Securing Hardware Random Number Generators against Physical Attacks

Yang CAO

Thesis submitted for the degree of Master of Science in Electrical Engineering, option Electronics and Integrated Circuits

Thesis supervisor: Prof. dr. ir. Ingrid. Verbauwhede

Assessors: Prof. dr. ir. Wim Dehaene Dr. Josep Balasch

Mentors: Bohan Yang Vladimir Rozic

Academic year 2015 – 2016
Preface

After an intensive year of reading, studying papers, experimenting and writing thesis, I would like to thank everybody who has made this possible.

First of all, I would like to thank Prof. Ingrid Verbauwhede for giving me the opportunity to do the research on COSIC.

Secondly, I want to thank my daily supervisors, Yang Bohan, Vladimir Rozic and Josep Balasch for your careful guidance and motivation on the research.

I also want to thank all the master students in ESAT. During the master years, you provide a warm environment in ESAT. I will always remember these two years in Belgium.

Finally, special thanks to my families, without their support and encouragement, I won’t have the opportunity to study at KU Leuven and reach my dream to be an electronic engineer.

Yang CAO
# Contents

Preface ................................................................. i  
Abstract ................................................................... iii  
List of Figures and Tables ........................................... iv  
List of Abbreviations and Symbols ................................... vii  
1 Introduction ............................................................... 1  
  1.1 Contributions ......................................................... 2  
  1.2 Thesis outline ......................................................... 3  
2 Background ................................................................ 5  
  2.1 Random number generators ........................................... 5  
  2.2 RNG designs ............................................................ 7  
  2.3 Statistical tests for RNGs ............................................. 14  
  2.4 Physical attacks on RNGs ............................................ 18  
3 TRNG design, implementation and tests ......................... 21  
  3.1 Set up design tools and hardware platform ................. 21  
  3.2 TERO based TRNG design ........................................ 22  
  3.3 Test designed TRNG ................................................ 32  
  3.4 Conclusion ............................................................. 37  
4 Physical attacks on designed TRNG .............................. 39  
  4.1 Isolation test suite ..................................................... 39  
  4.2 Freezing ................................................................. 40  
  4.3 Under/over-powering ............................................... 44  
  4.4 Conclusion ............................................................. 56  
5 On-the-fly tests ......................................................... 59  
  5.1 Test module design ................................................... 59  
  5.2 Parameter selection .................................................. 60  
  5.3 Test result .............................................................. 61  
  5.4 Conclusion ............................................................. 62  
6 Conclusions and future work ....................................... 63  
Bibliography ............................................................... 65
Abstract

In the first part of this work, we designed and implemented a True Random Number Generator (TRNG) using Transition Effect Ring Oscillator (TERO) as random source on Atlys Spartan-6 FPGA board. We proposed a new dynamic oscillation stop checking and then common constant Ctrl signal is replaced by adaptive Ctrl signal. We implemented TERO on different locations of FPGA board. Base on the oscillation occurrence situation, two locations are selected for TERO and two TRNGs both pass the evaluations of randomness including FIPS 140-2 and NIST SP800-22 tests.

We applied Freezing and underpowering attack experiment in the second part. An isolation test suite is proposed and tested with these attacks before applying attacks to designed TRNG to distinguish influence from other components on board. TERO is slightly influenced by freezing but strongly influenced by underpowering. The number of oscillation occurrence on transition stage as well as the randomness is significantly reduced by decreasing the supply voltage.

In the final part, an on-the-fly test module for detecting threats of attacks is proposed on designed TRNG during operation. On-the-fly test module is consisted of 3 basic statistical tests including mean value test, autocorrelation test and entropy test as well as oscillation test. We set critical bounds of \((\mu - 3\sigma, \mu + 3\sigma)\) of normal distribution for mean value and autocorrelation coefficient, the minimum entropy of 0.97 per bit for entropy test and 75 for oscillation test. On-the-fly test module is tested with different length of test sequence and finally we suggest longer test sequence due to experimental results.
# List of Figures and Tables

## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Basic block diagram of TRNG</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>The First Intel TRNG Design</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>General Structure of a Ring Oscillator based TRNG</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Galois Ring Oscillator</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Fibonacci Ring Oscillator</td>
<td>10</td>
</tr>
<tr>
<td>2.6</td>
<td>Galois &amp; Fibonacci Ring Oscillator based TRNG</td>
<td>11</td>
</tr>
<tr>
<td>2.7</td>
<td>New dynamic delay reconfiguration method</td>
<td>12</td>
</tr>
<tr>
<td>2.8</td>
<td>Transition Effect Ring Oscillators with different control gates</td>
<td>13</td>
</tr>
<tr>
<td>2.9</td>
<td>TERO based TRNG structure</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>General block diagram of TERO TRNG</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>First published version of TERO</td>
<td>23</td>
</tr>
<tr>
<td>3.3</td>
<td>TERO used in our design</td>
<td>24</td>
</tr>
<tr>
<td>3.4</td>
<td>Proposed placement for TERO</td>
<td>26</td>
</tr>
<tr>
<td>3.5</td>
<td>Four inverters combined in one slice</td>
<td>27</td>
</tr>
<tr>
<td>3.6</td>
<td>Final placement and routing for TERO</td>
<td>27</td>
</tr>
<tr>
<td>3.7</td>
<td>Export user constraints from FPGA editor</td>
<td>29</td>
</tr>
<tr>
<td>3.8</td>
<td>Asynchronous counter placed close to TERO</td>
<td>30</td>
</tr>
<tr>
<td>3.9</td>
<td>Constant control signal compare with dynamic control signal</td>
<td>30</td>
</tr>
<tr>
<td>3.10</td>
<td>Finite state machine for TERO</td>
<td>31</td>
</tr>
<tr>
<td>3.11</td>
<td>Oscillation occurrence test result of 8 different locations on FPGA</td>
<td>33</td>
</tr>
<tr>
<td>3.12</td>
<td>Glitches of oscillation occurrence in X8Y84, X20Y6 and X42Y42</td>
<td>34</td>
</tr>
<tr>
<td>3.13</td>
<td>Mean value test result</td>
<td>34</td>
</tr>
<tr>
<td>3.14</td>
<td>Auto correlation test result</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Isolation test suite</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Servisol Freeze it 20</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>Oscillation occurrence test result</td>
<td>42</td>
</tr>
<tr>
<td>4.4</td>
<td>Mean value test result on normal situation and freezing</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>Autocorrelation test result on normal situation and freezing</td>
<td>44</td>
</tr>
<tr>
<td>4.6</td>
<td>Bypassing the regulator of FPGA chip</td>
<td>46</td>
</tr>
<tr>
<td>4.7</td>
<td>Oscillation occurrence test result of location X10Y90</td>
<td>48</td>
</tr>
</tbody>
</table>
List of Figures and Tables

4.8 Oscillation occurrence test result of X44Y108 .................. 49
4.9 Average number of oscillation occurrence curve .................. 50
4.10 Mean value test result of X10Y90 .............................. 52
4.11 Mean value test result of X44Y108 .............................. 53
4.12 Autocorrelation test result of X10Y90 .......................... 54
4.13 Autocorrelation test result of X44Y108 .......................... 55
4.14 Average value of random bit and autocorrelation coefficient .... 56
4.15 Estimated Shannon entropy and minimum entropy ............... 57

5.1 On-the-fly test module ........................................... 60

List of Tables

2.1 Examples of the maximum-length polynomial .................... 8
2.2 Requirements for Runs test ..................................... 17
3.1 Settings for Xilinx ISE ........................................ 22
3.2 Estimated routing delays ..................................... 28
3.3 Distribution of variable X .................................... 35
3.4 FIPS 140-2 and NIST SP800-22 test results ..................... 36
4.1 Freezing attack isolation test result ............................. 41
4.2 Total average mean value and autocorrelation coefficients ...... 43
4.3 Distribution of variable X and entropy estimation ............. 45
4.4 Isolation test result: minimum effective supply voltage for each module 46
5.1 Parameters selected for on-the-fly test ......................... 61
5.2 On-the-fly test result (represented in alarm rate) for TERO based TRNG in X10Y90 at 0.70V, 0.75V and 1.20V supply voltage .......... 62
## Listings

3.1 Verilog source for and gates ........................................ 24
3.2 Verilog source for inverters ....................................... 25
3.3 Verilog source for user constraints .............................. 26
3.4 Exported user constraints ........................................... 28
List of Abbreviations and Symbols

Abbreviations

- FPGA: Field Programmable Gate Arrays
- PRNG: Pseudo Random Number Generator
- TRNG: True Random Number Generator
- TERO: Transition Effect Ring Oscillation
- LFSR: Linear Feedback Shift Register
- LUT: Look Up Table
- Inv: Inverter

Symbols

- $C$: Autocorrelation coefficient
- $H$: Shannon entropy
- $H_{\text{min}}$: Minimum entropy
- $\mu$: Mean of the distribution
- $\sigma$: Standard deviation
Chapter 1

Introduction

Random numbers are widely used in many applications such as electronic entertainments, mathematics or statistics science. They are also very important in cryptography applications. Session keys, password reset cookies and web application in secure communication all rely on large sets of random numbers. Having high security level requires strong randomness. Random numbers can be divided into two types: pseudo random and true random. Pseudo random numbers exhibit as statistical randomness but they are generated by an entirely deterministic causal process. This means they are predictable and duplicable. True random numbers are generated by physical phenomenon e.g. noise in the circuit which guarantees the unpredictable property.

A Random Number Generator (RNG) is a kind of device used to generate a sequence of random numbers. It can be built in both software and hardware. Software based RNGs are mostly used in computer science today because they are easy to be implemented or embedded in applications. They use mathematical algorithms to generate random numbers by expending short seeds into long bit sequence. Such sequence looks random but it is not truly random at all. They are pseudo random numbers and these software RNGs are referred to as Pseudo Random Number Generators (PRNGs).

Random numbers can also be generated from hardware. It is possible to implement mathematical algorithms like polynomial principle in hardware to generate random numbers. Such kind of hardware random number generators are similar to software RNGs and they are referred to as PRNGs, too. However, different from software random number generators, some hardware RNGs use physical phenomenon such as thermal noise and intrinsic noise from hardware circuit as their random source. These RNGs can generate true random numbers. With effective post processing algorithms, such kind of hardware RNGs can generate enough strong true random numbers. Random number generators which performs nondeterministic are referred to as true random number generators (TRNGs). Although such kind of TRNGs usually have lower bitrate than PRNGs, TRNGs are more secure in cryptographic applications because of its nondeterministic property. Therefore, in this thesis, such kind of hardware TRNGs will be implemented instead of PRNGs.
1. Introduction

In the past, hardware TRNGs were always designed based on some analog property. That means, whenever a cryptography application needs a TRNG, analog IC design is required. Therefore, if RNG is required in early digital designs, PRNGs are normally used. Otherwise additional analog part has to be implemented. With the development of Field Programmable Gate Arrays (FPGAs), nowadays it becomes more and more popular in digital design including cryptography applications because of its advantages in performance, flexibility or cost. Obviously, a TRNG is normally required in high secure level applications instead of a PRNG. However, adding analog parts into such design will increase the cost and complexity significantly. Therefore, it is necessary to develop pure digital TRNGs on FPGA. A pure digital TRNG should be designed and implemented by standard design tools like Xilinx ISE. It is also possible to make TRNGs as IP cores, which brings lots of flexibility in FPGA designs.

As mentioned above, random numbers perform an important role in cryptography applications. It is dangerous if an adversary knows the random numbers by getting control of the RNG inside a cryptography application. For instance, in encrypted communication, adversaries can easily get the secret keys once they successfully attack the RNG used to generate keys which means such encrypted communication is not secure for them any more. Therefore, an effective random number generator should have not only good statistical properties but also reliability against such attacks. There are many different attacks can be applied on TRNGs. They can be divided into two different types: passive attacks and active attacks.

Passive attacks are based on the results gathered from side channel information like power analysis [12] and EM analysis [9]. Such analysis can provide information inside circuit without modifying it. However, this kind of attacks are difficult to implement and can be defended by special secure design. Sometimes it is more effective to perform active attacks.

Different from passive side-channel analysis, active attacks are implemented in a more direct way to influence the TRNGs. Active attacks try to control the outputs of RNG rather than retrieve the outputs by analyzing observations. There are many different physical attacks that can be implemented on TRNGs. Typical attacks could be temperature changing [14], EM injection [3] or some other attack. Active attacks are destructive to electronic circuits, so, assailants also need to prevent destroying the circuit when applying active attacks.

