INDIAN INSTITUTE OF INFORMATION TECHNOLOGY NAGPUR

Project Report
On
Random Bit Generator

Submitted By:

Mundru Abhiram BT20ECE011

Submitted To:

Dr. Paritosh Peshwe

Department of Electronics and Communication Engineering

April, 2023

ABSTRACT

A hardware-based random bit generator based on ring oscillators is a device that generates random bits using the phase jitter of ring oscillators. This abstract explores the working principle of such generators, the construction of the ring oscillator circuit, and the methods used to extract random bits from the phase jitter. The statistical properties of the generated random bits, such as entropy and bias, are discussed, and various tests for randomness are introduced. Techniques for evaluating the quality of the generated random bits are presented. The advantages and disadvantages of using ring oscillator-based random bit generators are also discussed, including their simplicity, low cost, and vulnerability to side-channel attacks. Overall, the importance of hardware-based random bit generators based on ring oscillators in modern computing and communication systems is highlighted, as they provide a higher level of security than software-based approaches and are essential for ensuring the confidentiality, integrity, and availability of information.

TABLE OF CONTENTS

List of Figures

1. INTRODUCTION

Ring Oscillator based TRNG Jitter in Ring Oscillators (RO) provides a simple and effective way to extract randomness. RO based TRNG are popular as they can be implemented efficiently in both ASIC/custom as well as FPGA based designs [4][5][6][7]. A ring oscillator is built using an odd number of inverters and feeding back the output of final inverter to the input of first inverter. Variation in power supply in the form of IR drop or supply noise causes a variation in the inverter delays and hence the frequency of oscillation. This uncertainty of the oscillator signal (output) in time domain is termed as Jitter. Since the jitter depends on various factors, some of which are random, it can be used as a source of randomness in the design of an RO based TRNG. The basic TRNG circuits use two or more ring oscillators as shown in figure 2 and XOR the outputs. The output signal of the XOR is sampled during the transition zone to get random values because of the jitter in the two oscillator rings.

