IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
Research Paper **Journal Volume 11, Iss 07, 2022**

TRUE RANDOM NUMBER GENERATOR IMPLEMENTATION IN AN FPGA-BASED SYSTEM

[#1]MAVURAPU SWAPNA, *Assistant. Professor,*

[#2]KOTICHINTHALA NEETHIKA, *Assistant. Professor,*

Department of Electronics Communication Engineering,

SREE CHAITANYA INSTITUTE OF TECHNOLOGICAL SCIENCES, KARIMNAGAR, TS.

Abstract: The need for security in industries such as communication, computerized payment systems, and disk encryption has resulted in the widespread use of cryptographic technologies. Random numbers are used in a variety of cryptographic applications, ranging from key generation and encryption to masking protocols and online gambling, to improve and protect the privacy of electronic communications. Predictable random numbers are a significant flaw in cryptography systems that generate secret keys. TRNGs (true random number generators) are required for the operation of many different cryptographic systems. These are used to generate PINs and passwords, authentication tools, keys, random padding, and nonces. Electrical noise, a type of randomness in electronics, is mostly to blame for the failure. Most of these security measures can be implemented in hardware using field-programmable gate arrays (FPGAs). The TRNG recommended for Xilinx-FPGA applications is based on the pulse frequency detection method.

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

1. INTRODUCTION

Encryption is now an essential part of keeping our computers and networks secure. Cryptography is a secure means of concealing information. It is a typical component of the security armory used by a wide range of businesses to protect critical information. As a result of human connection via the Internet and other modes of communication, new security vulnerabilities have evolved. Cryptography protects information from potential threats by providing a number of methods for converting it into an unreadable format. The basic goal of cryptography is to keep sensitive information out of the wrong hands. Encrypting the information contained in data frames necessitates a specialized implementation strategy. Another application is to ensure that the sender of a message always confirms receipt of the data.

Any cryptographic system requires secret information that is only accessible to authorized users and cannot be guessed by unauthorized parties. Random strings are widely used in keys, salts, nounces, challenges, initialization vectors, and other one-time amounts to ensure that they cannot be guessed.

This requires encrypted random number generators to function properly. A random number generator is a piece of software that can be used to produce random numbers. True random number generators have been in use since the dawn of time. Dice, cash, a deck of cards, a stalk of yarrow, and other items appear. IT security systems employ a variety of approaches and random number generators. Security can be jeopardized if the generated random numbers are not truly random. They must be difficult to crack. It must include strong security procedures. It must be distributed evenly within a specified territory without interfering with one another. This emphasizes the significance of having a flawless RNG that meets these conditions. The key to producing secure encrypted data is to select

IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES
ISSN PRINT 2319 1775 Online 2320 7876
Research Paper Journal Volume 11, Iss 07, 2022

random numbers that correspond to the criteria of cryptographic technologies.

There are two kinds of random number generators: authentic and fraudulent. Pseudorandom number generators generate sequences of numbers that appear to be chosen at random but actually follow a pattern. True random number generators generate numbers that do not cluster or repeat.

True Random Number Generators:

Methods such as (i) oscillator sampling, (ii) direct amplification, and (iii) discrete time chaos are widely used. It is feasible to build a low-frequency clock with a low quality factor (Q) by extracting samples from a higher-frequency oscillator. As a result, there will be period fluctuations, often known as oscillator jitter. Direct amplification uses an amplifier and a comparator to transform analog heat or discharge noise into digital signals. Finally, chaos systems can be used to create TRNGs.

An LFSR is a mathematical notion that can be used to generate a PRNG that delivers the same randomness test results as a good TRNG. The previous generator's output determines the output of this one. It has the potential to become the system's single largest weakness in the vast majority of scenarios. As a result, as shown in Fig. 1, there are three basic components that comprise a TRNG. 1.

