State of the Art in True Random Number Generation

Berk Sunar
sunar@wpi.edu
http://crypto.wpi.edu

Worcester Polytechnic Institute
Department of Electrical & Computer Engineering
Worcester, Massachusetts 01609, USA
Visting Scholar, Ruhr-Universität Bochum, Germany

Presented at UCLA IPAM

Components of TRNGs

- **Entropy Source:** Thermal and shot noise in circuits, brownian motion, nuclear decay, quantum phenomena. Determines the available entropy.
- **Harvesting Mechanism:** The entropy source is tapped using a harvesting mechanism that does not disturb the randomness and collects as much entropy as possible. The harvesting mechanism should come with a rigorous justification with the most basic assumption of the source explicitly stated.
- **Post-Processing:** A post processor masks imperfections in the entropy source or harvesting mechanism, or may provide resistance to environmental changes and tampering, e.g. a von Neuman corrector [IntelRNG], extractor functions [PA-BST03], SHA-1 [IntelRNG].

Testing TRNGs

- Typically tests designed for PRNGs, e.g. DIEHARD Battery of tests, [M-96] or NIST Test Suites [N-00] are used.
- The sampling frequency is decreased until the output sequence starts to pass the NIST or DIEHARD tests.
- Alternatively, cryptographic postprocessing techniques such as hashing are used.
- Recently, the German BSI [AIS]and later NIST made some efforts to design tests specifically for TRNGs
- Serious Paradigm Shift: The designers should also provide a testing methodology.
- However, AIS does not propose concrete means for testing.
- Killman & Schindler surveyed several techniques in [KS01][SK-02]: tot, startup, online testing.

Baggini and Bucci

The design introduced in [BB-99] uses a combination of analog and digital components for amplification and sampling of white noise.

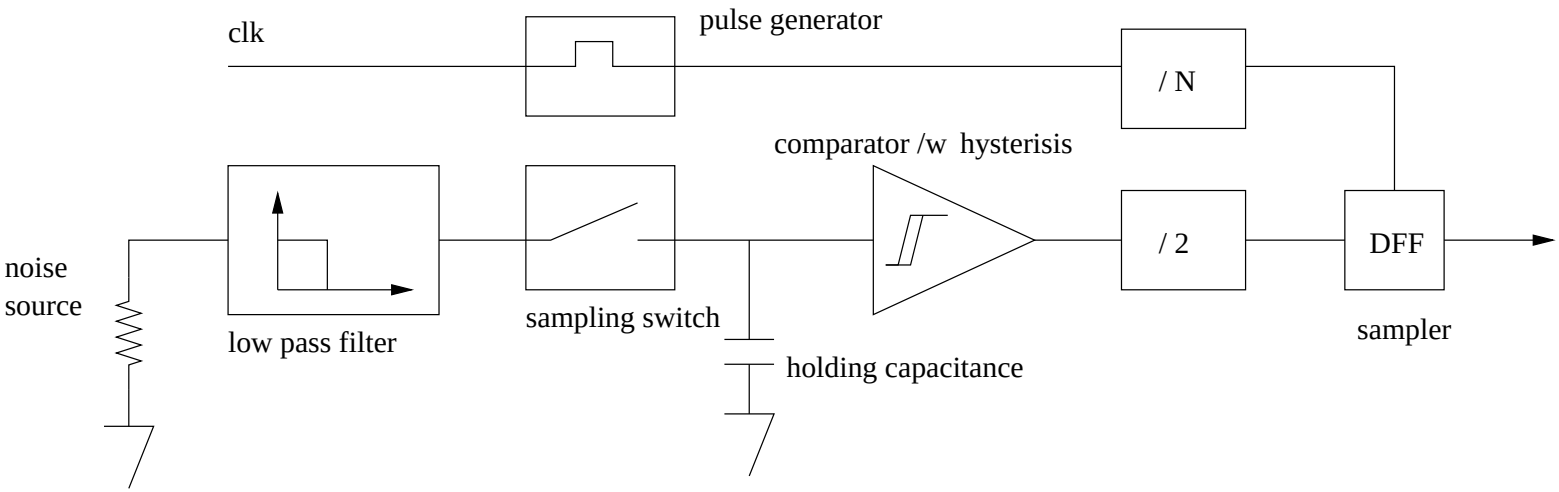


Image reproduced from [BB-99]

The Intel TRNG

Intel Corp. [IntelRNG], where the thermal noise on a junction is amplified and used to drive a voltage controlled oscillator which is sampled by another oscillator. The output sequence is postprocessed using the von Neumann corrector and then hashed using SHA-1.