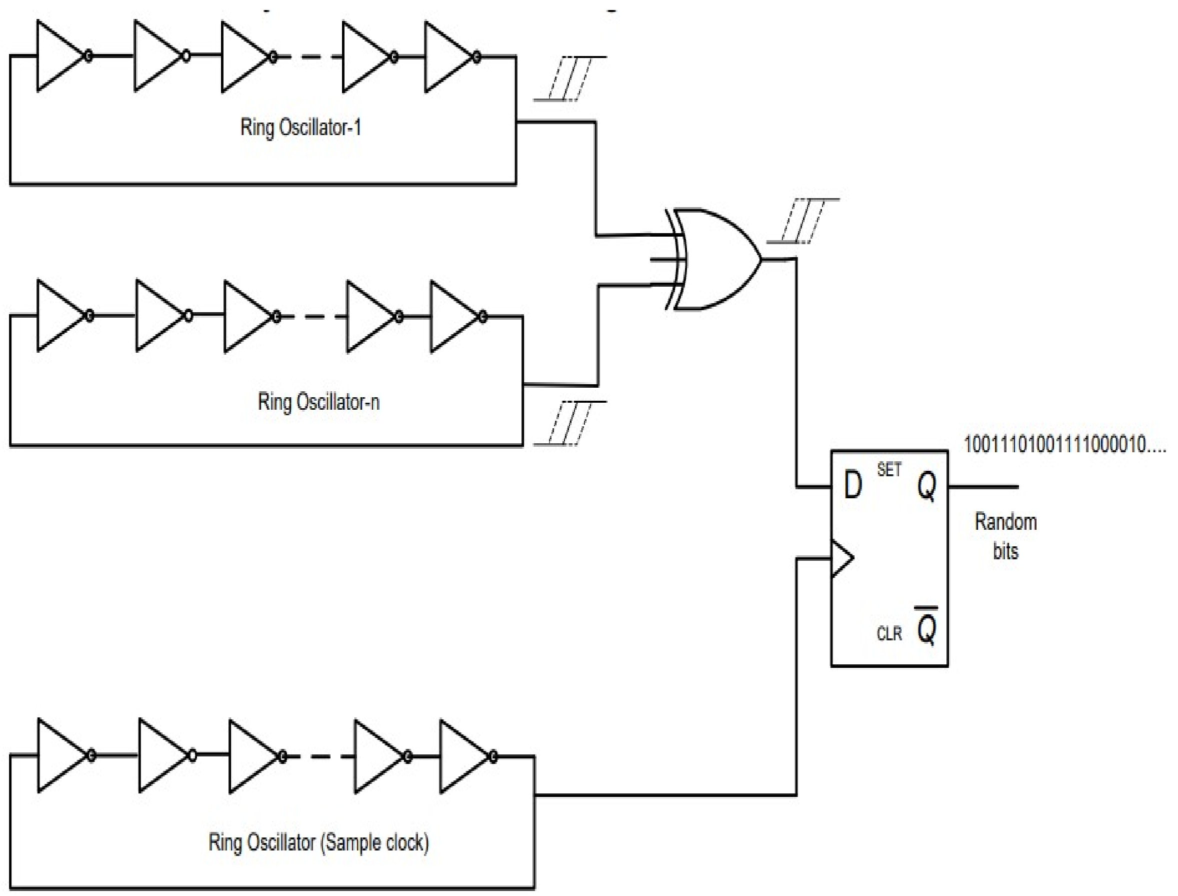


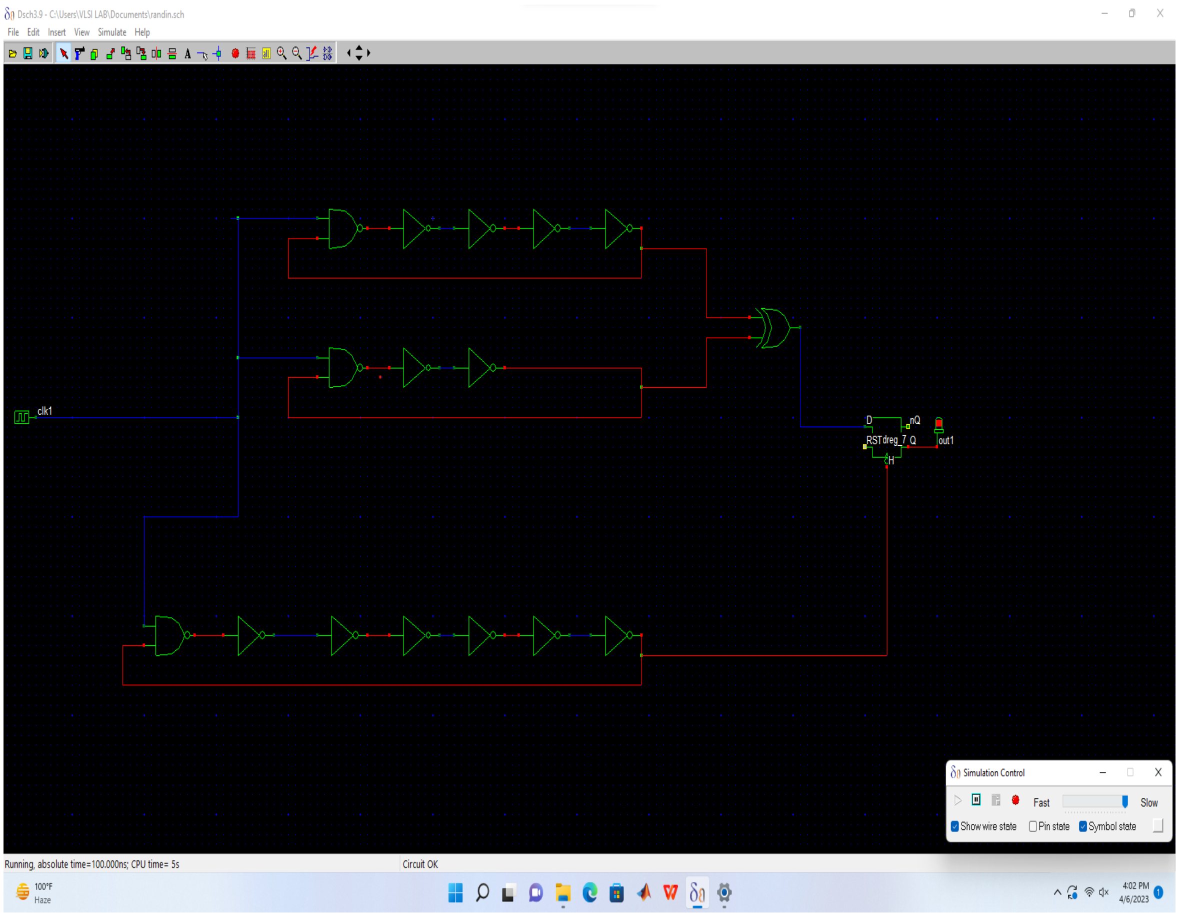
Figure 1.1: Random Bit Generator Circuit

1.1. Implementation

In practical use case, we will be requiring a random bit at every instance of clock. The circuit we've seen earlier generates random output without any control parameter. So we add clock dependency to output by using a NAND gate at the beginning of each ring oscillator for which one input will be clock and the other input will be output of last inverter.
The output of each ring oscillator is fed back into the oscillator and output is generated when the clock is triggered. Clock streamlines the random bit generation with other devices of which this circuit is a part of. Having clock control also decreases power consumed.

1.2. Implementation in DSHC

The above stated circuit can be implemented in DSHC as follows:


Dsch3.9 - C:\Users\VLSI LAB\Documents\randin.sch
File Edit Insert View Simulate Help
clk1
D
nQ
RST dreg_7 Q
H
out1
Simulation Control
Fast
Slow
100°F
Haze
4:02 PM
4/6/2023

[✓ Show wire state] [ ] Pin state [✓ Symbol state]