1.1 Contributions

The main contribution of this thesis are:

1. To design and implement a high speed TRNG on FPGA as well as evaluating the TRNG using security tests.

Digilent Atlys FPGA board is used as the experimental FPGA platform in this thesis. This platform contains a Xilinx Spartan-6 XC6SLX45 FPGA chip and some
other useful external ports. Total system will be designed with Xilinx ISE 14.7 software. The TRNG implemented in this thesis will be chosen from several available TRNGs suitable for FPGA. Generated random numbers are uploaded via uart-usb to personal computer (PC). NIST test suite as well as several simple statistics functions will be used to test the random number sequence. NIST test gives a general overview of the randomness.

2. To evaluate the resistance of the TRNG against physical attacks and to design suitable protections

Other simple statistical functions help to compare the output of TRNG during experimental evaluation stage of attacks. Moreover, countermeasures will be discussed based on the experimental results after experimental stage. Finally, an on-the-fly test is proposed inside the TRNG system.

1.2 Thesis outline

This thesis includes 6 chapters:

Chapter 2 gives theoretical background. Examples of existing RNGs and physical attacks will be introduced in this chapter.

Chapter 3 explains the TRNG design and implementation flow on the FPGA board. After designing the TRNG, we will also examine it with several randomness tests including NIST 800-22 test. To make a comparison, the TRNG will be tested in different location on FPGA with different routings. Test results will be demonstrated and discussed in this chapter.

Chapter 4 covers the experiments of designed TRNG against physical attacks. We attempt to attack the TRNG in different situations e.g. different locations and evaluates the impacts made by attacks.

Chapter 5 presents a kind of on-the-fly test of designed TRNG. This on-the-fly test will be experimented with different situations and suitable parameters for the malfunction will be selected based on the experiment results.

Chapter 6 gives a summary of this thesis work and discussion of possible future directions.
Chapter 2

Background

This chapter will give an introduction to RNGs and active attacks.

2.1 Random number generators

RNGs can be divided into two main types: pseudo random number generator (PRNG) and true random number generator (TRNG).

2.1.1 Pseudo random number generators

A PRNG is a device that generates pseudo random number sequence. Such PRNG exhibits random-like behavior but is actually not truly random. For many pseudo random number algorithms, an initial state named seed is required. The output random number sequence is based on the initial seed. In other words, if the initial seed and the algorithm is known, a same RNG can be duplicated by others. Duplicated RNG will have the same performance as the original one. Such property makes the RNG pseudo.

Another important property of the PRNG is periodicity. Once the initial seed is fixed, the output of PRNG will always repeat after numerous intervals. For instance, a PRNG contains n-bit internal state is not able to have a period larger than $2^n$. Moreover, the period can be very short due to a bad initial seed. So it is always required to increase the length of state to make a PRNG strong enough. Adding more bits requires more hardware. And no matter how many bits are added for PRNG state, this RNG is always duplicable and periodic.

2.1.2 True random number generators

A TRNG generates nondeterministic true random numbers for use in high secure level applications. The most important property of hardware TRNG is true randomness. Unpredictability which means the output number always has no relation with previous output. For this reason, the behavior of hardware TRNG is not duplicable. As shown in Figure 2.1,a hardware TRNG contains 3 main parts: random source, digitalization and post processing [10].
2. **Background**

![Figure 2.1: Basic block diagram of TRNG](image)

**Random source**

Random source is the core part because it determines the randomness of the TRNG directly. Numerous random sources have been proposed such as thermal resistor noise [8], clock jitter [4] [6] [15] [17], meta-stability of the circuit [5] [11] [19], or even nuclear decay [7], which is obviously not suitable for common electronic circuits. The random sources generate analog signal that contains natural random property inside, e.g. the white noise.

**Digitalization**

The output of random source is analog signal. It is clear that such analog signal can not be used as output of a random number generator directly. Thus digitalization part is required to convert the analog signal into binary random numbers. There are different digitalization ways. For instance, a simple D flip-flop is enough for collecting clock jitter based random source. However a counter is required for collecting the oscillation time based random source. Obviously the digitalization part directly influences the speed of the TRNG. Using higher sampling frequency will enable higher bitrates of the TRNG. Unfortunately, higher sampling frequency sometimes can also decrease the randomness collected from the random source. This is why a TRNG is usually slower than a PRNG.

**Post processing**

Post processing is a method to compress the output numbers after digitalization. Post processing will increase the robustness of the TRNG but decrease the output bitrate at the same time. This part is not compulsory for a TRNG because sometimes the random source is strong enough for a random number generator. However, there is still possibility that the output of the random source has some bad properties, e.g. unnecessary bias. Then such bias can be eliminated by the post processing to ensure the randomness. A simple Von Neumann extractor [16] is the typical post
2.2 RNG designs

2.2.1 An example of PRNG design

Linear feedback shift register (LFSR)

A LFSR is a well-known PRNG which can be implemented in both software and hardware [20]. A LFSR is a kind of shift register which has a linear function as its feedback. The initial state of the shift register is called seed. Some bits of the shift register will take part in the linear feedback function and therefore influence the next state of the shift register. Positions of these bits are called taps on LFSR. By choosing different taps, LFSR will have different state transition which can be expressed in finite field arithmetic as a polynomial mod 2. These polynomials are called feedback polynomials. Most common feedback polynomial is based on XOR gate, which is easy to implement in digital circuits. A well-chosen polynomial function for a n-bit LFSR can guarantee the LFSR with the longest period of $2^n - 1$ in order to have the best pseudo randomness of LFSR. Such polynomial is called a maximal length polynomial. A polynomial is maximum length if and only if it is a primitive polynomial. Some maximal length polynomials are listed in Table 2.1. Moreover, each feedback polynomial can have two different kinds of LFSR structure: Fibonacci LFSR and Galois LFSR. Both structures with the same feedback polynomial will have the same length of state period.

- Fibonacci LFSR uses bits on taps as input of the feedback and generates a bit as the input bit of the shift register. It is noticed that 1 in the polynomial is the input bit of the shift register.

- Galois LFSR is also called one-to-many LFSR. Unlike Fibonacci structure, Galois LFSR only use the output bit of shift register as the input of feedback. The feedback is connected not only in the input of the shift register but also all the taps. Feedback with XORs are inserted in the taps. Note that the sequence of Galois structure’s taps is in reverse order compared to the Fibonacci structure.
2. Background

<table>
<thead>
<tr>
<th>Feedback polynomial</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$2^n - 1$</td>
</tr>
<tr>
<td>2</td>
<td>$x^2 + x + 1$</td>
</tr>
<tr>
<td>3</td>
<td>$x^3 + x^2 + 1$</td>
</tr>
<tr>
<td>4</td>
<td>$x^3 + x^2 + 1$</td>
</tr>
<tr>
<td>5</td>
<td>$x^5 + x^3 + 1$</td>
</tr>
<tr>
<td>6</td>
<td>$x^6 + x^5 + 1$</td>
</tr>
<tr>
<td>7</td>
<td>$x^7 + x^6 + 1$</td>
</tr>
<tr>
<td>8</td>
<td>$x^8 + x^6 + x^5 + x^4 + 1$</td>
</tr>
<tr>
<td>9</td>
<td>$x^9 + x^5 + 1$</td>
</tr>
</tbody>
</table>

Table 2.1: Examples of the maximum-length polynomial

LFSRs are widely used in digital system and software because of its simple structure and high speed. Also it is noticed that LFSRs can be used as some part of TRNGs. This will be discussed later.

2.2.2 An overview of TRNG designs

Intel’s first true random number generator

In the past, TRNGs use user random keystrokes, mouse movements or even nuclear delay as its random source. Such random sources are either very slow or not possible to implement in chip. In 1999, Intel proposed a new kind of TRNG\(^8\). Figure 2.2 shows its structure. The Intel TRNG uses thermal noise (also named Johnson noise) as its random source because thermal noise from a resistor is totally unpredictable. In this design, an amplifier is also added behind the noise resistor to make the thermal noise measurable. However, this amplifier may bring a problem that the output signal may have correlation with the environment parameters of the amplifier itself. For instance, lower supply voltage will have lower output signal. To minimize this problem, Intel samples two adjacent resistors at the same time and subtracts them as the output.

Another feature of the Intel first TRNG is the dual oscillator digitization part. Intel uses one fast and one much slower ring oscillator and suggests a ratio of 1:100 between these two oscillators. The output of amplifier is used to modulate the low speed oscillator. And thus the drift between these two oscillators provides the random number.

Finally, a post processing of digital extractor (Von Neumann extractor in this design) is used to eliminate the bias of the random number.

Multi-ring oscillator based TRNG

In digital designs, e.g. FPGA and CPLD designs, it is difficult and expensive to add additional analog parts. Therefore, other random sources are proposed. Clock jitter is one of them.
2.2. RNG designs

A common ring oscillator consists of an odd number of inverters. Due to the feedback, the output of inverter chain will oscillate from high to low and then back to high. In an ideal ring oscillator, the duty of the oscillation should be a constant which is determined by the delay of the chain. However, in real electronic circuits, there is always deviation called clock jitter between the real duty and the ideal duty. This jitter is usually random and can be used as a kind of random source. Figure 2.3 illustrates a block diagram of a ring oscillator based TRNG. There are two ring oscillators in the TRNG. The fast one is used as the random source and the slow one is used as sampling clock. In FPGA the slow one can be replaced by chip slow system clock. Such structure is very simple and only contains inverters and a D flip-flop.

Unfortunately, there are several problems with this design. Since jitter only takes place on the edge of clock, sampling at the edge of the ring oscillator will provide high randomness. However, the jitter is usually quite small compared to the clock duty which means sampling during the duty even cannot provide any randomness. Consequently, it is always required that sampling clock is well matched with the random source which is almost impossible in electronic circuit. A simple solution is combining several ring oscillators tighter as a random source to strength the
randomness. Schellekens et al [15] proposed a TRNG with 110 rings of 3 inverters and passed NIST tests. Sunar, Martin and Stinson [17] proposed another method that using multi-ring oscillators with different periods which are relative prime in number and extract them by using a XOR tree. They prove that this method will make the TRNG in highest efficiency compare to other multi-ring oscillators which uses the same number of rings. Also, base on the required randomness of the TRNG, they propose a complete design flow and reference tables consists all the parameters such as ring lengths and number of rings for the multi-ring oscillator TRNG.

**Galois ring oscillator and Fibonacci ring oscillator based TRNG**

Golić [6] proposed two new kinds of digital TRNG structures named Galois and Fibonacci ring oscillators. These two ring oscillators both have a feedback function like linear feedback shift registers. The only difference between new ring oscillators and LFSRs are the delay elements, which are replaced by the inverters.

A Galois ring oscillator of length $r$ is shown in Figure 2.4. This ring oscillator uses only output as the feedback to all the taps in the ring. Because of the same structure of Galois LFSR, the feedback function of them are also same. Based on the tap locations, the feedback function of the Galois ring oscillator can be represented as a binary polynomial: $x^r + \sum_{i=1}^{r-1} f_i \cdot x^i + 1$, where $f_i$ represents the state of the switch $i$ (on as 1, off as 0). Similar to the Galois LFSR, here coefficient $f$ is 1 when there is a feedback connection to the tap $i$, or 0 when there is no connection to the tap.

A Fibonacci ring oscillator of length $r$ is shown in Figure 2.5. This ring oscillator
uses output of taps as the feedback function inputs and generates feedback function output to the input bit. This kind of structure is similar to the Fibonacci LFSR of the same length. The feedback function of it is represented as the following binary polynomial: \( x^r + \sum_{i=1}^{r-1} f_i \cdot x^i + 1 \) which is the same with the Galois ring oscillator.

It is noticed that using inverters as the delay elements will introduce jitter into the output, which means the output will be truly random instead of pseudo random compared to the Galois/Fibonacci LFSR.

Moreover, Golić uses both two kinds of new ring oscillator as the random source and combines them by using a XOR as demonstrated in Figure 2.6. Such combination makes the random source stronger.

### Self-timed Ring Oscillator based TRNG

A Self-Timed Ring (STR) is a kind of oscillator which contains several same stages in loop. Each stage is consisted of a Muller C-element and an inverter. The Muller C-element provides a kind of 2-phase handshake protocol and such loop is named self-timed ring. Because of the 2-phase handshake protocol, signal transferred in the loop is never colliding, e.g. node \( F_n \) will keep the value until it is transferred into \( F_{n+1} \) node in STR.