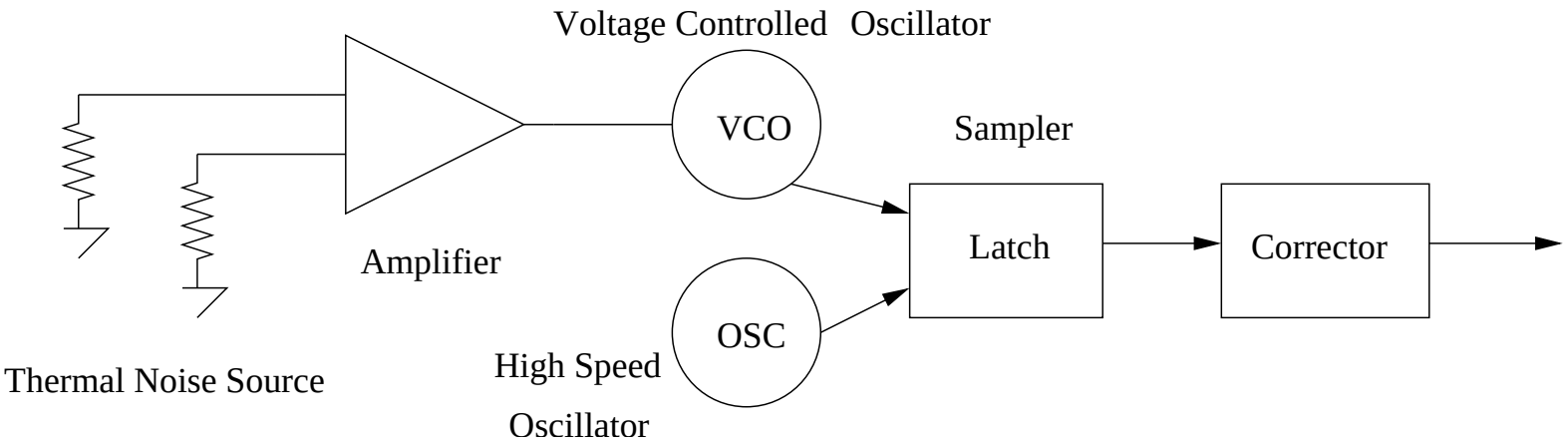


Image reproduced from [IntelRNG]

Fischer and Drutarovský

The design in [FD-03] samples the jitter in a phase locked loop (PLL) – an analogue component – on a specialized reconfigurable logic platform.