Running, absolute time=100.000ns; CPU time= 5s

Circuit OK

Figure 1.2: DSHC Circuit

1.3. DSHC Output

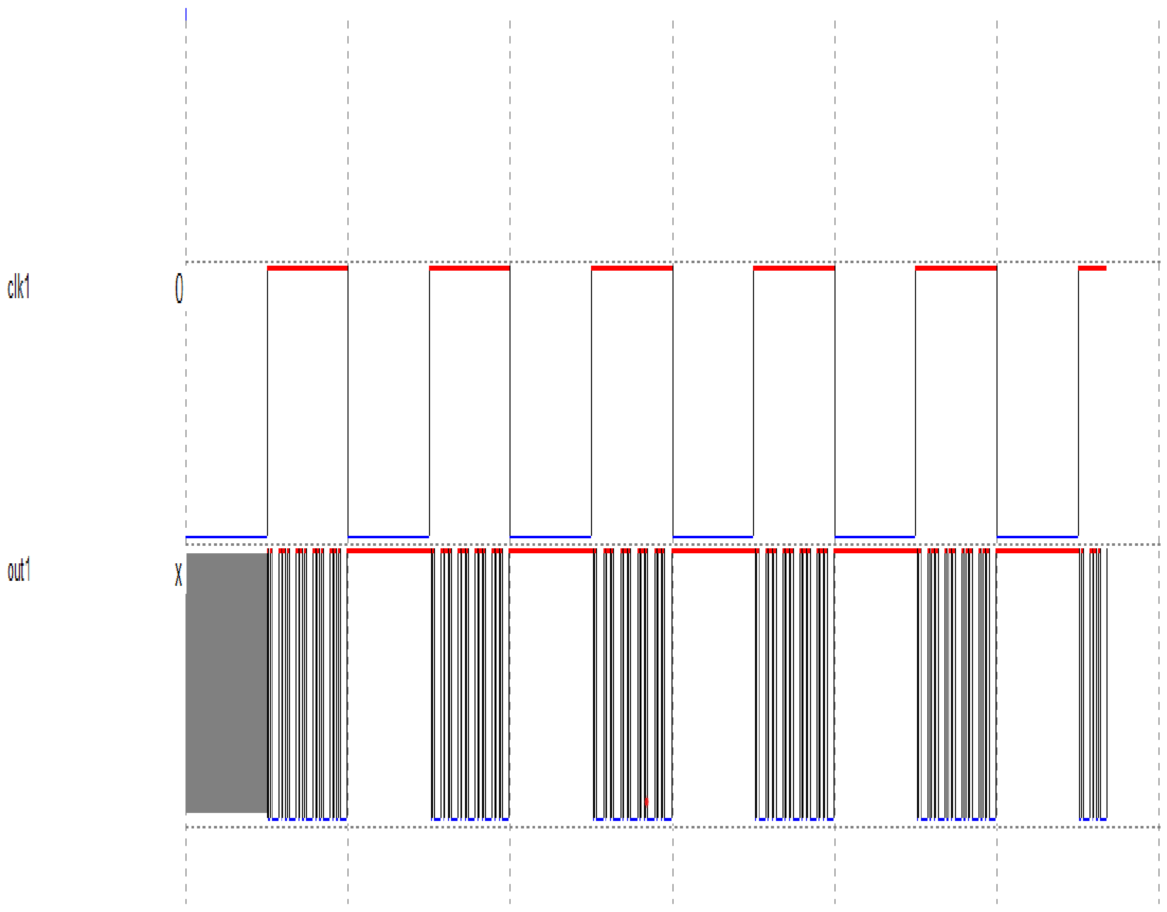


Figure 1.3: DSHC Output

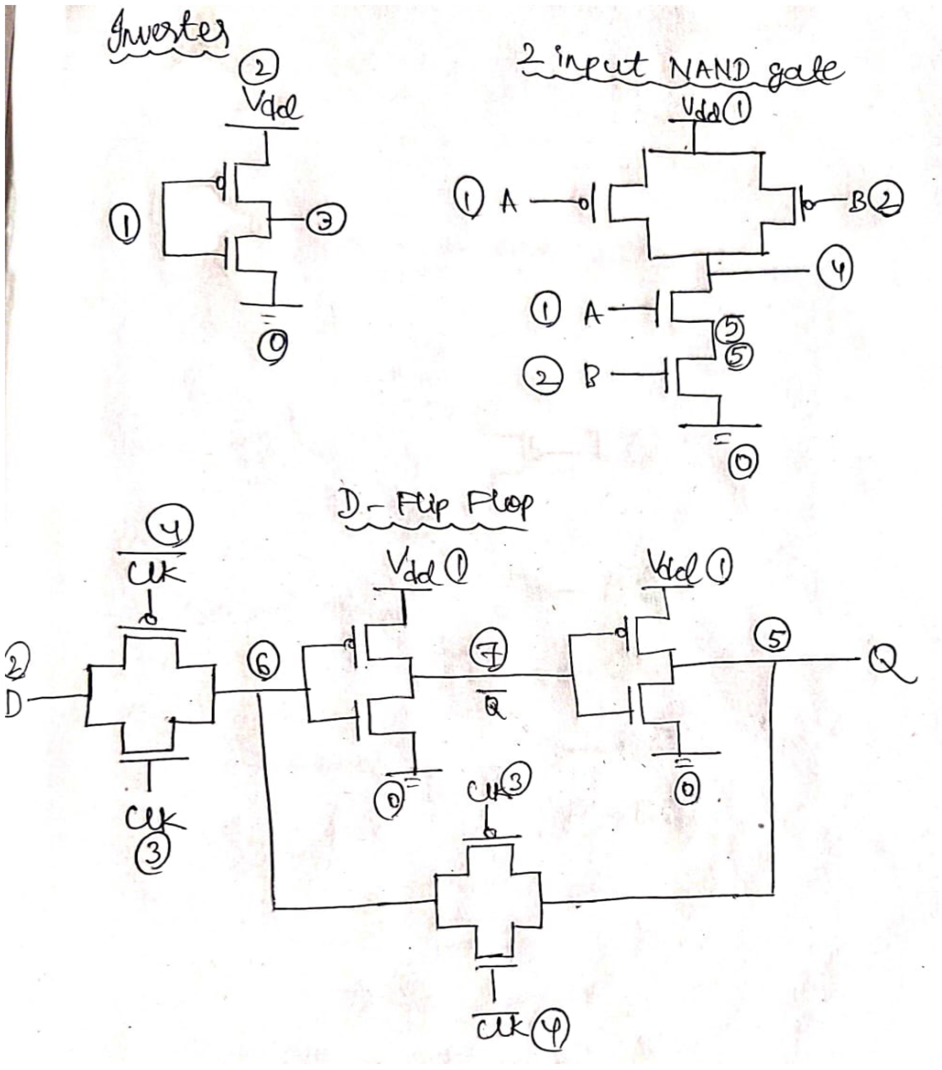
clk1 is the clock input
out1 is the sequence of random bits generated
Random bits are generated when the clock is high and thus we have achieved synchronization and control over the circuit.

2. NGSPICE IMPLEMENTATION

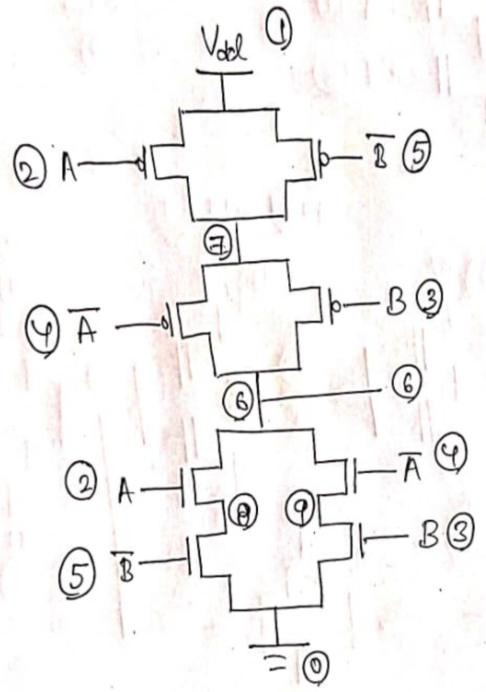
The components required to implement the circuit are circuit are:

- Inverter x 13
- 2 input XOR gate x 1
- 2 input NAND gate x 3
- D Flip-Flop x 1

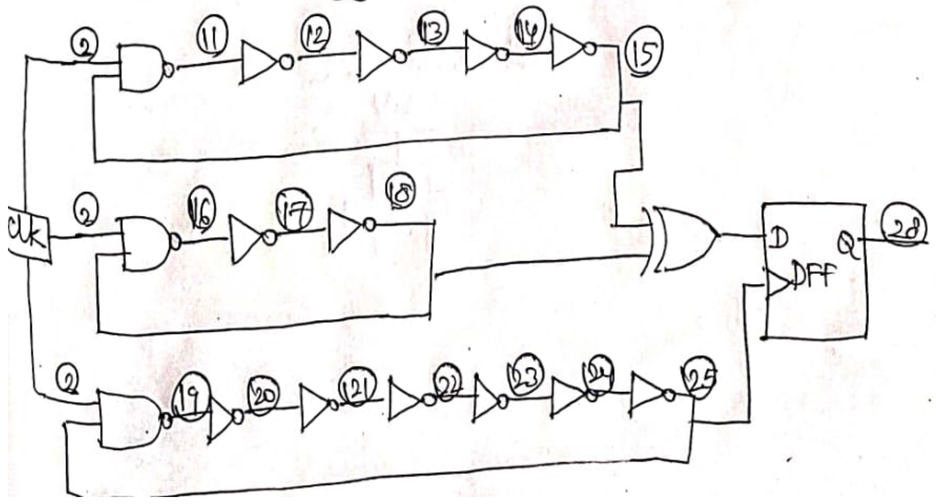
As there are multiple instances of same commponent, we create sub-circuit for each component and connect them together. Netlist for the circuit is as follows:



2 input XOR gate



TRNG circuit



2.1. NGSpice Code

```
***TRNG
.subckt inv 1 2 3
M1 3 1 2 2 pmod w=100u l=10u
M2 3 1 0 0 nmod w=100u l=10u
.model nmod nmos version=54 level=4.7
.model pmod pmos version=54 level=4.7
.ends

.subckt nandd 1 2 3 4
M1 4 1 3 3 pmod w=100u l=10u
M2 4 2 3 3 pmod w=100u l=10u
M3 4 1 5 5 nmod w=100u l=10u
M4 5 2 0 0 nmod w=100u l=10u
.model nmod nmos version=54 level=4.7
.model pmod pmos version=54 level=4.7
.ends

.subckt dff 1 2 3 4 5
M1 2 3 6 6 nmod w=100u l=10u
M7 2 4 6 6 pmod w=100u l=10u
M2 7 6 1 1 pmod w=100u l=10u
M3 7 6 0 0 nmod w=100u l=10u
M4 5 7 1 1 pmod w=100u l=10u
M5 5 7 0 0 nmod w=100u l=10u
M6 6 4 5 5 nmod w=100u l=10u
M8 6 3 5 5 pmod w=100u l=10u
.model nmod nmos version=54 level=4.7
.model pmod pmos version=54 level=4.7
.ends

.subckt xorr 1 2 3 4 5 6
M1 7 2 1 1 pmod w=100u l=10u
M2 7 5 1 1 pmod w=100u l=10u
```

```
M3 6 4 7 7 pmod w=100u l=10u
M4 6 3 7 7 pmod w=100u l=10u
M5 6 2 8 8 nmod w=100u l=10u
M6 8 5 0 0 nmod w=100u l=10u
M7 9 3 0 0 nmod w=100u l=10u
M8 6 4 9 9 nmod w=100u l=10u
.model nmod nmos version=54 level=4.7
.model pmod pmos version=54 level=4.7
.ends

Vdd 1 0 5v
Vclk 2 0 pulse (0 5 0 0 0 10m 20m)

Xnand1 15 2 1 11 nandd
Xa 11 1 12 inv
Xb 12 1 13 inv
Xc 13 1 14 inv
Xd 14 1 15 inv

Xnand2 18 2 1 16 nandd
Xe 16 1 17 inv
Xf 17 1 18 inv

Xnand3 25 2 1 19 nandd
Xg 19 1 20 inv
Xh 20 1 21 inv
Xi 21 1 22 inv
Xj 22 1 23 inv
Xk 23 1 24 inv
Xl 24 1 25 inv

Xm 15 1 26 inv
Xn 18 1 27 inv
Xxorr 1 15 18 26 27 28 xorr
```

```
Xclkbar 25 1 28 inv
Xout 1 28 25 28 29 dff

.tran 0.1m 30m
.control
run
plot V(2)
plot V(28)
.endc
.end
```

