Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 14, Issue. 01 : 2023
ISSN : **1906-9685**

DEVELOPMENT OF A TRULY RANDOM NUMBER GENERATOR USING AN FPGA-BASED SYSTEM

#1BOORLA SANTHOSH, ***Associate Professor,***
Department of Electronics and Communication Engineering,
#2G Y RAMYA KRISHNA, ***Assistant Professor,***
Department of Electronics and Communication Engineering,
MOTHER THERESA COLLEGE OF ENGINEERING AND TECHNOLOGY, PEDDAPALLY, TS.

Abstract: Cryptographic technologies have been widely implemented due to the necessity for security measures in a variety of industries, including communication, computerized payment systems, and disk encryption. Random numbers are used in a variety of cryptographic applications, including online gambling, masking protocols, encryption, and key generation, to improve and protect the confidentiality of electronic communications. The presence of predictable random numbers is a serious flaw in cryptography systems that generate private keys. TRNGs, or true random number generators, are necessary for the operation of many cryptographic systems. These elements are used to generate cryptographic keys, random padding, nonces, authentication techniques, passwords, and personal identification numbers (PINs). The fundamental source of the issue is electrical noise, a type of stochasticity present in electronic systems. The majority of these security controls are easily implemented with hardware field-programmable gate arrays (FPGAs). The pulse frequency detection approach is used by the True Random Number Generator (TRNG) intended for Xilinx-FPGA applications.

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

1. INTRODUCTION

Encryption has become critical to the security of modern computer networks and systems. Cryptography is a dependable method for ensuring the integrity and secrecy of sensitive data. Many firms use this technology to protect sensitive data on a regular basis. The Internet and other communication channels have facilitated the emergence of human connection, which has given rise to new security problems. Cryptography safeguards data against potential weaknesses by offering a number of methods for converting data into an unintelligible format. Cryptography's fundamental goal is to prevent unwanted access to sensitive data. The encryption of data frames necessitates a distinct implementation technique. Creating a system that ensures the message sender receives reliable confirmation that the data was received is another conceivable use case.

Only authorized parties should have access to secret data that is resistant to guessing by unauthorized parties in order to ensure the security of a cryptographic system. Numerous cryptographic applications, such as keys, salts, nonces, challenges, initialization vectors, and other unique variables, rely heavily on random strings. The goal of using random strings in these situations is to make them very unpredictable, defying any attempts to forecast or anticipate them.

This needs the usage of encrypted random number generators in order to perform properly. A random number generator is a piece of software that generates numbers with a high degree of randomness. Since antiquity, real random number generators have been widely employed. Dice, money, a deck of cards, a yarrow stem, and other objects arrive at random. Information technology security systems employ a variety of methodologies and random number generators. The reliability of security measures may suffer if generated random numbers lack genuine randomization. These are likely to be difficult to decipher. It is necessary to incorporate strong security protocols. To reduce interference, resources should be dispersed consistently across a specific area. This highlights the significance of having an ideal random number generator (RNG) that meets these parameters. The careful selection of random numbers that correspond to the unique requirements of cryptographic algorithms is the fundamental way for generating secure encrypted data.

There are two types of random number generators: trustworthy and dishonest. Pseudorandom number generators are ways for generating seemingly random but actually predefined numerical sequences. TRNGs, or true random number generators, are devices that create random numbers without the use of patterns or clustering.

True Random Number Generators:

Discrete time chaos, direct amplification, and oscillator sampling are all common applications. It is possible to generate a low-frequency clock with a low quality factor (Q) using samples from a higher-frequency oscillator. As a result, oscillations will exhibit periodic changes, often known as oscillator jitter. An amplifier and a comparator are used in direct amplification to transform analog heat or discharge noise into digital signals. Finally, True Random Number Generators (TRNGs) may incorporate chaos systems into their design.

The Linear Feedback Shift Register (LFSR) mathematical concept can be utilized to build a Pseudorandom Number Generator (PRNG) that delivers randomness test results comparable to those of a trustworthy True Random Number Generator (TRNG). The output of this generator is determined by the output of the preceding generator. Under most conditions, it has the potential to become the system's most critical vulnerability. As a result, as seen in Figure. A True Random Number Generator (TRNG) is made up of three main components. The purpose of this study is to look into how social media use affects mental health outcomes.