Similar to normal ring oscillator, signal in the node of self-timed ring oscillator also contains jitter inside. Such jitter also can be collected as the random source. Based on such property, Cherkaoui [4] proposed a kind of TRNG. The author proves that due to the 2-phase handshake protocol, the deterministic jitter, which appears in normal ring oscillators and reduces the randomness, will not propagate through the self-timed ring oscillator. Such property gives self-timed ring oscillators better quality than normal ring oscillators in RNG designs.

### Delay chain based TRNG

Delay chain based TRNG is proposed by Danger [5]. This is a new kind of FPGA TRNG based on the meta-stability of the digital circuits. It contains two coarse chains, two fine chains, several D-latches and an XOR-tree. The clock signal is
2. Background

Figure 2.7: New dynamic delay reconfiguration method

split into two coarse chains which are named as data coarse chain and clock coarse chain respectively. These two coarse chains are used to eliminate the difference of clock signal and data signal caused by routing. In other words, the outputs of data coarse chain and clock coarse chain are well-synchronized. The output of data coarse chain will be used as the input of a delay chain on data (fine chain on data). The output of clock coarse chain will be used as the input of fine chain on clock. Both two fine chains have N nodes and there is a delay element between each two adjunct nodes. The delay elements are designed by LUT directly to have the same property. By carefully placing and routing, the data fine chain will then have the same property with the clock fine chain which means signal on each node k of N in data fine chain will be almost the same with the signal on the corresponding node k in clock fine chain. Therefore, using signal on each clock fine chain node to sample the corresponding node on data fine chain will bring the D flip-flop into meta-stable state and give a random output bit. Two fine chains with N nodes can generate N random bits at the same time. Finally, an XOR-tree is used to compress all the N output bits into one random bit as the output of the TRNG.

Danger implemented this design in a Virtex-5 FPGA and successfully passed NIST and AIS-31 tests. He also claims that this design will give high throughput as well as high security against coupling attack on oscillator based TRNGs. However, this design still has a drawback that it is strongly technology depended. Normally it is required to be optimized for each different FPGA family.

Majzoobi et al. [11] proposed an adaptive feedback control for delay chain based TRNG. The new feedback mechanism is made by performing fine delay tuning using so-called high precision Programmable Delay Lines (PDLs) with picosecond resolution. Traditional PDL uses switch matrix of FPGA to configure delay. However, changing the switch connections points and routings require a new configuration, and doing so during the circuit operation is only possible by dynamic reconfiguration. Mehrdad etc. proposed the new PDL only using a single Look Up Table (LUT) as shown in Figure 2.7. By selecting different multiplexers, the signal will have different route and thus the delay is reconfigured. Close loop feedback with such delay reconfiguration
ensures that the clock signals and data signals all arrive simultaneously at the flip-flop to drive it into a metastable state.

**Transition Effect Ring Oscillator (TERO) based TRNG**

As shown in Figure 2.8, Transition Effect Ring Oscillator (TERO), proposed by Varchola et al [19], is a structure that consists of even number of inverters (can be zero) with two control gates (XOR/XNOR, AND/NAND or OR/NOR, note that AND/OR TERO needs odd number of inverters in each chain but NAND/NOR TERO needs even.) in a loop. Actually TERO can be seen as a kind of RS/RS latch, whose R and S inputs are connected with the same control signal, other than a ring oscillator. A TERO has both stable state and meta-stable state that depend on the input of control gate. When the control signal converts, the TERO will come into meta-stable state (Also note that XOR/XNOR TERO is meta-stable when ctrl converts either from 0 to 1 or 1 to 0 but AND/NAND TERO is meta-stable only when ctrl converts from 0 to 1 and OR/NOR TERO is meta-stable only when ctrl converts from 1 to 0). As a result, TERO will oscillate due to the meta-stability. Total structure of TERO based TRNG is demonstrated in Figure 2.9. The oscillations
occur on meta-stability are counted by an asynchronous counter connected to the output of TERO and [19] illustrates that the number of oscillations depends on the intrinsic noise in the loop. Therefore, the counting result will be a random value after oscillation stops. Then, the last bit, which is also the parity of the counting result, is used as the random output in TERO based TRNG. Varchola tested with not only the last bit but also last 2, 3 or 4 bits and he declares that sometimes TERO based TRNG even passes FIPS test with 4 bits selected as the random number outputs [18].

The randomness of TERO based TRNG comes from the number of oscillations and thus the oscillation determines the reliability of the TRNG. In fact, the time of oscillations depends mainly on the symmetry of the structure and the accumulation of the random jitter in the oscillations (ideally the oscillation will last forever when there is no jitter). Therefore, it is important to select placement and routing for TERO carefully to make it symmetrical, as well as the control signal to make sure the oscillations die down within the control signal period. Unfortunately, even different locations of TERO on the same FPGA may have different oscillation situations. In other words, they have different maximum allowed frequency of control signal. To solve this problem, a dynamic check can be implemented. Once the output of TERO remains the same value for several system clocks, then it can be considered that the oscillation has stopped because the oscillation of TERO normally has higher frequency than system clock. A TERO with dynamic check will be adaptive for different locations and different FPGAs.

2.3 Statistical tests for RNGs

2.3.1 Basic statistical tests

The basic tests are mean value test, auto correlation test and entropy estimation. These three tests are simple to apply and they can give a general view of the randomness property.

- Mean value test

  Mean value test calculates the mean value of the sequence in order to check if the number of zeros and ones in the sequence is approximately the same. In
this test, the random sequence will be divided into blocks of 500 bits and then calculate the mean value of each block. A histogram will be drawn base on the test result to give an intuitive view.

• Autocorrelation test
The autocorrelation test is implemented in order to check the correlation between random bits in the sequence at different time. Autocorrelation coefficients are calculated by the following expression:

\[ C_i = \sum_{k=1}^{n-i} a_k \oplus a_{k+i} \]  

(2.1)

Where \( a_1 \ldots a_n \) are the bits of test sequence. Number \( i \) determines the interval between selected two bits. This test is the basic statistic test thus \( i \) is set as 1 here. Then the mean value of the coefficients is calculated as the auto correlation test. The block size is also 500 bits and the test result will be plot in a histogram.

• Entropy estimation
In cryptography entropy is defined as the randomness collected by the application. In other words, the entropy of the random sequence directly represents the randomness of the designed TRNG. In information theory, Shannon defined the entropy of a discrete random variable \( X \) as:

\[ H(X) = \sum_{i=1}^{n} p_i \log_2(p_i) \]  

(2.2)

where \( x_i \) represents possible values of \( X \), \( n \) is the total number of possible values and \( p(x) \) is the corresponding probability function.

Besides Shannon entropy, minimum entropy (min-entropy) is also introduced in the entropy estimation. The min-entropy is a very conservative measurement and often used for the worst case measure of uncertainty base on the observation of random variable \( X \) [2]. It determines the lower bound of information contained in \( X \). Still let \( x_i \) represent possible values of \( X \) and \( P(x) \) as the corresponding probability function. Then the min-entropy of \( X \) is calculated by:

\[ H_{min}(X) = -\log_2(max(p_i)) \]  

(2.3)

This is the worst case of the random sequence and thus the best case for adversaries who is guessing the random variable \( X \).

In this test, we set the hex of output random sequence as \( X \). The random sequence is divided into blocks of 4096 Bytes and the probability the proportion of \( x \) is estimated as \( P(x) \). Both Shannon entropy and min-entropy are calculated for comparison.
2. Background

2.3.2 NIST tests

NIST (National Institute of Standards and Technology) published several documents introducing the standards and tests for random number generators in cryptography applications.

FIPS 140-2

FIPS (Federal Information Processing Standard) 140[1] is the first standard issued by the NIST specifies the security requirements that should be satisfied by a cryptography module. The newest version of FIPS 140 is FIPS 140-2 issued on May 25th 2001. FIPS 140 is widely used for testing the statistical properties of random number generators. There are four tests included in the test: Monobit Test, Poker Test, Runs Test and Long Run Test.

• Test 1: Monobit Test

Count the number \( N \) of ones in the 20,000 bit stream. If \( 9725 < N < 10275 \), then this test is passed. This test is similar to the basic mean value test.

• Test 2: Poker Test

Divide a sequence of 20,000 into 5,000 consecutive 4-bit segments. Denote \( f(i) \) to be the number of each 4-bit value \( i \) where \( 0 < i < 15 \). Then calculate the following:

\[
N = \frac{16}{5000} \sum_{i=1}^{16} f(i)^2 - 5000
\]

(2.4)

The test is passed if \( N \) satisfies \( 2.16 < N < 46.17 \).

• Test 3: Runs Test

Run is defined as the maximum sequence of consecutive bits of either all ones or all zeros that is the part of the 20,000 bitstream. Count and store the run bits >1. If all the length of each run is fallen into the interval listed in Table 2.2, this test is passed.

• Test 4: Long Run Test

Long run is defined as a run with the length 26 or more. This test is passed if no long run is found in the 20,000 bit stream.
2.3. Statistical tests for RNGs

<table>
<thead>
<tr>
<th>Length of Runs</th>
<th>FIPS 140-2 Required Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2343-2657</td>
</tr>
<tr>
<td>2</td>
<td>1135-1365</td>
</tr>
<tr>
<td>3</td>
<td>542-708</td>
</tr>
<tr>
<td>4</td>
<td>251-373</td>
</tr>
<tr>
<td>5</td>
<td>111-201</td>
</tr>
<tr>
<td>6 and 6+</td>
<td>111-201</td>
</tr>
</tbody>
</table>

Table 2.2: Requirements for Runs test

NIST SP800-22

NIST also published another special document SP800-22 discussing the randomness test for RNGs[2]. In SP800-22, 15 statistical tests including 4 FIPS tests (originally 16 tests, but Lempel-Ziv compression test was removed finally) are discussed. These tests are:

- Frequency test (referred as monobit test in FIPS),
- Frequency test within a block,
- Runs test,
- Test for the longest run in a block,
- Binary matrix rank test,
- Discrete Fourier transform test,
- Non-overlapping template matching test,
- Overlapping template matching test,
- Universal statistical test,
- Linear complexity test,
- Serial test (referred as poker test in FIPS),
- Approximate entropy test,
- Cumulative sums test,
- Random executions test,
- Random executions variant test.

All the NIST tests can be implemented by using the NIST test suite which is available on the Internet.
2. Background

2.4 Physical attacks on RNGs

Although nowadays RNGs are developed more and more secure, as we all know, a system is as secure as its weakest link thus we can never promise that a RNG is always secure. Actually, attacks are developed at the same time with RNGs. So there are still a wide class of physical attacks that RNGs are vulnerable to. And these physical attacks are separated in passive attacks and active attacks. Active attacks are more effective than passive attacks thus we consider only active attacks in this work.

2.4.1 Active attacks

In some attacks, the adversary exerts some influence on the behavior of the target RNG, then such attacks are called active attacks. Active attacks are always implemented in order to control the target RNG’s output rather than infer the output random numbers. As a result, active attacks are somehow harmful to the target circuits.

Temperature changing

Temperature changing is a normal kind of active attack. It can be heating or freezing. Both heating and freezing will bring uncertainty to the circuit. It is noticed that some chips, e.g. chips on satellites, are designed and manufactured with special processes to adapt extreme environment temperature. Applying heating or freezing may have limited impact on such chips.

Under/over-powering

Under/over-powering is a common kind of active attack. It is not difficult to implement under/over-powering on digital circuits. Actually, lower supply voltage will make digital circuit slower and reduce the noise margin. As a consequence, it usually has impact on digital circuit. Besides common under-powering, there is also another way of attack by changing power supply called power glitch injection. By injecting glitches inside supply voltage for different frequency and core voltage, Martin et al. [14] declared a successful attack on STRNGs.

EM injection

EM injection is a kind of attack that uses an EM probe to insert EM wave interference on chip. Poucheret et al. have proved the influence of EM wave over the CMOS integrated circuits. An advantage of EM injection is this kind of active attack can be implemented contactless, which means destruction of the chip can be avoided. A good example is that Bayon et al. [3] use EM injection to attack the ring oscillator based TRNG. The ring oscillator is successfully locked by certain frequency of injected EM wave. Jitter of such ring oscillator is also reduced a lot with the lock of frequency. Without enough jitter, ring oscillator TRNG is no longer secure.
2.4. Physical attacks on RNGs

Circuit modification

Circuit modification is a kind of destructive physical attack. The basic idea of circuit modification is to connect or disconnect several parts in security system. For instance, some TRNGs use thermal resistor as its random source. Cutting down the connection between thermal resistor and sampler and connecting the sampler to a fixed port will cause the RNG to stuck at a fixed value. Note that such kind of modification is difficult to be implemented on chips due to the high reliance on equipments. Normally laser or focused ion beam is required in order to cut or paste tracks and add probe pads inside the chip. TRNG is no longer secure.

