Lightweight Digital Hardware Random Number Generators

Teng Xu, Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{xuteng, miodrag}@cs.ucla.edu

***Abstract*— Random Number Generator (RNG) plays an essential role in many sensor network systems and applications, such as security and robust communication. We have developed the first digital hardware random number generator (DHRNG). DHRNG has a small footprint and requires ultra-low energy. It uses a new recursive structure that directly targets efficient FPGA implementation. The core idea is to place or extract random values in FPGA configuration bits and randomly connect the building blocks. We present our architecture, introduce accompanying protocols for secure public key communication, and adopt the NIST randomness test on the DHRNG's output stream.**

I. Introduction

The use of RNGs in sensor networks has a long history and results in significant variety of techniques, e.g., security, privacy and remote trust. Specifically, distributed sensing raises the importance of sensor security to an even higher level [1][2][3][4][5]. It is important for the sensor user to verify that the remotely received data is actually from the trusted sensors in the distributed system. Meanwhile, RNGs enable public key communication and other cryptographic protocols in sensor networks due to its intrinsic randomness.

Classical (algorithmic) RNGs have been widely used. However, they are not well suited for the sensor networks. They are relatively slow, require high energy, and large hardware footprints. They are also susceptible to side channel and physical attacks [6][7].

A number of hardware random number generators (HRNGs) are proposed in [8] to mitigate the above the problems. All of them employ analog mechanisms that cannot be controlled or reduplicated, such as with the use of physically unclonable functions (PUFs) in conjunction with Von Neumann data post processing as random number generator was done by a group in MIT [9]. However, traditional analog mechanisms are also high energy consuming and pose high implementation requirements in terms of measurement accuracy and environmental stability.

In this paper, we propose the DHRNGs on the platform of FPGA that inherit the general advantage of HRNGs, but employ a completely digital system, which is extremely lightweight and resilient to environmental conditions [10][11]. Furthermore, two variants of DHRNGs are discussed, which respectively targets different applications. The first variant targets easy replication, which enables creation of synchronized RNGs that produce identical streams, therefore, can be used for encryption/decryption and robust public key communication. The second variant employs random values after power-up of the circuit which makes each DHRNG to be unique and unclonable. The essence of the DHRNG variant II is that it is a new type of SRAM PUF that can be applied to all the classic HRNG applications because of the difficulty to reproduce or simulate the new RNG. Both variants have the same hardware architecture with only small modifications to the fabrication procedure.

II. Related Work

We now briefly survey the most directly related literature on physical unclonable functions and hardware random number generators.

A. PUF

Papu et al. demonstrated the first active physical unclonable function using optical mesoscopic systems in 2001 [12]. Devadas and members of his research group observed that intrinsic deep submicron process variation in silicon is an ideal practical and economical starting point to fabricate a large amount of PUFs [13][14].

More recently, many types of PUFs [15][16][17][18] are proposed and several research groups have demonstrated SRAM-based PUFs [19]. The key idea is that each SRAM cell has a high probability that it is initialized to some value, either 0 or 1, after each power-up. Although the "stability " of the SRAM PUF cells has always been a problem, recently, it has been experimentally demonstrated that the use of device aging [20][21][22], specifically hot carrier injection (HCI) [23][24], can completely eliminate this problem.

B. RNG

Pseudo number generation and its evaluation have a long history and have resulted in a significant variety of techniques and tools. However, they are rarely adopted in compact general purpose processors and FPGA implementations since they usually use 32 or 64 bit numbers as their variables. A list of FPGA random number generators are also proposed [25]. All of them employ analog mechanisms that cannot be controlled. Exactly the same situation exists with other hardware-based random generators [26], therefore, none of them are suitable to be applied in sensor networks. The use of PUFs in conjunction with standard pseudorandom generators and von Neuman data post-processing was analyzed by Devadas group [9].