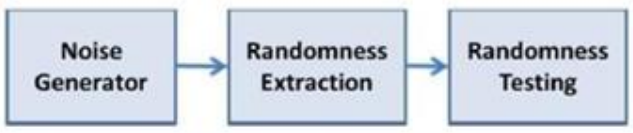

Fig 1. What You Should Know About TRNG

The noise generator is the "black box" that generates random sequences. It is based on a wide range of unpredictable physical phenomena, such as sound and light propagation over various media and the impacts of cosmic radiation. The Randomness Testing section includes several statistical analyses to establish how random the outcome is. The Randomness Extraction Box can be used to extract random bits of information from the generator's output to make it more uniform. The final two sections are entirely made up of mathematical exercises.

Pseudorandom Number Generators:

There are several methods for generating pseudorandom sequences, some of which are typical software-based methods that can be implemented in hardware. A linear feedback shift register (LFSR) can be produced in a compact configuration by stacking flip-flops and XOR gates, making it a frequent component for constructing pseudo-random number generators (PRNGs)[11]. It is common practice to generate a PRNG from the data provided by an LFSR. Despite this, the LFSR alone is insufficient to generate good random patterns. When used as a keystream generator, its linear property aids decryption. The Berlekamp-Massey algorithm can evaluate the LFSR output sequence to identify the connection polynomial. Despite its strong statistical properties and efficient hardware implementations, this approach is not suited for use in cryptography.

Classes of TRNG:

As seen in Figure 1. The two most prevalent types are random circuit TRNGs and thermal noise TRNGs.

The thermal noise generator amplifies the noise produced by electrons traveling through a resistor and converts it to a random number. Because the thermal noise level is less than 1mV, this method is more vulnerable to non-random, data-dependent digital switching noise in a large SoC. The TRNG, on the other hand, is more reliable since the white noise it makes from electrons flowing through a resistor is truly random.

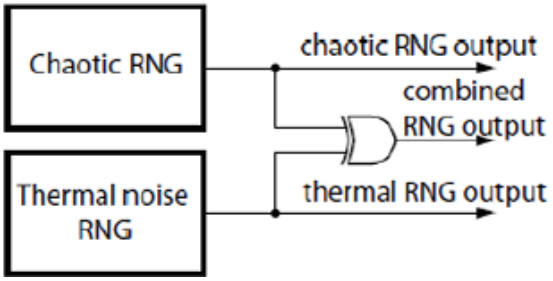

Fig 2. TRNG Varieties

A chaotic TRNG uses the unpredictable nature of chaotic oscillators to generate truly random numbers. We've learnt that chaotic circuits are utterly unexpected as our understanding of nonlinear systems has grown. The signal of a chaotic oscillator can be substantially stronger than that of a thermal TRNG. The signal-to-noise ratio (SNR) improves as a result, making the data more robust. Two circuit approaches that help to reduce supply and substrate noise include cascade and the use of an XOR gate to merge unconnected TRNG outputs (Figure 1). Each TRNG generates random bits using a different noise source, and each circuit has its own transfer function from input to output. This shows that no relationship exists between the two data streams.

FPGA-based systems outperform microprocessor, DSP, and VLSI-based systems in terms of performance, design time, power consumption, flexibility, cost, and device size. A random number generator constructed on an FPGA has numerous cryptography applications. TRNGs have proliferated throughout the last decade. Tsoi and Leung, for example, suggested an FPGA-based TRNG that uses oscillator phase noise as its seed. For a high-quality random bit stream, they recommend sampling an accurate high-frequency clock using an FPGA-gated ring oscillator, some external resistors, and capacitors. The generator's fastest output speed is 4.7Kbps, which is significantly too slow for most cryptographic applications. The TRNG is partially transparent, allowing for alterations. Epstein and Hars created a TRNG based on the observation that digital circuits are prone to instability. However, the proposed generator is incompatible with cutting-edge FPGAs and can only run smoothly on a limited number of low-cost digital ICs. Because of the tremendous speed of its CMOS circuitry, there is virtually no chance of a metastable event occurring in any of the gates of modern FPGAs.

Many embedded digital systems are shifting away from traditional computer platforms and toward programmable devices. Because of their potential to provide acceptable to high working rates at much lower prices and significantly faster design cycle times, reconfigurable systems, such as FPGAs, are frequently used in cryptography.