Other attacks

There are still many kinds of physical attacks such as UV and X-ray injection. They are much more dangerous, destructive and difficult to implement on circuit. All the attacks try to destroy the randomness of RNGs to alter or decrypt the secure system. It is necessary to experiment with these attacks for each designed RNG to achieve high security and reliability in cryptography applications.
Chapter 3

TRNG design, implementation and tests

This chapter describes the design and implementation flow presents the statistical test results of a TRNG on selected FPGA board. We select the transition effect ring oscillator (TERO) as its random source to generate random numbers.

3.1 Set up design tools and hardware platform

This design is developed by using Xilinx ISE design suite, coded by Verilog and implemented on Atlys Spartan-6 board.

3.1.1 Xilinx ISE Design Suite

This design is implemented on Spartan-6 FPGA by Xilinx ISE design suite. Xilinx ISE design suite is a software developed by Xilinx Inc. as design environment of FPGA products from Xilinx. ISE contains a complete design flow of FPGA applications. The design flow contains design, synthesis, simulation, implementation and generation of programming file as well as many useful IP cores.

The first part of design flow is coding. Synthesis is then introduced after coding. In this part, the codes, IP cores and user constraints will be compiled together by XTS to generate NGR, NGC and LOG files. NGC is the file needed for the following implementation. Such synthesis process also gives optimization which is controlled by synthesis options to the original codes when generating NGC file. Note that each component of the random number generator, even a dummy inverter without any load, may have impact on its randomness, so new settings of synthesis options is required to avoid unwanted optimization which may influence the quality of designed TRNG. Table 3.1 demonstrates the changes of settings in comparison with default settings. FSM encoding algorithm is removed because in experiments sometimes state machine will not work as expected after auto optimization. Others are set in order to avoid circuit modification by synthesis optimization such as duplicating Flip
3. TRNG design, implementation and tests

<table>
<thead>
<tr>
<th>Process</th>
<th>Option</th>
<th>Default setting</th>
<th>New setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis</td>
<td>FSM Encoding Algorithm</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Resource Sharing</td>
<td>Resource Sharing</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Register Duplication</td>
<td>Register Duplication</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Equivalent Register Removal</td>
<td>Equivalent Register Removal</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LUT Combining</td>
<td>LUT Combining</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Implementation</td>
<td>Trim Unconnected Signals</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.1: Settings for Xilinx ISE

Flops with high fanouts or reuses of counters. Then the designed TRNG will keep structure as described in its original codes.

After synthesis, the design has to be implemented. In implementation, the optimization option also needs modification. The option 'Trim Unconnected signal' should be removed in order to prevent circuit modification by optimization. The implementation process can be automatic or manual. Manual implementation in ISE is done by FPGA Editor which is a graphical application for displaying and configuring FPGAs. In this design, some parts of the TRNG are implemented manually because TRNG is really sensitive to the placement and route.

3.1.2 Atlys Board

The TRNG in this thesis is implemented on an Atlys board. The Atlys board, a product from Digilent Inc., contains a xc6slx45-2csg324 (Spartan-6 family) FPGA chip as its core. Spartan-6 FPGAs are composed of an array of configurable logic Blocks (CLBs), 2.1Mbits of fast block RAM, four clock tiles (eight DCMs and four PLLs), six phase-locked loops and 58 DSP slices. Each CLB consisted of 2 elements called slices which contains four 6- input LUTs and eight Flip-Flops in each slice. Atlys board also provides external components such as on-board 100MHz clock, LEDs, keyboards and switches.

3.2 TERO based TRNG design

The general block diagram of TERO based TRNG system in this thesis is as shown in Figure 3.1. It contains 4 parts: TERO randomness source, digitalization module, control module and communication module.

3.2.1 TERO design

**Determine TERO structure**

TERO is the randomness source and also the core part of the TRNG. As mentioned in Chapter 2, there are 3 main kinds of TERO based on the control gates. XOR/XNOR controlled TEROs have transition state (meta-stable state) when the control signal
3.2. TERO based TRNG design

Figure 3.1: General block diagram of TERO TRNG

Figure 3.2: First published version of TERO

changes whether from 1 to 0 or 0 to 1. But AND/NAND controlled TEROs only have such state when control signal converts from 0 to 1 and OR/NOR controlled TEROs from 1 to 0.

Although XOR/XNOR controlled TEROs have two times transition states with the same control signal compared with AND/NAND or OR/NOR controlled TEROs, there is a drawback of this TERO. This TERO does not contain a reset state so its starting condition is always random. Figure 3.2 is the first TERO structure which was proposed by Varchola et al. [19]. They added two ANDs after the XORs and then by controlling ANDs, such TERO would have known starting state. However, additional ANDs need additional controls which makes the control signal quite complex. Unlike XOR/XNOR controlled TERO, AND/NAND/OR/NOR controlled TERO doesn’t have such problem. When the control signal is set to 0 for AND/NAND or 1 for OR/NOR, the output is fixed as starting state. And the difference between NAND/NOR TERO and AND/OR TERO is the inverter chain. AND/OR TERO needs odd number of inverters in each chain while NAND/NOR TERO needs even. They are actually the same. Also note that in Xilinx FPGA, all the logic gates are represented as LUTs. Therefore, we simply select AND as the control gate of TERO in this design.

The length of inverter chain determines the delay between control signal and TERO output. This delay will determine the oscillation frequency in transition state and thus the length of inverter cannot be too long or too short. In this design, the
number of inverters in each chain is chosen as 7.

Moreover, the randomness source TERO has two outputs and one input. The control signal is connected with an inverter as input buffer. Two output nodes are connected with an inverter which are referred to as output buffer inverter and dummy inverter respectively to overcome the large difference of load capacitance caused by asymmetry output connection.

**Represent TERO in Verilog**

After determining the structure of TERO, then we have to represent it in ISE. According to the requirement of TERO and the structure of Spartan 6 FPGA, we designed the TERO in Verilog as follow:

First of all, all the elements of TERO, ANDs and inverters, are represented directly in LUTs. In Xilinx FPGAs, all the logic gates will be eventually represented as LUT-5/6s. During synthesis LUTs will be generated and distributed based on user options and constraints. Xilinx also provides a direct way to use LUTs in Verilog by instantiating LUT-6 component. Due to the symmetry requirements of TERO, we directly instantiate LUT-6 to represent logic gates in TERO instead to avoid the asymmetry caused by software synthesis. Listing 3.1 and 3.2 give the pieces of code to represent AND and inverter respectively by using LUT-6 directly. The initial value of LUT-6 for AND with inputs I0 and I5 should be set as x’AAAAA00000000 and for inverter with input I5 should be x’00000000FFFFFFFF. Then each LUT will have the same property as expected logics.

**Listing 3.1: Verilog source for and gates**

```
module And(
  input a,
  input b,
  output o);
  LUT6 #( .INIT ( 64 ’hAAAAA00000000 ) )
    And
```

---

Figure 3.3: TERO used in our design
3.2. TERO based TRNG design

Listing 3.2: Verilog source for inverters

```verilog
module Inv(input a, output o);
LUT6 #(.INIT(64'h00000000FFFFFFFF))
Not
(
  .O(o),
  .I0(1'b0),
  .I1(1'b0),
  .I2(1'b0),
  .I3(1'b0),
  .I4(1'b0),
  .I5(a)
);
endmodule
```

Secondly, we use a user constraint file to force the placement of LUTs. Although LUTs are instantiated directly, their placement is still unknown. ISE gives an auto placement and route during implementation stage. But such implementation is uncontrolled by user and obviously not suitable for TERO which is sensitive to placement. To solve this problem, user constraints are introduced here. User constraint file (UCF file) in ISE project usually contains two main constraints: pin constraint and timing constraint. Pin constraint is used to specify the external ports to match the design in specific evaluation board and general constraint file for Atlys board can be downloaded from Digiilentinc.com. Besides external pins, pin constraint also can be used to constrain NETs inside chip and thus we use pin constraints to lock specific LUTs for TERO in this design. As shown in Listing 3.3, common constraints for a LUT contains 3 lines: first line for slice location, second line for LUT number in selected slice and third line for locking the pins of the LUT (I0, I5 for And and I0 for inverter). In our design, each LUT is used as LUT-6 and only for one logic gate. Note that each slice contains four LUTs. Thus we can put at most four adjacent logic gates into one slice. In our design, there are in total one inverter as input buffer, 2 ANDs as control gates followed by 2 inverter chains combined into
3. TRNG Design, Implementation and Tests

![Diagram of TRNG design and implementation]

**Figure 3.4: Proposed placement for TERO**

A loop with 7 inverters in each and 2 inverters as output buffers in TERO. Inv₀ (Input buffer), each AND occupies one slice. Inv₀ is put in a slice_m. Then And₁ is located in a slice_x (a slice in odd column of Spartan 6 is referred to as a slice_x, and in even columns is referred to as a slice_m) of next CLB in the same row and And₂ is in the corresponding slice_x on the next row under And₁. To have better symmetry property, only slice_x is used for the following two inverter chains in loop. Each chain contains 7 inverters as well as an output buffer. Thus 2 slice_x's are occupied by each chain with four adjacent inverters in each slice_x.

**Listing 3.3: Verilog source for user constraints**

```verilog
INST "TERO/LUT6_not_0" LOC=SLICE_X10Y90;
INST "TERO/LUT6_not_0" BEL="C6LUT";
INST "TERO/LUT6_not_0" LOCK_PINS="ALL";
```

Finally, there are still some fine adjustments with routing in TERO. Similar to placement, manual routing can also be implemented by user constraints. But actually routing is much more complex than placement because it is impossible to have location of every line and switch matrixes. An alternative way to do routing manually is using FPGA editor. FPGA Editor is a graphical application used to display and configure FPGAs. It also can provide an estimation of delay of each line in FPGA design. The routes of And₁ and And₂ with corresponding inverter chain are adjusted to the same in order to have the same delay in each chain. It is not possible to route two feedback lines, from Inv₇₁ to And₂ and Inv₇₂ to And₁, as the same. Thus these two lines are routed to have the same delay. Also two input routes from Inv₀ to And₁ and And₂, are also tried to have similar delay. The final routes are illustrated in Figure 3.6 and the delay of the routes is shown in Table 3.2. The results show that this design is quite symmetrical.

The constraints of routes can also be exported into UCF file by FPGA editor. As
3.2. TERO based TRNG design

Figure 3.5: Four inverters combined in one slice

Figure 3.6: Final placement and routing for TERO
### Table 3.2: Estimated routing delays

<table>
<thead>
<tr>
<th>Starting point</th>
<th>End point</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Inv_0$</td>
<td>$And_1$</td>
<td>0.889</td>
</tr>
<tr>
<td>$And_1$</td>
<td>$Inv_{11}$</td>
<td>0.618</td>
</tr>
<tr>
<td>$Inv_{11}$</td>
<td>$Inv_{71}$</td>
<td>1.882</td>
</tr>
<tr>
<td>$Inv_{71}$</td>
<td>$And_2$</td>
<td>0.589</td>
</tr>
<tr>
<td>$Inv_{71}$</td>
<td>$Inv_{out_1}$</td>
<td>0.560</td>
</tr>
<tr>
<td>$Inv_0$</td>
<td>$And_2$</td>
<td>0.909</td>
</tr>
<tr>
<td>$And_1$</td>
<td>$Inv_{12}$</td>
<td>0.618</td>
</tr>
<tr>
<td>$Inv_{12}$</td>
<td>$Inv_{72}$</td>
<td>1.882</td>
</tr>
<tr>
<td>$Inv_{72}$</td>
<td>$And_1$</td>
<td>0.589</td>
</tr>
<tr>
<td>$Inv_{72}$</td>
<td>$Inv_{out_2}$</td>
<td>0.560</td>
</tr>
</tbody>
</table>

Listing 3.4: Exported user constraints

```plaintext
NET "TERO/inv_0"
ROUTE="{3;1;6 slx45csg324;ca56ac17!−1;−48856;87552;S!0;
−683;208!0;−8!1;" *−1749;−1355!2;−683;224!3;2917;
−693!4;1038;−2160!5;2982;2176!6;3112;−920!" '7:845:56:L!8:845:56:L!}";

NET "TERO/and_1"
ROUTE="{3;1;6 slx45csg324;510e485d!−1;−48696;100816;S!0;
−843;−376!1;1008;" *1264!2;3142;−656!3;−995;1301!4;995;−1501!5;845:144:144:L!}";

NET "TERO/and_2"
ROUTE="{3;1;6 slx45csg324;bd506685!−1;−48696;104016;S!0;
−843;−376!1;1008;" *1264!2;3142;−656!3;−995;1301!4;995;−1501!5;845:144:144:L!}";
```

shown in Figure 3.7, open directed routing constraints, select corresponding NETs and export constraints in relative or absolute location. Then the constraints will be generated in UCF file and thus we do not have to open FPGA editor to do such modifications each time after synthesis and auto implementation.