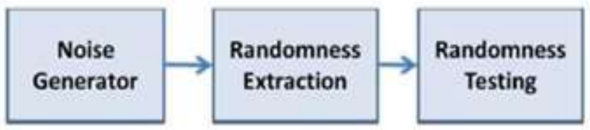

Fig 1. TRNGs (True Random Number Generators) Explained

The noise generator is a device that can generate random sequences. This phenomena involves a wide range of stochastic physical processes, including the impact of cosmic radiation and the transmission of sound and light via various media. The unpredictability Testing segment includes a number of statistical tests to analyze the unpredictability of the output. By making it easier to extract random information from the generator's output, the Randomness Extraction Box enhances consistency. Only arithmetic questions are covered in the latter two sections.

Pseudorandom Number Generators:

There are several methods for creating pseudorandom sequences, including software-based methods that may be implemented in hardware in the future. A linear feedback shift register (LFSR) is created by compactly arranging flip-flops and XOR gates. As a result, it is widely used in the development of pseudo-random number generators (PRNGs). It is a standard and accepted way to construct a pseudorandom number generator (PRNG) using data from a linear feedback shift register (LFSR). The Linear Feedback Shift Register (LFSR) alone is insufficient for generating high-quality random patterns. When the keystream generator is used, the linear property makes decryption easier. The Berlekamp-Massey approach can be used to find the connection polynomial of a linear feedback shift register (LFSR) by studying the LFSR output sequence. Despite its excellent statistical features and efficient hardware implementations, this approach is not appropriate for cryptography applications.

Classes of TRNG:

as illustrated in Figure 1. The most prevalent types are thermal noise True Random Number Generators (TRNGs) and random circuit True Random Number Generators (TRNGs).

The thermal noise generator amplifies and converts the noise generated by electrons passing through a resistor into a random number. This approach is particularly prone to non-random, data-dependent digital switching noise in big System-on-Chip (SoC) designs since the thermal noise level is less than 1 mV. The TRNG, on the other hand, is more dependable due to the inherent randomness of white noise generated by electrons going through a resistor.

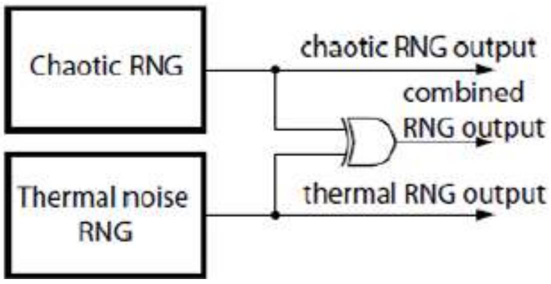

Fig 2. TRNG (True Random Number Generator) Varieties

A chaotic True Random Number Generator (TRNG) generates really random numbers by utilizing the inherent unpredictability of chaotic oscillators. The discovery of chaotic circuits in the study of nonlinear systems was a shocking breakthrough that called our current understanding into question. A thermal True Random Number Generator (TRNG) cannot generate a signal with the same magnitude as a chaotic oscillator. As the signal-to-noise ratio (SNR) increases, so does the data's reliability. Two successful circuit solutions for decreasing supply and substrate noise are using an XOR gate to join disconnected True Random Number Generator (TRNG) outputs and a cascade configuration (Figure 1). To generate random bits, each True Random Number Generator (TRNG) has a different noise source, and each input-to-output converting circuit has a different transfer function. This result implies that no relationship exists between the two data sets.

In terms of performance, design time, power consumption, cost, and device size, FPGA-based systems outperform microprocessor, DSP, and VLSI-based systems. Numerous cryptography applications stand to benefit considerably when a random number generator is built with a field-programmable gate array (FPGA). TRNGs (True Random Number Generators) have grown in popularity during the last decade. Tsoi and Leung (year) presented a TRNG based on field-programmable gate arrays (FPGA) that uses the intrinsic phase noise of oscillators as a source of entropy. It is recommended to sample a precise high-frequency clock using an FPGA-gated ring oscillator in conjunction with external resistors and capacitors to generate a high-quality random bit stream. The generator's output rate of 4,7 Kbps is regarded insufficient for most cryptographic applications. The True Random Number Generator (TRNG) is transparent, allowing for its modification. Epstein and Hars used the inherent instability of digital circuitry to build a True Random Number Generator (TRNG). The generator in question, however, is incompatible with the most advanced Field-Programmable Gate Arrays (FPGAs) and can only function optimally with a limited number of low-cost digital integrated circuits (ICs). The possibility of a metastable event occurring within the gates of current field-programmable gate arrays (FPGAs) is exceedingly low due to the fast speed of their complementary metal-oxide-semiconductor (CMOS) circuitry.

In the realm of embedded digital systems, programmable devices are replacing traditional computer platforms. Because of its potential to deliver adequate to high operating efficiency at reduced prices and significantly faster design cycle times, reconfigurable systems, such as Field-Programmable Gate Arrays (FPGAs), are frequently employed in the cryptography business.

