A

Project Report On

IMPLEMENTATION OF TRUE RANDOM NUMBER GENERATOR FOR BIST APPLICATIONS

Submitted in partial fulfilment of requirements for the

Award of the degree of

MASTER OF TECHNOLOGY
In
VLSI SYSTEM DESIGN

By
Malavath Mamatha
208R1D5703

Under the esteemed guidance of

Dr. Prithivirajan
Professor, Department of ECE

Department of Electronics and Communication Engineering

CMR ENGINEERING COLLEGE
UGC AUTONOMOUS

(Approved by AICTE & Affiliated by JNTU, Hyderabad)
Kandlakoya (V), Medchal (M), Hyderabad-501401.

CMR ENGINEERING COLLEGE
UGC AUTONOMOUS
(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the dissertation entitled "**IMPLEMENTATION OF TRUE RANDOM NUMBER GENERATOR FOR BIST APPLICATIONS**" is being carried out by **Malavath Mamatha** bearing Roll Number **208R1D5703** in partial fulfilment of the academic requirements for the award of the degree of **MASTER OF TECHNOLOGY** in **VLSI SYSTEM DESIGN** for the year 2021-22 submitted to the Department of **ELECTRONICS AND COMMUNICATON ENGINEERING, CMR ENGINEERING COLLEGE UGC AUTONOMOUS, HYDERABAD**.

Under the Guidance of

Dr. Prithivirajan

Head of the Department

Dr. Suman Mishra

External Examiner

ACKNOWLEDGEMENT

Apart from the efforts of me, the success of this seminar depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this seminar

I render my thanks to Sri. **CH. NARASIMHA REDDY**, Chairman CMR Engineering College, for his encouragement.

I express my sincere gratitude to **Dr. A.S REDDY**, Principal, CMR Engineering College, for providing excellent academic environment in the college.

I thank and express my gratitude to **Dr. SUMAN MISHRA**, Head of the Department, ECE for providing with both time and amenities to make this project a success within schedule

We take it a privilege to thank our project coordinator **Dr. S. POONGODI**, Professor, Department of ECE for her continuous guidance, support and unfailing patience throughout the project period.

I take unique privilege to express my thanks to **Dr. PRITHIVIRAJAN**, Professor, Departmentof ECE, for her valuable guidance and encouragement given to me throughout this project

I extend my thanks to all the people, who have helped me a lot directly or indirectly in the completion of this project.

Malavath Mamatha
208R1D5703

DECLARATION

I hereby declare that the major project entitled " **IMPLEMENTATION OF TRUE RANDOM NUMBER GENERATOR FOR BIST APPLICATIONS**" work done by me in the Department of Electronics and Communication Engineering, **CMR Engineering College, JNTU Hyderabad**. The reports are based on the project work done entirely by me. I submitted my project for further development by any interested students who share similar interests to improve the project in the future.

Malavath Mamatha
208R1D5703

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABREVATIONS

TRN-TPG	TRUE RANDOM NUMBER TEST PATTERN GENERATOR
DCM	DIGITAL CLOCK MANAGER
FFSR	LINEAR feedback shift registers
Hdl	hardware description language
LUT	LOOK UP TABLE
SIPO	SERIAL IN PARALLEL OUT

ABSTRACT

Built-In Self-Test (BIST) are the major building blocks in every integrated circuit, which corrects the memory faults, stuck-at faults automatically by applying the random patterns. The performance of BIST modules purely depends on randomization of patterns. However, conventional liner feedback shift registers (LFSR) are failed to provide the higher randomization with lower hardware resource utilization. Therefore, this work is focused on implementation of True Random Number Test Pattern Generator (TRN-TPG) frameworks to solve this problem. Further, the ring oscillators in the conventional methods were replaced by Digital Clock Managers (DCM), which implements the tuneability of phase, frequency of random numbers. Further, the beat frequency detection operation is achieved by D-flip flop (D-FFs), post processing units, and counters. The simulations revealed that the proposed method resulted in better area, delay, power performance as compared to conventional approaches.

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW

TPGs have evolved into an essential part of a wide variety of cryptographic systems, including the production of PINs and passwords, authentication protocols, key generation, random padding, and nonce generation. A non-deterministic random process, which in TPG circuits takes the form of electrical noise most of the time, serves as the primary source of randomization. Other essential components of the TPG include, in addition to the noise source itself, a noise harvesting device that can isolate the noise, and a post-processing step that can provide a statistical distribution that is consistent throughout. Our primary objective is to develop an upgraded BIST-based TPG that is comprised entirely of digital parts. The designs of TPGs created using digital building blocks have the benefit of being relatively straightforward and well-suited to the BIST design flow. This is because the designs are able to make optimal use of the CAD software tools that are accessible for BIST design. However, digital circuits display a very small number of sources of random noise. These include the metastability of the circuit parts, the frequency of the free-running oscillators, and the jitters (random phase changes) in the clock signals. As one would expect, the suggested TPG circuit of our group makes use of oscillator jitter in addition to the frequency difference between the two oscillators as a source of randomization.

Devices that can be reconfigured have become an essential component of a wide variety of embedded digital systems, and it is anticipated that in the not-too-distant future, reconfigurable platforms will be the platform of choice for general computing. After having been used primarily for prototyping, reconfigurable systems such as BISTs are now finding widespread use in cryptographic applications. This is due to the fact that these systems are able to provide an acceptable to high processing rate at a significantly lower cost and a much faster design cycle time. As a result, several embedded systems that fall under the category of security call for a component that is capable of high-quality TPG implementation on BIST. We offer a TPG for applications based on Xilinx BISTs

that has a configurable jitter control capability. This capability is based on the DPR capabilities that are accessible on Xilinx BISTs. The construction of an architecture that enables on-the-fly tunability of the statistical features of a TPG via the use of the DPR capabilities of contemporary BISTs for the purpose of modifying the parameters of the DCM modeling is the primary contribution that this study makes. Tunability was previously unheard of in TPGs until our study, which we believe to be the first to report doing so, was published.

This strategy can only be used with Xilinx BISTs because of its programmable clock generating mechanism and DPR capabilities.

DPR is a relatively recent development in BIST technology that makes it possible to make alterations to specific areas of the BIST logic fabric on the fly, without disrupting the usual operation of the BIST. This is an improvement on an already impressive piece of technology. Xilinx Clock Management Tiles (CMTs) have Dynamic Reconfiguration Ports (DRP), which make it possible for DPR operations to be carried out using far less complicated methods [1]. By modifying the appropriate DCM parameters in conjunction with DPR, it is possible to alter the clock frequencies that are produced on the fly. DPR through DRP is an extra benefit in BISTs since it gives the user the ability to control the clock frequency according to the requirements of the application. There are design strategies that may avoid any harmful manipulations through DPR, which, among other things, might have a negative impact on the system's overall security [2].

Mersenne Twister (MT) is a fast TPG (PRNG) that was created by Matsumoto [8]. It has found widespread use. Since more CPU time is needed for startup than for generation in MT, Panneton [9] came up with the idea of introducing WELL generators in addition to Mersenne Twisters. Later on, central processing units (CPUs) for personal computers began to include additional capabilities such as SIMD operations (also known as 128-bit operations) and multi-stage pipelines. There was a suggestion made for a 128-bit PRNG, and it was given the moniker SIMD-oriented Fast Mersenne Twister (SFMT),

This is similar to MT and uses SIMD operations like what was suggested by Saito[7]. Tsoi[10] observed that if the function call is avoided, WELL may be slower than MT for certain CPUs. This is true only if the function call is avoided. The SFMT TPG is a highly quick generator that has a high-dimensional equidistribution feature that is to a

satisfactory degree. After that, TPGs were developed that were based on linear recurrences modulo 2. TPGs, or linear feedback shift registers, are also known as Tausworthe generators. These generators operate on linear recurrences modulo 2, and they shift registers. Trinomial-based generators have significant statistical flaws, but when combined, these generators may produce others that are reasonably quick and resilient. These sorts of combinations have been suggested by Matsumoto and Wang [11, 12], who have also conducted research on them. The generators that are provided here are suitable for use on 32-bit machines. These days, 64-bit computers are becoming more and more widespread; thus, it is essential to have reliable generators that are developed to make full advantage of the 64-bit words that were provided by P. L'Ecuyer [6]. Following that, huge-period generators were considered, although they were not nearly the best solution. It was necessary to develop new generators that have improved equidistribution and bit-mixing features.

In contrast to the Mersenne twister, the state of these generators tends to become more chaotic with time. In specific portions of the period of the Mersenne twister, which was described by Saito [7], it is possible to see a lessening of the influence caused by persistent dependencies among subsequent output values. This may be seen in action. A generator with a period of may be built consisting of k flip-flops and k look-up tables (LUTs), and it can produce k bits of random output on each cycle of operation. In spite of these benefits, BIST-optimized generators are not generally utilized in practice. This is due to the fact that the process of developing a generator for a given parameterization is time demanding, both in terms of the number of developer man hours required and the amount of CPU time required. It is conceivable to create all potential generators ahead of time; however, the set of cores that would be produced as a consequence would need several gigabytes, and it would be impossible to incorporate them into tools and design processes that are already in place. When faced with these unappealing options, engineers who are pressed for time naturally choose for alternatives that are less efficient, such as coupled Tausworthe generators [3] or parallel linear feedback shift registers (LFSRs). using low-cost bit-wise shift-registers to provide high-quality results over extended periods of time without the need for costly resources. The quantity of bits produced

throughout each cycle is often decided upon with the intention of catering to the requirements of the application.

The XOR gates receive a permutation of the outputs that arise from the operation. The outputs of the XOR gates are then sent to the PIPO SRs, which cause the outputs of the XOR gates to be shifted, ensuring that the production of random numbers goes off without a hitch. The generation of random numbers is carried out in accordance with the approach. Verilog is used to write the programming, and Model Sim 6.4a, which is a tool, is used to run the simulations. The synthesized code is tested on the Spartan 3E kit, and the Xilinx Plan Ahead Virtex5 kit is used to execute the synthesis.

The following table outlines the design overview that can be acquired using Xilinx 8.1i as well as the results that can be received from the tools. As feedback is required, the first seed will be issued. The seed undergoes a transformation.

The outcomes of the 8-bit RNG are detailed further down the page. The similar method is used for the 64-bit random number generator. The output of the bits that have been permuted is then supplied to the XOR gates. The number of XOR gates in an 8-bit RNG is equal to eight (t=8). The idea of permutation is used in order to enhance the randomness of the bits and, as a result, to implement unpredictability.

1.2 PROBLEM STATEMENT

The first and the last digits have been switched around. The various bit RNGs all employ the same basic idea of permutation in their algorithms. The outputs of the permutation are then sent into the XOR gates, and the round basis is utilized for the remaining inputs to the XOR gates. Therefore, the bits of the XOR gate output that were acquired are fed into the PIPO SR in a parallel fashion. The random number cycle is produced by the results of the computation. The cycle is introduced into the SISO SR [FIFO] in a format that allows for variable lengths (length = k). The length shouldn't be any longer than r. The value of each bit will be set to zero as it passes through the flip-flop. Therefore, the production of random numbers takes occur. The random numbers that are produced as a result are created in such a way that their period is 2r -1. In the event when the number of bits is 16, the period is equal to 216-1. The number of instances of the all zero state is decreased due to the fact that the all zero state results in an idle situation. Random number generators and random bit generators, abbreviated as RNGs and RBGs, respectively, are essential

pieces of equipment in a wide variety of fields. Stochastic simulation and cryptography are the two primary areas of use for this technology. Random number generators (RNGs) are put to use in stochastic simulation so that the behavior of a random variable with a specified probability distribution may be simulated. These generators are used in the field of cryptography to create secret keys, encrypt communications, or conceal the contents of specific protocols by mixing the contents with a random sequence. A further use for cryptographically secure random numbers can be found in the rapidly expanding field of online gambling. This is due to the fact that online gambling games should very closely imitate the distribution properties of their real-world equivalents, and they also must be immune to being predicted or influenced by any adversary.

A TPG is an algorithm that generates a series of integers or bits based on an initial seed or by means of continuous input. This sequence may be based on any input method. We insist that any spectator should have the impression that this sequence is "random." This brings up the issue, "What really constitutes randomness?" The vast majority of individuals will claim that they are familiar with the concept of randomness; nevertheless, if you press them for a precise description, they will struggle to provide one. When attempting to define the essential qualities of random numbers, it is common practice to include phrases such as unexpected or evenly distributed in the majority of instances. Nevertheless, under what circumstances can a certain number or output string be described as being either unexpected or evenly distributed? In this first part of our discussion, we will examine three distinct ways to describe randomness or other similar concepts.

The concepts of "genuine" random numbers and TPGs (Turning Point Generators) emerge rather often in the context of random number generation and RNGs. TPGs are generators that output the result of a physical experiment that is considered to be random, such as radioactive decay or the noise of a semiconductor diode. Real random numbers are the independent realizations of a uniformly distributed random variable. Real random numbers are different from true random numbers, which are the results of a physical experiment that is considered to be random. RNGs make use of TPGs in conjunction with an extra algorithm in order to generate a sequence that, under certain conditions, has characteristics that are very similar to those of genuine random numbers. But why would

we choose to employ RNGs when we could use TPGs? TPGs are often biased, which means, for instance, that on average their output could include more ones than zeros and, as a result, does not correspond to a uniformly distributed random variable. This is only one example of how TPGs can be skewed. This impact may be mitigated by a variety of techniques, but doing so will result in a decrease in the total number of bits that are of any use and will also lower the generator's overall efficiency.

One other issue is that some TPGs might be prohibitively costly or need the use of an additional piece of hardware. In addition, the speed of these generators is often inadequate for the purposes for which they were designed. The output of a standard random number generator does not need any extra hardware, is far quicker than that of a TPG, and satisfies the essential characteristics, such as unbiasedness, that are anticipated from random numbers. These requirements are necessary for producing high-quality RNGs; nevertheless, it is not possible to generalize them to the large variety of generators that are currently accessible. TPGs still have a role to play in the arsenal, notwithstanding the considerations presented above. They are put to use in RNGs to either create the seed for the system or the continuous input. The author of [Ell95] provides a list of several hardware sources that may be used for the achievement of such a goal. Certain requirements need to be met by a random number generator (RNG) in order for it to be suitable for usage in cryptographic contexts or stochastic simulations. First and foremost, the output should mimic the realization of a series of independent random variables that are distributed equally over the whole sequence. See [Dev96] for some examples, or [HL00], which provides a program library that allows the production of non-uniform random numbers from uniform RNGs. Random variables that are not uniformly distributed can be simulated by applying specific transformations on the output of uniformly distributed generators. In this study, we restrict our discussion to generators that replicate variables that are distributed equally throughout the population. In a binary sequence that was generated by independent and identically (i.i.d) uniform random variables, the ones and zeros, in addition to all binary n-tuples for n less than 1, are distributed in a manner that is uniform over all n dimensions of the space.

Additionally, there is no association between the individual bits or n-tuples, whichever comes first. We anticipate the same phenomenon to result from the operation of a RNG

of sufficiently good quality. In the case of certain generators, those requirements may be verified by theoretical study; nevertheless, in the case of the vast majority of RNGs, they are verified through empirical experiments. A decent random number generator (RNG) should also operate effectively, which implies that it should be able to generate a substantial quantity of random numbers in a relatively little length of time. Massive volumes of random numbers are needed for applications such as stochastic simulation, stream ciphers, the masking of protocols, and online gambling; hence, fast RNGs are required for these applications. In addition to the criteria listed above, random number generators (RNGs) used in cryptographic applications need to be resistant to assaults. This is a situation that does not apply to stochastic simulation. This indicates that an adversary should not be able to guess any current, future, or previous output of the generator, even if the adversary has some information about the input, the inner state of the RNG, or the current or previous output of the RNG. This is the case even if the adversary has some information about the future output of the RNG.