#### 3.2.2 Digitalization module

In TERO based TRNG, the randomness comes from jitter in oscillation, which will affect the pulse length and results in a random number of oscillation occurrences during transition state. Thus the digitalization module is normally a counter connected with the output of TERO. The dummy output is also connected to a
3.2. TERO based TRNG design

Figure 3.7: Export user constraints from FPGA editor

counter in order to balance the load. Note that the lowest frequency of oscillation is actually determined by the delay of each chain. From delay estimated from FPGA editor, it is clear that the oscillation frequency is much higher than system clock frequency. Common synchronous counters cannot be used here because they are too slow. Hence asynchronous counters are used here to count the oscillation occurrence. The asynchronous counter used here consists of several rising edge triggered TFFs in serial. Each TFF represents a bit of the number and the last bit is used as the random output in our design. Moreover, the location of the asynchronous counter should also be considered in the design in order to keep symmetry on output. In our design, location of the counter is forced in the slice following the output inverter shown in Figure 3.8.

3.2.3 Control module

In the existing designs of TERO based TRNG, the TERO is always driven by a constant control signal. The structure of a constant control module is really simple however there is a drawback of constant control signal. One of the most important properties of TERO is that the random number should not be extracted before oscillation stops. If constant control module is used, then the pulse length of control signal must at least satisfy the worst case of oscillation (the largest number of
3. TRNG DESIGN, IMPLEMENTATION AND TESTS

Figure 3.8: Asynchronous counter placed close to TERO

Figure 3.9: Constant control signal compare with dynamic control signal

oscillation). There are two serious problems for constant control signal. Firstly, always keeping TERO on the worst case will absolutely decrease the output data rate of random number. Secondly, the oscillation is random, and the worst case in different location or different FPGAs are always different due to the process variations. This means the number of oscillation occurrence is placement and FPGA dependent. As a result, the constant control signal is also placement and FPGA dependent. Every time the placement is modified or another FPGA is used, the control signal always needs modification to fit the new worst case.

Therefore, a dynamic control signal is introduced to replace the constant control signal in our design. Dynamic control signal needs dynamic check on oscillation stop. Figure 3.9 demonstrates the comparison between constant control signal and dynamic stop check. By using dynamic stop we can achieve highest throughput and
avoid errors. The control module is implemented as a finite state machine as shown in Figure 3.10. Rst state represents the reset period of TERO, only one system cycle is enough for this state. The transition state is represented as Trans followed by dynamic oscillation stop check. The counting number of oscillation is always compared with the number in previous system cycle. Once the counting number retains for several cycles (5 in our design), then the oscillation is considered as stopped. As the oscillation frequency is much higher than system clock, actually 5 cycles contains enough time margin for dynamic check. The dynamic check guarantees the TRNG always work on the highest random number throughput and makes the TRNG placement adaptive.

After transition state, the state machine will work base on the mode selection. In Mode 0 the number of oscillation occurrence will be stored into the block ram directly while in Mode 1 the random bit will be shifted into a 16-bit shift register.
And random numbers are saved when shift register is filled by 16 new random bits in Mode 1. The random data will be sent to PC when the block ram is full. Note that the block ram is dispensable, the random data can also be sent directly to PC.

3.2.4 Communication module

Communication module in the TRNG is implemented in order to transmit data from FPGA to external equipment like PC for further analysis. In our design, UART-16650 (Universal Asynchronous Receiver Transmitter) is implemented as the communication module. UART is used for serial communication which sends one start bit, 8 data bits and one stop bit as one frame of data each time. The baud rate, the same as data rate, is selected in 230400 bits per second (the maximum baud rate supported by software in our PC), with no parity bits in this design to reach the highest speed.

3.3 Test designed TRNG

Several tests are applied to check and improve our designed TRNG and test results are demonstrated in this section.

3.3.1 Oscillation occurrence test result

In TERO TRNG, the randomness is extracted from the number of oscillation during transition state. So, the first test for TERO TRNG is the oscillation occurrence test. Placement and routing are the same as mentioned in Chapter 3.2. The only difference is the location coordinate. In this test, several different location coordinates for TERO (represented by the coordinate of Inv) are selected to compare the oscillation occurrence and then determine the suitable locations.

Figure 3.11 gives several typical results that we observed from oscillation occurrence test of 8 different locations. It is clear that not all of them are suitable for a TRNG.

As we can see, in X10Y4, X44Y108 or X32Y62, the oscillation occurrence is quite low. Either mean value or maximum value of oscillation occurrence of them are below 100. The lower the number of oscillation occurrence, the fewer possible values of oscillation occurrence can be obtained. In X32Y62, there are only four possible values, which will absolutely have impact on the randomness.

In X42Y42, X20Y6 and X8Y84, the number of oscillation occurrence is much higher. However, in these 3 locations, there is another problem. As shown in Figure 3.12, occurrence of each number is much higher or lower than its adjacent two numbers. This is called glitch. In this design, such glitches will influence the bias of random sequence because the parity of oscillation occurrence used as the randomness source is influenced by the glitches. So these situations also need to be avoided.

Two suitable situations of oscillation occurrence are observed in X10Y90 and X44Y6, both of them are perfect normal distributed with mean value over 100 and without glitches. Thus according to the test result location coordinates X10Y90 and
Figure 3.11: Oscillation occurrence test result of 8 different locations on FPGA
3. TRNG DESIGN, IMPLEMENTATION AND TESTS

Figure 3.12: Glitches of oscillation occurrence in X8Y84, X20Y6 and X42Y42

Figure 3.13: Mean value test result

X44Y6 are chosen to implement TERO and the following tests will be applied on these locations.

3.3.2 Basic statistical test results

In this part, we apply basic tests on the TERO TRNG in the designs located in X10Y90 and X44Y6.

Figure 3.13 gives the mean value distribution of two TERO TRNGs. Each TERO involves 40,000,000 random bits in this test. As we can see, both of them are similar to normal distribution with bias of 0.5. Also the mean value of total test sequence is calculated. The mean value of X10Y90 is 0.5001 and X44Y6 is 0.4999.

The test result of auto correlation test is illustrated in Figure 3.14. Both locations have similar distribution of auto correlation to normal distribution with bias of 0.5.

Table 3.3 shows the distribution and calculated Shannon entropy and minimum entropy based on equation 2.2 and 2.3. The entropy is calculated of 4-bit random variable X. The probability function is estimated by the distribution of each possible value of X. Both locations have high entropy.
3.3. Test designed TRNG

Figure 3.14: Auto correlation test result

<table>
<thead>
<tr>
<th>Variable X</th>
<th>Occurrence in X10Y90</th>
<th>Occurrence in X44Y6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>625215</td>
<td>630566</td>
</tr>
<tr>
<td>1</td>
<td>626642</td>
<td>629415</td>
</tr>
<tr>
<td>2</td>
<td>628181</td>
<td>630621</td>
</tr>
<tr>
<td>3</td>
<td>628172</td>
<td>631447</td>
</tr>
<tr>
<td>4</td>
<td>627794</td>
<td>630651</td>
</tr>
<tr>
<td>5</td>
<td>627996</td>
<td>629212</td>
</tr>
<tr>
<td>6</td>
<td>626801</td>
<td>630775</td>
</tr>
<tr>
<td>7</td>
<td>626627</td>
<td>630025</td>
</tr>
<tr>
<td>8</td>
<td>627021</td>
<td>630930</td>
</tr>
<tr>
<td>9</td>
<td>627693</td>
<td>628725</td>
</tr>
<tr>
<td>A</td>
<td>627556</td>
<td>629829</td>
</tr>
<tr>
<td>B</td>
<td>626767</td>
<td>630061</td>
</tr>
<tr>
<td>C</td>
<td>628630</td>
<td>630252</td>
</tr>
<tr>
<td>D</td>
<td>626327</td>
<td>630691</td>
</tr>
<tr>
<td>E</td>
<td>627087</td>
<td>630081</td>
</tr>
<tr>
<td>F</td>
<td>627715</td>
<td>629023</td>
</tr>
<tr>
<td>Total</td>
<td>10036224</td>
<td>10082304</td>
</tr>
<tr>
<td>Shannon entropy</td>
<td>4.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>Min-entropy</td>
<td>3.9973</td>
<td>3.9969</td>
</tr>
</tbody>
</table>

Table 3.3: Distribution of variable X

As seen in the test result above, the statistical property of designed TRNG is quite good.
3. TRNG Design, Implementation and Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>X10Y90 P-value</th>
<th>Proportion</th>
<th>X44Y6 P-value</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monobit</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Poker</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Runs</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Long Run</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Frequency</td>
<td>0.554420</td>
<td>100/100</td>
<td>0.739918</td>
<td>100/100</td>
</tr>
<tr>
<td>BlockFrequency</td>
<td>0.304126</td>
<td>99/100</td>
<td>0.779188</td>
<td>100/100</td>
</tr>
<tr>
<td>Runs</td>
<td>0.971699</td>
<td>100/100</td>
<td>0.455937</td>
<td>100/100</td>
</tr>
<tr>
<td>LongestRun</td>
<td>0.090936</td>
<td>100/100</td>
<td>0.181557</td>
<td>100/100</td>
</tr>
<tr>
<td>Binary matrix rank</td>
<td>0.455937</td>
<td>97/100</td>
<td>0.851383</td>
<td>99/100</td>
</tr>
<tr>
<td>Discrete Fourier transform</td>
<td>0.987896</td>
<td>100/100</td>
<td>0.595549</td>
<td>98/100</td>
</tr>
<tr>
<td>Non-overlapping templates</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Overlapping template</td>
<td>0.437274</td>
<td>98/100</td>
<td>0.350485</td>
<td>100/100</td>
</tr>
<tr>
<td>Universal statistical</td>
<td>0.911413</td>
<td>97/100</td>
<td>0.037566</td>
<td>98/100</td>
</tr>
<tr>
<td>Linear complexity</td>
<td>0.851383</td>
<td>99/100</td>
<td>0.834308</td>
<td>100/100</td>
</tr>
<tr>
<td>Serial</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Approximate entropy</td>
<td>0.798139</td>
<td>98/100</td>
<td>0.401199</td>
<td>97/100</td>
</tr>
<tr>
<td>Cumulative sums-Forward</td>
<td>0.191687</td>
<td>100/100</td>
<td>0.574903</td>
<td>100/100</td>
</tr>
<tr>
<td>Cumulative sums-Reverse</td>
<td>0.319084</td>
<td>100/100</td>
<td>0.249284</td>
<td>100/100</td>
</tr>
<tr>
<td>Random executions</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>Random executions variant</td>
<td></td>
<td>Success</td>
<td></td>
<td>Success</td>
</tr>
</tbody>
</table>

Table 3.4: FIPS 140-2 and NIST SP800-22 test results

3.3.3 NIST test results

In this part, both FIPS 140-2 and NIST SP800-22 tests are applied to the selected locations of TERO. NIST test suite is used as the test platform to perform both FIPS140-2 and NIST SP800-22 tests. A bitstream of 20,000 random bits are tested in FIPS 140-2 and 100 bitstreams of 1,000,000 random bits in each are tested in NIST SP800-22. Test result is shown in Table 3.4: TERO (in both locations) pass the FIPS 140-2 and NIST SP800-22(minimum pass required proportion 96/100) successfully.