Because of their programmability and adaptability in executing various algorithms and operations, Field-Programmable Gate Arrays (FPGAs) have been widely used in the development of cryptographic systems. They have a wide range of uses in research and security. FPGAs outperform application-specific integrated circuits (ASICs) in terms of speed and versatility. ASICs (Application-Specific Integrated Circuits) were commonly utilized in the past to overcome cryptography concerns. Reprogrammability for Field-Programmable Gate Arrays (FPGAs) has recently advanced to the point where writing new code and tweaking algorithms is now easier. Using a Field-Programmable Gate Array (FPGA) accelerates algorithm creation and exchange.

The goal is to improve TRNGs (True Random Number Generators) by developing digital variations based on FPGAs (Field-Programmable Gate Arrays). Using Computer-Aided Design (CAD) software tools created expressly for FPGA design, it is possible to create True Random Number Generators (TRNGs) that are user-friendly and compatible with the FPGA design process. It is possible for digital circuitry to experience random noise on occasion. Oscillations, which are defined as irregular phase shifts in clock signals, are an additional source of random noise in addition to the frequency of free-running oscillators and the metastability of circuit components.

The field-programmable gate array (FPGA) has become a popular alternative for the creation of cryptographic systems that rely primarily on true random number generators (TRNGs) due to its ability to scale and hasten market entry. Although FPGAs are capable of running a wide range of hardware true random number generator (TRNG) methods, several techniques are platform-specific and thus inapplicable. Existing FPGA TRNGs can have their throughput per unit area increased. TRNGs can generate unpredictable output bitstreams that are impacted by external conditions such as temperature and voltage changes. FPGAs can be used with their circuits undergoing a method known as partial reconfiguration (PR), which allows the insertion of modifications and, in some cases, new functionality. DPR approaches enable architects to build systems with fewer moving parts, smaller hardware footprints, and lower power consumption. The logical structure of a Field-Programmable Gate Array (FPGA) can be changed without compromising its functioning using Dynamic Partial Reconfiguration (DPR).

The TRNG circuit implementation's dynamic partial reconfiguration (DPR) capabilities can be utilized to adjust the jitter control behavior for Xilinx FPGA-based applications. It is possible to avoid harmful DPR manipulations that could have a negative impact on the system by making sensible design decisions. Creating a hardware system that can act as a Trojan horse while remaining undetected.

2. BACKGROUND OF PROPOSED WORK

In this section, we will present a high-level summary of the BFDTRNG Model, which contains only one phase.

The BFD-TRNG circuit is a completely digital True Random Number Generator (TRNG) that reduces jitter using the BFD approach. A 65-nanometer Complementary Metal-Oxide-Semiconductor (CMOS) Application-Specific Integrated Circuit (ASIC) was used in the initial implementation. There is no need to rewrite the user's text.

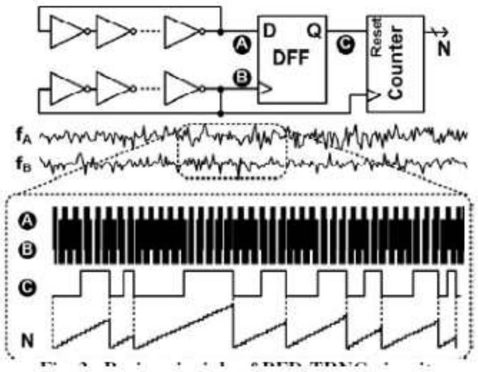

Fig 3. The BFD-TRNG requires a number of key components to function.

- ➢ The circuit includes two ring oscillators labeled A and B. The configurations and techniques of manufacture are virtually identical. It is revealed that oscillator A has a little greater oscillation frequency than oscillator B due to physical randomization caused by process variation effects in the manufacture of deep sub millimeter CMOS. To improve the precision of the oscillator's output frequencies, the authors proposed utilizing pruning capacitors.

- The output of one oscillator is compared to the output of another using the D flip-flop (DFF). For the sake of clarity, assume that the DFF's clock and D-inputs are connected to outputs A and B, respectively.
- The frequency difference between the two oscillators defines their temporal relationship, with the faster oscillator's signal going ahead of, aligning with, and then passing ahead of the slower oscillator's signal. The DFF generates logic-1 signals at irregular intervals with erratic behavior due to system disruption. As a result, documented occurrences take place at irregular intervals known as beat frequency intervals.
- During beat frequency gap periods, the DFF logic-1 output is used to increment from 0 to 1. Because jitter can occur at any time, the output of a free-running counter spikes before resetting at the end of each count-up cycle.
- A sampling device tracks the counter's output as it rises.
- A random bit sequence is produced by the serialized and sampled answer.
- The ring oscillator architecture makes it difficult for the BFD-TRNG circuit to create truly random numbers. Manufacturing flaws in a True Random Number Generator's (TRNG) ring oscillators could jeopardize the generated bit stream. The number of inverters in both systems was the same, but the maximal counters were different.
- The ideas listed below are feasible solutions to the previously outlined FPGA difficulty.