III. Architecture

We now present the micro-architecture of DHRNG. The basic idea is to use randomly connected lookup tables(LUTs) to generate the combinational logic to produce an output stream. The DHRNG architecture shown in Figure 1 is consisted of a LUT network of height h and width w, where h is the level of randomly connected LUTs and w is the number of k-input LUTs in each level. Each LUT randomly chooses its inputs to be either output of a LUT in the previous level or its own primary input in forming an output bit. By offering m bits of primary inputs, w bits of final outputs are expected. Note that the randomness of such structure mainly comes from the randomness of shuffling and the contents of each LUT, which can be easily achieved when h and w are large enough.

We use the structure shown in Figure 1 to generate a stream of random numbers. A stream is generated by first randomly choosing an m-bit input vector as the random seed, which is then used to generate a w-bit output vector. A bit-wise exclusive-or operation is performed between w-bit output vector and the current input vector to produce a binary vector. The generated binary vector would serve as the random seed for the next round. We repeat this procedure to acquire the output stream. We note that, although we allocate the LUTs in such a way that each cell has the same probability to be either 1 or 0 in each individual case, the number of 0's and number of 1's may not be the same. Therefore, we further adopt Von Neumann correction on our obtained output stream to guarantee equal number of 0s and 1s in the final bit stream.

IV. Two Variants

Two variants of the DHRNG are proposed in this section which respectively targets on different applications. Both variants use exactly the same architecture as discussed in previous section and the only difference is the way to generate the LUT contents.

A. Variant I

As for the Variant I DHRNG, the core idea is to manually allocate the contents of the LUTs and the connections so that the DHRNG can be easily replicated, which enables the creation of synchronized DHRNGs that produce identical streams. Note that matching procedure can either be configured by the sensor owner themselves or the trusted third party before the DHRNGs are put into use.

Now that since both devices contain the same LUT contents connected by the same network, they produce the same outputs for a given set of inputs, enabling such a design to be applied for multi-party communication. Only the party with the right-configured DHRNG can encrypt/decrypt the messages. The advantage of this system comes from three points. The first is the intrinsic randomness of each device's output stream, making it difficult to decrypt the message statistically. The second point is the easy replication. As long as the DHRNG configuration is known, it is very easy to configure and reproduce the same piece of device. Therefore, multi-party communication becomes simple which is particularly useful in the case when a sensor owner wants to communicate with his multiple sensors. The last point, as well as the most important point, the device can be custom configured. Therefore, even in the case when a malicious manufacturer or untrusted third-party exists, the device can still be made secure.

B. Variant II

Before explaining the concept of Variant II DHRNG, it is essential to introduce the HCI-based power-up. Hot carrier injection (HCI) is a phenomenon in which the electron or the hole in a transistor may be trapped in the gate oxide when provided with high enough energy. HCI-based power-up presents a PUF response reinforcement technique based on HCI, which can reinforce the PUF golden response in a short stress time without impacting the surrounding circuits. According to a recent study, powering up a circuit for a long enough duration (e.g., 125s) causes the content of each SRAM cell to be randomly altered to either a stable 1 or 0 according to its unique intrinsic ID regardless of the environmental condition.

The Variant II DHRNG is built by applying the HCI-based power-up on the SRAMs of LUT cells as shown in Figure 1. Two operations are required, the configuration and the power-up. Configuration is the process in which we randomly connect the LUTs on the FPGA. HCI-based power-up, which should be adopted after configuration, is used to assign the contents of the SRAM cells, in other words, the contents of the LUTs on the FPGA. Due to the properties of the HCI-based power-up, the contents of the LUTs are assigned in a completely random, unique and stable way without impacting the surrounding circuits. After the two operations complete, both the connection and the contents of the LUTs in the DHRNG are set in an unpredictable way.

The configuration and power-up creates a Variant II DHRNG that enables two intrinsic properties which we can take advantage of. First, the output stream cannot be predicted or controlled in any way. Therefore, such a design can be appied to generate the random seed for any secure protocols of sensor networks. Second, the HCI-based power-up makes our digital PUFS inherit the unclonable property of traditional PUFs.

V. Test Results

We adopt the NIST randomness test as well as the standard security test on our DHRNG structure in this section to validate the output randomness.