3.3.4 Throughput check

In addition, we apply a throughput test to check the bitrate of our TRNG design. A synchronous counter is used to accumulate the system cycles that TERO based TRNG occupied to generate a random bit. This test continues for one million times with system clock of 100MHz. The average number of system cycle consumption is 77 for X10Y90 and 80 for X44Y6. We consider this number as the normal requirement for generating one random bit and thus the throughput of designed TRNG is 100M ÷ 77 ≈ 1.29Mbps for X10Y90 and 100M ÷ 80 = 1.25Mbps for X44Y6.
3.4 Conclusion

In this chapter, we describe our design and implementation of a TERO based TRNG with dynamic oscillation stop check. The TERO is placed and routed manually in order to achieve high symmetry. Then several tests including NIST tests are applied in designed TRNG. We select the location of TERO based on the oscillation occurrence test. Results of following statistical tests show that the designed TRNG on selected location X10Y90 and X44Y6 can satisfy the requirements of FIPS 140-2 and NIST SP800-22 and thus the designed TRNG is proved effective. In addition, we also check the throughput of the TRNG which in this design is 1.29Mbps in X10Y90 and 1.25Mbps in X44Y6 on average.
Chapter 4

Physical attacks on designed TRNG

This chapter covers the experiments of physical attacks against designed TRNG. Using the available equipment in the lab, we perform freezing and under/over-powering attacks on TERO TRNG. To have a intuitive view of attack influence, experimental results are recorded in this chapter and analyzed by oscillation occurrence and effect on the statistical tests described in Chapter 2.

4.1 Isolation test suite

Physical attacks against TRNG designs may not only influence the random source of random number generator, but also other components on the FPGA. Thus an isolation test suite is designed here. We will firstly apply attacks on isolation test suite and check the impact on each module in the suite.

As shown in Figure 4.1, there are 4 main tests in the isolation test: block RAM test, shift register test, asynchronous counter test and UART communication test. In order to avoid interference of each test module, the test of each module will be isolated, too. For example, in the block RAM test, the block RAM works with attacks but test data is sent to PC when FPGA returns to normal situation.

Two block RAMs (16-bit data width) are implemented in block RAM test module. Test data is the value from 0000 to FFFF which will be stored and read in both block RAMs. If test module works fine, the received data of two block RAMs should be correct and the same.

In shifter register test, bit 1 and bit 0 will be shifted into tested register(16-bit) regularly. Also in this test two block RAMs are implemented to store the data from shift register, in order to distinguish the error caused by shift register from the block RAM. If shift register works fine but block RAMs not, then received data of two block RAMs should be different. If shift register does not work fine, received data of two block RAMs should have the same error.

The asynchronous counter test module contains 4 synchronous counters (16-bit) connecting with different clocks with the frequency of 50MHz, 100MHz, 200MHz.
and 300MHz, respectively. Clocks are generated by the IP core Clocking Wizard. In this test, 4 counters will count the corresponding clocks for 1us (100 cycles in 100MHz system clock). We would like to check whether the asynchronous counter works fine under attacks or not.

The communication test will be applied based on all the 3 test modules by generating test data on normal situation then sending data under attacks.

4.2 Freezing

4.2.1 Freezing spray

Freezing attack is executed by using Servisol freezing spray as shown in Figure 4.2. Servisol spray is used to reduce electro component temperature rapidly (Circuit freezer). Manufacturer claims that the temperature of component can be decreased to -50 degrees Celsius. Note that FPGA will also warm during working state, therefore the lowest temperature that can be achieved is always above -50 degrees Celsius.

4.2.2 Isolation test result

To avoid the impact on other components by temperature changing, first we checked the influence of freezing on isolation test suite. Table 4.1 gives the results of this experiment.

No error happens in test suite on block RAM, shift register and UART test under freezing attack. In counter test, the received data is 32, 64, C8 and 12C in Hex for each asynchronous counter. In Decimal, they are 50, 100, 200 and 300 respectively. This result is correct and the same as normal situation. Thus we can conclude that
freezing has no impact on other components. We can apply freezing attack on TERO TRNG directly.

### 4.2.3 Freezing attack test result

We applied temperature attack on the two designs (X10Y90 and X44Y6) described in Chapter 3. Note that the effective time for applying the freezing spray is a few seconds. Therefore only about 1,000,000 numbers of oscillation occurrences are captured in this test.

#### Oscillation occurrence test

Figure 4.3 demonstrates the oscillation occurrence test result of TERO TRNG under freezing. From this figure we can see that there is limited influence on TERO by freezing.
4. Physical attacks on designed TRNG

On location X10Y90, the number of oscillation occurrence is 155 in average in normal situation. However this number is increased into around 159 when under freezing. But the distribution of oscillation occurrence is still a normal distribution. In other words, TERO still satisfy the requirements of transition oscillation.

The situation obtained in X44Y6 is different from X10Y90. The number of oscillation occurrence is decreased from 116 in average into 111 in average. But similar to X10Y90, the distribution is still a normal distribution. And the difference of number of oscillation occurrence is quite small. Thus we cannot make a conclusion from oscillation occurrence test that TERO is influenced by freezing or not.

**Basic statistical test**

Figures 4.4 and 4.5 and Tables 4.2 and 4.3 illustrate the basic statistical test results under freezing. About 1 million bytes of random data are captured for each design during freezing. To have a good comparison between normal situation and under freezing, the random data captured of both designs on normal situation is also 1 million bytes. The block of mean value and autocorrelation calculation is set at 500 bits.
4.2. Freezing

Figure 4.4: Mean value test result on normal situation and freezing

<table>
<thead>
<tr>
<th>Total average value</th>
<th>X10Y90</th>
<th>X44Y6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Under Freezing</td>
</tr>
<tr>
<td>Mean value</td>
<td>0.4999</td>
<td>0.5013</td>
</tr>
<tr>
<td>Correlation coefficients</td>
<td>0.4999</td>
<td>0.5001</td>
</tr>
</tbody>
</table>

Table 4.2: Total average mean value and autocorrelation coefficients

The mean value test result is shown in Figure 4.4 and the calculated total mean value in average is listed in Table 4.2. There is no change on the distribution of the mean value. Actually there is little difference between normal situation and under freezing. On location X10Y90, the total mean value changes from 0.4999 to 0.5013 and in X44Y6, this number changes from 0.5001 to 0.5018.

The autocorrelation test result is shown in Figure 4.5 and the average value of correlation coefficients is also listed in Table 4.2. There is no change on the distribution of the mean value. Unlike the mean value test, the calculated total average values of correlation coefficients in each situation are almost same (0.5001 compared to 0.4999 in X10Y90, and 0.5001 compared to 0.5000 in X44Y6).
4. Physical attacks on designed TRNG

The distribution of 4-bit variable X and estimated entropy of test random bitstream is listed in Table 4.3. As we can see, the estimated Shannon entropy of the designs reaches almost 4, and the minimum entropy are over 3.97. On location X44Y6, the minimum entropy decreased from 3.9955 to 3.9787, a quite large difference. However, estimated minimum entropy even increased from 3.9933 to 3.9952 on location X10Y90 which is quite remarkable.

Based on the basic statistical test result, the difference between normal situation and under freezing are quite small. Thus we can conclude that freezing has little impact on the TERO based TRNG.

4.3 Under/over-powering

The second attack we applied on the TERO TRNG implementations is under/over-powering.
4.3. Under/over-powering

Variable X(HEX) | X10Y90 | | X44Y6 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Under Freezing</td>
</tr>
<tr>
<td>0</td>
<td>126654</td>
<td>126448</td>
</tr>
<tr>
<td>1</td>
<td>125697</td>
<td>126294</td>
</tr>
<tr>
<td>2</td>
<td>126339</td>
<td>127088</td>
</tr>
<tr>
<td>3</td>
<td>126813</td>
<td>127210</td>
</tr>
<tr>
<td>4</td>
<td>126932</td>
<td>126281</td>
</tr>
<tr>
<td>5</td>
<td>126788</td>
<td>126996</td>
</tr>
<tr>
<td>6</td>
<td>126741</td>
<td>126968</td>
</tr>
<tr>
<td>7</td>
<td>126456</td>
<td>127041</td>
</tr>
<tr>
<td>8</td>
<td>126558</td>
<td>127092</td>
</tr>
<tr>
<td>9</td>
<td>126796</td>
<td>127118</td>
</tr>
<tr>
<td>A</td>
<td>126029</td>
<td>126811</td>
</tr>
<tr>
<td>B</td>
<td>126802</td>
<td>126661</td>
</tr>
<tr>
<td>C</td>
<td>126068</td>
<td>126541</td>
</tr>
<tr>
<td>D</td>
<td>127118</td>
<td>126879</td>
</tr>
<tr>
<td>E</td>
<td>126608</td>
<td>126063</td>
</tr>
<tr>
<td>F</td>
<td>125989</td>
<td>127013</td>
</tr>
<tr>
<td>Total</td>
<td>202448</td>
<td>2028544</td>
</tr>
<tr>
<td>Shannon entropy</td>
<td>4.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>Min-entropy</td>
<td>3.9933</td>
<td>3.9952</td>
</tr>
</tbody>
</table>

Table 4.3: Distribution of variable X and entropy estimation

4.3.1 Experimental Setup

Modified FPGA board

There are always regulators to filter supply voltages on digital platforms. Thus before applying under/over-powering attack, modifications should be made on FPGA board. On Atlys Spartan-6 FPGA board, r274 is the regulator used to filter voltage for FPGA chip. We removed the regulator and replaced it by a common copper wire connected to an external power supply. Then the external power supply will give the supply voltage to the FPGA directly.

Agilent DC Power Supply

An Agilent E3610A DC Power Supply is used here for controlling the supply voltage for FPGA. The output range of this power supply can be 0-8V for voltage with 0-3A for current or 0-15V for voltage with 0-2A for current. It has two modes: constant voltage mode and constant current mode. Obviously, constant voltage is used in this test.
4. Physical attacks on designed TRNG

Figure 4.6: Bypassing the regulator of FPGA chip

<table>
<thead>
<tr>
<th>System clock</th>
<th>Block RAM</th>
<th>Shift register</th>
<th>Asynchronous counter</th>
<th>UART module</th>
</tr>
</thead>
<tbody>
<tr>
<td>100MHz</td>
<td>0.86V</td>
<td>0.86V</td>
<td>0.90V</td>
<td>0.86V</td>
</tr>
<tr>
<td>50MHz</td>
<td>0.70V</td>
<td>0.70V</td>
<td>0.73V</td>
<td>0.70V</td>
</tr>
<tr>
<td>20MHz</td>
<td>0.67V</td>
<td>0.67V</td>
<td>0.67V</td>
<td>0.67V</td>
</tr>
<tr>
<td>10MHz</td>
<td>0.67V</td>
<td>0.67V</td>
<td>0.67V</td>
<td>0.67V</td>
</tr>
</tbody>
</table>

Table 4.4: Isolation test result: minimum effective supply voltage for each module

4.3.2 Isolation test result

We have to apply isolation test before under/over-powering attacking TERO TRNG. At first, we applied the isolation test on test suite with common system clock 100MHz. This test result is not suitable because the UART module break down quickly on 0.9V. Moreover, BRAM and Shift register operations break down at 0.86V. Lower supply voltage does not influence the data stored in RAM, but influences the writing operation. Changing the supply voltage of digital platform influences the propagation time of the digital circuit. Thus we tried to decrease the system clock frequency and applied the isolation test 3 times with lower system clock frequencies: 50MHz, 20MHz and 10MHz.

The results of all isolation tests are shown in Table 4.4. We recorded the lowest voltage that each test can reach without any error. From the table we can see the minimum allowed voltage of UART module decreases from 0.9V to 0.67V when we reduce the system clock frequency from 100MHz to 20MHz. However, this value keeps the same on 10MHz clock which is lower than 20MHz. The same situations happen on other modules. Note that the recorded minimum voltage of BRAM, shift
register and asynchronous counters are the same, and this value is considered as the system break down voltage. This means it is considered that no component can work correctly under such voltage. On both 20MHz and 10MHz situation, the minimum allowed voltage of UART module is also the same with system break down voltage. Therefore we do not have to separate the generating stage and transmitting stage if we select one of them as the system clock frequency. Ultimately we select 20MHz as the frequency of the system clock.

4.3.3 Under/over-powering attack test result

Before applying the attacks, we also checked the lowest voltage that can be reached by TERO TRNG. The number is 0.68V, a little bit larger than isolation test suite. The normal supply voltage for Spartan-6 FPGA is 1.20V. For obtaining a comprehensive view of this test, we need to investigate both lower and higher supply voltage. Thus the adjusting range of power supply is determined from 0.68V to 1.59V.