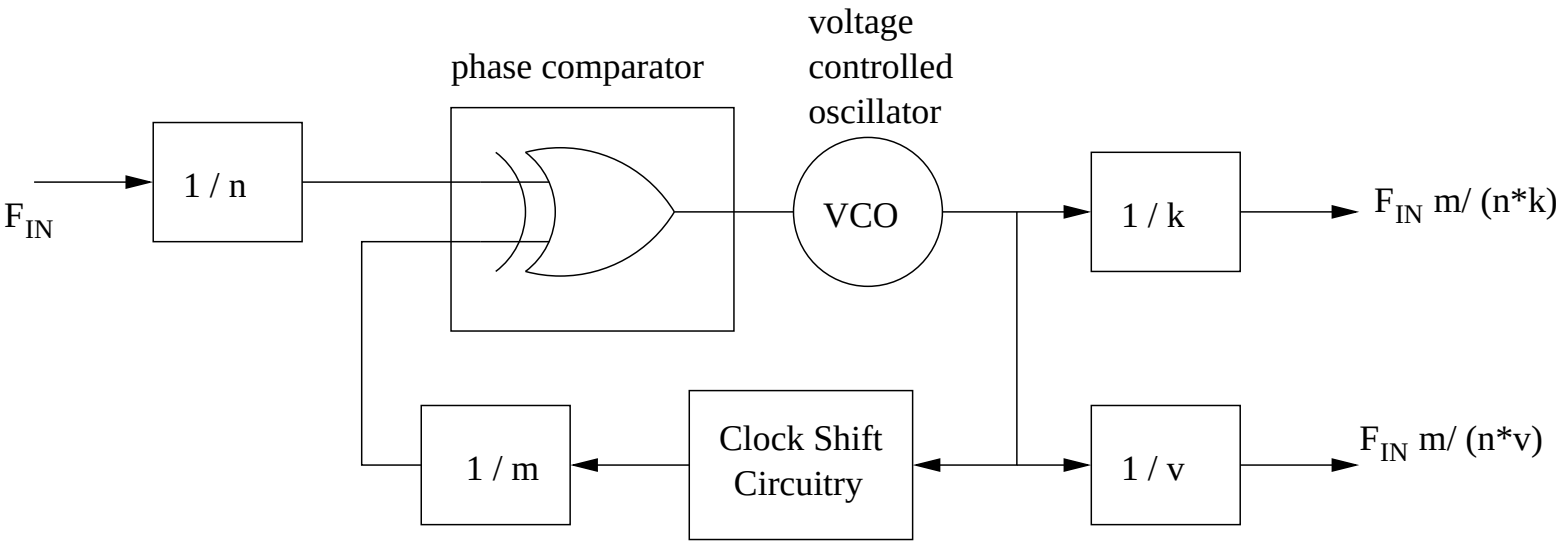


Image reproduced from [FD-03]

Tkacik

The innovative design introduced in [T-03] randomly samples the output of an LFSR and a cellular automata. The randomness comes from the jitter in the two oscillator circuits which are used to clock the two deterministic circuits.

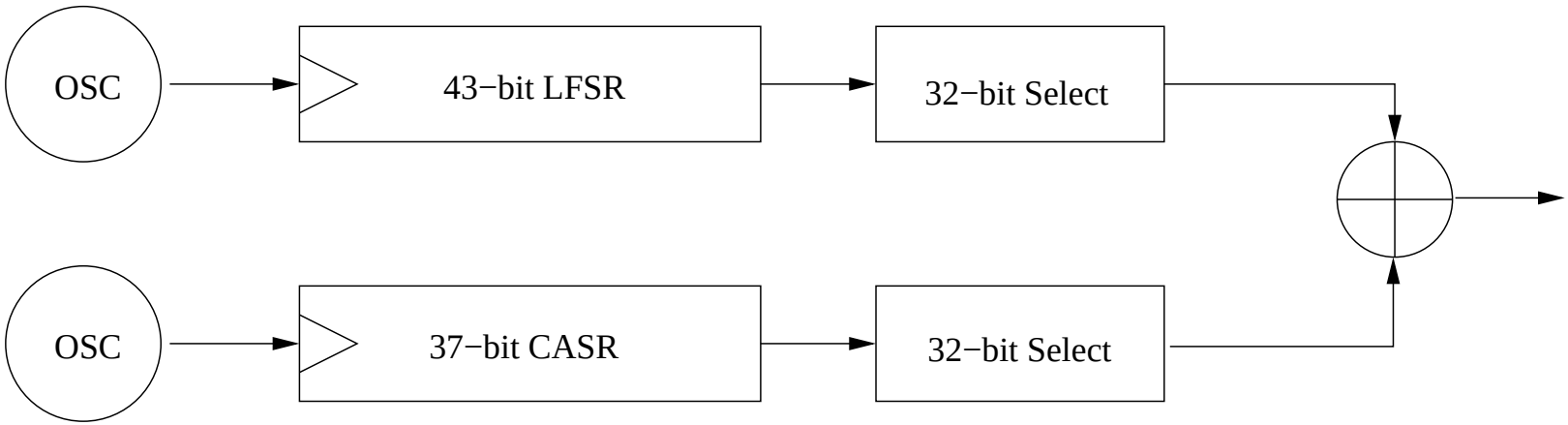


Image reproduced from [T-03]

Epstein et al.

[EHKRZ-03], a simple architecture based on metastable circuits is proposed. The design passes the statistical tests only when a large number of such circuits are combined.