A. NIST randomness test

The NIST randomness test [27] is a battery of standard statistical tests to detect non-randomness in binary sequences constructed using either random number generators or pseudo-random number generators.

We simulate our DHRNG with height $h = 20$ and width $w = 64$. The initial random seed is a randomly generated 64-bit input vector ($m = 64$). Every LUT cell in the DHRNG has an equal chance to be a 1 or 0. As mentioned before, we

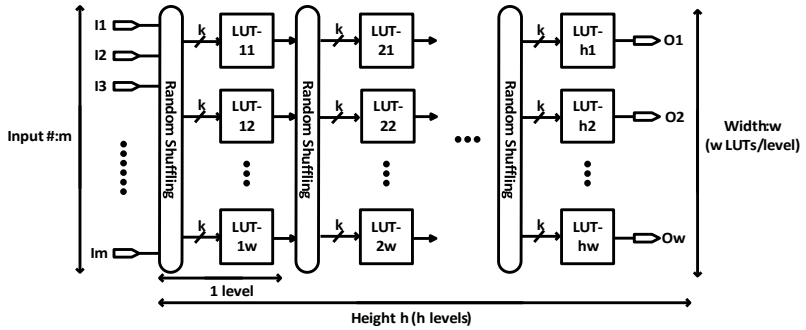


Fig. 1: DHRNG architecture with input number m, width w and height h.

use the DHRNG architecture iteratively to generate the output stream. For each single statistical test, we test for 1000 cases. Table I shows the average passing ratio of each NIST statistical test. We can see that the proportion of successful tests is high enough to indicate excellent randomness in the output stream.

Statistical Test	Avg. Success Ratio
Frequency	100%
Block Frequency (m=128)	99.5%
Cusum-Forward	99.2%
Cusum-Reverse	99.2%
Runs	97.2%
Longest Runs of Ones	98.1%
Rank	99.4%
Spectral DFT	98.5%
Non-overlapping Templates (m=9)	95.8%
Overlapping Templates (m=9)	97.7%
Universal	100%
Approximate Entropy (m=8)	98.1%
Random Excursions (x=+1)	98.8%
Random Excursions Variant (x=-1)	97.0%
Serial (m=16)	99.5%
Linear Complexity (M=500)	97.9%

TABLE I: NIST Statistical Test Suite average success ratio. 1000 arrays are tested for each test. Significance Level $\sigma = 0.01$. When $P\text{-}value \geq \sigma$, the array passes test.

B. Security test

While using the DHRNG to secure the sensor network, the resistance against malicious attack/prediction is important. In this section, we statistically analyse the security of the DHRNG system by assuming potentially different types of attacks. The attacks would be regarded successful if the output stream can be predicted. The simulation is conducted on a DHRNG architecture with height $h = 20$ and width $w = 64$. The statistical results are based on 1,000,000 input-output pairs.

One type of prediction is to predict outputs by the knowledge of the outputs from similar inputs. This attack is dangerous when the output vector with similar input vectors are highly correlated with one another. To test this, we summarize the hamming distance (the number of bits that are different between two vectors) between the output vectors by changing one bit of the input vectors at each iteration. In the ideal case, the distribution would be in the form of a binomial distribution with the peak on the half of the number of outputs. Figure 2a shows the accumulative results of 1000 test cases, and in each case, 1,000,000 instances of hamming distance are tested. The binomial distribution proves that the DHRNG output stream can not be predicted in this way.

Similar to the previously described attack, the other type of attack attempts to predict an output bit O_i according to the value of an input bit I_j or a corresponding output bit O_j. In either case, if the output bit has a strong correlation with either an input bit or another output bit, then the attacker can deduce the output vector by knowing input bits or a subset of the output bits. We present a conditional probability map of $P(O_i = 1|I_j = 1)$ in Figure 2b and $P(O_i = 1|O_j = 1)$ in Figure 2c depicting the low potential for prediction based on input to output correlation and output to output correlation.

