Lab 4 Extra Credit

Description:

There are two components to this extra credit. The first is to follow similar steps in Lab 2 with the linux image running on the ARM processor. The second component is collecting the samples of the LFSR and creating code to simulate the NIST run test.

Objective:

Our goal is to perform a run test to check the randomness of our Verilog code. We will refer to the NIST statistical test. To convert to ones and zeros, we will find the median of the numbers, then compare each number to the median. If it is less, it will be zero, otherwise, one. Then we will compute the p-value with the null hypothesis being the numbers are random. Thus, if the p-value is less than the set significance value, we can conclude with confidence that the numbers are not random from the run test.

References:

Please refer to slides 14-18 of Lecture 18: Random Number Generation. Check out nistrng · PyPI for inspiration and its github repo. Refer to NIST SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications section 3.3 on runs test and their theoretical background.

Steps:

1. Follow the beginning slides of Lab 2 to get the linux image in order to run python code. You can run python code on an IDE for now to test your code.

2. Create an array of size 10 and store 10 decimal numbers found from your LFSR simulation. You can also choose to collect from the TRNG binary output and convert those to decimal.
3. Write an algorithm that computes the median of the array and converts each element to 1 or 0 depending if it is less or more than the median.
4. Write an algorithm to calculate the observed runs (hint: if next element is different, new run)
5. Implement the p-value calculation based on the statistical knowledge provided in the NIST paper section 3.3 (There is a math.erfc() function).
6. Compare the p-value to the significance value of 0.01. If it is more, print “Random”, if it is less, print “Not Random”.

GOOD LUCK!