A 2.92μW Hardware Random Number Generator

Jeremy Holleman and Brian Otis
Department of Electrical Engineering
University of Washington
Seattle, Washington, 98195–2500
Email: {hollemj,botis}@ee.washington.edu

Seth Bridges, Ania Mitros, Chris Diorio
Department of Computer Science and Engineering
University of Washington
Seattle, Washington, 98195–2500
Email: {seth, ania, diorio}@ee.washington.edu

***Abstract*— This paper presents two novel hardware random number generators (RNGs) based on latch metastability. We designed the first, the DC-nulling RNG, for extremely low power operation. The second, the FIR-based RNG, uses a predictive whitening filter to remove non-random components from the generated bit sequence. In both designs, the use of floating-gate memory cells allows us to predict and compensate for DC offsets and other non-random influences while minimizing power consumption. We also present a simple post-processing technique for improving randomness. We fabricated both RNGs in a standard 2P4M 0.35μm CMOS process. The DC-nulling RNG utilized .031 mm^2 of die area, while the FIR-based RNG occupied 1.49 mm^2.**

I. Introduction

True random number generators (TRNGs) are required for many applications including RFID tag initialization, cryptography, and seeding pseudo-random number generators (PRNGs). TRNGs generate random sequences from physical phenomena such as nuclear decay, thermal noise, or cosmic radiation, with thermal noise being the most common entropy source for integrated TRNGs. Our RNGs use a latch to sample and amplify thermal noise originating in the transistors of the latch itself and use floating-gate analog memory cells to compensate for the latch DC offset and other corrupting non-random sources. For a good overview of TRNG architectures, see [1], [2].

Thermal noise is a good source of randomness because of its uniform spectrum, but it is easily corrupted by non-random influences including supply noise, $1/f$ noise, interference, and DC offsets caused by device mismatch. Both of our designs apply an adaptive, predictive signal to the sampling circuit to remove non-random components and improve the randomness of the resulting sequence.

II. Design

In this section, we describe our two RNG systems, and the design of their sub-components. The DC-nulling RNG uses a running estimate of the median system noise as the prediction. The FIR-based RNG uses a linear-prediction filter to remove correlations between bits separated by short time intervals.

A. Evaluation Latch

At the core of both RNGs is the evaluation latch, shown in Fig. 1(a), comprising a pair of cross-coupled, current-starved NAND gates. During the reset phase, when CLK is low, both latch outputs, BIT and $\overline{BIT}$, reset to V_{DD}. During the evaluation phase, CLK is high, and positive feedback will drive one output to 0 and the other to 1. The predictive signal V_{Pred} is applied to one of the current-limiting nFETs, and a reference voltage is applied to the other. An XOR gate compares the two latch outputs to verify that initial metastability has been resolved. Due to the relatively slow clock frequency ($\leq$ 100kHz), the probability of a metastable state lasting throughout half of a clock period is exceedingly low. We examined the XOR output for 100 million bits at 100kHz and found no unresolved metastable events.

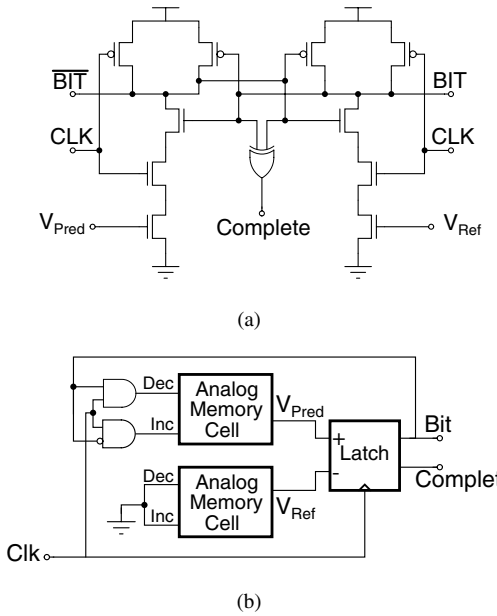


Fig. 1. (a) The evaluation latch used in both RNGs. (b) The DC-nulling RNG.