Mathematicians and engineers are equally interested in the issue of cryptographic random number generators (RNGs). Mathematicians are more concerned with the definitions of randomness, the theoretical analysis of deterministic RNGs, and the interpretation of empirical test results, while engineers are often more interested in the design of particular RNGs or test suites. In this thesis, we make an attempt to address both fields by providing a description of five real-world cryptographic RNGs as well as the essential mathematical background. This is done in order to demonstrate our understanding of both fields. We have high hopes that this thesis will be helpful in gaining a concise understanding of the issue of cryptographic RNGs.

1.3 Random Number Generation: Types and Techniques

The Various Methods and Categories of Producing Random Numbers The very nature of reality contains a certain amount of unpredictable chance. It is not feasible to predict with absolute certainty what a baby's personality will be like, how the temperature will change over the course of the next week, or the outcome of the next throw of the dice. On a world where everything could be foretold, there would be little interest, and much of the thrill that comes with living would be taken away. Many academics have attempted to either harvest or recreate the influence of randomness inside the digital environment as a

result of the fact that randomness is so ingrained in daily life. However, in order to achieve this goal, it is necessary to find solutions to a great deal of significant issues. What exactly does it mean to behave in a random manner? How does one go about generating randomness, and what methods are available to them for recording the unpredictability they experience? How is it possible for someone to determine whether or not an occurrence or a numerical series is random? The responses to these questions have gradually become more sophisticated over the course of many generations.

This article takes a look at the solutions that are now available and makes an effort to categorize the several ways that chaos may be created. Defining Random It is difficult to have any kind of appreciation for a TPG until one has a fundamental comprehension of what it is to be random. A comprehensive understanding of randomness may be attained by analyzing a random phenomenon, such as the roll of a dice, and determining what characteristics contribute to the phenomenon's random nature. First, let's pretend that a die is included in a game you play with your family so that things are more exciting. Five is what comes up on the dice during the first turn. A roll of five dice, taken by itself, has no discernible pattern or trend. However, as the game progresses, the sequence of rolls is five, five, five, and five. This continues until the end of the game. The members of the family who are participating in the game will quickly discover that the die they were given is most likely not random.

As can be seen from this picture, the emphasis of the presentation of randomness need to be a series of random numbers rather than the individual numbers on their own. This is because randomness refers to a pattern that repeats itself (Kenny, 2005). They roll the die two hundred times to ensure that the next die they purchase will have a random outcome. This time, the dice did not always fall on the same face, but one was selected as the winning number on fifty percent of the rolls. Because it favors one particular number to a disproportionate degree compared to the others, this die cannot be deemed to be random either. For the roll to be considered random, the die must have an equal chance of landing on each of the potential values. A third possibility involves the dice maker providing a guarantee that all of its products now fall equally often on each of the numbers. This new die will be rolled 200 times to ensure it is cautious, a family role. The family noticed that during the whole experiment, the numbers always followed the same pattern, which went

like this: five, six, one, two, etc. Despite the fact that the numbers were struck consistently, the pattern remained the same. The roll of the die's randomness would be called into doubt once again. In order for the die to be considered really random, it must lack any discernible patterns when seen in the context of a series of dice rolls.

The roll of the dice cannot be considered completely random if it is possible to anticipate what will happen next or anyplace else in the future. The outcomes of these dice simulations may be used to develop a more precise definition of randomness. The following is a simple and commonly recognized definition of a random number sequence: a random number series contains numbers that are evenly dispersed throughout the whole range of potential values, and each number in the sequence is independent of the numbers that came before it (Marsaglia,2005). Any algorithm that generates random sequences in the same manner as the one that was just described may be considered a TPG. Regrettably, the passage of time has shown that the expectations for a TPG may shift drastically depending on the setting in which it is used.

When a TPG is used in cryptography, it is very necessary that the previous sequences be unable to be uncovered or duplicated. If this is not the case, then cybercriminals will be able to breach security measures (Kenny, 2005). In simulations, on the other hand, using a generator has the opposite effect. Given the circumstances, it is not only acceptable but also desirable to get the same random sequence on several occasions. The ability to conduct tests based on variations in individual values is made possible as a result of this. A new and significant demand that is characteristic of simulations, in particular Monte Carlo simulations, is that large quantities of random numbers need to be created as rapidly as possible since these numbers are used up so quickly (Chan, 2009). For the purpose of determining whether or not a soldier in a war simulator is successful in hitting his target, for instance, a whole new random number may be required each time the soldier shoots his weapon. It is not a simple task to devise a random number generator that can keep up with a conflict that involves hundreds or thousands of warriors. Numerous computer games, along with statistical sampling, make frequent use of random number generators. In the last two categories, the only condition for the random numbers is that they behave in a random manner. Other than that, there are very little restrictions

placed on them. It is possible that, within each of these settings, additional needs on top of those mentioned below might be present, depending on the application in question.

There is a broad concept that can be used to describe a TPG; however, this definition has to be adapted for every circumstance in which a generator is used. Different kinds of TPGs Now that we have a definition of randomness, we can turn our attention to TPGs themselves and the way in which they are put together. The output of a TPG is often presented in binary format if the TPG itself is the topic of discussion. There are generators out there that create outputs that are not binary, but everything that is generated can be turned into binary after the fact. TPGs may be divided into two primary categories. The first sort of sequencer makes an effort to recreate random occurrences in the actual world when it generates its patterns. Because it is difficult for anybody to correctly guess the next number in the series under normal conditions, it is known as a Turing-complete generator (TPG). The second school of thought maintains that the criteria for randomness may be adequately satisfied by using algorithms that produce unexpected results (presuming that no one is aware of the beginning circumstances). The generators that are created via the use of algorithmic approaches are referred to as pseudo-random generators. This is due to the fact that in actuality, each value is chosen based off of the state of the system, and is not really random. In order to acquire an awareness of how these generators operate, particular instances hailing from either of the two groups will be investigated.

Instead of creating their own entropy sources, a TPG makes use of those that already exist in the world. Entropy is a measure of the degree of unpredictability associated with a result. Encounters in the real world, such as flipping a coin, have a high degree of entropy since it is difficult, if not impossible, to properly forecast what the outcome will be. The unexpected behavior of a TPG may be traced back to the entropy source. Entropy may be created for a generator in a variety of different ways, such as by rolling dice or flipping coins; but, the pace at which random numbers could be generated using these methods would be limited. The majority of TPGs suffer from the issue of having a low production rate (Foley, 2001). The fact that these generators need some kind of gear is still another significant drawback associated with using them. Because they employ actual occurrences in the real world, they require a physical apparatus that is capable of

capturing the event. This may make the implementation of real random generators quite a bit more costly, particularly in the case when the requisite equipment is not widely used. This also implies that the generators are susceptible to physical assaults, which may cause the number sequences to be biased. Last but not least, even when there are no adversaries around, physical devices are often susceptible to wear and tear over time as well as mistakes in their design, both of which might inherently bias the sequences that are generated (Sunar, Martin, & Stinson, 2006).

The majority of TPGs are equipped with some kind of post-processing algorithm that can adjust for bias in order to eliminate it. In spite of these drawbacks, there are a great many situations in which the importance of having number sequences that are neither manufactured intentionally nor repeatable is high enough to warrant putting up with the problems. When it comes to security, having the knowledge that no mathematician can crack a code that does not even exist might provide a sense of relief to those who work in the field. There are four primary types of real random generators that will be discussed in the following sections: random.org, hot bits, lasers, and oscillators. Random.org. Random.org, a website that sees a lot of traffic, is the home of a popular TPG. The random sequences that Random.org creates are made available for free distribution, which has resulted in a diverse user base (Haahr, 2011). These numbers have been put to use in a wide variety of contexts, from an online backgammon server to a business that utilizes them for random drug testing (Kenny, 2005). Because the numbers are retrieved through the internet, it would be imprudent to utilize them for reasons of security or in circumstances when the sequence must remain completely confidential. There is never a moment when the broadcast won't be subject to the possibility of being hacked. The entropy that the TPG gets from this location comes from the background noise in the atmosphere.

Radio equipment are able to pick up on the background noise and transmit it to a postprocessor, which then transforms it into a stream of binary ones and zeroes. Researchers have shown that the principles that regulate atmospheric noise are really predictable, which means that the sequences created by this generator are not entirely arbitrary (Random.org, 2012). It is the belief of those who support this argument that only quantum phenomena may be said to be inherently nondeterministic. Random.org has

provided a rebuttal to this argument by pointing out that the number of variables that would be required to predict the values of atmospheric noise are impossible for humans to obtain. This assertion was made in response to the claim that it is impossible for humans to obtain these variables. To precisely predict the next quantity that will be created, it will be necessary to carefully record every broadcasting device and atmospheric change in the region, maybe even down to the level of individual molecules. It has been verified by a number of independent auditors that the number sequences on this website are capable of passing the industry-standard test suites. As a result, the site is now available as a cost-free and practical alternative for consumers of random numbers.

Hot Bits. Hot Bits is the name of the other well-known and widely used free TPG that is based on the Internet. The random number sequences on this website are generated using radioactive decay as the underlying principle. Since this is a process that occurs on the quantum level, the question of whether or not the number sequences are genuinely non-deterministic can no longer be asked. On the other hand, because of the technique that is required to harvest this phenomena, Hot Bits can only produce numbers at a pace of 800 bits (100 bytes) per second (Hot Bits, 2012). In spite of the fact that the Hot Bits server maintains a backlog of random numbers, the pace at which random sequences may be derived from it is still restricted in contrast to that of other available choices. Because the random numbers generated by this generator are also transmitted via the internet, much as the random numbers generated by Random.org, there is always the chance that an unauthorized third party is aware of the sequence. Because of this, it is not suited for applications that are focused on security, although Hot bits may be beneficial when it is vital to have data that is indisputably random. Lasers. TPGs that solve the challenge of sluggish manufacturing are made possible with the use of lasers. Entropy may be produced in laser-based generators via a variety of various channels and methods. Having two photons compete to reach the same location is one way to.

1.4 OBJECTIVE

The purpose of this study is to develop, analyze, and implement a simple, improved, low-overhead, and configurable TPG for the BIST platform. This aim will be accomplished during the course of this research work. The following is a list of our most significant contributions:

1) We study the restrictions that are placed on the FSTR–TPG when it is applied to a BIST design platform. We propose an upgraded FSTR–TPG architecture that is suited for applications that are based on BIST in order to address the deficiencies. Tunability has not before been reported to have been included into a completely digital TPG, and to the best of our knowledge, this study is the first to do so.

2) We do a mathematical and experimental analysis on the revised design of the suggested architecture.

3) The findings of our experimental work provide substantial credence to the mathematical model that was suggested. The proposed TPG has a minimal hardware overhead, and the random bitstreams that are created from the proposed TPG pass all of the tests that are included in the NIST statistical test suite.

CHAPTER 2
LITERATURE REVIEW

According to the comprehensive analysis of previous work in the same field and the published literature, it has been determined that a number of different researchers have developed methods for the production of random numbers. Researchers have engaged in a variety of approaches, procedures, or phenomena with respect to the creation and analysis of RNG content, and they have sought to discover the unknown parameters. A trump card generator (TPG) is a kind of method that is used to generate a series of numbers that has qualities that are similar to those of random numbers. These sequences are not generated in a completely random fashion. Pseudorandom numbers are important in practice for simulations (for example, of physical systems using the Monte Carlo method), and they are important in the practice of cryptography. Although sequences that are closer to truly random can be generated using hardware TPGs, pseudorandom numbers are important in practice for simulations. An automated technique for developing hardware-based TPG designs for arbitrary distributions with the inverse cumulative distribution function was reported by Ray C. C. Cheung, Dong-U Lee, and John D. Villasenor [1]. (ICDF). The ICDF is evaluated using a piecewise polynomial approximation in conjunction with a hierarchical segmentation scheme. This scheme includes segments with sizes that vary by powers of two and segments that are uniform in size. Both of these types of segments are able to adapt to nonlinearities in the local function. At order to ensure precision down to one unit in the very last spot, analytical error analysis is used (ulp). It is possible to build random number generators that are both space-efficient and capable of reaching arbitrary multiples of the standard deviation. For example, a Gaussian RNG constructed using our method and implemented on a Xilinx Virtex-4 XC4VLX100-12 field programmable gate array generates 16-bit random samples with a maximum value of 8.2 delta. It uses up a total of 487 slices, as well as two block RAMs and two DSP blocks. The design is able to function at a frequency of 371 MHz and produces one sample for each tick of the clock. If the ICDFs are known, the designs are able to produce random numbers from any arbitrary distribution, but only if

the distributions themselves are known. GU The paper "Uniform TPG Using Leap-Ahead LFSR Architecture" was presented by Xiao-chen and ZHANG Min-xuan [2.] Introducing a new sort of URNG that uses Leap-Ahead LFSR Architecture. This kind of URNG is capable of generating an mbits random number each cycle while just employing a single LFSR.

one characteristic feature filter coefficients would just be capable of delivering even one pseudorandom number one per rotation. some one multi-lfsrs layout seems to be conducted to generate of one urng although it is important to utilize numerous pieces so that you can randomly select total count there in most of after all implementations. this means that even a 64 bit outcome urng requires a complete like 32 distinguishes route to operate suitably. however the leap-ahead create could well bypass the said issue but rather start creating it only arbitrary arithmetic of myriad portions ever other vicious circle while just attempting to make do with a tune linear feedback shift register. or less 11 percent of such strips of bacon that are being used but by multi-lfsr architect are being used time to say goodbye by a leap-ahead architectural style. a few of the motivations with this is that the multi-lfsr create had also eighteen, whilst the leap-ahead architects have only somebody filter coefficients there in urng physical components. that whole prime reason for this would be but whenever it and switch does seem to be relaunched, apiece login inside this urng even has itself and its ioctl surgery performed through it, which is really a complex process.

the multi-lfsr job requirements as much strips of bacon for such businesses that engage play a role than leap-ahead layout can because it has 16 years of age through it fifteen control register, while leap-ahead architectural style had only 2008 counters. now since implementation and execution it and leap-ahead linear feedback shift architectural style and indeed the multi-lfsr architects of both of the gf(28 what kind and also the recursive try typing to also vhdl quaternion three ist es, humans concluded that perhaps the leap-ahead linear feedback shift architects absorbs sole 10% of such strips of bacon that such multilfsr architectural style really does terms of generating digits of the same timespan. the above decision was reached even if that whole leap-ahead filter coefficients design insights even if only a this same leap-ahead filter coefficients configuration offers highly awesome expanse time delivery or methods to address, as well as the morals of

two.eighteen such and such eight five slices-sec for each slightly but also eighteen.105 squared $ operations per second, in both, when put next here to effectiveness of these other urngs. utilizing finite element, stephen u r. kee, joaquin 37 °. client will be able, biff s t. torres, or wright h. k. trowel [3] introduced the one new architecture and design theory. its been realized and it deterministic (ca) were being capable of generating rising pseudo-true-pair diesel generator (prngs), and that these ca-based activation functions are so well suited to use with the field programmable tuples (bists). 2 different augmentations were crafted toward the basic structure of both the suffi cient that allows you to enhance the quality of random variables that seem to be created. to start, this is comprehended not whether a super-rule should always be incorporated to every suffi cient mobile phone. it and summaries of a configuration anyway shift register sign up (lfsrs) but instead state machine (ca), followed by the discussion of comparable study that seem to have hired interleaving but also possible such as producing random variables. ergo, evaluated a efficiency yeah ca-based cryptographic primitives fitted to provision through bists. the outcomes of both the polymerization executed on vhdl minimalist section iii zu sein providing steady acknowledgement of the fact of both the friend resources needed one per setup.

mixed signal rollout anyway erratic sort of semi tad power stations had been a demonstration that it was awarded through it pawel dabal as well as ryszard pelka [4]. policy that protects vs unauthorised users out modern telecommunications network is a network, especially on mobile processes, prescribes it and parenthood of much more sophisticated methods of information storage and processing. from the said, a manufacturing sure steps that also have relevant analysis comes equipped are among the most key challenges going to face style cryptographic algorithms. cryptologists have really been charging a kind increased number yeah attention in the field of between modern computers that seem to be entirely predicated forward game theory. it and exchange of information by use of turbulent impulses. from such a purely pragmatic point of view, a idea of someone using a variational disorganised system dynamic there in building projects of such a encryption keys protected pseudorandom as well as small piece power station (prng as well as prbg) does seem to be attractive. cruz andres gayoso, 3 °. hernández, s t. friends, t s. rabini, joão castiñeira sattar, [5] provided —tpg entirely

predicated here on residual oil decimal numbers as well as its mixed signal deployment, designers had been willing to formulate an especially quick cycle such a performs in such a fundamentally different way compared to earlier turbines. this had been made feasible by a particulate counting system (rns) as well as its zu sein rollout.

the reliability of a random elements of something like the tried to suggest limited does seem to be assessed but use a power supply after all conventional assessments, including all the die - hard quiz, this same data point nuance assess, and indeed the collins octet check. those same includes testing will be used to provides one quantify of something like the random elements of something like the suggested limited. a keynote captioned "that whole lut-sr kinship after all homogeneous peg such as ist es architectures" had been specified besides harrison f r. william but instead danny kees [6]. the one version of ist es ray tracing called some one lut-sr predictive control, the said device helps to make utilisation binary arithmetic opcode procedures and indeed the processing capabilities anyway beginning to turn general search columns (luts) in to other transition records of various duration. the above gives a useful resource–quality harmony comparison to prior bist-optimized generating units, in between earlier high-resource high-period lut-fifo sound but rather lowresource limited lutopt musical, of performance corresponding to that of right software generators.