Also noticed, the TERO TRNG is dependent on both placement and FPGA chip because of the process variations. In this experiment, a second FPGA board is modified and used so there is possibility that selected locations on previous FPGA board do not work as expected. Therefore we checked the oscillation occurrence of different locations on new board. Location X10Y90 is proved still suitable for TERO but X44Y6 is not. To have a comparison, a new location X44Y108 is selected in under/over-powering attack experiments.

Oscillation occurrence test

The first analysis of under/over-powering attack is obviously the oscillation occurrence test. The supply voltage used in oscillation occurrence test from 0.68V to 1.59V varies in intervals of 0.13V. Figure 4.7 and 4.8 illustrate the distribution of oscillation occurrence at locations X10Y90 and X44Y108 for each selected supply voltage. It is clear that the oscillation occurrence number of TERO in two locations is influenced by supply voltage.

As mentioned before, the distribution of oscillation occurrence is quite important for TERO. Fortunately, changing the power supply does not change the distribution of oscillation occurrence and does not introduce any glitches in the distribution. They still look like normal distribution. However, the number of oscillations occurrence is also important for TERO. Unfortunately, the bias value of oscillation occurrence changes with voltage variations. As shown in Figure 4.7 and 4.8 when the supply voltage is below 1.20V, this number is always decreasing with the decreasing of supply voltage. It is further noticed that on location X10Y90, the number of oscillation is reduced even lower that 40 by 0.68V. This oscillation occurrence number is so few that only 5 possible numbers are present in the distribution. This is quite damaging for TERO TRNG. When the supply voltage increases, from 1.20V to 1.59V, numbers of oscillation occurrence are also influenced. At 1.33V, the oscillation occurrence is almost the same as 1.20V. However, when supply voltage changes into 1.46V
4. Physical attacks on designed TRNG

Figure 4.7: Oscillation occurrence test result of location X10Y90
4.3. Under/over-powering

Figure 4.8: Oscillation occurrence test result of X44Y108
and 1.59V, the number of oscillation occurrence decreases significantly compared to 1.20V.

Figure 4.9 shows the total average number of oscillation occurrence of our designs. On both location X10Y90 and X44Y108, the highest oscillation number occurs at 1.33V. But the differences between 1.20V and 1.33V are very small, so we can not confirm that number is influenced. Note that the wire used to connect power supply and FPGA also has impedance, which may cause that the voltage delivered to the chip is lower than the value displayed on the power supply. When supply voltage is reduced from 1.20V to 0.68V, the number of oscillation occurrence is suddenly decreased from 132 to 27 in X10Y90, and from 223 to 59 in X44Y108. Such decreasing process is almost linear and the final value is only about 20% (X10Y90) and 26% (X44Y108) in comparison to normal situation.

In conclusion, under/over-powering has impact on the oscillation occurrence of transition stage. But the random bit is extracted from the parity of oscillation occurrence number. Whether the randomness of extracted random bits is also influenced or not has to be determined with statistical tests.
Basic statistical tests

The basic statistical tests are performed after the oscillation occurrence test. Note that in the oscillation occurrence test, the number of oscillation decreased significantly in lower and higher supply voltage. Therefore, we have to focus more on the boundary. Several new test points near 0.68V and 1.59V are added for statistical tests.

Figures 4.10 and 4.11 give result of mean test we observed near 0.68V and 1.59V. From the figure we can see that the distribution of mean value in blocks of 500 bits does not change in higher supply voltages on both X10Y90 and X44Y108. But in very low supply voltage, the situation is different. In x44Y108, decreasing supply voltage does not influence the distribution of mean value. It is still normal distributed with a bias around 0.5. However in X10Y90, the distribution becomes worse with the decreasing of supply voltage. The mean value in 0.70V is actually far from a normal distribution. The distribution at 0.68V is even worse, although the bias is still around 0.5.

Results of autocorrelation test are shown in Figure 4.12 and 4.13. In X44Y108, all the distributions of average autocorrelation coefficient are normal distribution with a bias around 0.5. Such distributions in X10Y90 are also normal distribution. But the bias value changes when the supply voltage decreases lower than 0.8V. In 0.68V, such value changes from around 0.5 to around 0.4.

We also calculated total average value of mean test and autocorrelation test. Figure 4.14 illustrates the curve of total average value. As seen in the figure, average autocorrelation coefficient in X44Y108 keeps almost 0.5 during all tested supply voltages. The average mean value fluctuates between 0.8V and 1.1V, but it is never lower than 0.49. The same fluctuation of mean value in X10Y90 is observed but the average autocorrelation coefficient drops quickly when supply voltage decreases below 0.8V. Based on the result from mean value and autocorrelation test, TERO in X10Y90 is almost unusable under 0.8V, but TERO in X44Y108 is almost not influenced by under-powering.

The last test is entropy estimation. Shannon entropy and minimum entropy are calculated and plotted in Figure 4.15. In X44Y108, Shannon entropy is always around 4, but the minimum entropy decreases with the supply voltage. The drop of minimum entropy is only about 0.05, thus we believe the influence of low voltage here is very little. Different from X44Y108, TERO in X10Y90 is influenced a lot by supply voltage. The minimum entropy as well as the Shannon entropy are always decreasing with the supply voltage. The minimum entropy in X10Y90 in 0.68V is only 3 effective bits and the corresponding Shannon entropy is only around 3.9 bits which is even lower than the minimum entropy in X44Y108 in 0.68V. Also notice that entropy becomes better with supply voltage increasing higher than 1.3V. That means randomness is even increasing with the supply voltage.

In summary, under-powering does have impact on randomness of TERO. Considering the impact on oscillation occurrence observed above, we think the influence of randomness is caused by the reducing of number of oscillation occurrence. In X44Y108, since the fewest number of oscillation occurrence is still above 50, the randomness is only influenced a little. However in X10Y90, such number is only
4. Physical attacks on designed TRNG

Figure 4.10: Mean value test result of X10Y90
4.3. Under/over-powering

Figure 4.11: Mean value test result of X44Y108
4. Physical attacks on designed TRNG

Figure 4.12: Autocorrelation test result of X10Y90

54
4.3. Under/over-powering

Figure 4.13: Autocorrelation test result of X44Y108
4. Physical attacks on designed TRNG

Figure 4.14: Average value of random bit and autocorrelation coefficient

around 27. Obviously, randomness is therefore influenced. Moreover, TERO in X10Y90 in 0.81V has the number of oscillation occurrence distributed around 50. Corresponding statistical test result is better as well. So, it is proved that the number of oscillation occurrence is a very important parameter that has to be considered with TERO TRNG design.

4.4 Conclusion

So far,

In this chapter, freezing and under/over-powering attacks are performed. We firstly proposed a isolation test suite implemented on FPGA before attacking TERO based TRNG. By attacking test suite, we then found the influence of other components on FPGA and thus we can distinguish the influence of TERO from other components.

In freezing attack, no influence of other components is observed on the test suite. And when attacking TERO, only little influence of the oscillation occurrence is observed. Such difference even can be seen as fluctuation. No impact on randomness is found. We conclude freezing attack as we performed has no impact on TERO based TRNG.

A modified FPGA board is used in under/over-powering attack experiment. We
firstly checked the randomness of TERO based TRNG on selected location (X10Y90 and X44Y6) of new board. X44Y6 is not suitable for TERO on new board, thus another location X44Y108 is selected in under/over-powering attack. After new location selected, the minimum effective supply voltage of each module are obtained then by attacking the test suite. To have a larger range for adjusting supply voltage, the system clock frequency is decreased. In the test, we finally reached the lowest effective voltage of 0.67V by using a clock frequency of 20MHz. Then we obtained the minimum effective supply voltage of 0.68V for TERO. In addition, higher supply voltage may also influence the TERO. Therefore we applied under/over-powering attack of the range between 0.68V and 1.59V. Test result shows the TERO is successfully influenced by under/over-powering. With the supply voltage increasing or decreasing, the number of oscillation is decreased at the same time. Also if the number of oscillation occurrence in normal situation is not large enough, e.g. in X10Y90 in this test, the number of oscillation occurrence may be decreased to a dangerous level in very low supply voltage. With such low number of oscillation occurrence, the TERO will not be strong enough as a random source for a TRNG any more. This also proves that number of oscillation occurrence is an important parameter for TERO implementation.

Figure 4.15: Estimated Shannon entropy and minimum entropy
Chapter 5

On-the-fly tests

In Chapter 4, we perform freezing and under/over-powering attacks on designed TERO based TRNG and observe the result that TERO is successfully influenced by under/over-powering. So far, we only test the randomness after receiving bitstreams from the TRNG to confirm the quality used for a cryptography application. Obviously such offline test cannot prevent the attacks. Therefore, in this chapter, we propose a on-the-fly test that provides online randomness checking during operation of TRNG and alarms the corresponding cryptography application to handle the threat.

5.1 Test module design

The main purpose for the on-the-fly test is to detect threats to the TRNG instead of total statistical property analysis [21]. Complex statistical tests require a lot of resources and power consumption. Thus we should only implement several simple statistical tests on the test module. Our approach is to select basic statistical tests as the on-the-fly tests. We have proved that the number of oscillation occurrence in transition stage plays a very important role in TERO based TRNG in Chapter 4. Therefore, an additional oscillation check is implemented as well.

The general block diagram of the test module is illustrated in Figure 5.1. The test module captures the random number and sends it into a 4-bit shift register. Three tests will be performed based on the shift register:

- Mean value test and Autocorrelation test In mean value test the last bit of register will be sent to the accumulator and random bit 1 will be accumulated in a sequence of random bits. Similar to mean test, the autocorrelation test will accumulate the 1s of the autocorrelation coefficients in a sequence of coefficients which is generated by extracting last two adjacent bits of the shift register. These two tests have the same size of test sequence. Accumulated result will be compared with an upper bound and a lower bound. Once the result is out of range, the test module will output the alarm of corresponding test.

- Entropy test The entropy test here is the estimation of minimum entropy. We do not calculate the exact number of entropy in this test because it is
5. On-the-fly tests

Figure 5.1: On-the-fly test module

unnecessary and a waste of resources. We will detect the 4-bit segment for every four random bits and accumulate the occurrence of each 4-bit segment (in hex, 0,1 ...F) in the test sequence. The min-entropy limits the most occurrence of each segment which represents the worst case. Thus an upper bound of the number of occurrence will be set for this test and the alarm is detected when the number is over the bound. Also note this test shares the test sequence size with mean value and autocorrelation test.

- Oscillation occurrence test Oscillation occurrence test is very simple. We directly retrieve the number of oscillation occurrence from digitalization module and compare it with a lower bound to detect the threat. Note that this test is implemented once the oscillation stage finishes, unlike other tests require a sequence of bits.

5.2 Parameter selection

In this test module, the size of test sequence for mean value, autocorrelation and entropy tests and the bounds for each test are the parameters to be determined. For mean value and autocorrelation tests, it is clear that different size of test sequence need different critical bounds. Note the distribution of mean value of random bit and average autocorrelation coefficient are both normal distribution. Thus we consider the range \((\mu - 3\sigma, \mu + 3\sigma)\), which actually covers 99.7% in normal distribution, as allowed in our test. The \(\mu\) value for the tests is always 50% of total size but \(\sigma\) should be computed based on the size of test sequence. We test sequence sizes from 512 bits to 65536 bits, use Matlab function \textit{Normfit} to estimate the \(\sigma\) and get the corresponding allowed range. The results are listed in Table 5.1.
5.3. Test result

In the entropy test, we consider 3.88 effective bits as the critical minimum entropy for 4-bit segments (0.97 per bit). Then the allowed proportion of number of occurrence for each 4-bit segment is calculated as:

\[ 10^{-3.88 \times \log_{10} 2} \approx 0.07 \]  

Thus we also compute the critical bound \((0.07 \times \text{sequence size} / 4)\) corresponding to each sequence size by this proportion in Table 5.1.

From the result observed from chapter 4, we know that if the number of oscillation occurrence is always lower than 50, then the randomness of TERO based TRNG can not be guaranteed. Considering adding 50% margin to the limitation, thus we select 75 as the critical bound in this test.

### 5.3 Test result

In chapter 4, we detect impact of under/over-powering attack on TERO based TRNG. Especially, TERO in X10Y90 is significantly influenced under 0.80V. Thus we apply the on-the-fly test on TERO in X10Y90 at 0.70V and 0.75V to check the alarm rate of each test. In comparison, the normal situation is also tested and all the results are demonstrated by alarm rate in Table 5.2.

