as Hardware Trojan Insertion [4].

II. BACKGROUND OF PROPOSED WORK

This section briefly describes the basic Single-Phase BFDTRNG Model:

The BFD-TRNG circuit [5] is a fully digital TRNG, which relies on jitter extraction by the BFD mechanism, originally implemented as a 65-nm CMOS ASIC. The structure and working of the basic BFD-TRNG can be summarized as follows, in conjunction with Fig.3.

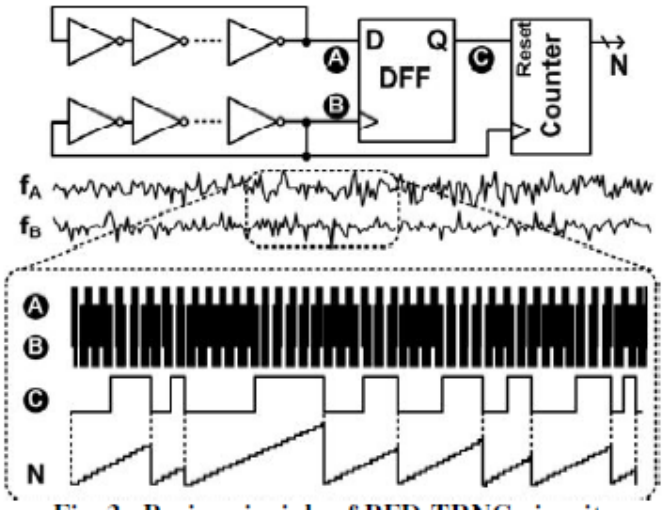

Fig 3. Basic principle of BFD-TRNG circuit.

1. The circuit consists of two quasi-identical ring oscillators (let us term them as A and B), with similar construction and placement. Due to inherent physical randomness originating from process variation effects associated with deep sub micrometer CMOS manufacturing, one of the oscillators (e.g., A) oscillates slightly faster than the other oscillator (B). In addition, the authors [5] proposed to employ trimming capacitors to further tune the oscillator output frequencies.

2. The output of one of the oscillators is used to sample the output of the other, using a D flip-flop (DFF). Without loss of generality, assume that the output of A is fed to the D-input of the DFF, while the output of B is connected to the clock input of the DFF.

3. At certain time intervals (determined by the frequency difference of the two oscillators), the faster oscillator signal passes, catches up, and overtakes the slower signal in phase. Due to random jitter, these capturing events happen at random intervals, called "beat frequency intervals." As a result, the DFF outputs a logic-1 at different random instances.

4. A counter is controlled by the DFF increments during the beat frequency intervals and gets reset due to the logic-1 output of the DFF. Due to the random jitter, the free running counter output rises to different peak values in each of the count-up intervals before getting reset.

5. The output of the counter is sampled by a sampling clock before it reaches its maximum value.

6. The sampled response is then serialized to obtain the random bit stream.

One limitation of BFD-TRNG circuit is[14] that its statistical random property is dependent on the design quality of the ring oscillators. Any design bias in the ring oscillators might critically affect the statistical random property of the bit stream generated by the TRNG. Designs with the same number of inverters but different placements resulted in changes in counter maximas.

The following are some techniques to eliminate the above problem during implementation on FPGA.

A. Random source:

Most FPGAs embed Digital Clock Manager (DCM) modules providing a wide range of powerful clock management features such as clock-deskew, frequency synthesis and phase shifting. In frequency synthesis, the DCM can generate a wide range of output clock frequencies by performing flexible clock multiplication and division of an input clock [13].Since Delay-Locked Loops (DLL) is used in the frequency synthesis, jitter in the period of the clock generated is inevitable. For example, according to the Xilinx FPGA data sheet, if the parameters for the frequency synthesis are chosen carefully, we can generate an output clock with a large period jitter which can be treated as a random source for the generation of random numbers.

B. Randomness extraction from the DLL-generated clock jitter:

The basic principle behind our method is to extract the randomness from the period jitter of a DLL-generated clock synthesized by a DCM in a FPGA. The jitter in the DLL generated clock, F1 is exploited by using it to sample an accurate high-frequency reference clock, Fh. Since the duty cycle of Fh may not be 50%, Fh will have unequal probability of being zero and one, leading the output random bit stream to be biased to either zero or one. Besides, if period jitter in F1 is not large enough (compared to the period of Fh), there will be correlation between the sampled bits and so the output random bit stream can be predicted to some extent from its previous bits. To remove the biases and correlation in the output, different kinds of de-skewing techniques such as parity filter, Von Neumann de-skew filter and strong mixing [13].

III. PROPOSED ARCHITECTURE

DCM can generate a wide range of output clock frequencies by performing flexible clock multiplication and division of an input clock. Since Delay-Locked Loops (DLL) is used in the frequency synthesis, jitter in the period of the clock generated is inevitable. For example, according to the Xilinx FPGA data sheet, if the parameters for the frequency synthesis are chosen carefully, we can generate an output clock with a large period jitter which can be treated as a random source for the generation of random numbers [10].