2.2. NGSpice Output

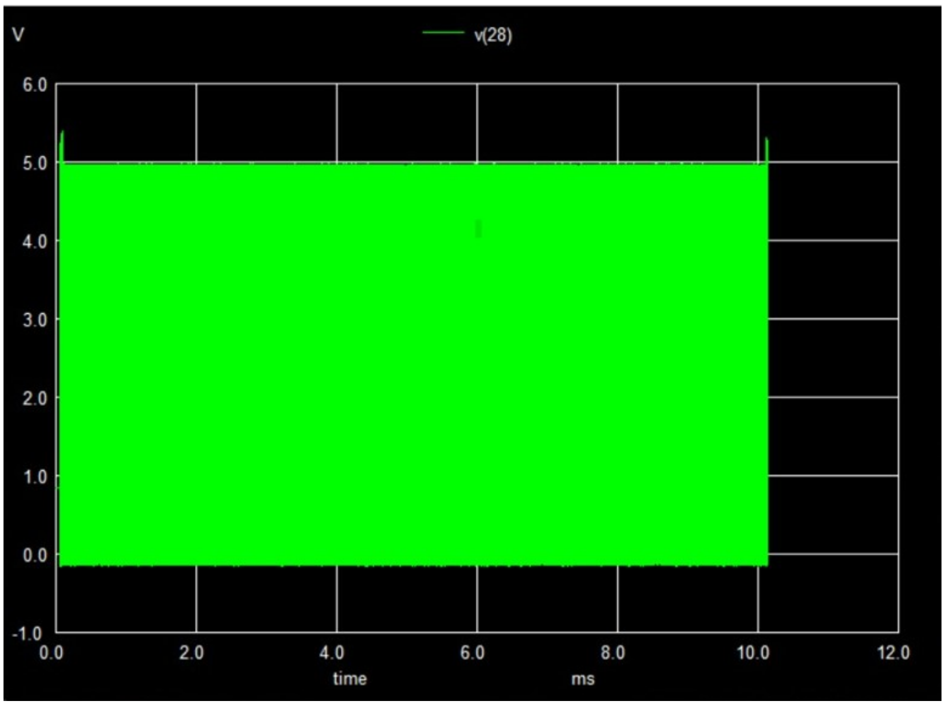


Figure 2.1: NGSpice Output

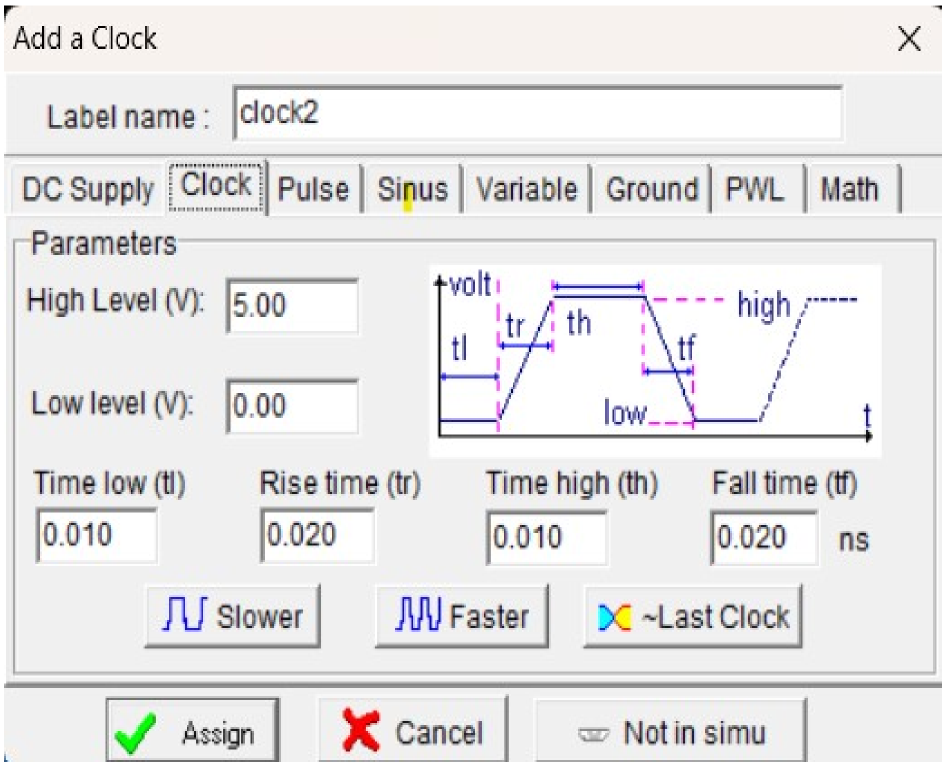
3. MICROWIND IMPLEMENTATION

3.1. Microwind Layout

The circuit is implemented in Microwind using CMOS Technology for Inverter, XOR gate and NAND gate. Transmission gates are used to implement D Flip-Flop.

3.2. Microwind Output

The following output is generated using 0.18um foundry, clock with the given values, step size of 9.90ps and timescale of 500ns. The two outputs attached are simulated using the same parameters and different output is observed.



Label name : clock2

High Level (V):	5.00
Low level (V):	0.00

Time low (tl)	Rise time (tr)	Time high (th)	Fall time (tf)	
0.010	0.020	0.010	0.020	ns