B. Memory Cell

To learn and store the characteristics of the system noise used for prediction, we use a floating-gate memory cell [3] to provide non-volatile analog storage. Values are stored as charge on a floating gate and adjusted using Fowler-Nordheim tunneling and hot electron injection. The tunneling and injection rates can be independently adjusted, ensuring that the voltage change produced by a tunneling pulse and by an

injection pulse are equal.

C. DC-nulling RNG

In the DC-nulling RNG, shown in Fig. 1(b), we adjust V_{Ref} to the desired common-mode level before operation. V_{Pred} is incremented or decremented by a fixed amount μ whenever a 0 or 1 is generated, respectively. Thus the corrective signal converges to the point where the bits are uniformly distributed. Adjustments to the predictive signal take place during the evaluation phase, so the reset phase is available for the associated transients to settle. Previous latch-based RNGs (e.g. [4]) have used switched-capacitor circuits to cancel DC offsets. The floating-gate memory cell is more power efficient because it avoids the slewing current required by switched-capacitor circuits. It also enables extremely low adaptation rates; faster adaptation can filter out too much low-frequency noise, spectrally coloring the bit sequence.

D. FIR-based RNG

The FIR-based RNG, shown in Fig. 2(a), uses a linear-prediction filter to remove correlated components of the total system noise. We predict the system noise based on previous bits and learned knowledge of the noise correlation structure. Previous bits are stored in a shift register such that at time t, the RNG output from time $t-k$ is stored in tap k. We use floating-gate memory cells to calculate and store correlation information. The weights are adjusted at every clock cycle according to the instantaneous autocorrelations at the corresponding lags:

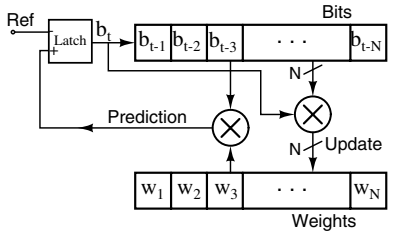
$$w_0(t+1) = w_0(t) + b(t)\mu \quad (1)$$

$$w_k(t+1) = w_k(t) + b(t-k)b(t)\mu, \quad (2)$$

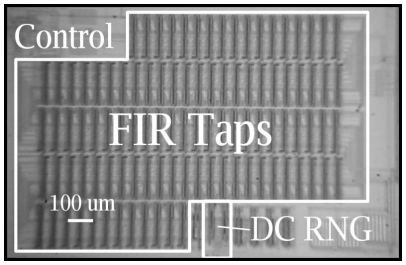
where $w_k(t)$ is the k^{th} tap weight at time t, and $b(t)$ is the RNG output at time t. One tap weight (w_0) is adjusted based on the current bit alone to account for DC bias. As long as the adaptation rates are symmetric (i.e. the magnitude of increments and decrements are equal), the weight for a given tap will converge to the value that causes the autocorrelation at the corresponding lag to be zero. Because we interpret the bits as ± 1, we perform the 1-bit multiplication $b(t-k)b(t)$ with an XNOR gate, the output of which determines whether the memory cell should be incremented or decremented. Tap weights w_{1-N} are multiplied by $\pm$ 1 using a cross-bar switch controlled by the bit stored in the respective tap, and the outputs of all taps are summed to form the prediction signal. The result is that the prediction signal is the inner product of the N tap weights with the past N bits plus a term to remove constant (or very slowly changing) bias.

$$V_{Pred}(t) = w_0 + \sum_{k=1}^{N} w_k(t)b(t-k) \quad (3)$$

The prediction signal is applied to the latch input V_{Pred} to remove the predictable component of the system noise. We implemented the summation using the circuit shown in Fig. 3(a), which is a well-known summation circuit, modified



(a)



(b)

Fig. 2. (a) Symbolic model of the FIR-based RNG. (b) Photograph of the RNG.

to use capacitors, instead of resistors. Both inputs of the op-amp are floating gate pFETs, with a DC level set by tunneling and injection. The use of floating gates on the input nodes allows the summing circuit to operate at arbitrarily low frequencies while using purely capacitive feedback.