Because FPGAs may be programmed to perform a wide range of algorithms and tasks, they have long been used to develop cryptographic algorithms. They are widely used in the sectors of security and scientific investigation. Field-programmable gate arrays (FPGAs) provide higher speed and versatility when compared to application-specific integrated circuits (ASICs). Previously, ASICs were extensively used to tackle cryptography challenges. Because of recent advancements in reprogrammability, it is now easier to modify algorithms and write new code for FPGAs. On an FPGA, developing and swapping algorithms is faster.

The goal is to improve TRNGs by developing ones that are totally digital and built on FPGAs. It is possible to construct TRNG designs that are both user-friendly and well-suited to the FPGA design flow using the CAD software tools available for FPGA design. Random noise can exist in digital circuitry, albeit it is uncommon. Along with the metastability of circuit components and the frequency of free-running oscillators, oscillations, which are unpredictable phase shifts in clock signals, are another source of random noise.

Because of its scalability and speed to market, the FPGA has become a popular alternative for designing cryptographic systems that heavily rely on TRNGs. While FPGAs can implement most hardware TRNG algorithms, a few are platform-specific and so cannot be used. Existing FPGA TRNGs can have their throughput-per-unit-area improved. Environmental factors such as temperature and voltage changes can potentially influence the unpredictability of TRNG output bitstreams. The FPGA's circuit can undergo "partial reconfiguration" (PR) during operation, allowing for changes (typically the addition of additional functionality). Architects can use DPR

ISSN PRINT 2319 1775 Online 2320 7876
Research Paper , Journal Volume 11, Iss 07, 2022

to develop systems with fewer moving components, smaller hardware footprints, and lower power needs. DPR enables rapid changes to the logic fabric of an FPGA without interrupting normal operation.

Jitter control behavior for Xilinx FPGA-based applications can be adjusted utilizing the TRNG circuit implementation's dynamic partial reconfiguration (DPR) capabilities. With the appropriate design decisions, bad DPR manipulations that could damage the system can be avoided. Setting up a Trojan horse piece of hardware.

2. BACKGROUND OF PROPOSED WORK

In this section, we will take a quick look at the BFDTRNG Model with a single phase.

The BFD-TRNG circuit is a totally digital TRNG that uses the BFD approach to eliminate jitter. The first implementation was a 65-nm CMOS ASIC. 3.

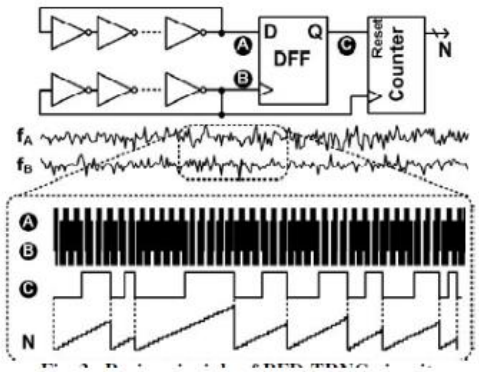

Fig 3. Elements critical to the operation of the BFD-TRNG.

- The circuit contains two ring oscillators, A and B. The building procedures and configuration possibilities are essentially comparable. Because of the physical randomness imposed by process variation effects in deep sub millimeter CMOS production, one of the oscillators (marked A) oscillates somewhat faster than the other (designated B). The authors suggested using pruning capacitors to improve the precision of the oscillator's output frequencies.
- The D flip-flop (DFF) compares the output of one oscillator to the output of another. Assume that the DFF's clock input and D-input are connected to outputs A and B for the purpose of clarity.
- The difference in frequencies determines the timing of the signal from the faster oscillator passing, catching up with, and passing the signal from the slower oscillator. Due to system noise, the DFF generates logic-1 at unpredictable intervals. As a result, capturing events occur at random intervals known as "beat frequency intervals."
- During beat frequency gaps, 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 reaches a distinct high point at the end of each count-up cycle before being reset.
- As the counter's output increases, a sampling clock reads it.
- The serialized sampled answer generates a random bit stream.
- Due to the construction of the ring oscillators, the BFD-TRNG circuit has certain difficulty producing truly random numbers. The randomness of a TRNG's generated bit stream is susceptible to manufacturing defects in the ring oscillators. Despite the fact that both systems had the same number of inverters, the counter maximums were distinct.
- The following are some potential techniques to dealing with the aforementioned FPGA issue.