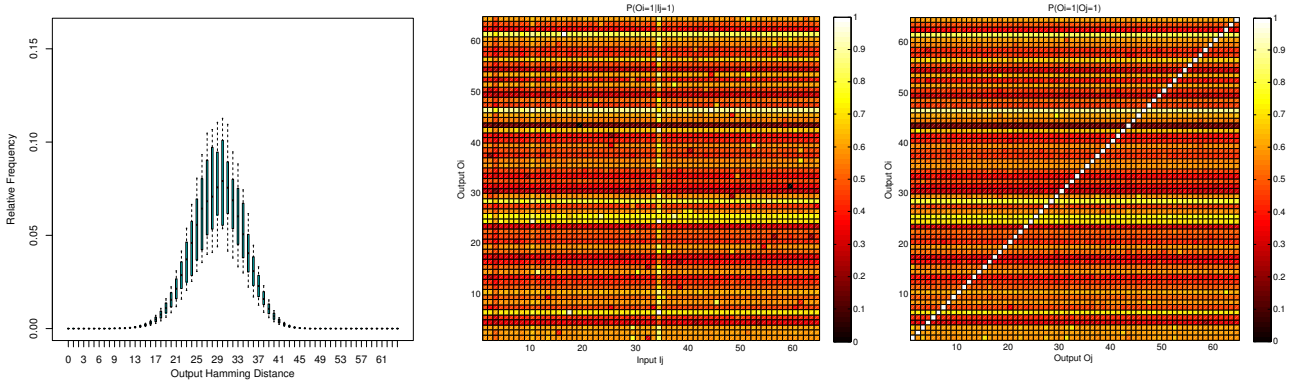
Protocol 1 Public Key Communication

1: The sensor chooses a random seed as input vector I and computes the corresponding output vector O.
2: The sensor XOR the output vector O with the message M to be sent and gets the result R.
3: The sensor sends I and R to the sensor owner.
4: The sensor owner computes output vector O with the received I and his/her identical DHRNG.
5: The sensor owner XOR O with the received R to get message M.

VI. Protocols

Public key communication is one of the most widely used and fundamental protocols for secure message exchange in sensor networks. We present how to use DHRNG to achieve this in this section. The detailed steps of public key communication using DHRNG are enumerated in Protocol 1. Note that both the sensor owner and the sensor are required to coordinate their Variant I DHRNG before communication.

Since both the sensor and the sensor owner only need very few clock cycle to encrypt/decrypt the message, the energy overhead of this protocol is very small compared to the traditional way of public key communication, therefore, especially suitable for the low-power required sensor networks.



(a) (b) (c)

Fig. 2: (a) The distribution of output hamming distance (the error bar shows the distribution of max, 75%, mean, 25% and min), (b) Conditional Probabilities between Output bits O_i and input bits I_j, (c) Conditional Probabilities between Output bits O_i and other output bits O_j.

VII. Conclusion

We have developed techniques for the creation of the first digital hardware random number generator (DHRNG) by randomly assigning the contents of look-up tables on FPGAs and connecting them to form a combinational network. In addition to inherit the intrinsic good properties of traditional HRNGs, the DHRNG is low-power, has small-footprint, and completely digital, as well as being resistant to environmental variations. DHRNG superiorly passes all standard NIST randomness test as well as a list of statistical test, indicating excellent randomness and resistance to a wide range of security attacks. Finally, we have explained how to use DHRNG to achieve public key communication in sensor networks.

References

[1] S. Wei, J. H. Ahnn, M. Potkonjak, "Energy attacks and defence techniques for wireless systems," *WISEC*, pp. 185-194, 2013.

[2] M. Potkonjak, S. Meguerdichian, J.L. Wong, "Trusted Sensors and Remote Sensing", *IEEE Sensors*, pp. 1104-1107, 2010.

[3] S. Meguerdichian, M. Potkonjak, "Security Primitives and Protocols for Ultra Low Power Sensor Systems," *IEEE SENSORS*, pp. 1225-1228, October 2011.

[4] J. B. Wendt, M. Potkonjak, "Nanotechnology-Based Trusted Remote Sensing," *IEEE SENSORS*, pp. 1213-1216, October 2011.

