Title of the Experiment: Hardware-based Security Primitives and Their Applications

Group Members' Names: Sujan Kumar Saha, Pankaj Bhowmik

Date: 10/23/2019

Abstract

In this experiment, we have implemented SRAM based Physical Unclonable Function (PUF), RO based PUF, SRAM based True Random Number generator(TRNG) and RO based TRNG. In the following experiment, two different types PUFs i.e. SRAM PUFs and RO-PUFs are analyzed and mapped on the FPGA of the Hardware Hacking board. As for the SRAM PUF, 64 bytes of the SRAM power-up states are read from the Microcontroller to generate a signature. Then, the signature is sent to the FPGA using the interconnections and extracted afterwards from an in-chip memory. In the second part of the experiment, a RO-PUF is implemented on the FPGA to generate a 128-bit key. The keys are collected at different operating voltages i.e. 2.7V and 3.3V. In the third part, a RO-based TRNG is developed using the RO-PUF modules. As per the instructions, the data are collected from the TRNG at varying voltages and different number stages of RO-based TRNG. All the data are processed afterwards using MATLAB.

Experimental Details

Goals: This experiment has three goals.

1. Implementing two Physical Unclonable Functions (PUF): SRAM-based PUF and RO-based PUF.
2. Implementing two True Random Number Generators using PUFs.
3. Evaluating the PUFs and TRNGs.

Experimental Setup: We used Quartus, ATMEL Studio and Analog discovery to do the experiments. We used Matlab for data processing. Also, we have used NIST statistical tool to evaluate the performance of the TRNGs.

Part I:

Experiment Steps:

1. We have written the microcontroller code using assembly language and load that code to the microcontroller. In this part, we send the SRAM cell data through the PORTD and generate a clock signal using PORTC pin 7. We implemented a delay function to make the clock signal frequency as expected.
2. We programmed the FPGA to read the data from PORTD which is sent by the microcontroller and save the data in a RAM.
3. We also have written a Matlab code to calculate the mean value and hamming distance of the data signatures.

Part II:

Experiment Steps:

1. We implemented the top module in verilog using the given VHDL files. We followed the figure 2 of the given document to implement RO-based PUF. To make placement and routing of the ROs symmetric, we manually find the coordinates and put commands in the tcl command window.
2. The 128 bit key output of the PUF is kept in RAM. We took the results for three different voltages and calculated hamming distance.

Part III:

Experiment Steps:

1. In this part, a SRAM based True Random Number Generator is implemented. We used the same implementation of part I. We took a couple of signatures and found a bit value which changes at different runs. So, the 8 bit value from those SRAM cells can be used as true random number generator.

Part IV:

Experiment Steps:

1. A RO-based TRNG has been implemented with ten 5-stage ROs. We changed the number of stages and number of ROs according to the given conditions. We did the placement of ROs symmetric. Then, we took the 100000 bit number and save it in a RAM.

Part VI:

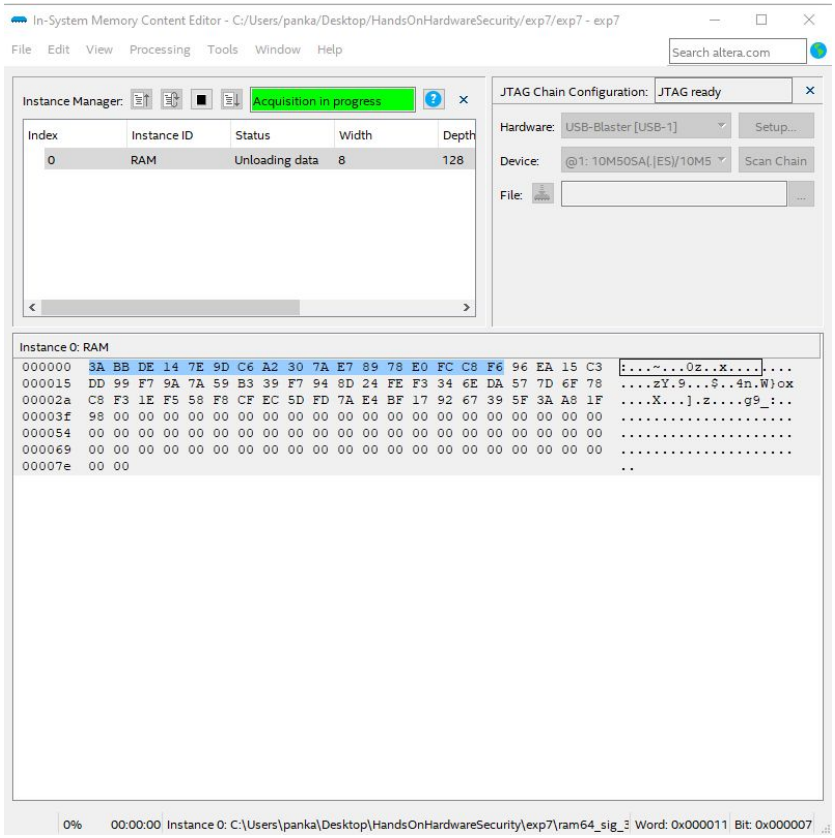
Experiment Steps:

1. We have done the optional follow-up which is randomness evaluation of the RO based TRNG. We downloaded the NIST source code, built it and use the binary bit data file to evaluate randomness of TRNG using 15 test suits.

