Research Journal for Preston Maness during Senior Design I: A Hardware Random Number Generator

Preston Maness

Abstract—**Just a dummy abstract here. Probably won't actually use an abstract for the journal.**

CONTENTS

I. Sunday September 15 2013

Was able to get dieharder compiled and running. Can feed in both /dev/urandom and a binary file of random, unsigned, 32-bit int's generated from Perl's rand() function. While /dev/urandom is passing most tests –it is marked WEAK in others– it seems that generating even a million random ints is not sufficient for dieharder. It will generate output stating that it "rewound" the file anywhere from tens of times to hundreds of times. Of course when it rewinds the file it is no longer "random" and so the tests fail. Good to know.

As well, it is becoming apparent that I will need to develop a rigourous statistical understanding of randomness. I should also familiarize myself with the GNU Scientific Library, as dieharder integrates tightly with it.

Regardless, I have the RNG stress-tester up and running. Now I need to focus on perhaps making a randomness bitstream a la /dev/customRandom, as our RNG will ultimately be speaking over a serial line and into the host that will then put the raw bits here.

II. Thursday September 19 2013

Found an outstanding link on random noise generation using avalanche breakdown. He even utilized ngspice in the simulation process! This should prove to be an outstanding guidepost. Judging by his work, there's a good chance that avalanche noise might be the best source to work with if construction by hand is a requirement.

http://holdenc.altervista.org/avalanche/

Still reading. It looks like he's covered all the fundamental bases I wanted to cover. If his work proves to speed up mine considerably, I should consider investigating parallelizing these designs and having a microcontroller for de-skewing and de-biasing this semester, rather than second semester.

Going to test his netlist on my installation of ngspice now. Getting similar results. That's good.

However, looking at his schematic it looks like he's throwing enough voltage at the 2N3904's to kill them over time, though I could be reading it wrong. I suspect you'd need some heat sinking to keep them cool enough, but he's not exactly throwing a lot of current at them either.

When the current is changed, Vazzana states that "the noise 'pattern' seems to change on the oscilloscope." Will need to investigate this fully.

http://http.developer.nvidia.com/GPUGems3/gpugems3_ch37.html

The NVIDIA link has a nice list of references I should look into.

III. September 23 2013

Been thinking about how to get this rapidly prototyped. The end of September is a lot closer than it seems. I don't want to fall behind schedule. For now, I'm looking into having an externally powered RNG that outputs its serial bitstream to something like a Rasberry Pi or an Arduino UNO. I own both of these pieces of hardware. I'm curious to see if I can accomplish debiasing and decorrelating with either of these bits of kit. I can then push out the purified bitstream over USB. Both of these devices have libraries available to take an input bitstream and send out a modified stream to a host via USB.

https://github.com/infomaniac50/Random

It looks like infomaniac50 is several steps ahead of me. Sidenote: It seems everyone is several steps ahead of me. I should think about how I can distinguish my work. The availability of the work of others means I should be able to get up and prototyping within the week. How can I take advantage of the additional time to produce a unique addition to this body of work? I'm leaning toward using several RNGs in parallel and then using a suitably fast uC to manipulate it into a single, high-speed bitstream over USB 2.0.

Anyway... infomaniac50's got an arduino library that takes the input from an RNG and debiases it. It should be simple enough to extend this library and have it output the bitstream over USB to a host. As well, it's licensed under the CC-AT-SA 3.0 Unported license. I don't know if this is compatible off the top of my head; my understanding is that CC licenses aren't really meant to cover code.

After that, it's just a matter of familiarizing myself with the internals of the Linux kernel to get a bitstream from a /dev/USB device into /dev/urandom or some other custom entropy pool (like /dev/myCustomEntropyPool).

Having some pretty graphs that show how much entropy is in this pool at any given time would be nice. Have a graph that shows how much entropy is used up during entropy-intensive processes. Ideally, we shouldn't be losing much entropy –that is to say, the hardware RNG should be able to keep up. In reality, I suspect my first Marks to not be able to keep up with dieharder's entropy demands.

A. *Later in the day inbetween classes...*

The title for this paper seems appropriate enough: http://dasgupab.faculty.udmercy.edu/Dasgupta-JSfinal.pdf "Mathematical Foundations of Randomness." Good source.

IV. Monday November 4 2013

Who am I kidding. October flew by with hardly any work getting done. The other four courses have been eating up all my time. However, I'm doing well in them and plan on shifting gears for a little bit. I can afford to let a few things slide. But really, I need to get a physical prototype up and running ASAP. I've been running through simulations, and thought of a rather interesting chicken and egg issue:

If I'm building a hardware RNG, then how can I trust any simulation designed to test such an RNG without having a hardware RNG providing the noise to the simulator? How does the simulator handle noise at all? Going to add this to the research log. I think this is a good example of how simulation only goes so far.

