Physical Unclonable Function and True Random Number Generator : a Compact and Scalable Implementation

Abhranil Maiti
ECE Dept, Virginia Tech
Blacksburg, VA -24061
1-(312)-752-6235
abhranil@vt.edu

Raghunandan Nagesh
ECE Dept, Virginia Tech
Blacksburg, VA -24061
1-(540)-808-9556
rnagesh@vt.edu

Anand Reddy
ECE Dept, Virginia Tech
Blacksburg, VA -24061
1-(540)-808-3133
anandrdy@vt.edu

Patrick Schaumont
ECE Dept, Virginia Tech
Blacksburg, VA -24061
1-(540)-231-3553
schaum@vt.edu

ABSTRACT
Physical Unclonable Functions (PUF) and True Random Number Generators (TRNG) are two very useful components in secure system design. PUFs can be used to extract chip-unique signatures and volatile secret keys, whereas TRNGs are used for generating random padding bits, initialization vectors and nonces in cryptographic protocols.

This paper proposes a scalable design technique to implement both a delay-based PUF and a jitter-based TRNG using ring oscillators. By sharing and reusing a significant amount of hardware resources, we achieve nearly 50% area reduction as compared to discrete implementations. We also propose and demonstrate a co-processor-based design that renders the circuit portable across various embedded processor platforms on FPGAs. Multiple scaled designs using 32 to 128 ring oscillators have been implemented and verified on Xilinx Spartan3S500E FPGA. A representative design uses 32 3-inverter ring oscillators, 64 flip-flops/latches, 31 2-input XOR gates and control circuitry giving a 3.2Mbps truly random stream and 31-bit unique device signature.

Categories and Subject Descriptors
K.6.5 [Management of Computing and Information Systems] Security and Protection

General Terms
Measurement, Experimentation, Security.

Keywords
TRNG, Ring oscillators (RO), jitter, PUF, FPGA, Scalable, Macro

1. INTRODUCTION
A PUF is a platform-unique function which, when supplied with an input challenge, produces an output response determined by the behavior of a complex, unclonable physical system. PUF can be used for authentication of chips and can generate secret keys required for cryptographic operations without the need for expensive non-volatile memories [1,2].

On the other hand, random numbers are very important in cryptographic systems and algorithms. They are used as secret keys, padding bits, initialization vectors, nonces, and seeds. The security of the entire system is dependent on the unpredictability of these random numbers.

PUFs and TRNGs together provide solutions to many security problems, but implementing both of them in a system increases the chip area. Moreover, most security applications or devices have embedded processors and associated firmware controlling peripherals. Hence, it is beneficial to develop a cross platform compatible coprocessor architecture which can be easily integrated into different systems to provide unique device ID using PUF, or a high quality truly random stream on demand. Additionally, a scalable architecture is needed as the device signature length from the PUF and throughput from the TRNG may vary depending on the security requirements of the application.

In this work, we propose a ring-oscillator (RO) based PUF and TRNG design technique to address all of the above problems while ensuring that both PUF and TRNG operate within quality requirements in terms of random bit throughput, uniqueness and reliability. The main contributions of our work are:

- *Area reduction*: We implement both PUF and TRNG sharing and reusing common hardware resources resulting in smaller size.

- *Scalability*: Different configurations/orders of PUF and TRNG can be implemented in an easy and efficient manner.

- *Portability*: Coprocessor based architecture is flexible and portable across different platforms.

- *Bridge between circuit level and system level*: Coprocessor based architecture with a 32-bit bus interface enables software access of PUF and TRNG outputs, thereby allowing easy integration into existing firmware.

The rest of the paper is organized as follows. Section 2 briefly talks about related work on PUF and TRNG circuits. Section 3 describes the design architecture and section 4 discusses detailed design implementation. Results from experiments are analyzed in section 5 and we conclude the paper in section 6.

2. BACKGROUND
2.1 PUF
A PUF is a platform-unique challenge-response function. Different PUF implementations in integrated circuit have been proposed so far. Our design is based on a ring-oscillator PUF proposed by Suh et al [1]. This PUF exploits the fact that manufacturing process variations cause random but static

GLSVLSI'09, May 10–12, 2009, Boston, Massachusetts, USA.

variations in the frequency of identically laid-out ring oscillators. Figure 1 below shows how a RO-based PUF is implemented.