Figure 3.1: Clock

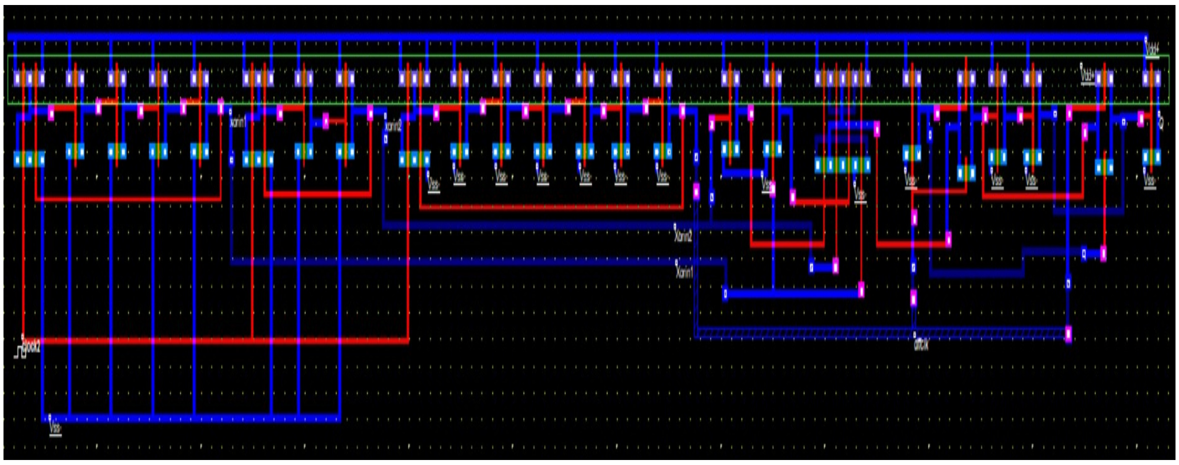


Figure 3.2: Layout of Random Bit Generator

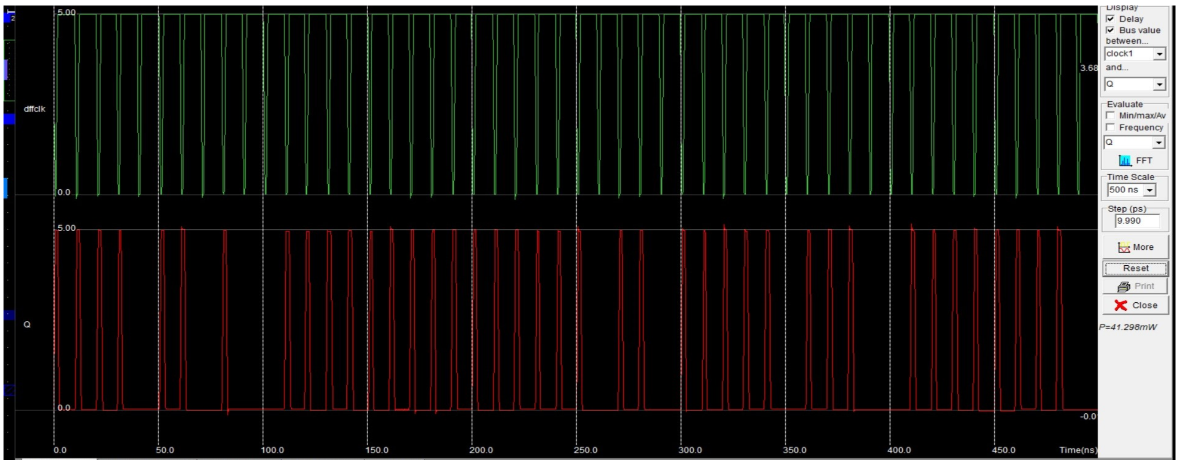


Figure 3.3: Test Run 1 Output of Random Bit Generator

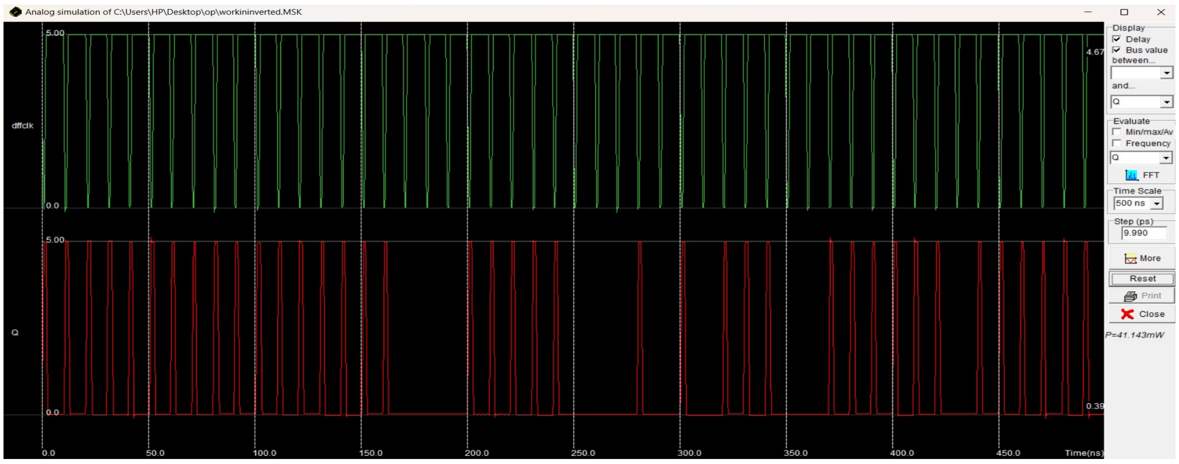


Figure 3.4: Test Run 2 Output of Random Bit Generator

4. STATISTICAL ANALYSIS OF RANDOMNESS

4.1. Shannon's Entropy Test

In information theory, the entropy of a random variable is the average level of randomness, uncertainty inherent to the variable's possible outcomes.
Shannon's entropy test is commonly used in cryptography and random number generation to evaluate the quality and randomness of generated bits. However, it has some limitations, such as being sensitive to biases and patterns in the sequence and not detecting some kinds of non-randomness. Therefore, it is usually used in conjunction with other statistical tests to ensure the quality and security of random data.
For a binary sequence with p being the probability of bit being '1', Shannon's entropy can be calculated by:

$$H(p) = -p*\log(p) - (1-p)*\log(1-p)$$

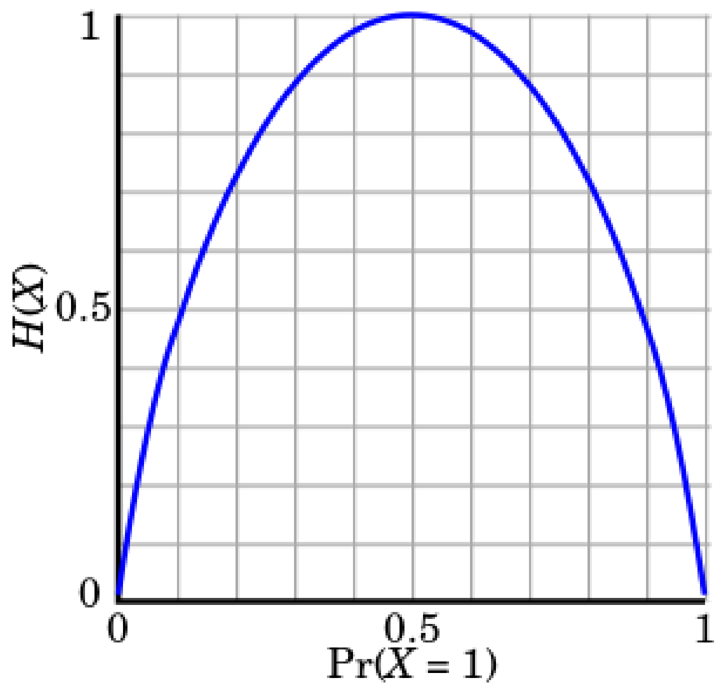


Figure 4.1: Shannon's Entropy Function

The function attains maxima at p=0.5 and minima at p=0 and p=1 i.e when both the bits are non-uniformly distributed throughout the sequence.