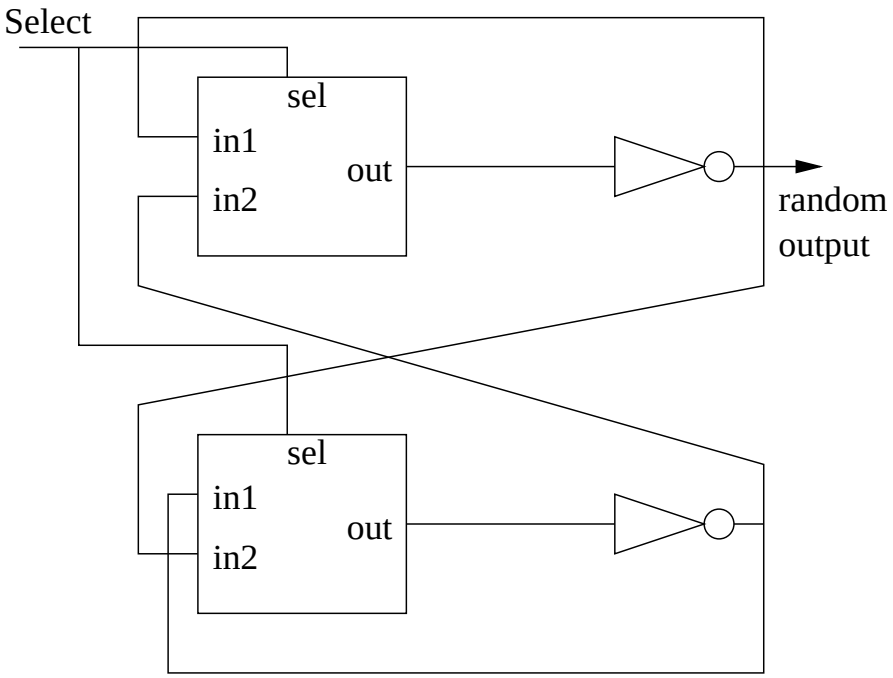


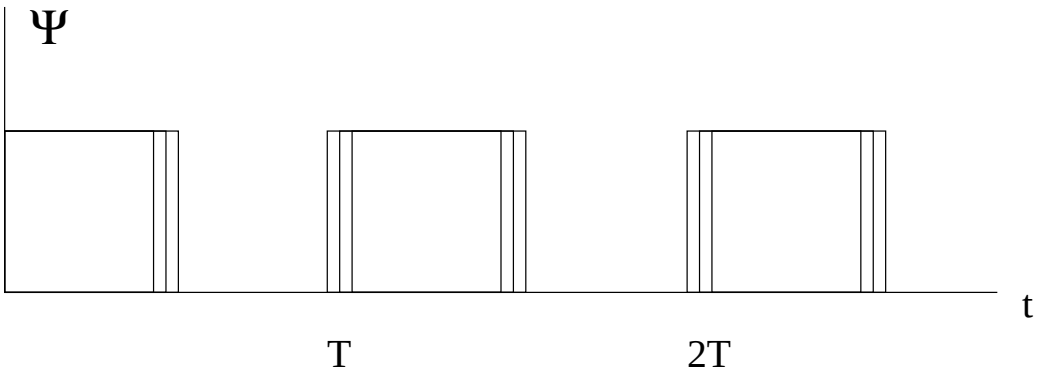
Image reproduced from [EHKRZ-03]