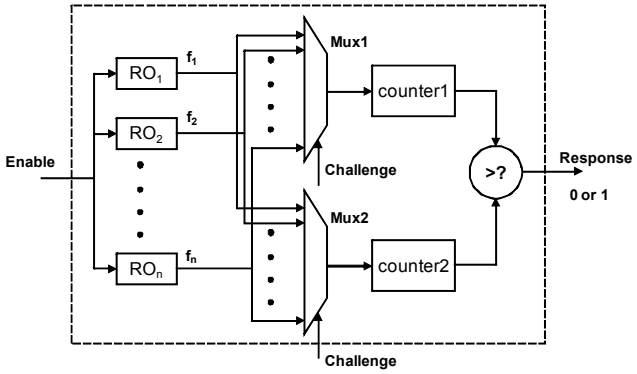


Figure 1. Ring Oscillator based PUF

The PUF output is created by pair-wise comparison of the ring oscillator frequencies. These comparisons can be represented as a challenge/response function, where the chosen ring oscillator pair is the challenge, and the comparison result is the response.

2.2 TRNG

Several TRNG designs have been proposed so far using analog, digital and mixed signal components [3,4,5].

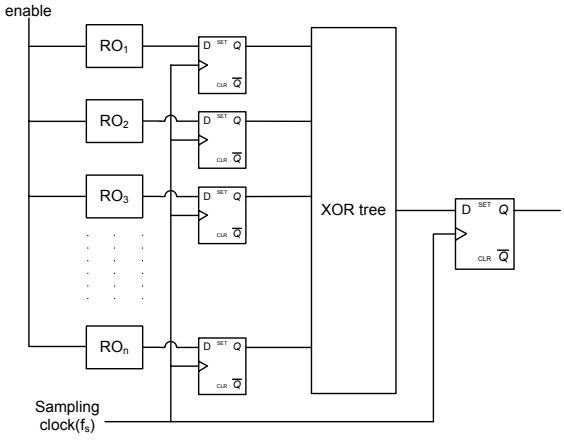


Figure 2. Ring Oscillator based TRNG

We started from the design proposed by [3]. This TRNG is based on jitter obtained from ring oscillators where jitter is defined as the "deviation in a signal transition from the expected value". To harvest the jitter and generate a stream of truly random bits, the outputs of several ring oscillators are fed to an XOR tree. The output of the XOR tree is sampled using a D-flip-flop and subjected to post processing to obtain a random stream

This design has certain drawbacks highlighted by [4],[6] and [8]. The issues include narrow pulse rejection and attenuation of high frequency components in the XOR tree. In order to solve this problem, [6] proposes adding an extra sampling flip flop at the output of each ring, so that the data fed into the XOR tree is synchronized and held constant for one full clock cycle. The scheme is shown in Figure 2

3. BASIC PUF -TRNG COMBINATION

Our core contribution is the merging of a PUF and a TRNG into a single module by using ring oscillators as basic elements. Figure 3 shows the overall design. In TRNG mode, all the ROs are enabled simultaneously and the outputs of the intermediate sampling flip-flops are selected using a 2:1 multiplexer. This multiplexer enables the selection of the un-sampled RO output needed for PUF or the sampled RO output (from intermediate D Flip-Flop) for TRNG. The multiplexer outputs are fed into the XOR tree whose output is sampled by a flip-flop to generate a random bit stream. In PUF mode, only one RO is enabled at a time and its output is selected using the 2:1 multiplexer and fed into the counter through XOR tree. In this design, we need only one counter and the frequency comparison operation to evaluate response of a PUF is handled in software.

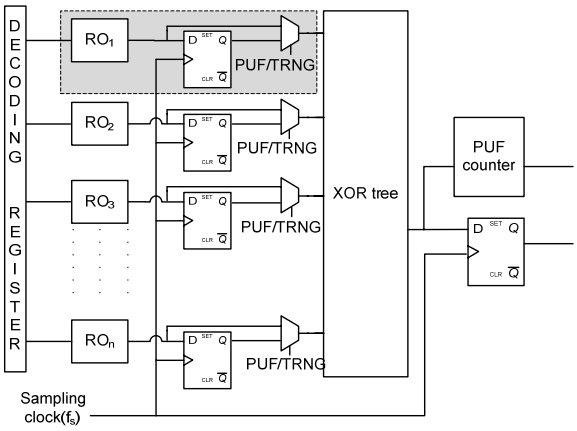


Figure 3. Combination of PUF and TRNG

The RO selection is done using an n-bit wide memory-mapped decode register. The selection between PUF and TRNG functionalities is also done using a bit in the control register.