Results and Observations

Part I: SRAM PUF

1. The size of the SRAM of Atmega16U4 is 1.25KB.
2. The address ranges from $0100 to $05FF for the SRAM. The lower addresses in that range are occupied by our program. We choose address from $0500 to $053F to get 64 byte signature
3. We generate a square wave by using PORTC bit PC7. We made the bit 1, added a delay and then set the bit to 0. This way we generate the square wave which is sent to the FPGA. The frequency of the square wave is 1.08Hz.
4. We have attached the assembly code with the report.
5. We have attached the FPGA code with the report. A screenshot of the In-system Memory content is given below.



6. We exported the signatures in hex file and processed those using Matlab. The S1 has 512 bits. The hex and binary values of S1 is given below.

S1

9 2 6 7 3 9 5 F 3 A A 8 1 F 9 8 3 A B B D E 1 4 7 E 9 D C 6 A 2 3 0 7 A E 7 8 9 7 8 E 0 F C C 8 F 6 9 6 E A 1 5 C 3 D D 9 9 F 7 9 A 7 A 5 9 B 3 3 9 F 7 9 4 8 D 2 4 F E F 3 3 4 6 E D A 5 7 7 D 6 F 7 8 C 8 F 3 1 E F 5 5 8 F 8 C F E C 5 D F D 7 A E 4 B F 1 7	1 0 0 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1

7. 295 bits are 1 and 217 bits are 0. So , 42.38% bits are 0 and 57.61% bits are 1. The mean value is 0.5762
8. We took the S2 and S3 for voltage 3V and 2.7V. The values are given below.

S2

F 7 9 A 7 A 5 9 B 3 3 9 F 7 9 4 8 D 2 4 F E F 3 3 4 6 E D A 5 7 7 D 6 F 7 8 C 8 F 3 1 E F 5 5 8 F 8	1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 1 1 1

C F E C 5 D F D 7 A E 4 B F 1 7 9 2 6 7 3 9 5 F 3 A A 8 1 F 9 8 3 A B B D E 1 4 7 E 9 D C 6 A 2 3 0 7 A E 7 8 9 7 8 E 0 F C C 8 F 6 9 6 E A 1 5 C 3 D D 9 9	0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1

S3

9 4 8 D 2 4 F E F 3 3 4 6 E D A 5 7 7 D 6 F 7 8 C 8 F 3 1 E F 5 5 8 F 8 C F E C 5 D F D 7 A E 4 B F 1 7 9 2 6 7 3 9 5 F 3 A A 8 1 F 9 8 3 A B B D E 1 4 7 E 9 D C 6 A 2 3 0 7 A E 7 8 9 7 8 E 0 F C C 8 F 6 9 6 E A 1 5 C 3 D D 9 9 F 7 9 A 7 A 5 9 B 3 3 9 F 7	1 0 0 1 0 1 0 0 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1

9. No, S2 and S3 are not the same as S1. S2 has 248 different bits compared with S1. So fractional hamming distance is 0.48. S3 has 244 different bits compared with S1. So, fractional hamming distance is 0.47.
10. We have attached the Matlab code for calculating mean value and hamming distance with this report.

Part II: RO-PUF

1. The codes are attached with the report.
2. Here is the K1 which has been taken for voltage value 3.3V.

K1

0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0

3. We took the results for voltage value 3V and 2.7V. The corresponding K2 and K3 are given below.

K2

1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 0 0 1

K3

0 1 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1

4. No, K2 and K3 are not the same as K1. K2 has 61 different bit with compared to K1. The fractional hamming distance is 0.4766. K3 has 65 different bits with compared to K1. The fractional hamming distance is 0.5078. As the hamming distance 0.5 is better, K3 has better hamming distance with K1 compared to K2.

Part III: SRAM as a TRNG

1. We ran the part I several times with same voltage value and got the signature S1 values. We checked the values manually and observed that the value at SRAM address $0504 changes randomly. So, we can use that 8 bit data for TRNG.
2. We send the address data of $0504 to RAM using the same procedure in part I to find the 8-bit random number. The code is added with the report.

Part IV: RO-based TRNG

1. We implemented the RO-based TRNG using ten 5-stage ROs. The 100000 bit numbers are collected for the given cases with voltage 3.3Vand attached with the report.
2. We changed the stages of ROs to 9 and took the results.
3. The number of ROs have been changed to 20 and 30 and results have been taken.
4. The operating voltage has also been changed to 3V and 2.7V and results have been taken.
5. The robustness analysis is given in the table below. We measured the fractional hamming distance compared with the base design of 10 5 stage ROs with voltage 3.3V.

Voltage	**# of ROs**	**# of Stages**	**#of Zeros**	**# of ones**	**Mean**	**Fractional Hamming Distance**
3.3V	10	9	56906	43094	0.43094	0.4912
3.3V	20	5	58772	41228	0.41228	0.4634
3.3V	30	5	57570	42430	0.42430	0.4678
3.0V	10	5	57727	42273	0.42273	0.4899
2.7V	10	5	55113	44887	0.44887	0.4784

Summary and discussion:

We learnt about PUF and TRNG and We faced difficulty to write microcontroller code and placement and routing of ROs to make those symmetric. But we were able to solve those. We also evaluated the robustness of the PUF and TRNG.