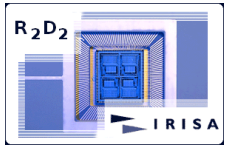
Accelerating Statistical Tests for Real-Time Estimation of Randomness

Renaud Santoro, Olivier Sentieys, Sébastien Roy

santoro@irisa.fr



June 19 - 22nd 2007

Random Numbers Generators (RNG)

Random number generators are necessary in many applications

- Cryptography
- Communication
- VLSI testing
- Probabilistic algorithms

Binary random numbers must be :

- Statistically independant
- Uniformly distributed
- Unpredictible

Random numbers generators and statistical tests

Two families of RNG exist

- True Random Generators (TRNG)
  - Chaotic process : radioactive decay, thermal noise, free running jitter oscillators
- Pseudo Random Generators (PRNG)
  - Deterministic algorithms
  - Initialized by TRNG : seed

Statistical tests

- Used to detect certain kind of weaknesses a RNG may have [Knu97]
- Some batteries of statistical tests are reported in the literature :
  - NIST [RSN+01]
  - Diehard [Mar96]
  - FIPS 140-2 [FIP99]
  - AIS 20, AIS 31 [Ais99]
  - Maurer [Mau91]

Measuring RNG quality

RNG quality can vary in time

- TRNG depends on physical process
  - Difficult to predict physical behavior
  - Depends on implementation quality
  - Chaotic processes can vary and become more deterministic
    - Power noise
    - Thermal noise
- Hacker attacks
  - The randomness of an RNG can be compromised by an attack
  - Statistical test can detect attacks

Consequence

- RNG must be tested in real-time conditions

Outline

I) Batteries of statistical tests

Principle of statistical test

- Detect certain kinds of possible defects [Mau91]
- Test a sequence of N symbols $S \in \mathbf{B} = \{0, 1\}$
- Deterministic algorithm

$$T : B^N \rightarrow \{H_0, H_1\} \quad (1)$$

- Statistical hypothesis
  - H_0 : the RNG is truly random
  - H_1 : the RNG is not truly random
- Significant level of test (Type I error) :

$$\alpha = Prob(H_0 = \text{rejected} | H_0 \text{ is true}), \alpha \in [0.001, 0.05] \quad (2)$$

- Type II error :

$$\beta = Prob(H_0 = \text{accepted} | H_0 \text{ is false}) \quad (3)$$

Batteries of statistical tests

The most recognized batteries of statistical tests

Batteries	Number of tests
NIST [RSN+01]	16
Diehard [Mar96]	15
Diehard [MT02]	3

- Implementations consume memories and arithmetic units
- Too costly to be been implemented in embedded circuits

Consequences on implementation choices

- Choose less demanding (in terms of resources) statistical tests
- $\Rightarrow$ implementation of standard statistical tests

II) Statistical test implementation

Standard statistical tests

- Less powerful tests
- High implementation efficiency
- Allow testing of RNG in real time
- Prevent RNG deviation from their ideal behavior

The four selected statistical tests

- Analyzed a 2×10^4-bit stream
- *Frequency* test, *poker* test, *run* test and *long-run* test
- Used in FIPS 140-1 and FIPS 140-2 (with greatest significant level of test) [FIP99]
- Currently in AIS 21 and AIS 31 [Ais99]

Selected statistical tests [MVO96]

Monobit test

- Digits are uniformly distributed
- Check the RNG bias
- Count the number of 1's n_1 and 0's n_0 in the sequence

$$X_1 = \frac{(n_0 - n_1)^2}{n} \rightsquigarrow \chi_1^2 \quad (4)$$

Poker test

- Bit stream is divided into 4-bit non-overlapping subsequences
- Number of occurrences of each subsequence is counted
- Tests if subsequences are uniformly distributed

$$X_2 = \frac{16}{5.10^3} \sum_{i=0}^{15} n_i^2 - 5.10^3 \quad (5)$$

Selected statistical tests

Run test

- n_i^1 : number of i length 1's runs (respectively n_i^0), $1 \leq i \leq 6$
- Verify if n_i^0 and n_i^1 are as expected for a random sequence

Length of run	Required interval (FIPS 140-1, AIS 21, AIS 31)
1	2267-2733
3	1079-1421
3	502-748
4	223-402
5	90-233
≥ 6	90-233