Random source:

Many FPGAs already have DCM (Digital Clock Manager) modules installed. Clock-deskew, frequency generation, and phase shifting are only a few of the complex clock management tasks handled by these modules. The DCM can generate a wide range of clock frequencies needed for frequency synthesis by multiplying and dividing an input clock in various ways. Because of the use of Delay-Locked Loops (DLL) in frequency

synthesis, jitter in the resulting clock time is unavoidable. According to the Xilinx FPGA data sheet, if the frequency synthesis parameters are selected correctly, we can build an output oscillator with a lot of period jitter that can be used to generate random numbers.

Randomness extraction from the DLL-generated clock jitters:

The main idea behind our technique is to combine a DLL and a DCM to minimize the inherent unpredictability in the period jitter of an FPGA-based oscillator. The jitter in F1 is used to sample a DLL-generated clock F1 from a precise high-frequency reference clock Fh. Because the duty cycle is not guaranteed to be exactly 50%, Fh's likelihood of being 0 or 1 will fluctuate. As a result, the resulting random bit stream will be biased toward either 0 or 1. Furthermore, if the period jitter in F1 isn't great enough (compared to the period of Fh), the output random bit stream can be partly predicted by looking at the bits that came before it. To eliminate output bias and association, several de-skewing techniques are used, including the parity filter, the Von Neumann de-skew filter, and strong mixing.

3. PROPOSED ARCHITECTURE

DCM may generate a wide range of output clock speeds by dividing and multiplying an input clock in a variety of ways. The resulting clock period is intrinsically unpredictable due to the use of Delay-Locked Loops (DLL) in frequency synthesis. In other words, the Xilinx FPGA data sheet indicates that by carefully selecting the frequency synthesis settings, an output clock with a lot of period jitter can be formed, which can then be used to generate random numbers..

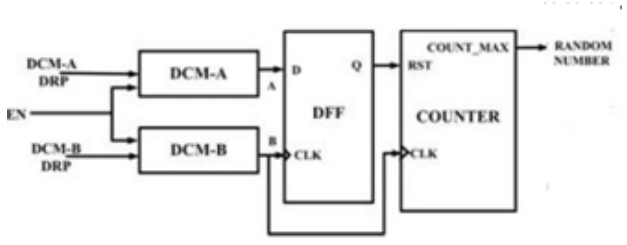

Figure 4: Construction Planning and Analysis

Fig. Figure 4 depicts the general design of the proposed TRNG. Instead of two ring oscillators, two DCM modules are employed to generate the oscillation patterns. Future outcomes are highly unpredictable due to the inherent volatility of the proposed system. DCM modules enable the designer to further customize clock waveforms and do not require initial calibration. Real-time tuning is available by altering DCM settings via DPR ports and DPR capabilities. The DFF identifies the difference between the two clock signals generated, making it more adaptive than the ring-oscillator-based BFD TRNG, and calculates the beat frequency interval at which the faster oscillator completes one cycle more than the slower. The DFF is initialized by utilizing a clock signal to reset the value. The counter significantly reduces the time necessary to produce random numbers.

4. CONCLUSION

The construction of the ring generators influences the unpredictability of the BFD-TRNG. Because ring oscillators run autonomously, planning and building the circuit on an FPGA platform with the same number of inverters spread out in different locations is difficult. We want to create a better, low-overhead TRNG that is easy to configure and deploy on the FPGA platform. Using the DPR capabilities of current FPGAs, the suggested architecture re-models digital clock management (DCM). This enables dynamic modifications to the randomization properties of the TRNG. The dynamic reconfiguration port (DRP) present on Xilinx clock management tiles (CMTs) simplifies 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.