Wishlist

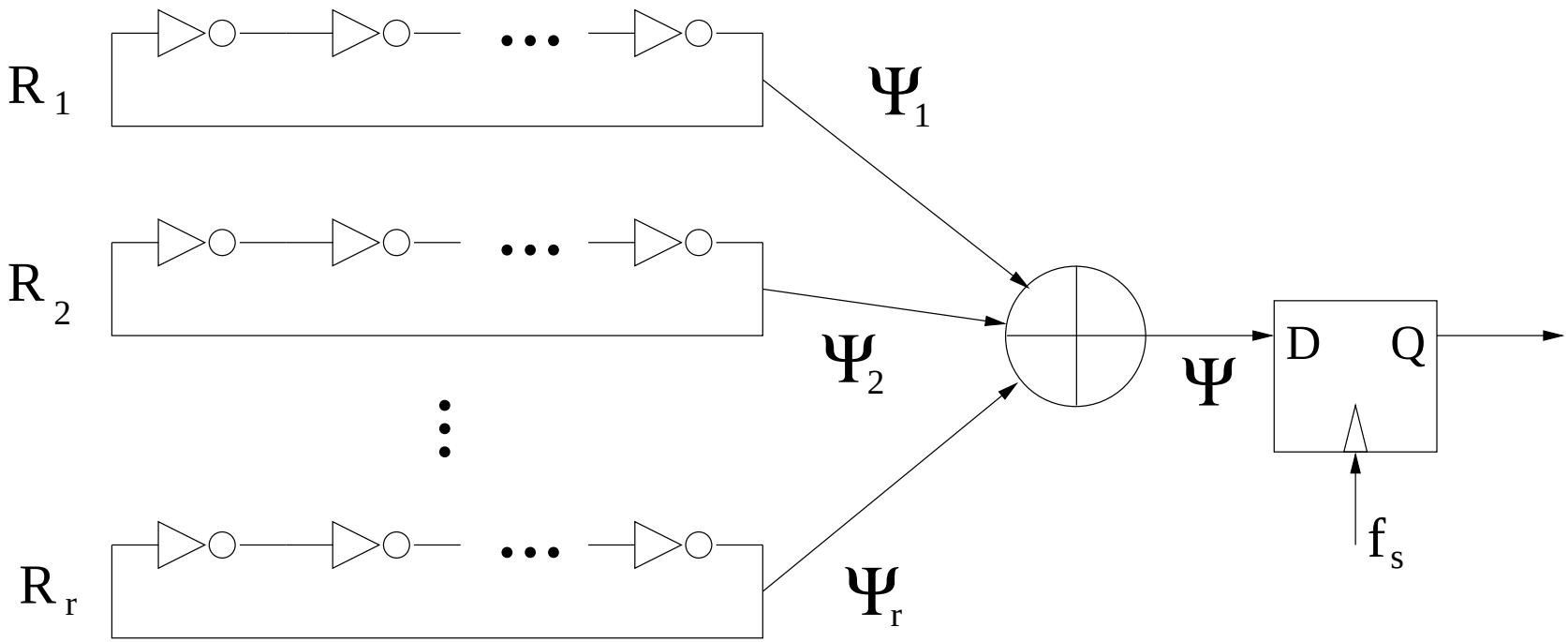
- The TRNG is the root of trust in cryptographic architectures.
- Extremely difficult to test proposed designs.
- Design must be sufficiently simple to allow analysis.
- Must come with security proof (passing DIEHARD or NIST tests is not sufficient)
- No complicated post processing
- Preferably digital (no analog components)

A Fault Resilient True Random Number Generator [Sunar,Martin,Stinson]

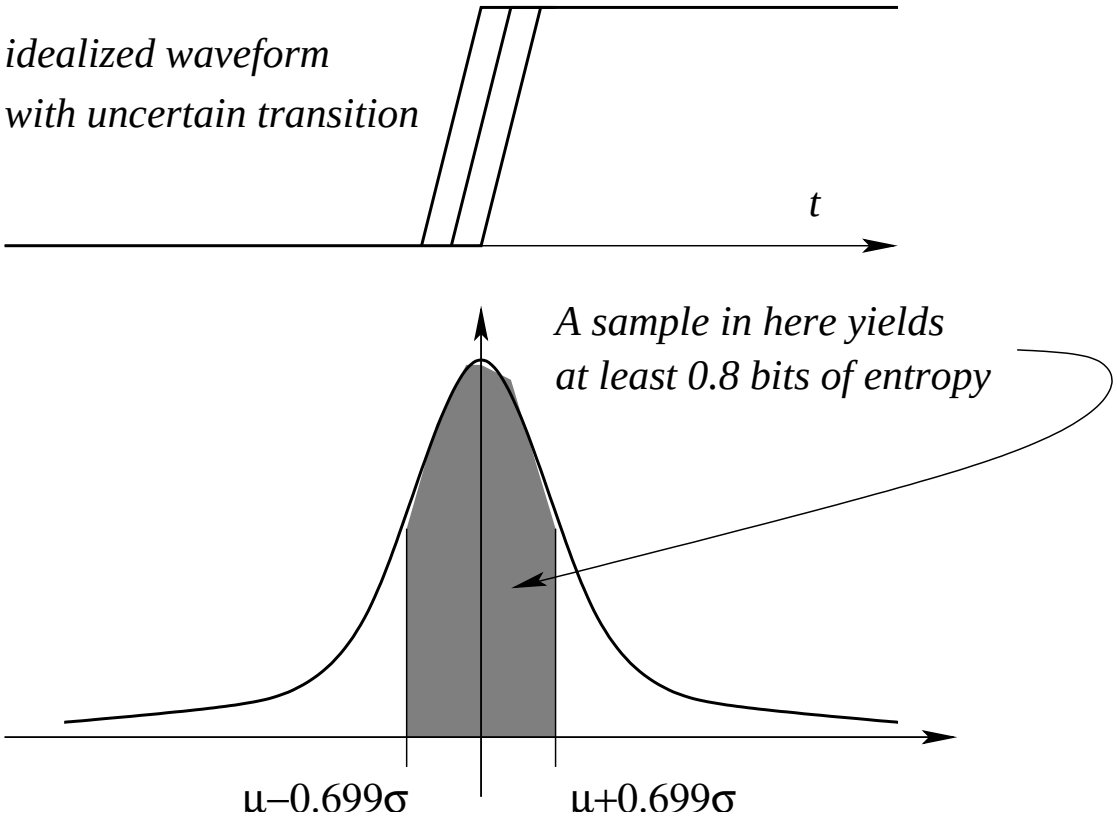
Jitter in oscillators is a good source of randomness.