Random source:

Pre-installed Digital Clock Manager (DCM) modules are found in a considerable fraction of Field-Programmable Gate Arrays (FPGAs). These modules efficiently perform a wide range of extremely sophisticated clock management tasks, including as phase shifting, frequency creation, and clock-deskew. The Digital Clock Manager (DCM) is capable of generating a wide range of clock frequencies required for frequency synthesis. This is accomplished by dividing and multiplying an input clock. When Delay-Locked Loops (DLL) are used in frequency synthesis, the resulting clock signal is inherently unstable. Using the information on the Xilinx FPGA data page, an output oscillator that generates random numbers can be built by appropriately setting the frequency synthesis parameters. This oscillator has a significant amount of period jitter.

Randomness extraction from the DLL-generated clockjitters:

The combination of a Digital Clock Manager (DCM) and a Dynamic Link Library (DLL) is the key concept of our methodology for reducing the inherent uncertainty in the timing variations of an FPGA-implemented oscillator. The jitter of the F1 signal is used to sample a clock signal (F1) generated by a DLL from a precisely scheduled high-frequency reference clock (Fh). Because there is no duty cycle guarantee of 50%, the probability that Fh will take on the values 0 or 1 varies. As a result, the generated random bit stream will have a bias toward 0 or 1. Furthermore, if the period variation in F1 is too big in comparison to the period variation in Fh, it is possible to forecast the output random bit stream in part by scrutinizing the bits that came before it. To reduce output bias and association, several de-skewing techniques are used, including robust mixing, the Von Neumann de-skew filter, and parity filtering.

3. PROPOSED ARCHITECTURE

The DCM may generate a wide range of output clock frequencies by executing repeated division and multiplication operations on an input clock. Using Delay-Locked Loops (DLL) during frequency generation reduces the predictability of the clock period. According to the Xilinx FPGA data sheet, by carefully setting the frequency synthesis parameters, it is feasible to create an output clock with observable period jitter. This turbulence can then be used to produce random numbers.

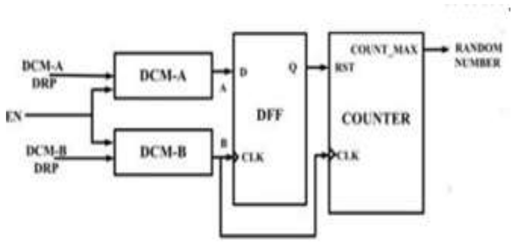

Figure 4: It is interesting to study the planning and analysis of construction projects.

As an example. Figure 4 displays the general layout of the proposed True Random Number Generator (TRNG). Oscillation patterns are formed using two DCM modules rather than two ring oscillators. The potential future consequences are highly unpredictable due to the intrinsic volatility of the suggested system. Designers can further alter clock waveforms using DCM modules without requiring initial calibration. DCM parameters can be changed in real time thanks to the DPR ports and DPR capabilities. The BFD TRNG is less versatile than the Differential Frequency Filter (DFF), which can distinguish between the two produced clock signals. The DFF also computes the beat frequency interval, or the time it takes the faster oscillator to complete one full cycle more than the slower oscillator. A clock signal is used to reset the DFF's value and therefore initialize it. When a counter is used, it takes significantly less time to produce random numbers.

4. CONCLUSION

The BFD-TRNG's randomness is affected by the design of the ring generators. Because ring oscillators are self-contained, designing and building the circuit on an FPGA platform may be tricky, especially if you want to install the same number of inverters in several locations. The goal of this project is to create a more efficient True Random Number Generator (TRNG) that is simple to configure and implement on Field-Programmable Gate Array (FPGA) devices. With the help of modern field-programmable gate arrays (FPGAs) and their dynamic partial reconfiguration (DPR) capabilities, the proposed architectural strategy aims to rethink the digital clock management (DCM) system. This feature allows for dynamic customization of the True Random Number Generator's (TRNG) randomization parameters. The dynamic reconfiguration port (DRP) offered on Xilinx clock management tiles (CMTs) simplifies dynamic partial reconfiguration (DPR).

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.