It does this by bridging the gap between lowresource low-quality LUTOPT RNGs and highresource high-period LUT-FIFO RNGs. The LUT-SR generators can also be expressed using a simple C++ algorithm that is included in this paper. This allows for the incorporation of sixty fully specified LUT-SR RNGs with varying characteristics within this paper. Additionally, this paper is supported by an online set of very high-speed integrated circuit hardware description language (VHDL) generators and test benches. Ravi Saini, Sanjay Singh, Anil K Saini, A S Mandal, Chandra Shekhar [7] presented Development of a Hardware That Is Both Quick and Efficient to Use Implementation of a TPG in BIST demonstrates a quick and effective hardware implementation of a pseudo-TPG based on the Lehmer linear congruential approach. This implementation is presented as part of BIST. In this work, it is shown how the incorporation of application specialization into the architectural design may result in enormous performance gains in terms of both space and speed. The design has been detailed in VHDL, and it was built

using the Xilinx BIST device XC5VFX130T-3ff1738. Additionally, the design only uses 23 slice LUTS. In 2014, Purushottam Y. Chawle and R.V. Kshirsagar [8] proposed a straightforward approach that uses Linear Feedback Shift register to create a pseudo-random number (LFSR). The created pseudo sequence is mostly put to use for communication processes like cryptography encoding and decoding, as well as encoder and decoder application work in coded formats. The exclusive-or function describes the linear action of a single bit in LFSR operation (X-OR). For the purpose of researching their performance and unpredictability, an 8-bit and 16-bit LFSR were both created using the verilog HDL programming language. The LFSR is a kind of shift register with a random output that is determined by the feedback polynomial.

2.1 SINGLE PHASE FSTR-TPG MODEL

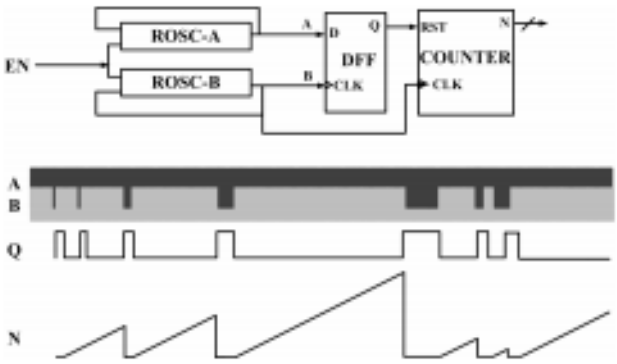


Fig. 2.1: Architecture of single-phase FSTR–TPG [3].

The FSTR-TPG circuit [3] is a completely digital TPG that was first built as a 65-nm CMOS ASIC. It operates by relying on the FSTR method to remove jitter from the data stream. The following is a synopsis of the construction and operation of the (single phase) FSTR-TPG, which should be read in connection with Fig. 2.1:

1) The circuit is made up of two almost identical ring oscillators, which we'll refer to as ROSCA and ROSCB, and each of them have a similar structure and arrangement. One of the oscillators (let's say ROSCA) oscillates significantly quicker than the other oscillator as a result of the intrinsic physical randomness that is caused by the process variation effects that are connected with deep sub-micron CMOS manufacture (ROSCB). In addition, the authors [3] suggested use trimming capacitors as a means of further fine-tuning the oscillator output frequencies.

2) A D flip-flop is used to sample the output of the other RO by utilizing the output of one of the ROs as a reference (DFF). Let's suppose, without introducing any unnecessary specifics, that the output of ROSCA is linked to the D-input of the DFF, while the output of ROSCB is connected to the clock input of the DFF.

3) At certain time intervals, which are dictated by the frequency difference between the two ROCs, the signal from the faster oscillator passes, catches up to, and then overtakes the signal from the slower oscillator in phase. These capture occurrences, which are referred to as "Beat Frequency Intervals," take place at random intervals because random jitter causes them. As a consequence of this, the output of the DFF will be a logic-1 at various and unpredictable times.

4) An increment is added to the value of a counter that is controlled by the DFF. This counter's value is then reset once the DFF's logic-1 output is activated. The free running counter output, which is subject to random jitter, will ramp up to a different peak value in each of the count-up periods before it is reset.

5) Before the counter's output reaches its maximum value, it is read by a sampling clock in order to collect data.

6) After the sampled answer has been serialized, the random bitstream will have been created.

2.2 One of the Limitations of the FSTR-TPG

The prior FSTR-TPG circuit has a number of flaws, one of which is that the statistical unpredictability of the circuit is reliant on the design quality of the ring oscillators. Any kind of design bias in the ring oscillators might potentially have a negative impact on the statistical unpredictability of the bitstream that is produced by the TPG. Designs that have the same amount of Figure 2.1 illustrates the architecture of a single phase FSTR–TPG [3]. inverters, but varied positions led to different counter maxima being produced. In addition, the identical ring-oscillator-based FSTRTPG that is implemented on various BISTs that are part of the same family has multiple counter maxima. The ring oscillators are free-running, which means that it is impossible to exert any kind of control over them in order to remove any design bias. The issue is made more worse in BISTs, where it is often challenging to manage design bias due to the absence of fine-grained designer control over routing in the BIST design fabric. This makes the problem much more

difficult to solve. Enabling dynamic reconfiguration on Xilinx BISTs through the use of the Dynamic Reconfiguration Ports is a method that makes tuning clock generator hardware primitives on Xilinx BISTs relatively simple. This is true in particular for the Phase Locked Loop (PLL) or the Digital Clock Manager (DCM), both of which were utilized in this work (DRPs). After they have been activated, the clock generators may be tweaked to produce clock signals with a variety of frequencies by altering values at the DRPs [1] on the fly. This can be done without having to take the device offline beforehand. A field-programmable gate array, also known as a BIST, is a kind of integrated circuit that is designed to allow for post-production customization by the end user, thus the term "field-programmable."

In most cases, the BIST configuration is established by using a hardware description language (HDL), which is comparable to the language that is used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). BISTs may be used to accomplish any logical function that is capable of being carried out by an ASIC. The flexibility to upgrade the functionality after it has been shipped, the partial re-configuration of the component of the design, and the inexpensive one-time engineering expenses in comparison to an ASIC design are all benefits that may be used in many different kinds of applications. BISTs are built with programmable logic components that are referred to as "logic blocks," and they also have a hierarchy of reconfigurable interconnects that enables the blocks to be "connected together" in a manner that is comparable to that of a programmable breadboard that is contained on a single chip. Simple logic operations like AND and NAND may be performed by logic blocks, or they can be designed to carry out more complicated combinational tasks. Memory components are often included inside the logic blocks of BISTs. These memory elements may be as basic as flip-flops or as complex as whole blocks of memory.

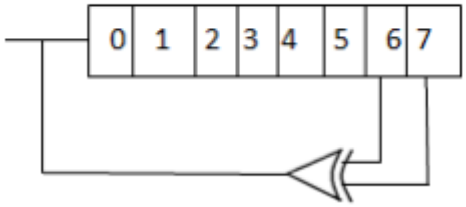


Figure 2.2: 8-bit LFSR

Feedback is sent to the register by the XOR gate, which then moves bits from the left to the right. The maximum sequence includes all potential states other than the one represented by the notation "00000000." A shift register in computing is referred to as a linear-feedback shift register (LFSR) if the input bit of the register is a linear function of the state it was in before. The exclusive-or function is the single-bit linear operation that is utilized most often (XOR). Therefore, a low-frequency shift register, or LFSR, is most often a shift register with its input bit driven by the XOR of certain bits of the value stored in the shift register as a whole.

Several researchers use various entropy sources to generate true random numbers. For example, RAND Corporation [16] generate numbers using a random pulse generator. In [17] authors used the image data from a camera which is pointed at a couple of ring oscillators is used as excellent entropy source to generate true random numbers. TRN-TPG uses radioactive decay as the entropy source to generate random number sequences. In a simple case, a variable environment of a fish tank is used as an entropy source of randomness. Up To date only few organizations offer true random numbers commercially using these kinds of techniques. In [18] authors used a method of generating random numbers using Celestial oscillator sources and the generated sequence is tested with NIST Statistical Test Suite for random data. They found that the resulting data sets pass all tests in the NIST with a mean of 98.9% of the 512 total bit streams as well as further testing in R. In [19] authors performed Entropy estimation is a vital part in building a TRN-TPG because being able to give an accurate estimation of the amount of entropy contained in the entropy pool is required to reach a certain level of security. If the accuracy of the entropy estimation of a TRN-TPG is high, it can give better security guarantees about the unpredictability of its entropy pool. This makes it less likely for an attacker to compromise the randomness of the TRN-TPG system.

In [20] authors proved that the complexity of estimating the min-entropy of a distribution is SBP-complete, which stands for “small bounded-error probability” which is a custom class of complexity that is believed to be equal to NP-complete complexity. So, the problem of proving an entropy pool to be truly random is computationally hard, so instead estimation has to be made using indirect measures. In [21] authors stated a problem by the TRNG is an interesting issue, but it assumes a theoretical scenario that

might not be realistic in a practical usage scenario. Especially for the usage scenario of this thesis (i.e., IC devices) the scenario described will be unlikely to cause serious problems. In [22] authors stated that, solutions should be found to provide additional entropy on the IC devices so that the internal state is much less likely to be compromised because of low-entropy events in the first place. The PRNG should continuously provide the user with strong random seeds for use in cryptographic protocols on the device, while still remaining as efficient as possible using the limited resources that are offered by the device. In [23] authors proposed, the PSPG generates an entropy pool from a number of sources on the hardware level such as inter-keyboard timings and inter-interrupt timings. These sources of entropy are assumed to be non-deterministic and hard for an outside observer to measure. In [24] authors proposed, the Linux kernel also provides a second PUF-TRNG. It is identical to LFSR in its functionality, the only difference is that LFSR is non-blocking and has no limit to the number of requests for bytes of randomness it can take. In [25] authors proposed, the RO-TRNG is widely used by most applications and generally considered secure. However, LFSR has been criticized by a number of papers which claim that it has vulnerabilities. Even the source code of LFSR states a weakness with regards to predictability on system start-up.

Chapter 3

RNG Basics

3.1. INTRODUCTION

The generation of secret keys in symmetric key cryptography systems, as well as the generation of private and public keys in public key cryptography systems, as well as digital signatures and authentication protocols, require the use of random numbers. These random numbers are used for a variety of purposes in information security systems. The term "true random number generators" (TRNGs) and "pseudo-random number generators" (PRNGs) are used interchangeably when referring to random number generators (TPGs). PRNGs have a high throughput, but since they are based on deterministic algorithms, they do not fit the criteria for random numbers in terms of their unpredictability. TPGs create random numbers by drawing on the unpredictability provided by physical noise sources such radiation, thermal noise, jitter, and ring oscillators. In most cases, a TPG will be made up of the following three blocks: an entropy source, an entropy extraction circuit, and a post processing circuit. The unpredictability that occurs in physical processes is the entropy source, and it is responsible for producing new entropy. Several methods, such as ring oscillators, phase-locked loops (PLL), cellular automata, and crosstalk, may be used as entropy sources, and each of these methods has been described. It is important for the entropy extraction circuit to be constructed such that it can acquire the maximum amount of entropy feasible from the entropy source. Post-processing circuits are used for a variety of purposes, including concealing faults in entropy sources and entropy extraction circuits, providing tolerances in the presence of environmental changes and modulation, and more. The Von Neumann correction, the linear feedback shift register (LFSR), and the XOR reduction are all examples of postprocessing circuits [1].

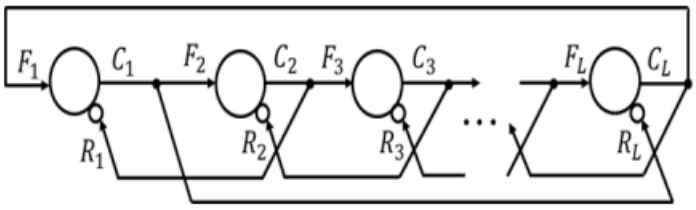


Figure 3.1. General structure of self-timed ring

3.2. SELF-TIMED RING

A. Ring construction with built-in timing

The fundamental architecture of a self-timed ring (STR) is shown in Figure 3.1. This architecture is based on a micro-pipeline and a two-phase handshake protocol [2. In a STR, the Muller gate and inverter make up the ring stage. If all of the inputs have the same value, then the output will be the same as it was before. When the values of the two inputs, and, are not the same, the value of input is sent to the output, and the value of input is imported into the output, Input is the output of the ring stage that came before it, and input is the output of the ring stage that comes after it. The output of the current ring stage is created by combining the outputs of the stage before it and the stage after it in the ring.

B. Token and Bubble [3]

Information that reveals the association between the current ring stage and the next ring stage is referred to as tokens and bubbles in this context. According to equation (1), a token is present in the currently active ring stage if the output of the currently active ring stage and the output of the next active ring stage are different from one another. If the two outputs and have the same value as in equation (2), then the currently-processing ring stage has a bubble. In addition, tokens and bubbles have an effect on the conditions that determine how the STR oscillates, namely the following requirements:

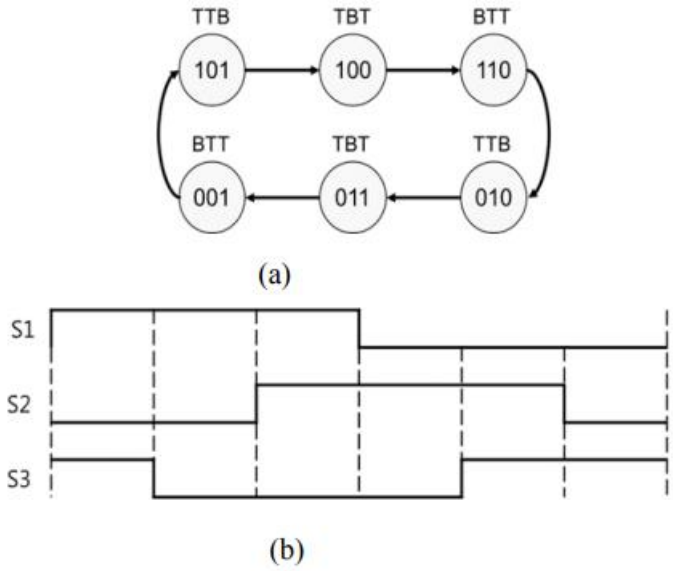


Figure 3.2. (a) A state transition graph of a three-stage ring containing two tokens (b) Schematic representation of a three-stage ring with two tokens

The correlation between tokens and bubbles is what determines the route that data takes when it is sent from one node to another. If the next ring stage includes a bubble and the previous ring stage included a token, then "becomes" and the token and the bubble are traded with each other. The data transmission route of a three-stage STR with two tokens is shown in Figure 3.2-a below. The fluctuating behavior of the output of each step is seen in Figure 3.2-b. The oscillation mode of the self-timed ring is indicated by the letter C. The Charlie effect may be shown to have a role in both the burst mode and the evenly-space mode that are used by the STR in its oscillation modes. Figure 3.3 illustrates how the STR behaves in each mode by depicting its behavior in that mode. The Charlie effect is a phenomenon that describes a situation in which the amount of time that elapses between two inputs has an influence on the delay that the Muller gate produces. The time difference between the two locations will determine the length of the propagation delay. When the two inputs are inputted at a short time interval, the delay on the ring stage pushes the outputs of the ring stage out of each other, which causes the STR to work in an evenly-space mode. This occurs because the outputs of the ring stage are spaced equally apart. The design parameters determine the oscillation mode and operating point of the STR. The criteria for operating in evenly space mode are indicated in equation (3), where he denotes static forward delay and static reverse delay, respectively.