Results of a test run are shown below and Shannon's Entropy is calculated:
Output: 10101010001010010000101010101010101010101010101000010100010101010100
Number.of ones: 40
Number of zeros: 58
Entropy: 0.9755259511264971
Entropy close to one is truly random and values towards zero shows less randomness.
Our test run data is close to being truly random.

4.2. Limitations of Shannon's Entropy Test

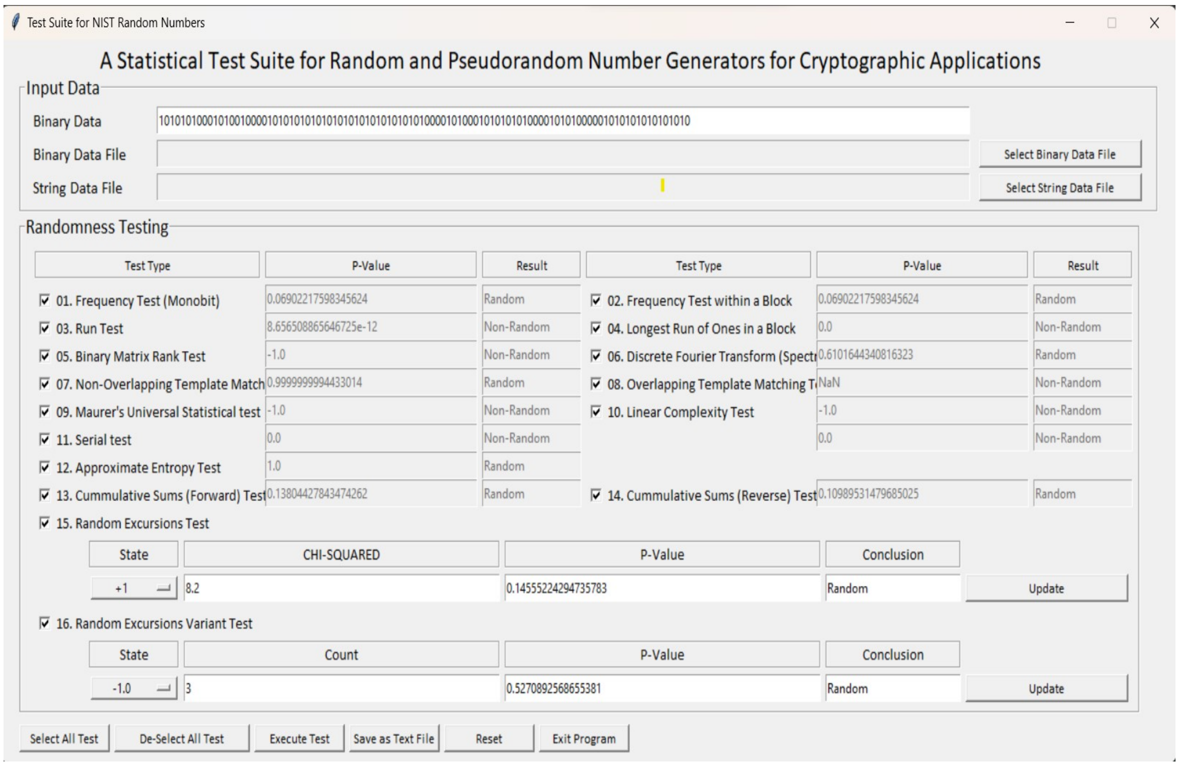
Shannon's entropy test, also known as the Shannon-Weaver entropy, is a measure of the amount of uncertainty or randomness in a system. While it is a widely used method for analyzing information and data, it has several limitations. Some of the limitations of Shannon's entropy test are:

- It assumes independence: Shannon's entropy test assumes that the events or symbols in a system are independent of each other, which may not always be the case. If the symbols or events are interdependent, the entropy calculation may not accurately reflect the system's randomness.
- It assumes a fixed probability distribution: Shannon's entropy test assumes that the probability distribution of the symbols in a system is fixed and known, which may not always be the case. In real-world applications, the probability distribution may change over time, making it difficult to use entropy as a measure of randomness.
- It may not be suitable for certain types of data: Shannon's entropy test is primarily designed for discrete data, such as text or binary code. It may not be suitable for continuous data, such as audio or images, or for complex data structures, such as graphs or networks.

  While the Shannon's entropy is a good measure of randomness of data as a whole, it fails to recognize the patterns that might be occurring in the sequence. Therefore we perform another extensive test mandated by NIST (National Institute of Standards and Technology).

4.3. NIST Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications

NIST's Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications is a collection of various tests for randomness. A Python implementation of the test suite can be found at link. The following is result of tests for generated output.



Test Suite for NIST Random Numbers

A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications

Input Data

Binary Data	1010101000101001000010101010101010101010101010101000010100010101010100001010100000101010101010
Binary Data File	
String Data File	

Randomness Testing

Test Type	P-Value	Result	Test Type	P-Value	Result
✓ 01. Frequency Test (Monobit)	0.06902217598345624	Random	✓ 02. Frequency Test within a Block	0.06902217598345624	Random
✓ 03. Run Test	8.656508865646725e-12	Non-Random	✓ 04. Longest Run of Ones in a Block	0.0	Non-Random
✓ 05. Binary Matrix Rank Test	-1.0	Non-Random	✓ 06. Discrete Fourier Transform (Spectr	0.6101644340816323	Random
✓ 07. Non-Overlapping Template Match	0.9999999994433014	Random	✓ 08. Overlapping Template Matching T	NaN	Non-Random
✓ 09. Maurer's Universal Statistical test	-1.0	Non-Random	✓ 10. Linear Complexity Test	-1.0	Non-Random
✓ 11. Serial test	0.0	Non-Random		0.0	Non-Random
✓ 12. Approximate Entropy Test	1.0	Random			
✓ 13. Cummulative Sums (Forward) Test	0.13804427843474262	Random	✓ 14. Cummulative Sums (Reverse) Test	0.10989531479685025	Random

✓ 15. Random Excursions Test

State	CHI-SQUARED	P-Value	Conclusion
+1	8.2	0.14555224294735783	Random

✓ 16. Random Excursions Variant Test

State	Count	P-Value	Conclusion
-1.0	3	0.5270892568655381	Random

Figure 4.2: NIST Test Suite Results

4.4. NIST Test Suite Observations

The test suite is a collection of benchmarks that gives an estimate of randomness. Each test has it's own parameters and one test result can't be universally valid. It is observed that a random binary sequence generated by software can't pass all the tests in the suite. So decision of sequence being random shouldn't be taken based on one test data and should never be based on one verdict of one test. Other sequences generated by the circuit may perform differently in the test suite. It is advised to look into parameters and decision making rules of each test before drawing conclusions.