At the input, we split the clock signal into two clocks, separated by 90°. One clock controls evaluation of the RNG, while the other clocks the shift register for the FIR, an arrangement which prevents supply noise from the shift register from interfering with evaluation of the RNG.

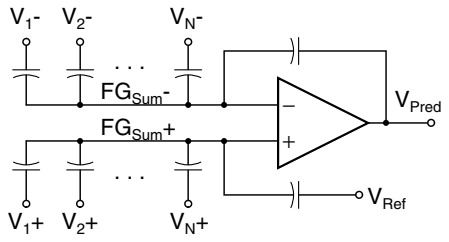
E. Post-processor

We post-processed the output of both RNGs to remove residual bias. Our post-processor uses minimal computational resources and only reduces the output rate by a constant factor of two. It comprises two non-linear feedback shift registers (NFSRs) that are modified to accept an input from the hardware RNG and combined in an alternating-step arrangement [5]. The modified NFSR, shown in Fig. 4(a), comprises a shift register and logic for computing a feedback bit. The feedback bit is computed as

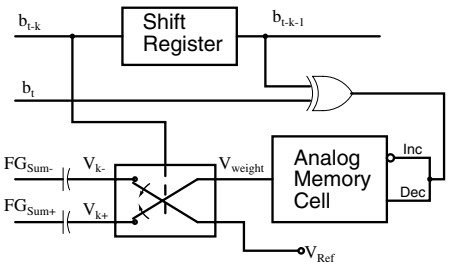
$$f[n] = \left(\sum_{i=1}^{L} c_i f[n-i] + \prod_{i=1}^{L} f[n-i] + \prod_{i=1}^{L} \bar{f}[n-i] + x[n] \right),$$

where $x[n]$ is the input at time n, taken from the physical RNG; c_i is the i^{th} coefficient of the feedback polynomial, chosen to be primitive [5]; L is the length of the shift register; and addition is defined as modulo-2 addition. The output of the NFSR is taken at the end of the shift register, $f[n-L]$.

The alternating step generator is shown in Fig. 4(b). Unprocessed bits are taken in pairs, with the first bit in each pair selecting one of the two NFSRs, and the second added to the

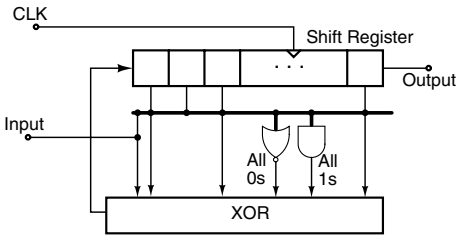


(a)

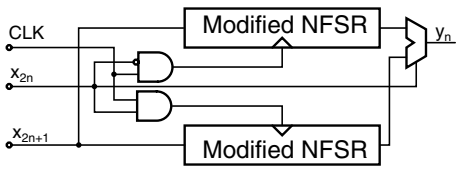


(b)

Fig. 3. (a) The capacitively coupled summation circuit. (b) A single tap of the FIR filter.



(a)



(b)

Fig. 4. (a) The modified NFSR. (b) Alternating-step post-processor.

feedback polynomial of the selected NFSR. The output of the selected NFSR is the final output. While conventional LFSRs have a periodic output, the output of our NFSR-based post-processor is not periodic, because the state of the NFSR is constantly perturbed by the the output of the hardware RNG. We implemented the post-processor off-chip, but we included its power consumption, based on Spectre^{TM} simulations, in our results. Using FSRs of length 15 and 17, the post-processor consumes 19 nW/kHz, with a supply of 5V, using the same process.

Existing post-processing techniques include parity/XOR, the von Neumann corrector [6], and hash functions [5]. Parity post-processors require large decimation ratios to effectively

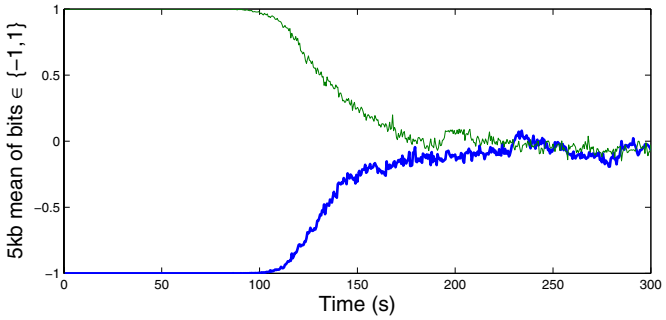


Fig. 5. The DC-nulling RNG converges to a near-uniform distribution from initial conditions that caused the output to be all 1s or all 0s.