It is evident from Figure 3 that there is a regular structure (shown in gray colored box) which is used multiple times. We exploit this regularity and propose a modular structure which renders the design scalable by multiple instantiation.

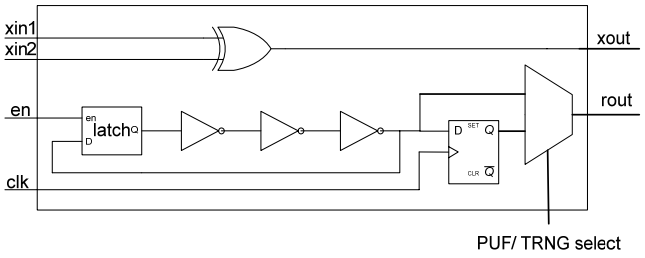


Figure 4. Scalable PUF/TRNG macro cell

To make the XOR tree scalable, we opted for a macro-cell approach. The smallest block of the design is a simple macro cell which has a ring oscillator composed of 3 inverters and a latch, a D Flip flop and an XOR-2 as shown in Figure 4. We implemented the 2 input XOR gate inside the macro cell to create the XOR tree of the TRNG.

To demonstrate the scalability, we can devise a simple PUF-TRNG using 8 macro cells (numbered 1 to 8) as shown below in Figure 5. The outputs of two ROs are fed as the inputs to an available XOR gate in any macro. For a structure with 2^n ROs, we need 2^{n-1} 2 input XOR gates.

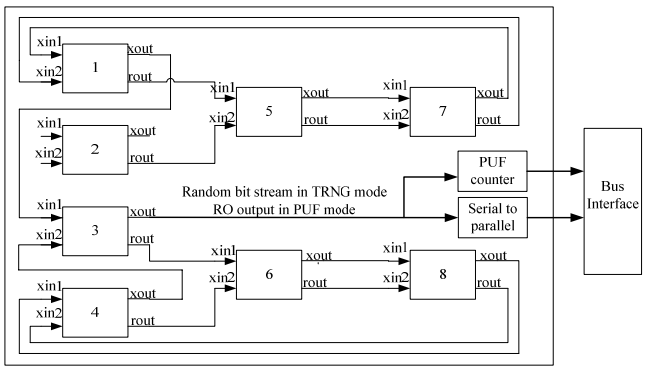


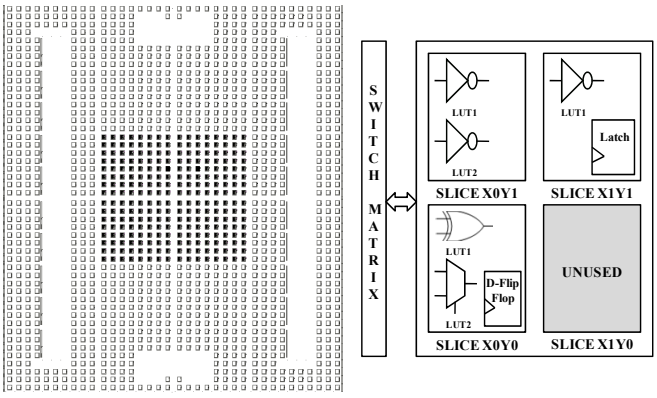
Figure 5. PUF-TRNG with 8 ROs using macro cells (enable and clk inputs are not shown for clarity)

At anytime, the output of the final XOR gate (from macro cell 3 in the Figure 5) will be a TRNG bit stream or single ring oscillator output depending on the software register setting. In PUF mode, the final output would be fed to a counter for frequency measurement and then to the bus interface for analysis. In TRNG mode, the output will be parallelized and stored into a memory mapped register and read each sampling clock cycle to be stored as a random bit stream in an off chip memory.