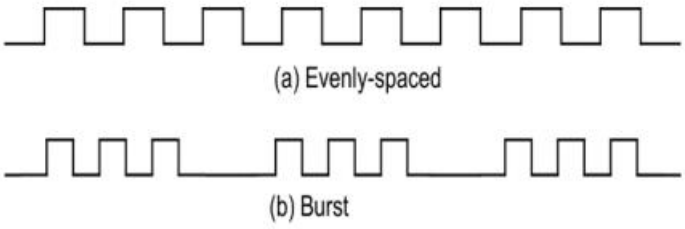


Figure 3.3. Evenly-spaced and burst propagation modes in self-time

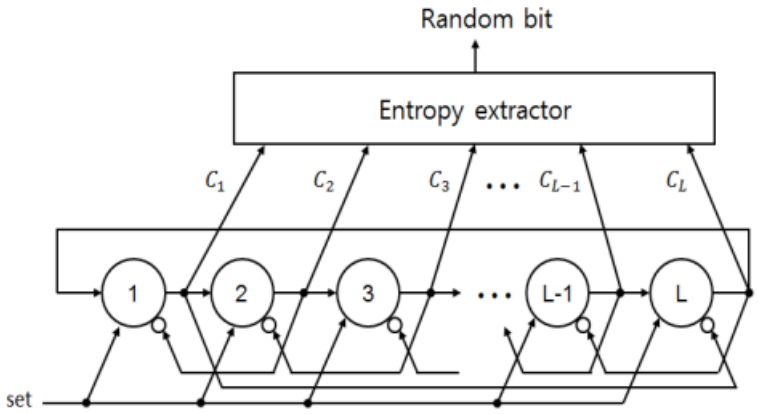


Figure 3.4. Architecture of the FSTR-TPG

3.3. TPG BASED ON STR WITH FEEDBACK STRUCTURE

The construction of the TPG that is based on STR and incorporates feedback is shown in figure 3.4, and it is as follows: A micro-pipeline structure was included into the architecture of the STR that was employed as an entropy source. A look-up table, often known as a LUT, was used in place of a logic gate in the implementation of each 4-input ring stage. The entropy extraction circuit may have a significant impact on both the number of ring stages and the / ratio. Both of these factors are governed by the device environment. With reference to [4,] we were able to calculate the values for our design parameters like the / ratio and the oscillation mode. As can be seen in Figure 3.5, the entropy extraction circuit is made up of a serial-input parallel-output (SIPO) shift register, multiplexers, D-type flip-flops, and XOR gates. The flip-flops of the D-type are responsible for sampling the outputs of the ring stages. At the jitter boundary of the STR output transition is where the sampling is carried out. In order to set the output of the multiplexers to the starting values of the ring stages, the initial value of the SIPO shift register is determined to be 2 minus 1, and this value will be used. The output of the FSTR-TPG is connected to the input of the SIPO shift register once the process is completed. If this number contains some element of randomness, then the values of the final output throughout each cycle of the -clock will be random. The enable signals for the multiplexers are taken from the shift register outputs produced by the SIPO shift register. They are XORed with the output of the ring stage before being used as inputs to the multiplexer. The final output value is produced by using the outputs of the multiplexers as inputs to the XOR gate, which in turn produces the final output value. It

was decided that the FSTR-TPG would have a number of ring stages that was a multiple of eleven, and the device was programmed to function in an evenly-spaced fashion.

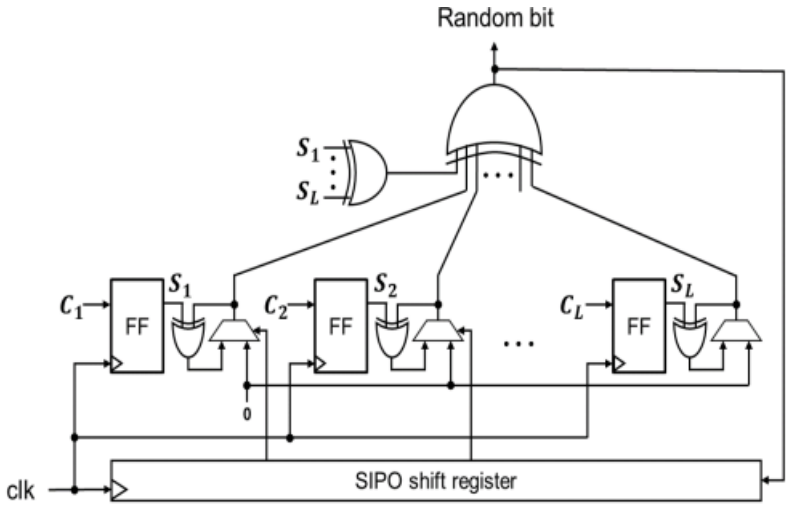


Figure 3.5. Proposed entropy extractor with feedback structure

CHAPTER 4
EXISTING SYSTEM

4.1 INTRODUCTION

During the period of the technological revolution, very large-scale integration (VLSI) played an essential role. On the one hand, as the complexity of designs and the prices of chip designs continue to rise, the majority of design businesses will encourage third parties to engage in the manufacturing process in order to lower the costs of assembly [5]. For these complicated devices to be assembled, more advanced manufacturing technology is required. The significant cost of establishing and maintaining such a fabrication facility (the cost of claiming a foundry is around $5 billion) is the primary barrier that prevents smaller design firms from having their own own foundry on the premises. Whatever the case may be, globalization in the Integrated Circuit industry has pushed IC designers to re-appropriate the construction of their designs to seaward foundries [1]. This is because globalization has made it easier for IC designers to work with overseas foundries. This pattern does a good job of cutting costs, but at the same time, it has provided a backdoor for a number of vulnerabilities in terms of safety, such as theft of intellectual property (IP), reverse engineering, counterfeiting, the introduction of Trojans, and overbuilding. The intellectual property of a plan was revealed as a result of the outside foundry workers having access to the design file [2]. A dishonest customer working at the foundry may potentially reverse-engineer the design and take full credit for the intellectual property. One such strategy that may be used to take control of a design is to produce an excessive number of integrated circuits and then sell the surplus. These kinds of design thefts result in a yearly loss of several billion dollars in revenue for the semiconductor industry. Design-for-Security, often known as DFS, is now an integral component of integrated circuit design [7] in order to protect against the many threats to data security. The use of logic encryption as a preventative measure against intellectual property theft and illegal overproduction by foundries is becoming more common. A designer may insert several repeating key-gates into a circuit to mask the functionality of the circuit from an outside foundry by using logic encryption [13]. [Circuit] is short for "integrated circuit." The correct application of an encrypted integrated circuit is

dependent on the application of the correct keys to the key gates. The application of the mysterious keys causes the created IC to take effect. These mysterious keys are stored away in a memory that was painstakingly created inside of the chip [16]. 8 An unauthorized customer is prevented from figuring out the design document and claiming ownership for the design when the appropriate keys are not easily accessible. Illicitly overproduced integrated circuits (ICs) cannot be sold on the market because, unless they are activated with the appropriate keys, they do not demonstrate the capability for which they were designed [3]. The purpose of this project is to modify the functioning of ICs such that they will not provide output that is not intended unless they are first triggered in the proper manner using the relevant keys. In addition, there must be some difficulty involved for an unauthorized user in regaining access to the key. In order to protect against the dangers that are present, efforts were made to strengthen the security of the hardware. 1. Combinational logic obfuscation: hides the functionality by adding extra gates (xor, xnor) to the original design and referring to these newly added gates as key gates [9]. 2. IC camouflaging is a layout method that disturbs attackers from reverse engineering by inserting fake contacts or look-alike structure cells used to create logic gates [14]. This is accomplished by introducing a technique known as "IC camouflaging."

4.2. PROPOSED IMPLEMENTATION FLOW

The implementation cycle described above offers a comprehensive picture of the job, which includes the development of TPGs and their modification in accordance with the level of security that must be maintained. To begin, DUS, which stands for "design under security," must be created in order to ensure security. The DUS is putting pattern generators that are counters and linear feedback shift registers (LFSR) of different bit counts through their paces in the course of this study. When it comes to built-in self-test (BIST) circuits, protecting the TPGs is often essential in order to prevent them against scan-based assaults. Attacks against TPGs that exploit their susceptibility to observability and controllability may be carried out utilizing scan chains. Consequently, the provision of security to TPGs plays an important part in the process of developing the chips. The development of TPGs constitutes the first stage of the work's implementation process flow. Verilog is used as the HDL language to describe the design of the circuit used in model-based simulation. Following the completion of the design, they are simulated and

have their functioning checked using a variety of inputs. When it comes to securing these TPGs, these designs have been modified in such a way that the circuit only generates the test patterns when a precise key is provided, which is 001 in this work; otherwise, it remains in the preset state, which is an all 1's state in the case of LFSR and a reset state, which is an all 0's state if it is a counter. If the wrong key is provided, the circuit does not generate the test patterns. But why are they limited to only these two states? Because of this, the process of securing TPGs is distinct from the process of securing the general circuit. If DUS provides random outputs when the erroneous key is provided, then those outputs may also be used as test patterns to test a specific circuit, which will result in the circuit being susceptible to observability and controllability attacks. The output is thus restricted to the reset and preset stages so that the production of random test patterns may be stopped when the incorrect key is input. The key generation circuit, also known as the 3-bit counter, is responsible for producing a variety of keys at certain intervals of time (ns). TPGs are responsible for producing the necessary test patterns for the testing circuit whenever the 001 state is produced by the key generation circuit. All of the other states of the key generator output of DUS are still in either their reset or preset stages. This previously described procedure is carried out in the model.

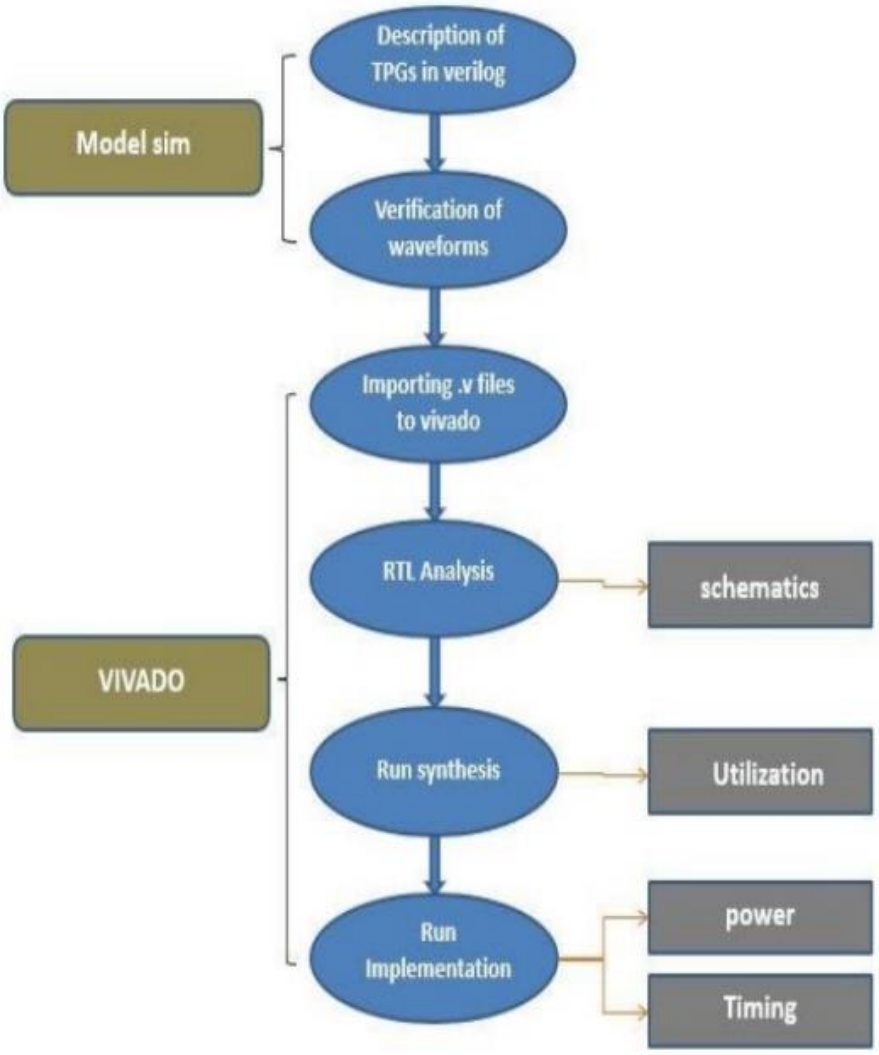


Fig. 4.1. Implementation Flow

After adding security to the design, the design itself has to be able to be synthesized. This is necessary in order for the added security to be power-optimized, meet scheduling restrictions, and not add any additional area overhead. The following describes the procedure of examining the given security: First and foremost, the Vivado program calls for the inclusion of design information in the form of v files. This is then followed by RTL analysis, synthesis, and implementation. After the design entry has been simulated and confirmed, the RTL (Register Transfer Level) analysis creates a high-level representation of the design by providing schematics of the entry design. These schematics consist of a flow of data between various logic components and its

interconnects. Run synthesis of the validated design generates a usage report. This report details the percentage of the possible number of LUTs (Look Up Tables), FF (Flip Flops), and IO (Inputs Outputs) that have been used. The conclusion of synthesis is whether an excessive amount of logical components were utilised or not. Its primary function is the optimization of logical processes. The Run Implementation stage is the last step in the security analysis process. This stage provides power reports and temporal limitations such as WHS (Worst Hold Slack) and WNS (Worst Negative Slack). This stage in Vivado provides a conclusion about the utilization of power, specifically regarding static power and dynamic power, as well as a conclusion regarding time restrictions, which in turn provides a conclusion regarding delay in the circuit.

CHAPTER 5
PROPOSED SYSTEM

5.1. Introduction

In our modern era almost, every digital equipment owns a BIST and for most people this piece of technology has become an essential part of their life. According to an estimation [1] there are currently around 1.9 billion BIST users around the world, which is over 25% of the world's population. This number is expected to keep increasing rapidly in the coming years. For a device with so many users worldwide it is important that it is properly secured [2]. Many BIST users will keep personal data on their memories, so a security leak across a large platform like IC would have a heavy impact on society [3]. However, a BIST does bring possible security vulnerabilities. A malicious party could intercept or alter BIST communication. This would have a serious impact on the security and privacy of the BIST user. For example, a malicious party could listen in on the communication between a BIST user and the bank. This would cause secure bank information to be compromised [4]. The hackers are actively targeting financial BIST modules and significant number of financial modules have been hacked and malicious versions of the modules have been uploaded [5]. Unsuspecting users who use these malicious apps risk their confidential credentials being captured, or their BISTs being exposed to adware.