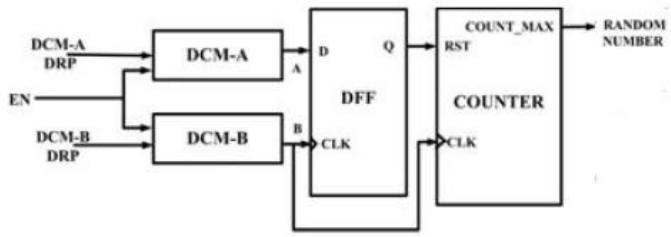

Figure 4: Proposed Architecture

Fig. 4 shows the overall architecture of the proposed TRNG. In place of two ring oscillators, two DCM modules generate the oscillation waveforms. In the proposed design, the source of randomness is the jitter The DCM modules allows increased designer control over the clock waveforms, and their use eliminates the need for initial calibration [5]. Tunability is established by setting the DCM parameters on-the-fly using DPR capabilities using DRP ports. This capability provides the design greater flexibility than the ring-oscillator-based BFD TRNG. The difference in the frequencies of the two generated clock signals is captured using a DFF. The DFF sets when the faster oscillator completes one cycle more than the slower one (at the beat frequency interval). One of the generated clock signals applied to counter and the counter is reset when the DFF is set. Effectively, the counter increases the process of the generated random numbers.

CONCLUSION

The randomness of BFD-TRNG depend on the design quality of the ring oscillators. As the ring oscillators are free running it is difficult to design and implement the circuit on FPGA platform with same number of inverters at different placements. Our goal is to design, analyze, and implement an easy-to-design, improved, low- overhead, and tunable TRNG for the FPGA platform. Proposed architecture allows on-the-fly tuning ability of statistical qualities of a TRNG by utilizing DPR capabilities of modern FPGAs for changing the digital clock manager (DCM) modeling parameters. Xilinx clock management tiles (CMTs) contain a dynamic reconfiguration port (DRP) which allows DPR to be performed through much simpler means.

References

[1] Sergio Callegari “Evaluation of a couple of True Random Number Generators with liberally licensed hardware,firmware, and drivers”, IEEE, 2015.
[2] Andrei Marghescu, Paul SvastaInto “Generating True Random Numbers - a Practical Approach using FPGA”,IEEE 21st SIITME, 2015.
[3] Prassanna Shanmuga Sundaram, “Development of a FPGA-based True Random Number Generator for Space Applications,” Master thesis in Electronics Systems at Linköping Institute of Technology.
[4] P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, “A PUF enabled secure architecture for FPGA- based IoT applications,” IEEE Transactions on Multi- Scale Computing Systems., vol. 1, no. 2, April–June 2015.
[5] Q. Tang, B. Kim, Y. Lao, K. K. Parhi, and C. H. Kim, “True random number generator circuits based on single- and multi-phase beat frequency detection,” in Proc. IEEE Custom Integr. Circuits Conf., Sep. 2014.
[6] J. Von Neumann, “Various techniques used in connection with random digits,” Nat. Bureau Standards Appl. Math. Ser., vol. 12.
[7] Mehrdad Majzoobi and Farinaz Koushanfar and Srinivas Devadas, “FPGA-based True Random Number Generation using Circuit Metastability with Adaptive Feedback Control”, Massachusetts Institute of Technology, CSAIL Cambridge.
[8] Juan C. Cerda, Chris D. Martinez, Jonathan M. Comer, and David H.K. Hoe, “An Efficient FPGA Random Number Generator using LFSRs and Cellular Automata”, IEEE, 2012.
[9] Vincent von Kaenel, Toshinari Takayanagi, “Dual True Random Number Generators for Cryptographic Applications Embedded on a 200 Million Device Dual CPU SoC”, IEEE Custom Intergrated Circuits Conference (CICC), 2007.
[10] Sammy H. M. Kwok, Edmund Y. Lam, “FPGA-based High-speed True Random Number Generator for Cryptographic Applications”, IEEE, 2016.
[11] K. H. Tsoi and K. H. Leung, "Compacted FPGA-based and Pseudo Random Generators", Proc. of the 11th Annual IEEE Symposium on Field- Programmable Custom computing Machines (FCCM'03), 2003.
[12] Michael Epstein and Laszlo Hars, "Generator based on Digital Circuit Artifacrts", Proc. of CHES 2003, LNCS 2779, pp. 152-165, 2003.
[13] Eastlake, Corker & Schiller, "Randomness recommendations for Security", RFC 1750, Dec. 1994.
[14] Anju P. Johnson, Rajat Subhra Chakraborty, “An Improved DCM Based Tunable True Random Number Generator for Xilinx FPGA”, IEEE transaction on circuits and systems -II, vol. 64, no. 4, april 2017.