[5] S. Wei, M. Potkonjak, "Wireless security techniques for coordinated manufacturing and on-line hardware trojan detection," *WISEC*, pp. 161-172, April 2012.

[6] F. Koushanfar, M. Potkonjak, "CAD-based Security, Cryptography, and Digital Rights Management", *Design Automation Conference*, pp. 268-269, San Diego, 2007.

[7] S. Wei, S. Meguerdichian, M. Potkonjak, "Gate-level characterization: foundations and hardware security applications", *ACM/IEEE Design Automation Conference*, pp. 222-227, 2010.

[8] James, F., "A review of pseudorandom number generators," *Computer Physics Communications*, vol. 60, pp. 329-344, 1990.

[9] C. W. O'Donnell, G. E. Suh, and S. Devadas, "PUF-based random number generation," *MIT CSAIL CSG Technical Memo 481*, 2004.

[10] J. Zheng, M. Potkonjak, "Securing Netlist-Level FPGA Design through Exploiting Process Variation and Degradation," *FPGA*, pp. 129-139, February 2012.

[11] J. X. Zheng, E. Chen, M. Potkonjak, "A benign hardware Trojan on FPGA-based embedded systems", *IEEE International Conference on Field Programmable Logic and Applications*, pp. 464-470, 2012.

[12] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026-2030, 2002.

[13] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," *ACM Conference on Computer and Communications Security*, pp. 148-160, 2002.

[14] S. Devadas, V. Khandelwal, S. Paral, R. Sowell, E. Suh, T. Ziola, "Design and Implementation of PUF-Based Unclonable RFID ICs for Anti-Counterfeiting and Security Applications," *IEEE RFID*, 2008.

[15] N. Beckmann, M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," *Information Hiding Conference*, pp. 206–220, 2009.

[16] S. Meguerdichian, M. Potkonjak, "Matched public PUF: ultra low energy security platform," *IEEE/ACM ISLPED*, pp. 45–50, 2011.

[17] T. Xu, J. B. Wendt, M. Potkonjak, "Digital Bimodal Function: An Ultra-Low Energy Security Primitive," *ISLPED*, 2013.

[18] M. Potkonjak, S. Meguerdichian, A. Nahapetian, Sheng Wei, "Differential Public, Physically Unclonable Functions: Architecture and Applications", *ACM/IEEE Design Automation Conference*, pp. 242-247, 2011.

[19] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Computers*, vol. 58, no. 9, pp. 1198-1210, 2009.

[20] S. Wei, A. Nahapetian, M. Nelson, F. Koushanfar, M. Potkonjak, "Gate Characterization Using Singular Value Decomposition: Foundations and Applications", Vol. 7, No. 2, pp. 765-773 , *IEEE Transactions on Information Forensics and Security*, 2012.

[21] S. Meguerdichian, M. Potkonjak, "Device Aging-Based Physically Unclonable Functions," *Design Automation Conference*, pp. 288-289, June 2011.

[22] M. A. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Reliability*, vol. 45, pp. 71-81, 2005.

[23] M. Bhargava, C. Cagla, and M. Ken, "Comparison of bi-stable and delay-based Physical Unclonable Functions from measurements in 65nm bulk CMOS," *Custom Integrated Circuits Conference*, IEEE, 2012.

[24] K. Miyaji, T. Suzuki, S. Miyano, and K. Takeuchi, "A 6T SRAM with a carrier-injection scheme to pinpoint and repair fails that achieves 57% faster read and 31% lower read energy," *In Solid-State Circuits Conference Digest of Technical Papers, 2012 IEEE International*, pp. 232-234, IEEE, 2012.

[25] Bucci, Marco, and Raimondo Luzzi, "Design of testable random bit generators," *Cryptographic Hardware and Embedded Systems - CHES 2005*, pp. 147-156, 2005.

[26] B. Barak, R. Shaltiel, E. Tomer, "True random number generators secure in a changing environment," *Cryptographic Hardware and Embedded Systems - CHES 2003*, pp. 166-180, 2003.

[27] "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," National Institute of Standards and Technology (NIST) Special Publication 800-22, Rev. 1a, Apr. 2010.