remove bias. The von Neumann corrector discards pairs of equal bits and keeps the first bit of unequal pairs. In [6] the von Neumann corrector reduced the output rate by an average factor of 6, but the output rate is dependent on the input. Post-processors using hash functions can have arbitrary decimation ratios, but the computational requirements are relatively high.

III. Results

We fabricated both RNGs in a 0.35µm, four-metal, double-poly process available through MOSIS. The DC-nulling RNG consumes 0.031mm^2 of die area and can be fabricated in a standard digital process, while the FIR-based RNG occupies 1.49mm^2 and uses double-poly capacitors in the differential summing circuit. Both RNGs run on a single 5V supply.

To test the convergence of the DC-nulling RNG, we programmed the memory cells to incorrect values, so that the RNG would output all 1s or all 0s. We then enabled adaptation and allowed the RNG to converge. The results are displayed in Fig. 5. In Fig. 6(a), the plot of means of 100kb blocks, taken over 24 hours, with and without adaptation enabled on the DC-nulling RNG, show that the DC-nulling adaptation works well to keep the bit distribution even. There is some residual variation in the distribution due to the temperature dependence of the tunneling and injection processes.

Fig. 6(b) illustrates the effectiveness of the FIR-based RNG at removing correlations. The top trace shows the correlations present in bits from the DC-nulling RNG due to the influence of $1/f$ noise and a slight alternating tendency, arising from charge kick-back into the finite source impedance of V_{Pred} and V_{Ref}. In the next two traces the FIR taps were externally programmed to the correct values, because of asymmetry in the adapation rates. Current work includes development of a memory cell with automated calibration of the adaptation rates. The second trace show that the 25-tap FIR removed auto-correlations for lags up to 25, but there is a small positive correlation at lags of 42 and 90, resulting from a layout artifact at taps 42 and 90. The third trace shows that the 100-tap FIR compensated for all three effects.

We tested the processed bits using the NIST test suite [7] and the proportion of sequences passing each test is shown in Table I. Every set of sequences passed every test at a rate exceeding the minimum pass rate, shown in the far right

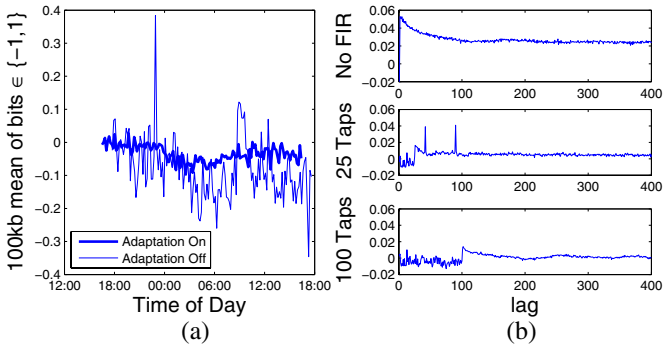


Fig. 6. (a) Bit distribution variation over 24 hours with and without adaptation. (b) Auto-correlation sequences for bits taken with the DC-nulling RNG, and the FIR-based RNG with 25 and 100 taps enabled.

column. Bit rate refers to bits produced per second after post-processing. All tests were performed on ten one-million-bit sequences, using the 1% significance level recommended by NIST.

Of all TRNGs we have found that report power dissipation and pass the complete NIST test suite, our DC-nulling RNG has the lowest energy consumption per bit. Table II compares our TRNGs to other integrated TRNGs. The RNG in [2] uses off-chip capacitors to implement a VCO with an FPGA. We estimated their power consumption based on their reported FPGA usage and device specifications. The RNG in [8] presented results for the FIPS tests [9], but not for the more rigorous NIST test suite.

IV. Conclusions

We presented two true random number generators with statistically validated randomness based on a novel bias compensation scheme using analog floating-gate memory cells. We also described a novel post-processor suitable for compact VLSI implementation. Their low power consumption makes our RNGs attractive for embedded applications with severe power constraints and modest bit rate requirements.