A 32 RO-based PUF-TRNG would require 32 Macro cells with 100% utilization of all the elements except XOR gates of which we need 31 in total. In general, if a higher order PUF or higher throughput TRNG is needed, increasing the number of macros proportionately would suffice along with increasing the width of decode register in software to enable larger selection of ring oscillators.

4. IMPLEMENTION DETAIL

To ensure that the PUF responses are purely based on random process variation, all the ring oscillators are required to be identical in terms of circuit components, their placement and routing. In [1], it is proposed that ring oscillators be created as a hard macro to meet the above requirement.



6(a) **6(b)**

Figure 6(a). Array configuration of macro cells of a PUF-TRNG circuit on Spartan XC3S500E FPGA. 6(b) Mapping of a macro cell into a Xilinx CLB

In Figure 6(a), black squares in the middle of the FPGA fabric represent hard macro cells with each one implemented in one CLB as demonstrated in Figure 6(b).

One of the objectives of this work is to have a software configurable PUF-TRNG with a power saving/disable mode which can be added to an existing 32 bit peripheral bus for an embedded processor.

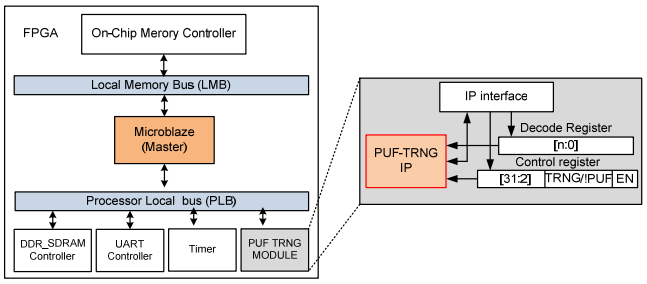


Figure7. System architecture with memory mapped registers

We used the Xilinx Microblaze 32 bit soft-core processor with Core Connect (PowerPC) bus architecture from IBM. The regular macro structure along with a serial-to-parallel converter, PUF counter and the wrapper is attached to the 32 bit processor local bus (PLB). An n-bit wide decoder register is used during PUF mode to select a ring oscillator for frequency measurement. A simple memory mapped register bit in the system selects PUF or TRNG and another bit enables/disables the ring oscillators. The overall system on FPGA is shown in Figure 7.

5. RESULTS

The PUF has been characterized based on quality metrics such as uniqueness and reliability, whereas the TRNG has been characterized in terms of randomness of the output bit stream.

5.1 PUF Characterization

1) Uniqueness (U) - We define uniqueness as a measure of how clearly a PUF can distinguish an FPGA from another. When a PUF is implemented on k FPGA chips; we define its uniqueness U as the average of the percentage hamming distance between responses from every pair of PUF implementations.

This is an estimate of the inter-die variation. It does not represent the actual probability of inter-die variation.

2) Reliability (R) - It expresses how consistently a response 'r' is reproduced by a PUF for a challenge 'c'.

R = % of number of times a response r is exactly reproduced by a PUF when a challenge c is applied to it n times.

In this experiment, we collected data for uniqueness and reliability under a constant operating environment and without any post processing to characterize PUF. Experiments on PUF are done based on 8 Xilinx Spartan3S500E FPGA chips.

Table 1: Uniqueness and reliability of PUF in different PUF-TRNG implementations

No of ROs	Uniqueness	Reliability
32	42.8%	80.6%
64	38.2%	79.5%
128	39.9 %	77.9%

It can be observed in table 1 that the PUF behaves with a consistent and high value of uniqueness for all the implementations while giving a moderately high value of reliability.

5.2 TRNG Characterization

We first tested the quality of the bit stream from TRNG using DIEHARD [9] and NIST [10] test suites - the most common test suites used by contemporary TRNG researchers [3,6,7,8]. These alone do not guarantee true randomness [4, 5], hence a restart test is also used to ascertain that the stream is not pseudorandom.

We carried out randomness test on large sample of 100 Million bits and the result is shown in Table 2. The results were positive for all the designs from 32 rings to 128 ring structures. Note that the number of rings has been scaled by powers of 2 to represent the vast range of configurations. Bias values (% of 1's in the stream) approached the ideal value of 50%.