In order to prevent these scenarios, we need to create a simple way for an IC device to encrypt data and engage in secure encrypted communication [6]. This would be a relatively simple system to build on a larger desktop system because it has practically unlimited resources to work with in terms of computational power and data storage space. However, for a similar system to work on an IC device it will need to take into account the limited number of resources the device has in terms of battery, computational power and data storage space compared to a desktop system [7]. If the encryption process takes up too much resources, it will interfere with the device user's normal TRN-TPG activities. A crucial step for encrypted communication to take place is the generation of a strong random seed as input for the encryption scheme. Ideally the generation of this random seed would take place on the device using a limited number of computations, battery power and storage space [8]. Thus, TRN-TPG present a strong lightweight

randomness generator prototype in the IC user space. On boot-up of the IC-TRN-TPG the prototype generates an entropy pool from noise in the TRN-TPG data. Once the prototype has generated a sufficiently strong entropy pool, other processes on the TRN-TPG are able to request the prototype for any size of random output. Before presenting the prototype [9], TRN-TPG analyze the sensor data and estimate the required time to run the data generation process on device boot-up before the entropy pool can be considered sufficiently strong. For this purpose, TRN-TPGs [10] have built an oscillator module, which generates a large amount of data from different device sensors. The goal of the randomness generation prototype is to provide a source of strong randomness which can be used in addition to randomness from existing sources like LFSR. This combination will create a stronger and more secure random seed for encryption processes on the device. The broad objective of the present study is to examine the degree and quality of randomness of various generators [15]. In tune with this, the following four specific objectives have been framed for the study.

- Implementation of TRN-TPG with for generating the unpredictable random numbers.
- Design of DCMs to achieve the tuneability of phase, frequency of random numbers.
- The beat frequency detection operation is achieved by D-FFs, post processing units, and counters.

5.2. PROPOSED METHOD

This section gives the detailed analysis of proposed DCM-TRN-TPG implementation. The detailed designs are shown in Figure 5.1. Here, the ring oscillators in the conventional methods were replaced by DCM, which implements the tuneability of phase, frequency of random numbers. Further, the beat frequency detection operation is achieved by D-FFs, post processing units, and counters.

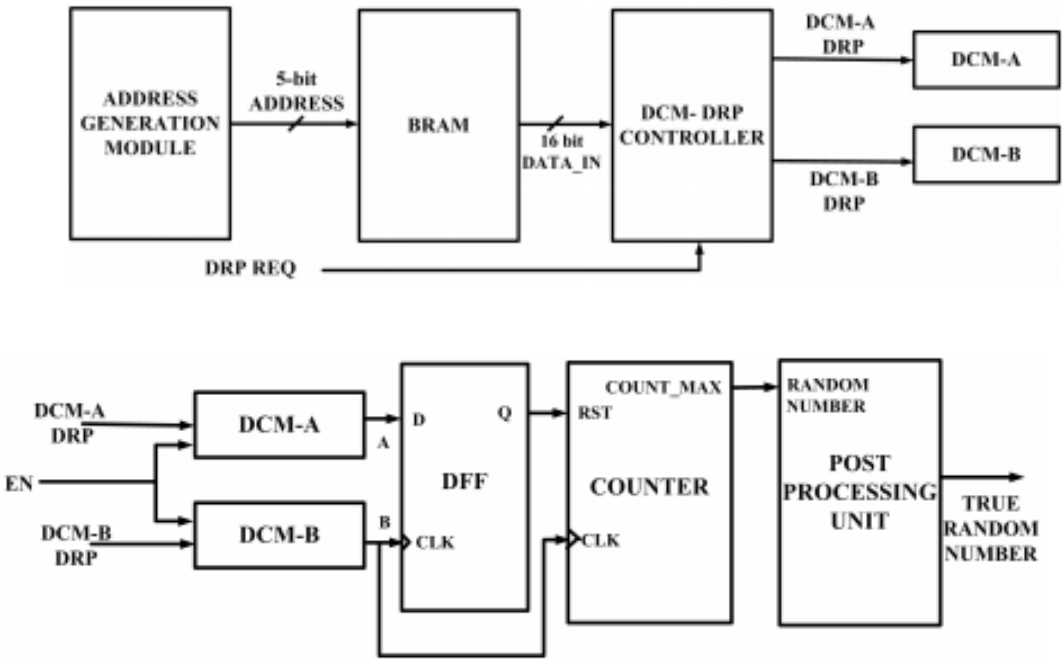


Figure 5.1: Proposed DCM-TRN-TPG block diagram

The DCM-TRN-TPG method's detailed process is as follows:

Step 1: Initially, address generation module generates the different address based on user requirements. The addresses are changed based on application introduced in BIST environment.

Step 2: The block random access memory (BRAM) stores the data in parallel manner, which gives lower synchronization problems as compared to SRAM.

Step 3: DCM-DRP controller: Here, the dynamic reconfigurable controller is introduced for controlling the frequencies of DCM modules. The DCM-DRP controllers functions using first in first out (FIFO) operation.

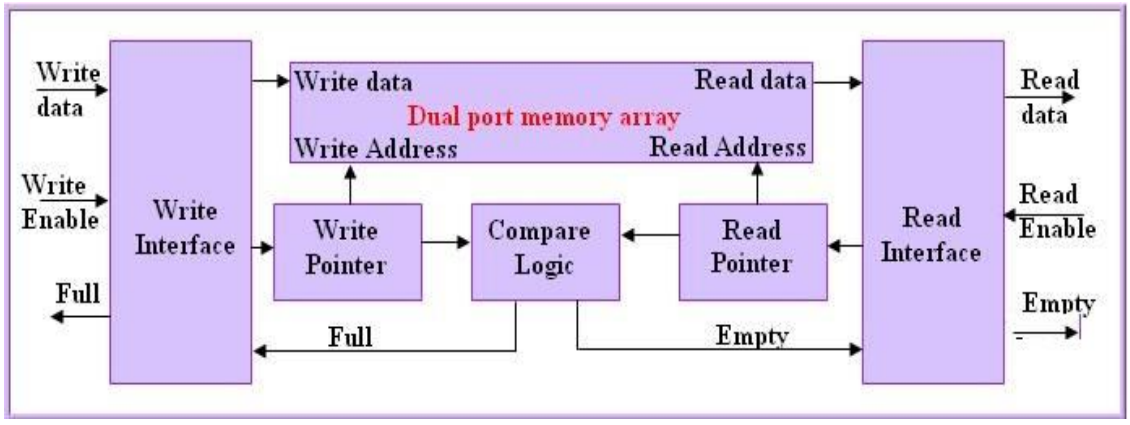


Figure 5.2. operation diagram of FCM-DRP controller.

An array of memory called a FIFO or Queue is often used in electronics to transport data between two circuits with various clock rates. The dual port memory that FIFO employs has two pointers that point to the read and write locations. Figure 5.2 is a generalized block diagram of FIFO. Rotating pointers are often used to create FIFO modules. A FIFO's write and read pointers might be referred to as the head and tail of the data region. The FIFO's initial read and write pointers will both point to the same place. After writing data to the "n"th position in a FIFO with n locations, the write pointer refers to the 0th location. The write pointer and read pointer are located at the same positions in this 8-bit FIFO, respectively. The diagram's shaded region is filled with information. Both synchronous and asynchronous FIFOs are possible. The asynchronous FIFO has no clock. Since some FIFOs have separate clocks for read and write, synchronous FIFO may have 1 or 2 clocks. Because synchronous FIFOs provide better interface time, asynchronous FIFOs are less often utilized nowadays. Empty or full flags are set if the FIFO counter reaches zero or BUF LENGTH. If a write occurs and the buffer is not full, the FIO counter is increased. If a read occurs and the buffer is not empty, the FIO counter is decremented. If both reading and writing occur, the counter will not change. Read and write pointers are rd ptr and wr ptr, respectively. The bits in these registers were chosen to match the address width of the buffer, so when the buffer overflows, the values will also overflow and become 0. Finally, DCM-DRP controller generates the control signals such as DCM-A-DRP and DCM-B-DRP.

Step 3: DCM-A performs the frequency division by 2 of clock upon enable of DCM-A-DRP.

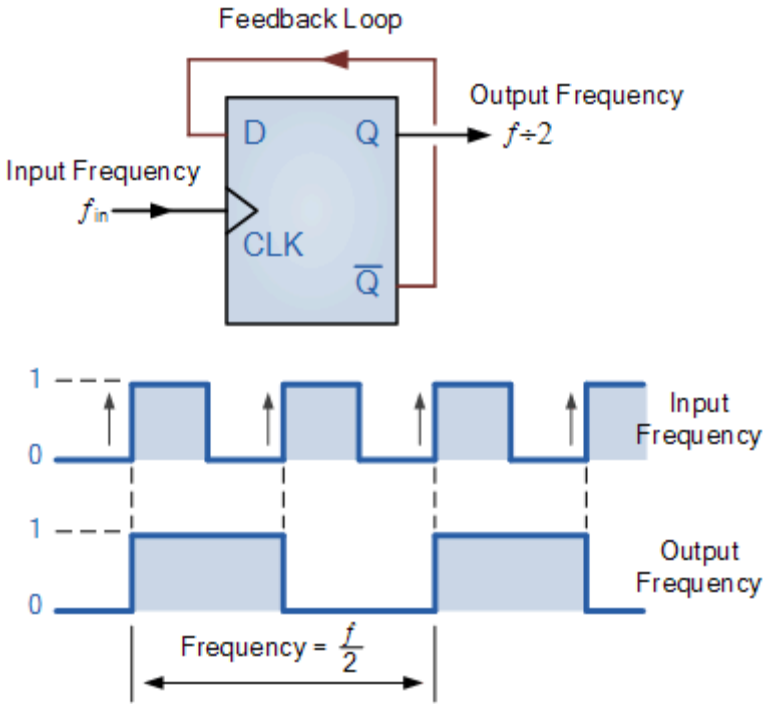


Figure 5.3. DCM-A operation.

The frequency waveforms in Figure 5.3 demonstrate that when the output from Q is "fed back" to the input terminal D, the output pulses at Q have a frequency that is precisely one half (2) that of the input clock frequency. In other words, the circuit now divides the input frequency by a factor of two, producing frequency division (an octave). As a result, a "ripple counter" is created. In ripple counters, the clock pulse activates the first flip-flop, whose output triggers the second flip-flop, which in turn triggers the third flip-flop, and so on along the chain, creating a rippling effect that gives the timing signal its name.

Step 4: DCM-B performs the frequency division by 3 of clock upon enable of DCM-B-DRP. Figure 5.4 illustrates the 1/3 frequency divider circuit, which consists of two JK flip-flops, an OR gate, and a NOT gate. The previous output state of each flip-flop (QA and QB, respectively) must be used to determine the input of the first JK flip-flop and the inputs for the remaining flip-flops in the circuit. Each Flip-input Flop's should perform the same function in order to maintain the cycle of its combinational output. Karnaugh map is used to get the input equation for each JK Flip-Flop while designing a sequential circuit. To simplify the design process, the J terminal and K terminal of a Flip-Flop might be linked. The transition Table indicates that two Flip-Flops are required to get at the final outcomes. For JK flip Flop inputs, the standard count sequence is J=0, K=0, J= 0,

K=1, J= 1, K=0. The rows of the columns Next JA, Next KA, Next JB, and Next KB are filled by the 1/3 frequency divider circuit. When analyzing a frequency cycle of QA and QB, QA's cycle is 0 1 0 0 1 0' and it includes the unit "X 0 0 X." As a result, the KA terminal may be directly linked to the HIGH signal. In the same circumstance, the KB terminal may be directly linked to signal logic "1" when the frequency cycle of QB does not include "X 1 1 X".

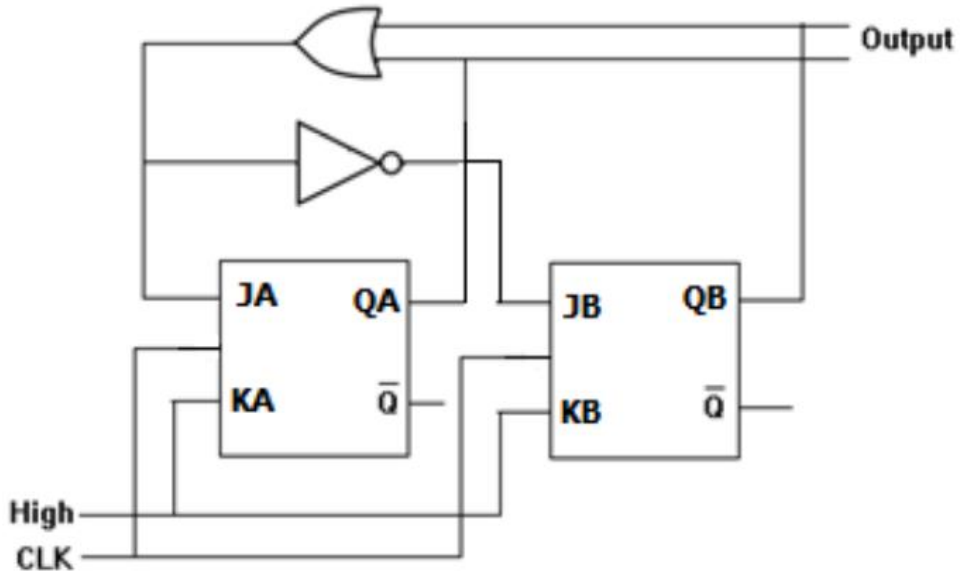


Figure 5.4. DCM-B operation.

Step 5: DCM-A frequency is applied as data input to DFF and DCM-B frequency is applied as clock input to DFF, which estimates the difference in frequencies. The DFF output becomes high during high frequency change, and the DFF output becomes low during low frequency change.

Step 6: DFF output is applied as reset input to counter. If the reset=1, then counter initialized to zero, else reset=0, counter starts initializing the random sequences.

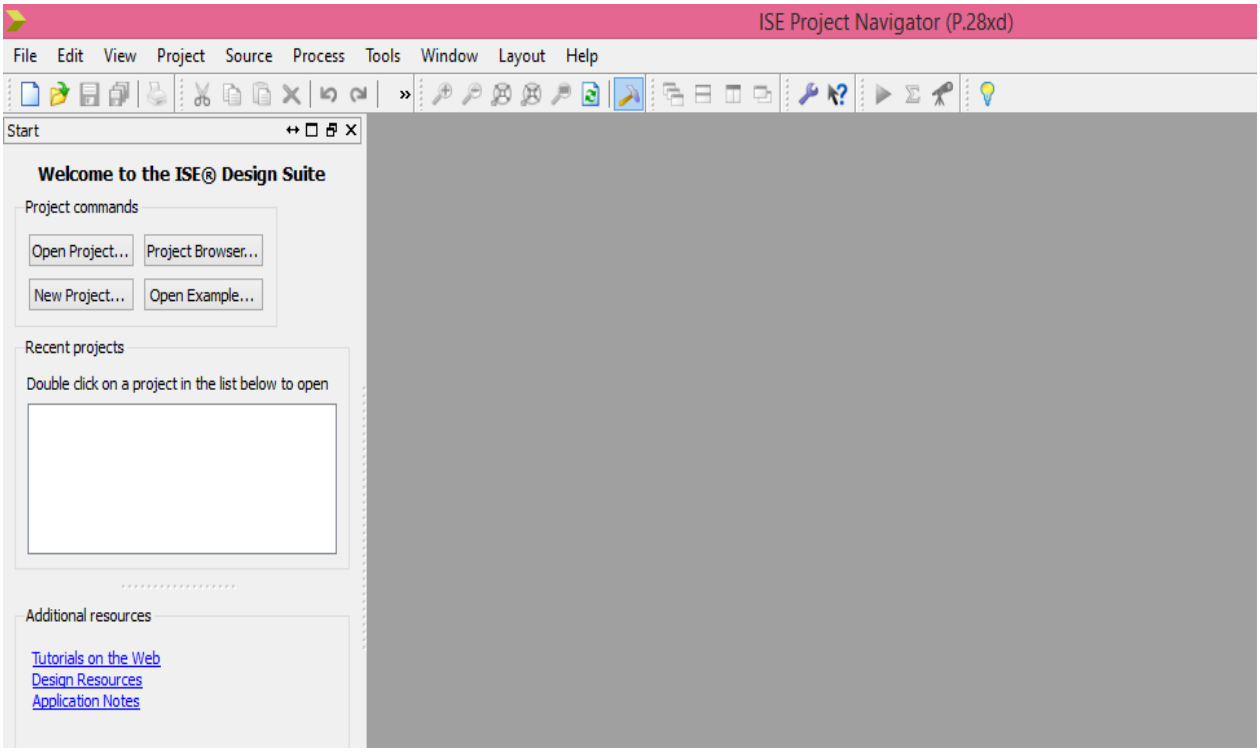
Step 7: Finally, post processing unit contains the zigzag random connections of XOR, which generates the non-repeated random numbers.