As demonstrated in Table 5.2, some alarm rates of mean value test of TERO based TRNG at 1.2V are over 0.3%. It is because that the output of TRNG always contains a bias. The alarm rate of autocorrelation test is as expected. However, estimation of entropy needs large sequence of variables. Thus in this test, lower test sequence size result in false alarms. There are two ways to solve this problem, increasing the critical bound or using large test sequence. Although large test sequence will decrease the test speed, we still suggest the large test sequence size for entropy test due to its high accuracy.

At both 0.75V and 0.7V, the alarm rate of mean value test and autocorrelation test rises with the increase of test sequence size. Therefore, we consider larger test sequence provides more accurate alarm for threats. But the test speed will be decreased by the large sequence.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test sequence size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>512 1024 2048 4096 8192 16384 32768 65536</td>
</tr>
<tr>
<td>Mean</td>
<td>(\mu - 3\sigma)</td>
</tr>
<tr>
<td></td>
<td>(\mu + 3\sigma)</td>
</tr>
<tr>
<td>Correlation</td>
<td>(\mu - 3\sigma)</td>
</tr>
<tr>
<td></td>
<td>(\mu + 3\sigma)</td>
</tr>
<tr>
<td>Entropy</td>
<td>-</td>
</tr>
<tr>
<td>Oscillation</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.1: Parameters selected for on-the-fly test

In the entropy test, we consider 3.88 effective bits as the critical minimum entropy for 4-bit segments (0.97 per bit). Then the allowed proportion of number of occurrence for each 4-bit segment is calculated as:

\[ 10^{-3.88 \times \log_{10} 2} \approx 0.07 \]  

Thus we also compute the critical bound \((0.07 \times \text{sequence size} / 4)\) corresponding to each sequence size by this proportion in Table 5.1.

From the result observed from chapter 4, we know that if the number of oscillation occurrence is always lower than 50, then the randomness of TERO based TRNG can not be guaranteed. Considering adding 50% margin to the limitation, thus we select 75 as the critical bound in this test.

### 5.3 Test result

In chapter 4, we detect impact of under/over-powering attack on TERO based TRNG. Especially, TERO in X10Y90 is significantly influenced under 0.80V. Thus we apply the on-the-fly test on TERO in X10Y90 at 0.70V and 0.75V to check the alarm rate of each test. In comparison, the normal situation is also tested and all the results are demonstrated by alarm rate in Table 5.2.

As demonstrated in Table 5.2, some alarm rates of mean value test of TERO based TRNG at 1.2V are over 0.3%. It is because that the output of TRNG always contains a bias. The alarm rate of autocorrelation test is as expected. However, estimation of entropy needs large sequence of variables. Thus in this test, lower test sequence size result in false alarms. There are two ways to solve this problem, increasing the critical bound or using large test sequence. Although large test sequence will decrease the test speed, we still suggest the large test sequence size for entropy test due to its high accuracy.

At both 0.75V and 0.7V, the alarm rate of mean value test and autocorrelation test rises with the increase of test sequence size. Therefore, we consider larger test sequence provides more accurate alarm for threats. But the test speed will be decreased by the large sequence.
5. On-the-fly tests

<table>
<thead>
<tr>
<th>$V_{dd}$</th>
<th>Test</th>
<th>Test sequence size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>512 1024 2048 4096 8192 16384 32768 65536</td>
</tr>
<tr>
<td>1.20V</td>
<td>Mean</td>
<td>0.26% 0.19% 0.45% 0.80% 1.49% 1.32% 1.20% 1.35%</td>
</tr>
<tr>
<td></td>
<td>Correlation</td>
<td>0.26% 0.21% 0.29% 0.39% 0.31% 0.21% 0.19% 0.34%</td>
</tr>
<tr>
<td></td>
<td>Entropy</td>
<td>100% 100% 98.8% 95.8% 75.6% 29.4% 3.40% 0.40%</td>
</tr>
<tr>
<td></td>
<td>Oscillation</td>
<td>0.00% 0.00% 0.00% 0.00% 0.00% 0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>0.75V</td>
<td>Mean</td>
<td>6.7% 11.7% 5.01% 5.91% 15.6% 41.2% 38.1% 39.7%</td>
</tr>
<tr>
<td></td>
<td>Correlation</td>
<td>11.4% 35.4% 22.9% 56.6% 94.7% 100% 100% 100%</td>
</tr>
<tr>
<td></td>
<td>Entropy</td>
<td>100% 100% 100% 100% 100% 100% 100% 100%</td>
</tr>
<tr>
<td></td>
<td>Oscillation</td>
<td>100% 100% 100% 100% 100% 100% 100% 100%</td>
</tr>
<tr>
<td>0.70V</td>
<td>Mean</td>
<td>32.4% 65.4% 54.9% 75.6% 74.9% 81.3% 82.6% 92.7%</td>
</tr>
<tr>
<td></td>
<td>Correlation</td>
<td>53.6% 99.7% 99.5% 100% 100% 100% 100% 100%</td>
</tr>
<tr>
<td></td>
<td>Entropy</td>
<td>100% 100% 100% 100% 100% 100% 100% 100%</td>
</tr>
<tr>
<td></td>
<td>Oscillation</td>
<td>100% 100% 100% 100% 100% 100% 100% 100%</td>
</tr>
</tbody>
</table>

Table 5.2: On-the-fly test result (represented in alarm rate) for TERO based TRNG in X10Y90 at 0.70V, 0.75V and 1.20V supply voltage

5.4 Conclusion

In this chapter, we proposed a on-the-fly test module consisted of mean value test, autocorrelation test, entropy test and oscillation occurrence test. We estimate the $\sigma$ value of each test sequence size and set the $(\mu - 3\sigma, \mu + 3\sigma)$ as allowed range of mean value and autocorrelation test. The entropy test bound is calculated by using minimum allowed min-entropy 3.88/4-bit(0.97 per bit). And the critical bound of oscillation test is set as 75.

We use the on-the-fly test module to test designed TERO based TRNG with different test sequence size at supply voltage of 1.20V, 0.75V and 0.70V. From the result we find that small size of test sequence will cause very high entropy false alarm rate. And larger test sequence also contributes to the mean value test and autocorrelation test. Thus we suggest to use larger test sequence size instead of small size in order to have higher accurate of threat detecting.
Chapter 6
Conclusions and future work

In this dissertation, we design, implement and evaluate a kind of physical true random number generator (TRNG) on FPGA board against physicals as well as the on-the-fly tests.

Firstly we use Xilinx ISE design suite to design and implement the TRNG on Atlys spartan-6 FPGA board. Transition effect ring oscillator (TERO) is selected as the random source of this design. The logic element used in TERO is represented by Look-Up-Table-6 (LUT-6) directly. Also we use the Xilinx FPGA Editor to manually place and route the TERO and eventually a very symmetry design. The control signal of TERO is generated by dynamic oscillation stop check instead of a constant control signal in order to have the higher throughput. By dynamic oscillation stop check, we also successfully avoid the error that transition oscillation stage is stopped by constant control signal other than itself. After implementing the TERO based TRNG, we check the oscillation occurrence of TERO with several different locations. And base on the oscillation occurrence we select X10Y90 and X44Y6 as effective locations for TERO based TRNG in our board. Then several statistical tests are applied on designed TRNG. The mean value of random bit and autocorrelation coefficient are both in normal distribution with \( \mu = 0.5 \) and the estimated Shannon and minimum entropy of output random bit sequence are both quite high (3.99 effective bits for 4-bit segment). Moreover, the designed TRNG even passes the FIPS 140-2 and NIST SP800-22 tests which proves the randomness of designed TRNG. In addition, the throughput of random bit is about 1.29Mbps (X10Y90) and 1.25Mbps (X44Y6) in average.

After designing and testing the TRNG, we evaluated reliability of designed TRNG against physical attacks. Existing research on physical attacks mostly focuses on RO based TRNGs such as [3], [13] and [14]. They all reduce jitter of RO successfully by locking the frequency of RO based TRNG. Unlike RO based TRNG, TERO uses metastability as randomness and so far, no paper claims the reliability of TERO against physical attacks. Thus we then performed physical attacks on designed TRNG to evaluate its reliability.

We performed freezing and under/over-powering attack on designed TRNG. Freezing attack is performed by freezing spray and another Atlys board is modified.
by replacing regulator of FPGA chip to a wire for applying under/over-powering attack. Before attacking designed TRNG, we firstly implement an isolation test suite contains block RAM, shift register, asynchronous counters and UART modules for the attacks to check the impact on other components under such attacks. Freezing has almost no impact on other components but under/over-powering has. To obtain lower supply voltage, the system clock frequency is changed from normal 100MHz to 20MHz and consequently the lowest supply voltage of 0.68V is achieved. Then we apply attacks on TERO based TRNG. Also note that another FPGA is used in under/over-powering attack thus we check the randomness before experiment. Location X10Y90 is still suitable for TERO but X44Y6 is replaced by another location X44Y108. Experimental results show that freezing has limited influence on TERO however under/over-powering has quite great impact. The number of oscillation occurrence is reduced by decreasing supply voltage under 1.20V or increasing supply voltage over 1.33V. And the lower oscillation occurrence does influence the property of randomness of the TRNG. In X10Y90, the distribution of either average random bit or autocorrelation coefficient are not normal distribution under 0.8V any more. Both Shannon entropy and minimum entropy estimated decrease quickly under 0.8V. Situation in X44Y108 is better because the number of oscillation occurrence in X44Y108 is much higher than X10Y90. This number is still over 50 when the supply voltage is even 0.68V.

Finally, we proposed an on-the-fly test module to detect the threat of TRNG during it operation. The on-the-fly test module contains 3 statistical tests and an additional test for oscillation occurrence. The statistical tests are mean value test, autocorrelation test and entropy test. The critical bound of mean value test and autocorrelation test is set by $(\mu - 3\sigma, \mu + 3\sigma)$ of the distribution. This range should be different with different test sequence size thus we used the Matlab function `Normfit` to process captured data and estimate the corresponding critical bounds. The critical bound of entropy test, the maximum allowed occurrence of each possible variable, is computed by using 3.88-bit for 4-bit segment (0.97 per bit) as the minimum entropy. From previous attack test result, we consider 50 as the critical number of oscillation occurrence. Therefore the critical bound of oscillation test is set as 75, which is critical number 50 with 50% margin. We test the on-the-fly test module with selected parameters of different test sequence size and eventually find that larger sequence size will provide higher accuracy for detecting threats. Also the false alarm rate is reduced at the same time. So, we suggest using larger test sequence size in the on-the-fly test module.

So far, we have used spartan-6 FPGA board as the test platform. As mentioned before, TERO will have different performance on different FPGA families. Thus we can focus on influence of different FPGA family for further study. Also only freezing and under/over-powering attacks are implemented on TERO. We can do more attacks on TERO such as clock glitch injection and EM injection and obtain more information about the influence of different attacks. Moreover, the on-the-fly test module only generates alarms. TRNG still need a mechanism for processing such alarms.
Bibliography


Abstract:
In the first part of this work, we designed and implemented a True Random Number Generator (TRNG) using Transition Effect Ring Oscillator (TERO) as random source on Atlys Spartan-6 FPGA board. We proposed a new dynamic oscillation stop checking and then common constant Ctrl signal is replaced by adaptive Ctrl signal. We implemented TERO on different locations of FPGA board. Base on the oscillation occurrence situation, two locations are selected for TERO and two TRNGs both pass the evaluations of randomness including FIPS 140-2 and NIST SP800-22 tests. We applied Freezing and underpowering attack experiment in the second part. An isolation test suite is proposed and tested with these attacks before applying attacks to designed TRNG to distinguish influence from other components on board. TERO is slightly influenced by freezing but strongly influenced by underpowering. The number of oscillation occurrence on transition stage as well as the randomness is significantly reduced by decreasing the supply voltage. In the final part, an on-the-fly test module for detecting threats of attacks is proposed on designed TRNG during operation. On-the-fly test module is consisted of 3 basic statistical tests including mean value test, autocorrelation test and entropy test as well as oscillation test. We set critical bounds of \((\mu - 3\sigma, \mu + 3\sigma)\) of normal distribution for mean value and autocorrelation coefficient, the minimum entropy of 0.97 per bit for entropy test and 75 for oscillation test. On-the-fly test module is tested with different length of test sequence and finally we suggest longer test sequence due to experimental results.