Rings Design



Jitter Signal Exhibits Gaussian Distribution



The Urn Model

- Assume identical ring lengths (identical periods)
- To simplify analysis discretize the spectrum where each ring fills an urn.
- Before we can sample we want to make sure the spectrum is populated with jitter.
- Main question: How many rings should we use.
- Similar to: How many balls do we have to throw to have all urns filled with high probability.
- Identical to the *Coupon Collector Problem* from discrete math which says the expected number of rings is

$$r = \sum_{s=1}^{N} \frac{N}{s} = N \sum_{s=1}^{N} \frac{1}{s} \approx N \log N.$$

- We also need to identify the confidence level to form lower bounds on the output entropy.

Number of Rings with Confidence Level

- For given N urns, and fill rate $0 < f \leq 1$, and confidence level $0 < p < 1$, determine the minimum $r = M(N, f, p)$ of rings necessary so that, among the N urns, the event that at least fN are filled has probability at least p.
- For $f = 1$, this is the Coupon Collector's Problem.
- Unfortunately, there is no known closed form expression for $M(N, f, p)$.
- Let $P(N, r, f)$ denote the probability that at least fN out of N urns are filled with exactly r rings.
- Then

$$M(N, f, p) = \min \{r : P(N, r, f) \geq p\}$$

Further analysis

- There are N^r functions from an r-set to an N-set.
- This is the total number of ways to associate each ring in our design to one of the N urns.
- The number of surjections (onto functions) among these N^r is given by inclusion-exclusion:

$$J(r, N) := \sum_{h=0}^{N} (-1)^h \binom{N}{h} (N-h)^r.$$

Further analysis - continued

- The number of functions $f : [r] \to [N]$ with an image of size at least fN is
$$\sum_{k=\lceil fN \rceil}^{N} \binom{N}{k} J(r,k).$$
- The probability that the image has size at least fN is then this value divided by N^r, the total number of functions. This gives us our desired value $P(N, r, f)$.

Number of Rings with Confidence Level

For the case $N = 36$ urns shows the number of rings necessary to fill at least fN of the urns with probability at least p

	f									
p	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
.50	26	30	35	41	48	52	61	74	94	142
.55	27	31	36	42	49	53	62	76	96	147
.60	27	31	37	42	50	54	64	77	99	153
.65	27	32	37	43	51	55	65	79	101	159
.70	28	33	38	44	52	56	66	81	104	165
.75	29	33	39	45	53	57	68	83	108	173
.80	29	34	40	46	54	59	70	86	111	182
.85	30	35	41	48	56	61	72	89	116	193
.90	31	36	42	49	58	63	75	93	122	208
.95	33	38	45	52	61	67	80	99	132	233
.99	36	42	50	58	68	75	90	113	153	291

Making it more efficient and fault resilient

Idea: Use less fewer rings than necessary, postprocess output using a *resilient function*

Definition

An (n, m, t)-resilient function is a function

$$F(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)$$

from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2^m$ enjoying the property that, for any t coordinates $i_1, \ldots, i_t$, for any constants $a_1, \ldots, a_t$ from $\mathbb{Z}_2$ and any element y of the codomain

$$\text{Prob}[F(x) = y | x_{i_1} = a_1, \ldots, x_{i_t} = a_t] = \frac{1}{2^m}.$$

In the computation of this probability all x_i for $i \notin \{i_1, \ldots, i_t\}$, are viewed as independent random variables each of which takes on the value 0 or 1 with probability 0.5.