5. CONCLUSION

In conclusion, a ring oscillator-based random bit generator is a promising approach for generating high-quality and secure random bits. In this project, we have explored the principle behind a ring oscillator and its use as a random bit generator.

We have seen that a ring oscillator consists of an odd number of inverters connected in a closed loop, where the oscillation frequency is determined by the delay of each inverter. The output of a ring oscillator exhibits a high degree of jitter and randomness due to the non-linear behavior of the inverters and the environmental noise.

We have also implemented a ring oscillator-based random bit generator using an FPGA board and evaluated the randomness of the generated bits using statistical tests such as the NIST tests. The results showed that the generated bits passed the tests and exhibited good randomness properties.

Overall, a ring oscillator-based random bit generator can be a low-cost and efficient solution for various applications that require high-quality and secure random bits, such as cryptography, communication systems, and IoT devices. Further research can be conducted on improving the quality and security of the generated bits, optimizing the design parameters of the ring oscillator, and integrating it with various systems and applications.

6. FUTURE ENHANCEMENTS

6.1. Circuit Combinations

Ring oscillator based random bit generators (RO-RBGs) are widely used in many applications that require a source of random numbers. However, they have several limitations, including susceptibility to environmental variations and lack of statistical randomness. While we could generate random bit output using 3 ring oscillators, based on area available and applications the idea can be extended to multiple ring oscillators with multiple input XOR gate. Here are some future improvements that can be made to RO-RBGs:

Non-linearity: One way to improve the randomness of RO-RBGs is to introduce non-linearity into the ring oscillators. Non-linearity can help to reduce correlation between the oscillators, and hence improve the entropy of the output.

Post-processing: The output of RO-RBGs can be further processed using statistical techniques such as whitening or conditioning to improve the quality of the random numbers generated. This can help to remove any bias or non-randomness in the output.

Testing and validation: RO-RBGs should be rigorously tested and validated using standard statistical tests to ensure that the output meets the required randomness criteria. This can help to detect any weaknesses or vulnerabilities in the generator.

Integration with other random sources: RO-RBGs can be combined with other sources of randomness, such as thermal noise or quantum noise, to improve the overall quality and randomness of the output. This can help to reduce the impact of any weaknesses or vulnerabilities in the individual sources.

6.2. Circuit Optimisation

The circuit that we've implemented can be further made compact by reducing the inverters required for XOR input. Inverted output can be directly taken from previous inverter of ring oscillators. Doing so drastically reduced power consumed.

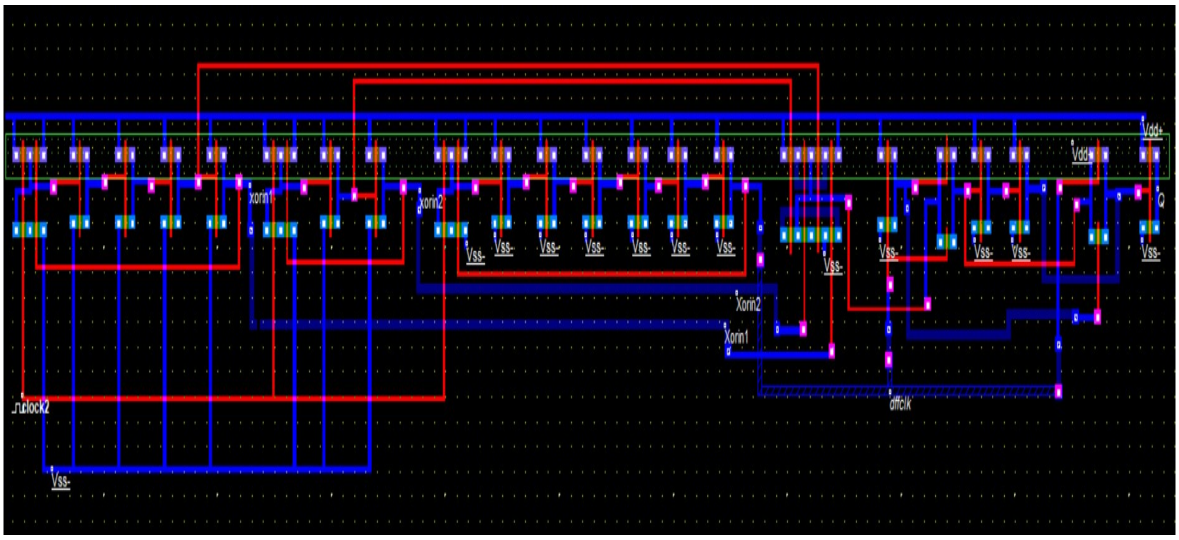


Figure 6.1: Optimised Circuit in terms of power, area and number of transistors

Since we have used ring oscillators with 4, 2 and 6 inverters the number of transistors required can't be further reduced because we've already taken the minimum possible arrangement of oscillators. Optimisation can be made at XOR gate and the results are tabulated below.

Table 6.1: Comparison between circuits implemented

Number of Transistors	**Power Consumed**
40	41 mW
36	2 mW

7. REFERENCES

- Suresh, Vikram Belur. ”On-chip true random number generation in nanometer cmos.” (2012).
- Cover, T. M., & Thomas, J. A. (2012). Elements of information theory. John Wiley & Sons. (Chapter 2.3 discusses limitations of entropy.)
- Yarosh, S. (2018). A critical evaluation of entropy as a measure of password strength. Proceedings of the 2018 Workshop on Theory and Practice of Provable Security (pp. 51-65). Springer. (Section 2.2 discusses limitations of entropy in the context of password strength.)