TABLE I

Results of NIST Tests

RNG Type	FIR	DC	DC	DC	Min. Pass Rate
Bit Rate	50k	0.5k	2.5k	5k	
Adaptation	Off	On	On	On	
Taps Enabled	25	—	—	—	
Unprocessed Bits					
Mean	0.0172	0.0283	0.0204	0.0140	–
Max. Corr.	-0.097	-0.077	0.0475	0.0579	–
Processed Bits					
Test Name	Pass Rates				
Frequency	1.0000	1.0000	1.0000	1.0000	0.896
Block-Frequency	1.0000	1.0000	1.0000	1.0000	0.896
Cusum	1.0000	1.0000	1.0000	1.0000	0.923
Runs	1.0000	0.9000	0.9000	0.9000	0.896
Long-Run	1.0000	0.9000	1.0000	1.0000	0.896
Rank	1.0000	1.0000	1.0000	1.0000	0.896
FFT	1.0000	1.0000	1.0000	1.0000	0.896
Aperiodic-Template	0.9892	0.9878	0.9878	0.9939	0.982
Periodic-Template	1.0000	1.0000	1.0000	1.0000	0.896
Universal	1.0000	0.9000	0.9000	1.0000	0.896
Apen	1.0000	1.0000	1.0000	1.0000	0.896
Random-Excursion	1.0000	1.0000	1.0000	1.0000	0.929
Random-Excursion-V	1.0000	1.0000	1.0000	0.9778	0.949
Serial	1.0000	1.0000	1.0000	1.0000	0.923
Lempel-Ziv	1.0000	1.0000	1.0000	0.9000	0.896
Linear-Complexity	1.0000	1.0000	1.0000	1.0000	0.896
$I_{DD,RNG}(\mu A)$	35.87	0.58	1.16	1.86	
$I_{DD,Post}(nA)$	197	1.97	9.85	19.7	
Power(μW)	180.3	2.92	5.87	9.39	
nJ/bit	3.61	5.85	2.35	1.88	

TABLE II

Comparison of TRNGs

RNG	Bit Rate	Power	Passed all NIST Tests	Joules/bit
Tsoi [2]	29kb/s	52.8mW	Yes	1.8μ
Petrie [1]	1.4Mb/s	3.9mW	Yes	2.78n
Bucci [8]	10Mb/s	2.3mW	No	.23n
Brederlow [10]	200kb/s	50μW	No	.25n
This work (DC)	500 b/s	2.92μW	Yes	5.85n
This work (DC)	5kb/s	9.39μW	Yes	1.88n
This work (FIR)	50kb/s	180μW	Yes	3.61n

References

[1] C. Petrie and J. Connelly, "A noise-based IC random number generator for applications in cryptography," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 47, pp. 615–621, May 2000.

[2] K. Tsoi, K. Leung, and P. Leong, "Compact FPGA-based true and pseudo random number generators," in *Field-Programmable Custom Computing Machines, 11th Annual IEEE Symposium on*, 2003, pp. 51–61.

[3] M. Figueroa, S. Bridges, and C. Diorio, "On-chip compensation of device-mismatch effects in analog VLSI neural networks," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 441–448.

[4] D. Kinniment and E. Chester, "Design of an on-chip random number generator using metastability," in *Proceedings of the 28th European Solid-State Circuits Conference.*, 2002, pp. 595–598.

[5] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC, 1997.

[6] B. Jun and P. Kocher, "The Intel random number generator," Intel, Tech. Rep., 1999. [Online]. Available: http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf

[7] A. Ruhkin *et al.*, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[8] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo, "A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 403–409, April 2003.

[9] *Security Requirements for Cryptographic Modules*, National Institute of Standards and Technology Std. FIPS 140-1, 1994.

[10] R. Brederlow, R. Prakash, C. Paulus, and R. Thewes, "A low-power true random number generator using random telegraph noise of single oxide-traps," in *International Solid-State Circuits Conference Digest of Technical Papers*, Jan. 2006, pp. 422–423.