Ok. Now it's officially Monday. I've got the analog noise floor up and amplified to 500 mV, with what looks at least in passing to be random oscillation. Great! See the "analog_only.sch" using gschem or view the net at "analog_only.net". It should run fine in ngspice.

This serves as my initial verification of the work done by Vazzana.

Useful links follow:

- http://www-mdp.eng.cam.ac.uk/web/CD/engapps/geda/geda-doc/spice-sdb/netlist.html
- http://www.brorson.com/gEDA/SPICE/x496.html
- http://web.jfet.org/hw-rng.html

V. Tuesday March 04 2014

It's been quite a while since I updated this research log. Much of my work has been logged in the "weekly-reports" section of the repository. In any case, I'm running into problems with what I'm assuming is a biased bitstream.

The following paper appears VERY relevant to my interests –that is, it addresses many methods of debiasing/decorrelating bitstreams– and the link is below:

http://www1.spms.ntu.edu.sg/~kkhoongm/Entropy.pdf

VI. Initial Research - What I've Found

BLAH blah blah... List of sources below. Going to keep adding text in here so that the itemize list doesn't get shoved up into the fancy 'B' character of "Blah."

- [1] Evaluating a TRNG in hardware.
- [2] Noise resistant TRNG. Stochastic model for parameter choicing.
- [3] Importance of RNG choice in GIS applications.
- [4] Ring-based RNG. Huh? What's that?
- [5] Investigating LFSR, LCG, and Blum Blum Shub on Xilinx FPGA
- [6] Non-Uniform RNG, Statistics of.
- [7] "Data-oriented" RNG? Not sure what this is about, but it mentions making distributions of random numbers with different characteristics (uniform, chi-squared, etc)
- [8] GPU Accelerated Scalable Parallel RNG.

References

[1] M. Soucarros, J. Clediere, C. Dumas, and P. Elbaz-Vincent, "Fault analysis and evaluation of a true random number generator embedded in a processor." *JOURNAL OF ELECTRONIC TESTING-THEORY AND APPLICATIONS*, vol. 29, no. 3, pp. 367 – 381, n.d. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000321520700011&site=eds-live&scope=site

[2] T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "A worst-case-aware design methodology for noise-tolerant oscillator-based true random number generator with stochastic behavior modeling." *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, vol. 8, no. 8, pp. 1331 – 1342, n.d. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000322026900007&site=eds-live&scope=site

[3] s. Barry, Simon C.1, "How much impact does the choice of a random number generator really have?." *International Journal of Geographical Information Science*, vol. 25, no. 4, pp. 523 – 530, 2011. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=60827948&site=eds-live&scope=site

[4] M. Ayat, Mehdi1, m. Atani, Reza Ebrahimi2, and r. Mirzakuchaki, Sattar1, "On design of puf-based random number generators." *International Journal of Multimedia & Its Applications*, vol. 3, no. 3, pp. 30 – 40, 2011. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=61025496&site=eds-live&scope=site

[5] j. Kumar, Jay1, s. Shukla, Sudhanshu1, e. Prakash, Dhiraj1, p. Mishra, Pratyush1, and s. Kumar, Sudhir1, "Random number generator using various techniques through vhdl." *International Journal of Computer Applications in Engineering Sciences*, vol. 1, no. 2, pp. 127 – 129, 2011. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=82881326&site=eds-live&scope=site

[6] s.-k. de Schryver, Christian1, D. Schmidt, N. Wehn, E. Korn, H. Marxen, A. Kostiuk, and R. Korn, "A hardware efficient random number generator for nonuniform distributions with arbitrary precision." *International Journal of Reconfigurable Computing*, pp. 1 – 11, 2012. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=87045573&site=eds-live&scope=site

[7] r. Farjami Nezhad, Rasoul1, e. Effatparvar, Mehdi2, and m. Rahimzadeh, Mohammad3, "Designing a universal data-oriented random number generator." *International Journal of Modern Education & Computer Science*, vol. 5, no. 2, pp. 19 – 24, 2013. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=87626745&site=eds-live&scope=site

[8] S. Gao and G. Peterson, "Gasprng: Gpu accelerated scalable parallel random number generator library." *COMPUTER PHYSICS COMMUNICATIONS*, vol. 184, no. 4, pp. 1241 – 1249, n.d. [Online]. Available: http://libproxy.txstate.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edswsc&AN=000315974100018&site=eds-live&scope=site