Resilient Functions from Linear Codes

Theorem

Let G be a generator matrix for an $[n, m, d]$ linear code C. Define a function $f : \{0,1\}^n \mapsto \{0,1\}^m$ by the rule $f(x) = xG^T$. Then f is an $(n, m, d-1)$-resilient function.

Output entropy without resilient function for large fN

$$H \approx pfN$$

and with resilient function

$$H \geq mp \ .$$

Overdesigning the resilient function, will gave safety margin against fault insertions into the TRNG.

Example Design

- We use rings use 13 inverters, then experimental evidence shows that $T \approx 25$ ns and $\sigma = 0.5$ ns [*MQPRNG*].
- Assume tolerance $(\mu - \frac{1}{4}\sigma, \mu + \frac{1}{4}\sigma)$, we can say that 1% of the spectrum is filled with jitter for each ring.
- So we have $N = 100$ urns in our combinatorial model
- Tolerance of $\frac{1}{4}\sigma$ ensures that each generated bit will yield at least 0.97 bits of entropy per sampled bit
- Set of $r = 114$ rings fill $> 0.60N$ of urns with prob. > 0.99.
- The resilient function is obtained from an X-BCH $[256, 16, 113]$-code $[B - 97]$.
- Out of the 256 input bits $(1 - 0.60) \cdot 256 = 102.4 \approx 103$ bits will be deterministic
- Our resilient function can tolerate up to 112 corrupted bits, the design has an additional margin to faults of up to 9 bits.

Independent Verification

- D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic True Random Number Generator," In 16th International Conference on Field Programmable Logic and Applications (FPL 2006), IEEE, 6 pages, 2006.
- The FPGA implementation results confirmed our design/analysis in the analytical model.
- Could be sampled at much higher frequencies. (overdesignd?)

Conclusions

- We presented a TRNG design based on sampling phase jitter in oscillator rings.
- Under mild assumptions, will generate *provably* random bits
- Provides some tolerance to adversarial manipulation
- A large increase in the number of oscillators is required to obtain a constant factor improvement in the fill rate.
- We overcome this problem by introducing a post-processing step which consists of an application of an appropriate resilient function.
- The resulting architecture is scalable and is capable of producing randombits in the megabit-per-second range.
- There is much room for improvement.

Bibliography

AIS Anwendungshinweise und Interpretationen zum Schema (AIS). AIS 32, Version 1, Bundesamt fr Sicherheit in der Informationstechnik, 2001.

SK-02 W. Schindler and W. Killmann. Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications. *CHES 2002*, LNCS 2523, pages 431–449, Springer-Verlag Berlin Heidelberg, August 2002.

KS01 W. Killmann, W. Schindler: A Proposal for: Functionality Classes and Evaluation Methodology for True (Physical) Random Number Generators. Version 3.1 (25.09.2001), mathematical-technical reference of [2] (English translation); www.bsi.bund.de/zertifiz/zert/interpr/trngk31e.pdf

M-96 Marsaglia, G. DIEHARD: A Battery of Tests of Randomness, http://stat.fsu.edu/ geo, 1996.

N-00 NIST Special Publication 80022 A Statistical Test Suite for Random and Pseudorandom Numbers. December 2000.

IntelRNG Benjamin Jun and Paul Kocher. The Intel random number generator, April 1999. White Paper Prepared for Intel Corporation.

PA-BST03 Boaz Barak, Ronen Shaltiel, and Eran Tomer. True Random Number Generators Secure in a Changing Environment. *CHES 2003*, pages 166–180, Berlin, Germany, LNCS 2779 2003. Springer-Verlag.

BB-99 Vittorio Bagini, and Marco Bucci A Design of Reliable True Random Number Generator for Cryptographic Applications. *CHES 1999*, pages 204–218, Berlin, Germany, LNCS 1717 1999. Springer-Verlag.

FD-03 Viktor Fischer, and Miloš Drutarovský True Random Number Generator Embedded in Reconfigurable Hardware, *CHES 2002*, pages 415–430, Berlin, Germany, LNCS 2523 2003. Springer-Verlag.

EHKRZ-03 Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts, Michael Epstein, Laszlo Hars, Raymond Krasinski, Martin Rosner, and Hao Zheng, CHES 2003, pages 152–165, LNCS 2779, Springer-Verlag.

MQPRNG Abcunas Brian, Sean Patrick Coughlin, Gary Thomas Pedro, and David C. Reisberg. Evaluation of RNGs Using FPGAs, May 2004. WPI, MQP Report.