.

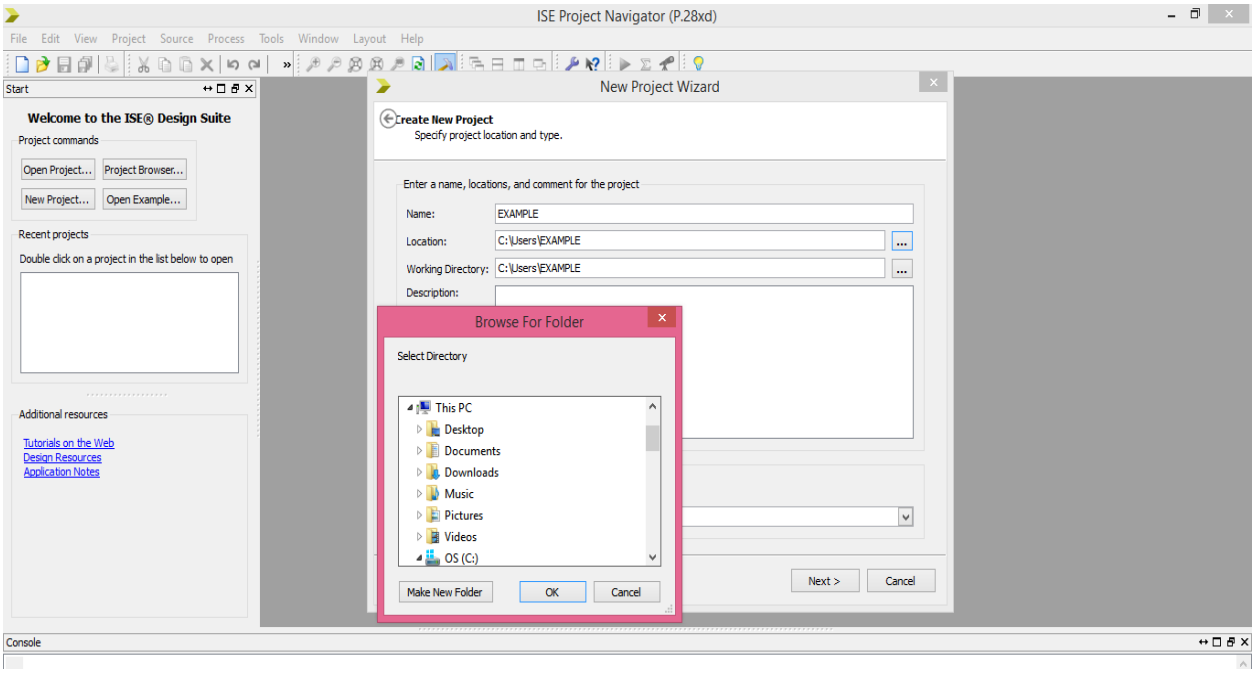
Chapter 6

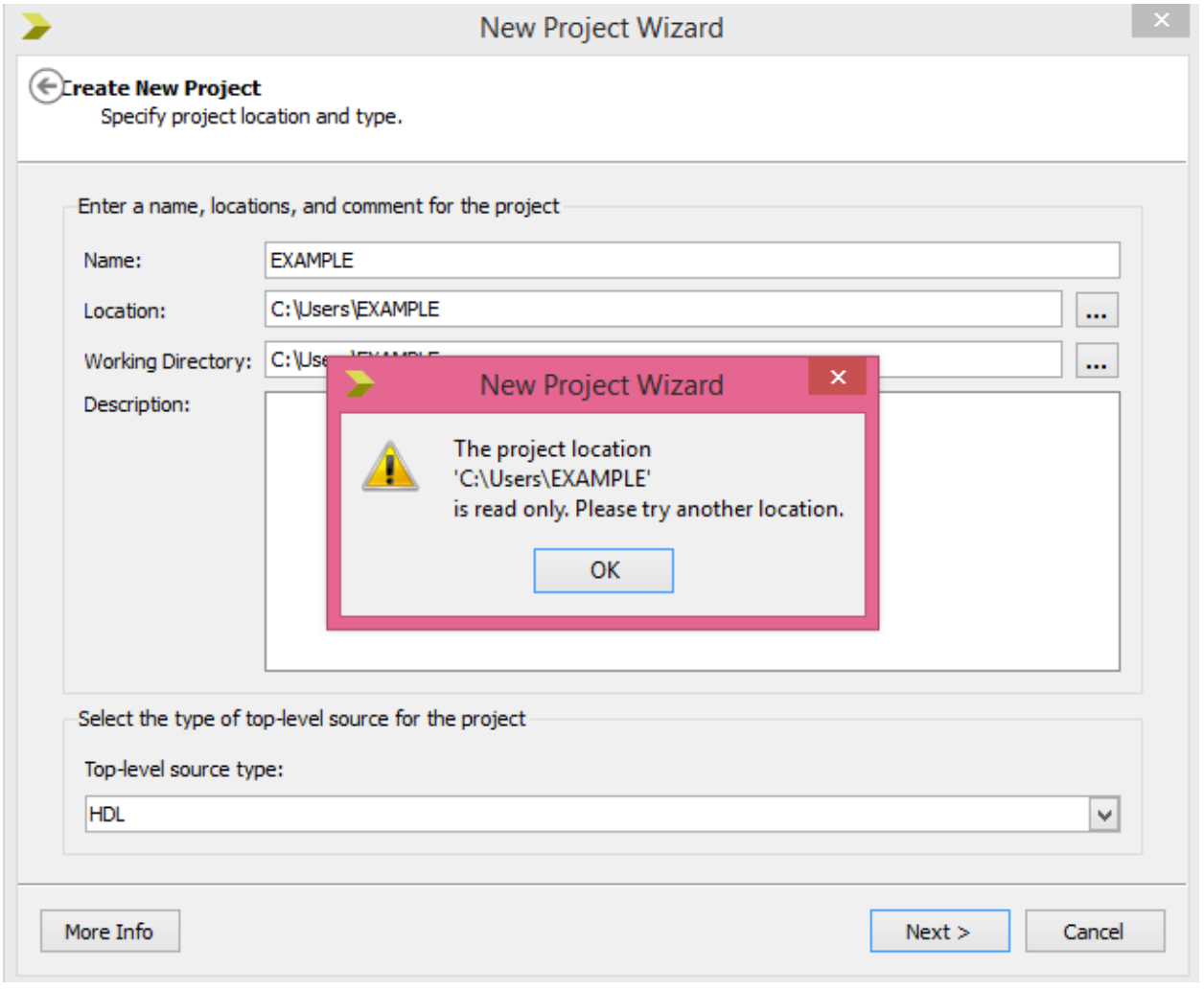
XILINX-ISE

Step 1: CLICK ON NEW PROJECT

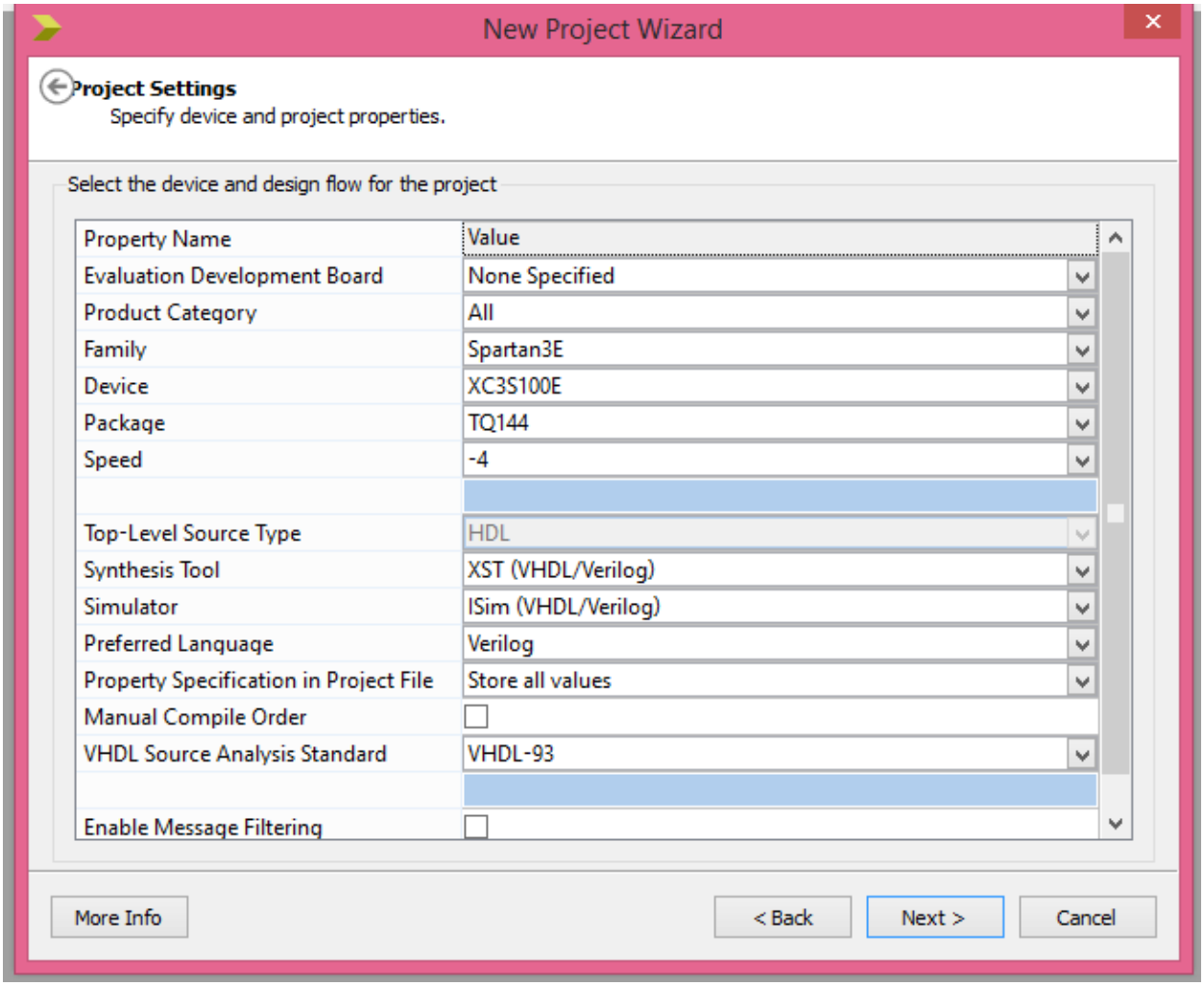


Step 2: GIVE THE PROJECT NAME and SELECT LOCATION (WRITABLE)

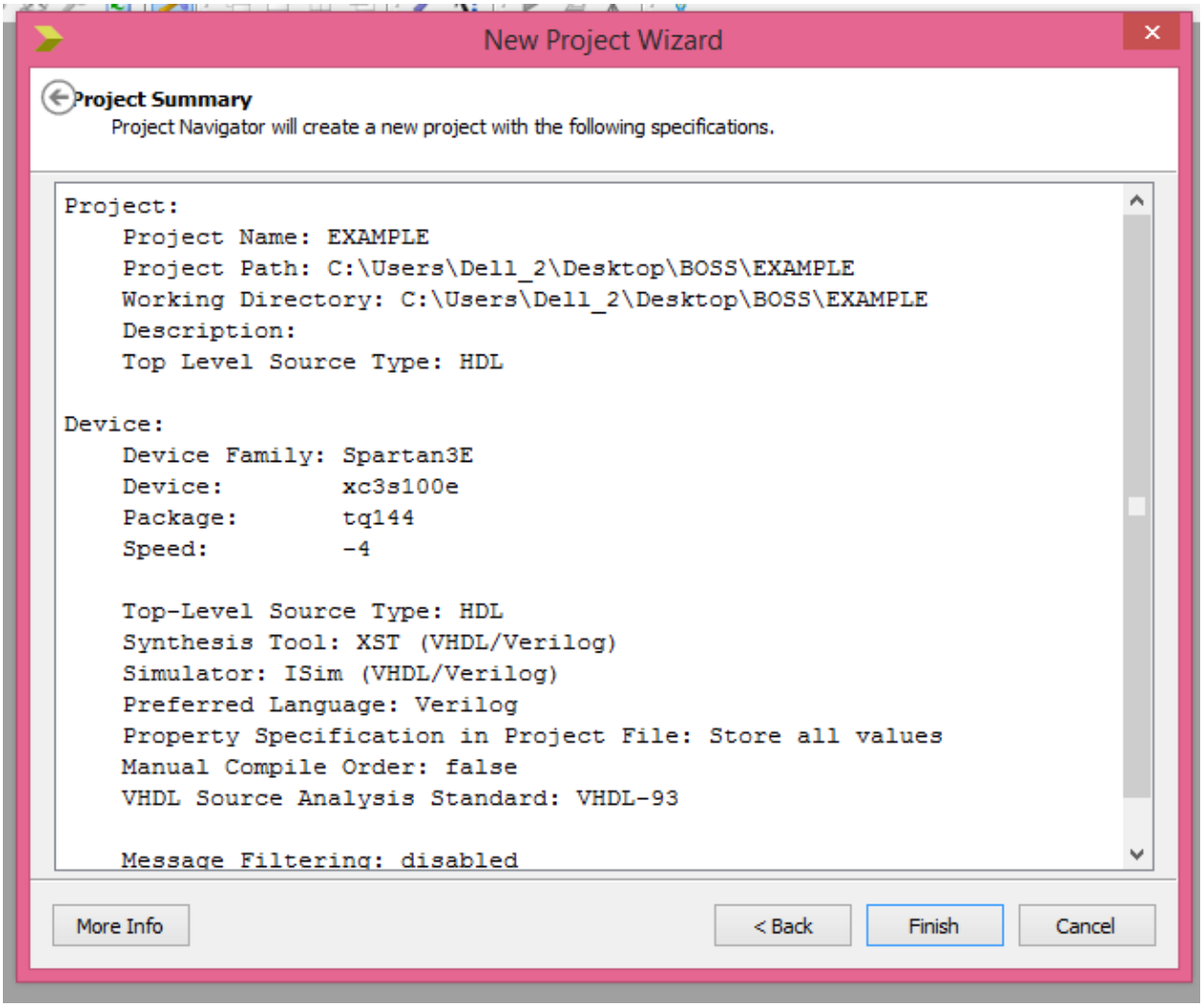

ISE Project Navigator (P.28xd)
File Edit View Project Source Process Tools Window Layout Help
Start
Welcome to the ISE® Design Suite
Project commands
Open Project...
Project Browser...
New Project...
Open Example...
Recent projects
Double click on a project in the list below to open
Additional resources
Tutorials on the Web
Design Resources
Application Notes
New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project
Name: EXAMPLE
Location: C:\Users\EXAMPLE
Working Directory: C:\Users\EXAMPLE
Description:
Browse For Folder
Select Directory
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Make New Folder
OK
Cancel
Next >
Cancel
Console