Table 2: Result of NIST and DIEHARD test on TRNG outputs

Number of ROs	Bias	DIEHARD	NIST
32	49.92	Pass	Pass
64	49.96	Pass	Pass
128	49.94	Pass	Pass

All the designs have been tested successfully from 400kbps up to 3.2Mbps throughput. It has been proven by [3,6,7,8] that increasing the number of rings increases the throughput.

A test was carried out to verify the start up sequences for 10 restarts for a representative design with 32 rings and 3 inverters per ring. First 24 sampled bits after every restart were observed and plotted in MATLAB (shown in Figure 8). We observe that they are different each time, thereby discarding any pseudo randomness possibilities.

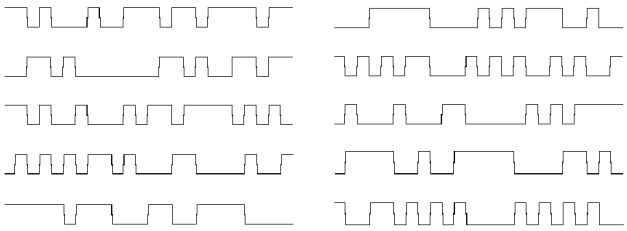


Figure 8. TRNG output for 10 restarts

Dichtl in [4] questioned the effects of RO coupling which makes closely spaced ROs operate at same frequency and hence, we decided to observe the average frequencies for closely placed RO structures. We observed that the frequency varied significantly and the jitter accumulation overcame any locking effects.

5.3 Resource Utilization

Table 3: FPGA resource utilization on Spartan XC3S500E by PUF-TRNG (including control circuitry)

Number of ROs	LUT4s	FFs /Latches
32	712	753
64	904	817
128	1288	945

The logic utilization of our proposed design is much lower compared to a discrete implementation. A 32 RO-based PUF and TRNG implemented separately with control circuitry and bus interface utilized 1339 LUT4s and 1474 Latches/Flip Flops – as compared to the 712 LUT4s and 753 latches/FFs (table 3) for the combined implementation leading to a resource reduction of almost 50%.

6. CONCLUSIONS

We proposed a design technique to implement both PUF and TRNG circuit using reduced area with resource savings of up to 50%. The validity of our design is supported by multiple implementations and their experimental data. This highly scalable, purely digital architecture is easy to implement and is controlled entirely using software on a standard 32 bit bus. In the light of increasingly compact single chip solutions, this architecture would be ideally suited for security device or trusted computing modules.

7. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation Grant No 0644070.

8. REFERENCE

[1] G. E. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of Design Automation Conference*, June 2007.

[2] J. Guajardo, S. S. Kumar, G.-J. Schrijen and P. Tuyls. Physical unclonable functions and public-key crypto for FPGA IP protection. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, August 2007.

[3] B. Sunar, W. J. Martin, D. R. Stinson. A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks. In *IEEE Transactions on Computers, vol 58, no 1, pages 109-119, January 2007.*

[4] M. Dichtl, and J. Dj. Golic. High-Speed True Random Number Generation with Logic Gates Only. In *Proceedings of the Cryptographic Hardware and Embedded Systems - CHES 2007.*

[5] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim and Bohdan Karpinskyy Fast Digital TRNG based on Metastable Ring Oscillator. In *Proceedings of the Cryptographic Hardware and Embedded Systems - CHES 2008.*

[6] Knut Wold, Chik How Tan. "Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings", In *Proceedings of the International Conference on ReConFigurable Computing and FPGAs, Reconfig 2008*.

[7] D. Schellekens, B. Preneel, and I. Verbauwhede. FPGA Vendor Agnostic True Random Number Generator *International Conference on Field Programmable Logic and Applications, FPL'06. pages 1-6, August 2006.*

[8] Sang-Kyung Yoo, Berk Sunar, Deniz Karakoyunlu, Berk Birand. A Robust and Practical Random Number Generator, under review. Pre-print

[9] G. Marsaglia. DieHard: A Battery of Tests of Randomness, http://stat.fsu.edu/_geo,1996.

[10] NIST Special Publication 800-22 A Statistical Test Suite for Random and Pseudorandom Numbers, 2000.