New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project
Name: EXAMPLE
Location: C:\Users\EXAMPLE
Working Directory:
Description:
New Project Wizard
The project location
'C:\Users\EXAMPLE'
is read only. Please try another location.
OK
Select the type of top-level source for the project
Top-level source type:
HDL
More Info
Next >
Cancel

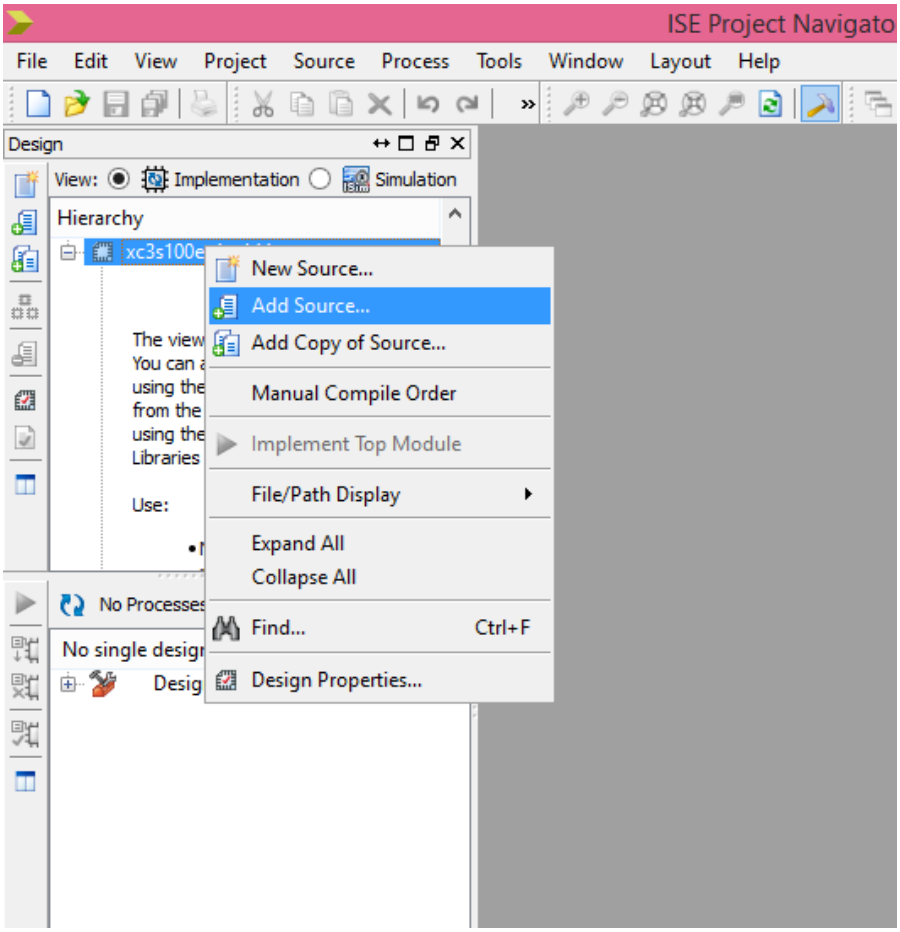
Step 3: CLICK ON NEXT and NEXT



Step 4: CLICK ON FINISH

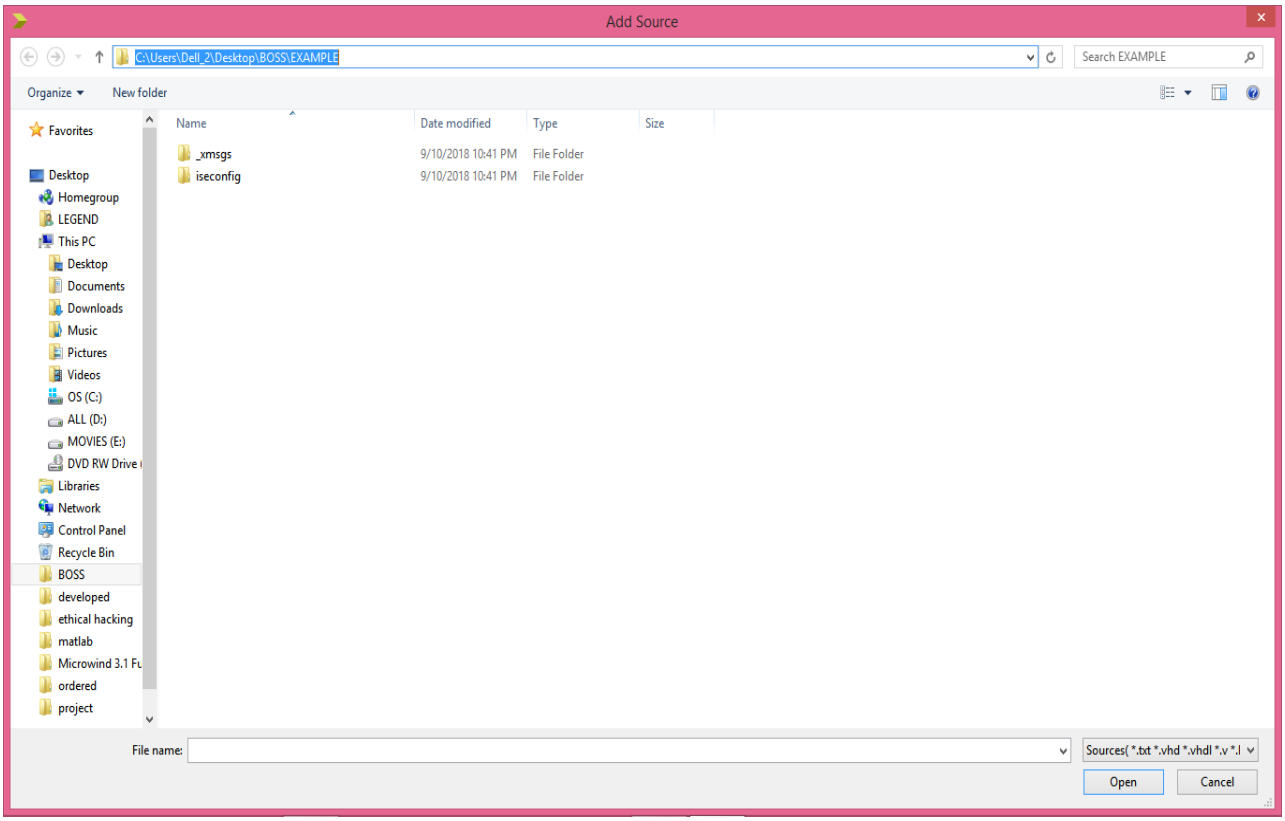


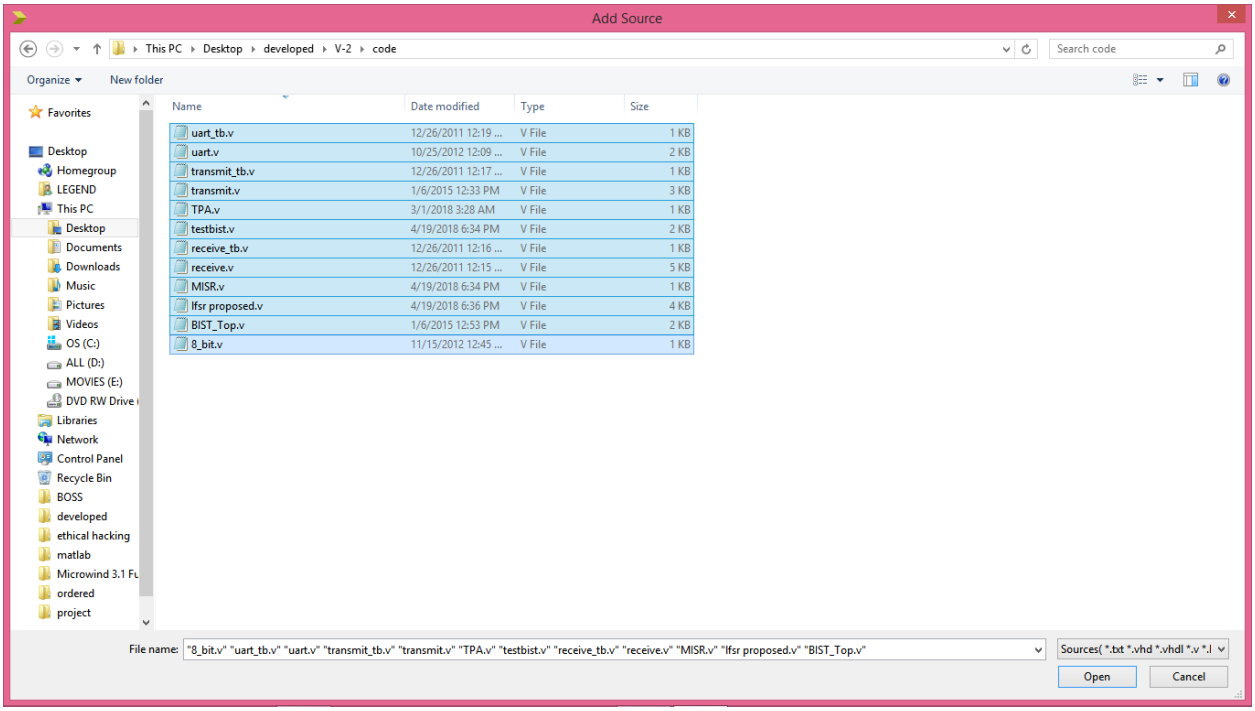
Step 5: CLICK ON CHIP (XC…) then MOUSE RIGHT CLICK then CLICK ON ADD SOURCE



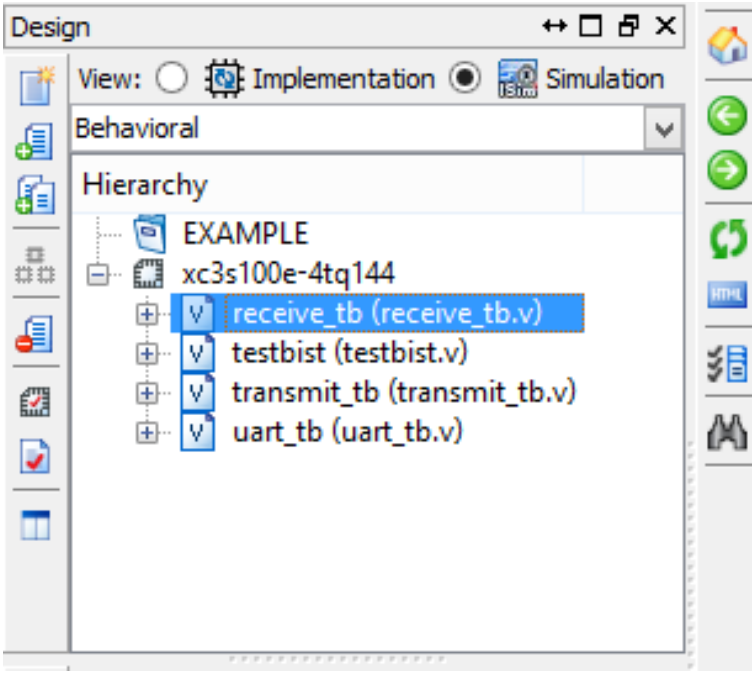
Step 6: SELECT THE CODE LOCATION GIVEN BY DEVELOPER AND ADD CODE

(Note ALL FILES) AND CLICK OPEN



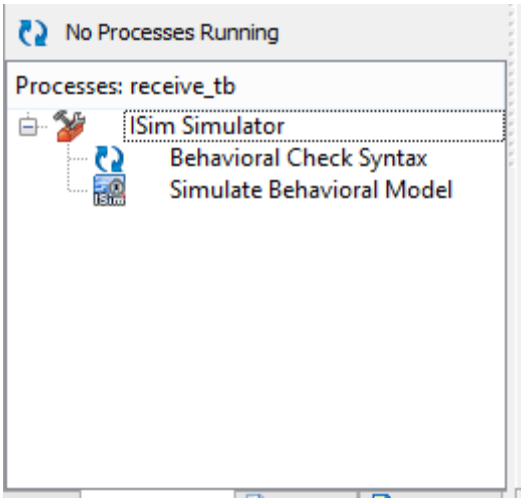


Step 7: SELECT THE SIMULATION and select files to RUN



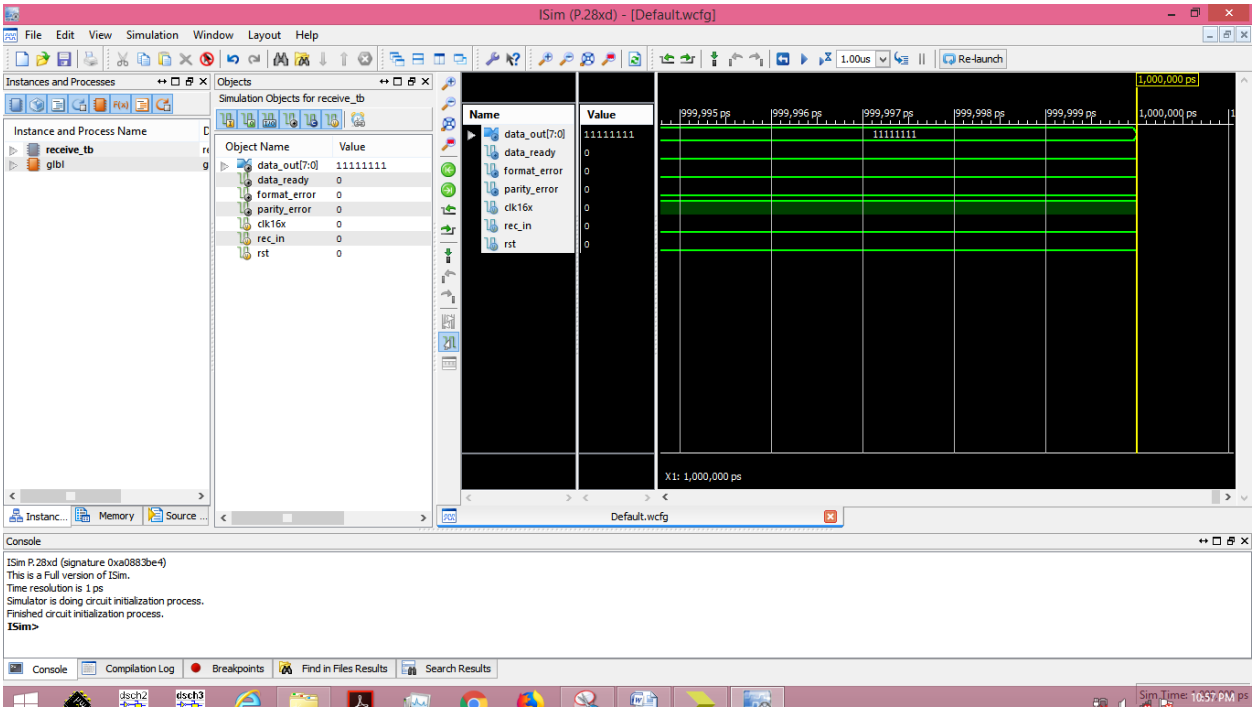
Step 8: SELECT ISIM SIMULATOR and SIMULATE BEHAVIORAL MODEL

If no errors isim window will open



Step 9: ISIM WINDOW

select zoom to full view



CHAPTER 7
SIMULATION RESULTS

Xilinx ISE software was used to create all of the DCM-TRN-TPG designs. This software programmed gives two types of outputs: simulation and synthesis. The simulation results provide a thorough examination of the DCM-TRN-TPG architecture in terms of input and output byte level combinations. Decoding procedure approximated simply by applying numerous combinations of inputs and monitoring various outputs through simulated study of encoding correctness. The use of area in relation to the transistor count will be accomplished as a result of the synthesis findings. In addition, a time summary will be obtained with regard to various path delays, and a power summary will be prepared utilizing the static and dynamic power consumption.

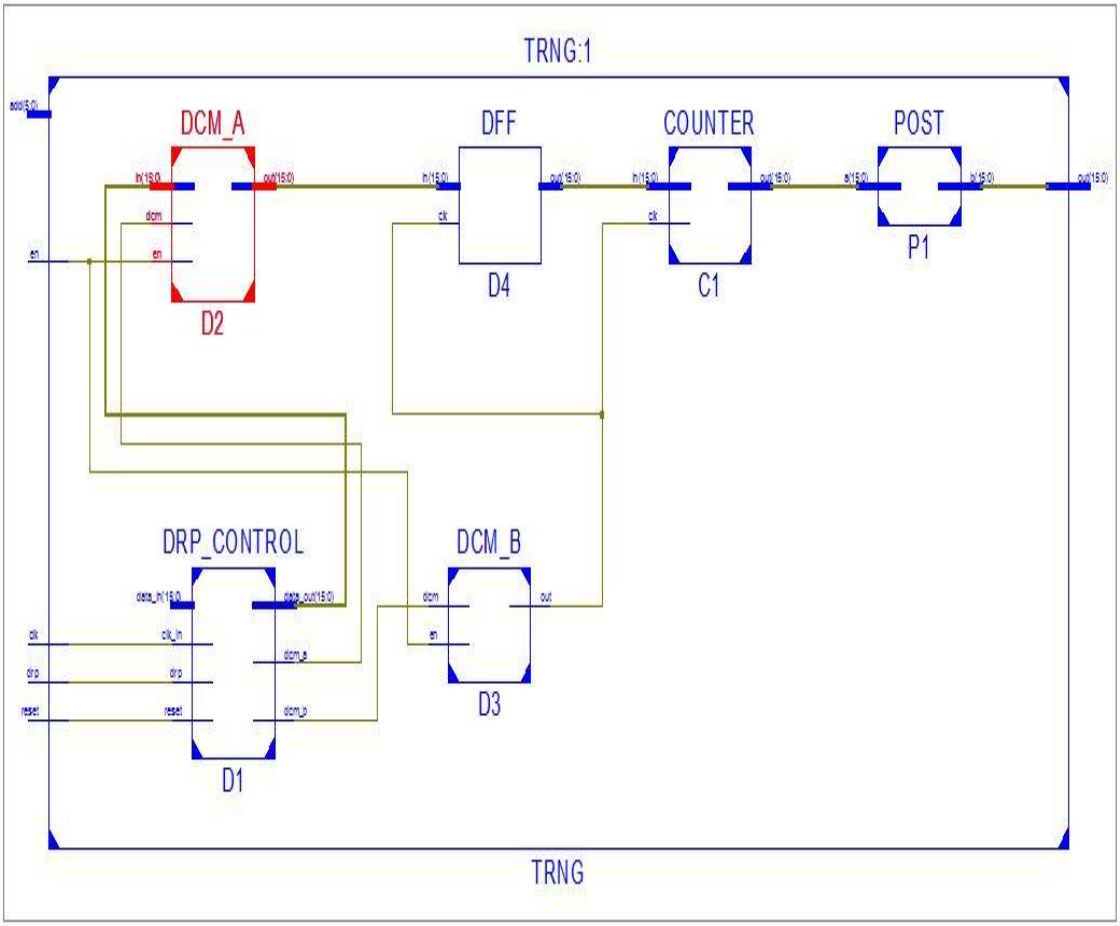


Figure 7.1. RTL schematic

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	16	35200	0%
Number of Slice LUTs	16	17600	0%
Number of fully used LUT-FF pairs	16	16	100%
Number of bonded IOBs	35	100	35%
Number of BUFG/BUFGCTRLs	1	32	3%

Figure 7.2. Design summary.

Figure 7.2 shows the design (area) summary of proposed method. Here, the proposed method utilizes the low area in terms of slice LUTs i.e., 16 out of available 17600. Further, the proposed method utilizes the slice registers as 16, out of available 35200. Further, the proposed method utilizes fully used LUT-FF as 16, out of available 16.

```
Data Path: load to r_LFSR_1
                           Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             16   0.000   0.497  load_IBUF (load_IBUF)
   LUT4:I1->O             1   0.043   0.000  mux161 (r_LFSR[15]_seed[15]_mux_3_OUT<0>)
   FDE:D                     -0.001          r_LFSR_1
  ----------------------------------------
  Total                      0.540ns (0.043ns logic, 0.497ns route)
                                     (8.0% logic, 92.0% route)
```

Figure 7.3. Time summary

Figure 7.3 shows the time summary of proposed method. Here, the proposed method consumed total 0.316ns of time delay, which is entirely route delay.

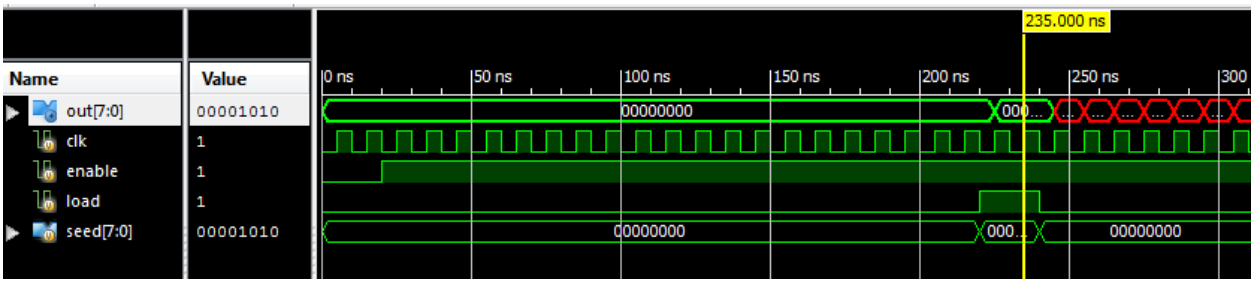


Figure 7.4. Simulation outcome.

Figure 7.4 presents the simulation outcome of proposed system. Here, clock (clk), enable, load, and seed are the input data pins. Further, out is the output pin. Initially, a logic high enable causes the system will function and a logic high load resulted in loading of seed value. So, 8 bit seed value (example 1010) is loaded into TRNG and resulted out as 1010. Further, the random numbers are generated for each clock cycle.

2. Summary
2.1. On-Chip Power Summary

On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	1.30	3	---	---
Logic	0.00	10	11776	0
Signals	0.00	20	---	---
IOs	0.00	20	372	5
Quiescent	31.52			
Total	32.83			

Figure 7.5. Power summary.

Figure 7.5 shows the power consumption report of proposed DCM-TRN-TPG. Here, the proposed DCM-TRN-TPG consumed power as 32.83 milli watts. Table 7.1 compares the performance evaluation of various TRNG controllers. Here, the proposed DCM-TRN-TPG resulted in superior (reduced) performance in terms of LUTs, slice registers, LUT-FFs, time-delay, and power consumption as compared to conventional approaches such as PRNG [22], PUF-TRNG [24], and RO-TRNG [25]. Further, the graphical representation of performance comparison is presented in Figure 7.6.

Table 7.1. Performance evaluation.

Metric	**PRNG [22]**	**PUF-TRNG [24]**	**RO-TRNG [25]**	**Proposed DCM-TRN-TPG**
Slice Registers	56	45	32	16
LUTs	67	55	42	16
LUT-FFs	74	52	39	16
Time delay (ns)	0.927	0.837	0.735	0.540
Power consumption (mw)	82.61	73.41	58.26	32.83

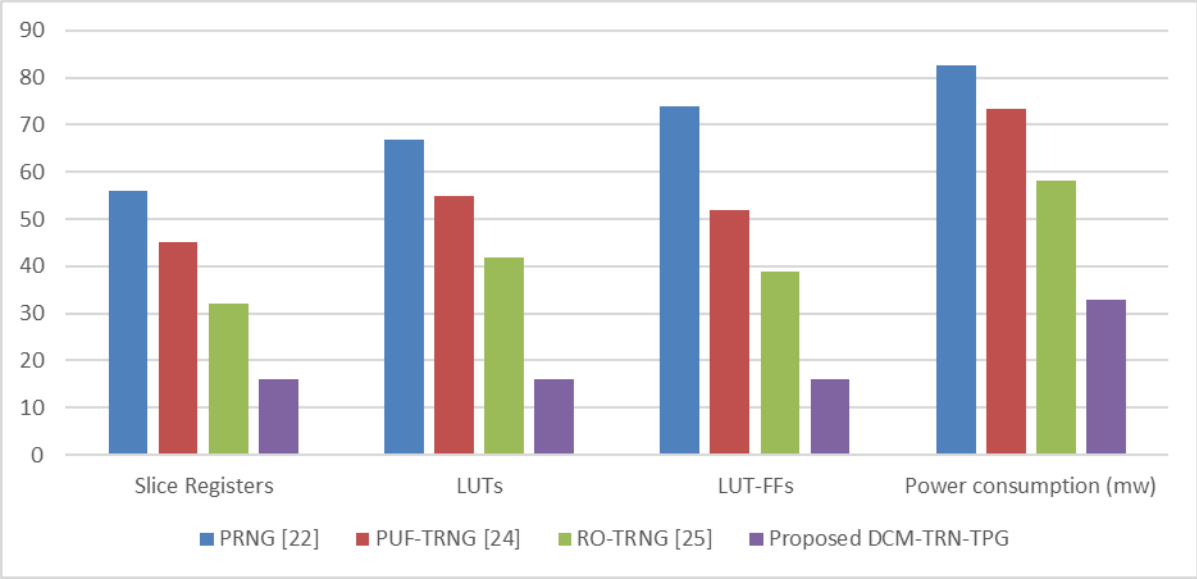


Figure 7.6. Graphical representation of performance evaluation.

CONCLUSION

The introduction of TRN-TPG frameworks to address this issue is the main goal of this effort. Furthermore, DCM, which enables the tuneability of phase and frequency of random numbers, took the role of the ring oscillators in the traditional approaches. Furthermore, D-FFs, post processing units, and counters are used to carry out the beat frequency detecting operation. The simulations showed that the suggested technique outperformed existing methods in terms of area, latency, and power. The work may be expanded to build real-time safe protocols including public key cryptography, RSA, ECC, and HECC cryptography methods using the DCM-TRN-TPG outputs as both public and private keys.

FUTURE SCOPE

Within the context of this project, we are now working with the numeric OTP system that consists of four digits. Therefore, in order to make the system more secure, we need to create an OTP system that uses alphanumeric symbols as its foundation.

REFERENCES

[1]. Tseng, Po-Hao, et al. "ReRAM-Based Pseudo-True Random Number Generator With High Throughput and Unpredictability Characteristics." *IEEE Transactions on Electron Devices* 68.4 (2021): 1593-1597.

[2]. Talukdar, Jonti, et al. "A BIST-based Dynamic Obfuscation Scheme for Resilience against Removal and Oracle-guided Attacks." *2021 IEEE International Test Conference (ITC)*. IEEE, 2021.

[3]. Saha, R., Geetha, G., Kumar, G., Buchanan, W. J., & Kim, T. H. (2021). A Secure Random Number Generator with Immunity and Propagation Characteristics for Cryptography Functions. *Applied Sciences*, *11*(17), 8073.

[4]. Singh, Vikram, and Kishor P. Sarawadekar. "FPGA Implementation of Chaos based Pseudo Random Number Generator." *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. IEEE, 2021.

[5]. Liu, B., Chang, Y. F., Li, J., Liu, X., Wang, L. A., Verma, D., ... & Lai, C. S. (2022). Bi2O2Se-Based True Random Number Generator for Security Applications. *ACS nano*, *16*(4), 6847-6857.

[6]. Bostancı, F. N., Olgun, A., Orosa, L., Yağlıkçı, A. G., Kim, J. S., Hassan, H., ... & Mutlu, O. (2022, April). DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 1141-1155). IEEE.

[7]. Garipcan, Ali Murat, and Ebubekir Erdem. "DESSB-TRNG: A novel true random number generator using data encryption standard substitution box as post-processing." *Digital Signal Processing* 123 (2022): 103455.

[8]. Liu, Jing, et al. "Min-entropy estimation for semiconductor superlattice true random number generators." *Scientific Reports* 12.1 (2022): 1-9.

[9]. Lu, Yi-Fan, et al. "A High-Performance Ag/TiN/HfOx/HfOy/HfOx/Pt Diffusive Memristor for Calibration-Free True Random Number Generator." *Advanced Electronic Materials* (2022): 2200202.

[10]. Zhou, Xue, et al. "Impact of relaxation on the performance of GeSe true random number generator based on Ovonic threshold switching." *IEEE Electron Device Letters* (2022).

[11]. Liao, Teh-Lu, Pei-Yen Wan, and Jun-Juh Yan. "Design and synchronization of chaos-based true random number generators and its FPGA implementation." *IEEE Access* 10 (2022): 8279-8286.

[12]. GALLI, DAVIDE. "On the effectiveness of FPGA implemented True Random Number Generators." (2022).

[13]. Addabbo, Tommaso, et al. "Low-Level Advanced Design of True Random Number Generators Based on Truly Chaotic Digital Nonlinear Oscillators in FPGAs." *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, Cham, 2022.

[14]. Xu, Mingtao, et al. "Voltage and temperature dependence of Random Telegraph Noise and their impacts on random number generator." *Microelectronics Journal* 125 (2022): 105450.

[15]. Gomez, Ana I., Markus Kiderlen, and Florian Pausinger. "Improved entropy bounds for parity filtered self-timed ring based random number generators." *Information Processing Letters* 174 (2022): 106212.

[16]. Lv, Yang, Brandon R. Zink, and Jian-Ping Wang. "Bipolar Random Spike and Bipolar Random Number Generation by Two Magnetic Tunnel Junctions." *IEEE Transactions on Electron Devices* 69.3 (2022): 1582-1587.

[17]. Shafi, K. M., Chawla, P., Hegde, A. S., Gayatri, R. S., Padhye, A., & Chandrashekar, C. M. (2022). Multi-bit quantum random number generator from path-entangled single photons. *arXiv preprint arXiv:2202.10933*.

[18]. Cirauqui, David, et al. "Quantum Random Number Generators: Benchmarking and Challenges." *arXiv preprint arXiv:2206.05328* (2022).

[19]. Zia, Unsub, et al. "A novel pseudo-random number generator for IoT based on a coupled map lattice system using the generalised symmetric map." *SN Applied Sciences* 4.2 (2022): 1-17.

[20]. Kong, Dexuan, et al. "Methodology for Random Number Generation Based on FPGA." *Proceedings of Sixth International Congress on Information and Communication Technology*. Springer, Singapore, 2022.

[21]. Bezuidenhout, Riaan, Wynand Nel, and Jacques M. Maritz. "Embedding tamper-resistant, publicly verifiable random number seeds in permissionless blockchain systems." *IEEE Access* 10 (2022): 39912-39925.

[22]. Gupta, Mangal Deep, and Rajeev K. Chauhan. "Hardware Efficient Pseudo-Random Number Generator Using Chen Chaotic System on FPGA." *Journal of Circuits, Systems and Computers* 31.03 (2022): 2250043.

[23]. Ryan, C., Kshirsagar, M., Vaidya, G., Cunningham, A., & Sivaraman, R. (2022). Design of a cryptographically secure pseudo random number generator with grammatical evolution. *Scientific reports*, *12*(1), 1-10.

[24]. Gao, B., Lin, B., Li, X., Tang, J., Qian, H., & Wu, H. (2022). A unified PUF and TRNG design based on 40-nm RRAM with high entropy and robustness for IoT security. *IEEE Transactions on Electron Devices*, *69*(2), 536-542.

[25]. Chu, Wei-Yu, et al. "Security Study of RO-TRNG under Fault Disturbance Scenarios." *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*. Vol. 6. IEEE, 2022.