Long Run test

- Number of runs of length greater than 34

Implementation results

Implementation in a Virtex 5 LX50

Number of batteries	1	10
Number of Slice Register	409 (1%)	3884 (13%)
Number of Slice LUT	777 (2%)	7104 (24%)
Maximum frequency (MHz)	188.629	188.331

Total area, total dynamic power and critical path in 130 *nm*, 1.2V CMOS technology

battery Number	Area (μm^2)	Power (mW)	Critical Path (ns)
1	32186	7.5	4.43 (225.733 MHz)

Real-time testing of RNG randomness

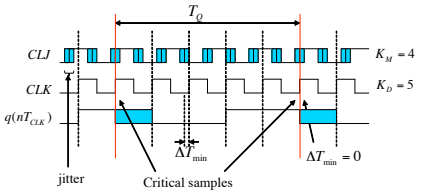


FIG.: Sampling of CLJ signal by CLK.

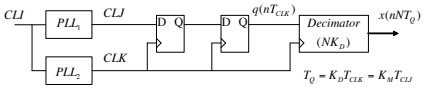


FIG.: Structure of the oscillator sampling.

Simple stochastic model of the RNG bias as a function of K_D^p value

- $x(nT_Q) = q(nT_q) \oplus q(nT_q - T_{CLK}) ... \oplus q(nT_q - (K_D - 1)T_{CLK}))$
- $K_D = K_D^p + K_D^1 + K_D^0$
- $K_D^p \rightarrow$: number of critical samples
  - Critical sample probability distribution is measured in FPGA
- $K_D^1, K_D^0 \rightarrow$ deterministic

Real-time testing of configuration proposed in [SFDF06]

♯	K_M	K_D	K_D^p	$E[x(nT_Q)]$
1	144	119	61	0.829
2	144	175	89	0.729
3	486	119	55	0.553
4	486	161	74	0.524
5	250	203	95	0.526
6	270	203	96	0.496

TAB.: Predicting value of $E[x(nT_Q)]$ via the simple stochastic model.

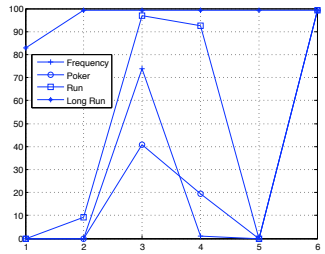


FIG.: Success percentage of 1×10^3 statistical tests (2×10^7 bits) for each configuration.

Conclusion of [SFDF06]

- ♯1, 2, 5 : statistical test deviation from TRNG
- ♯3, 4 : model can not be applied due to dependence between critical samples
- ♯6 : efficient configuration but not always random (99.35 %)

Cellular automata (CA)

Applicability to VLSI

- Array of cells : highly parallel, regular
- Cell :
  - Locally interconnected
  - Very simple logic functions
  - Have only a finite number of possible states (here $\{0, 1\}$)
- Evolution is defined by the state of the neighbor cells and the CA rule [Ila01]
- Depending on the rule, CA can exhibit highly chaotic behavior
- Predicting CA behavior can be extremely difficult [Ila01]

Finding the optimal CA rule

Problem

- If a cell has N neighbors : 2^{2^N} rules are possible

Solutions

- Genetic algorithm [TSP00] (not guaranteed to find the optimal rule)
- Exhaustive search is a time-consuming task
- [STCS02] have done an exhaustive search for a 4-neighbor CA using a hardware entropy measure

Objective

- Find the best rule for one-dimensional uniform CA with 64 cells
- Each cell has 5 neighbors $\Rightarrow 2^{32}$ possible rules exist
- Cyclic boundary conditions are applied

Finding the optimal CA rule

Methodology

- Exhaustive search : accelerated in hardware
  - Four hardware statistical tests
  - Entropy measure computed on 2500 8-bit non-overlapping blocks

$$H_8 = -\sum_{i=0}^{2^8-1} p_i \log_2 p_i \quad (6)$$

- Each CA rule is analyzed in real time
- Number of possible rules (2^{32}) can be reduced : $\begin{pmatrix} 32 \\ 16 \end{pmatrix}$
- Due to the complexity of cutting-edge FPGAs :
  - Several rules can be analyzed simultaneously
- In a Virtex 5 LX50, the exhaustive search takes about 4 hours and 28 minutes

Global architecture

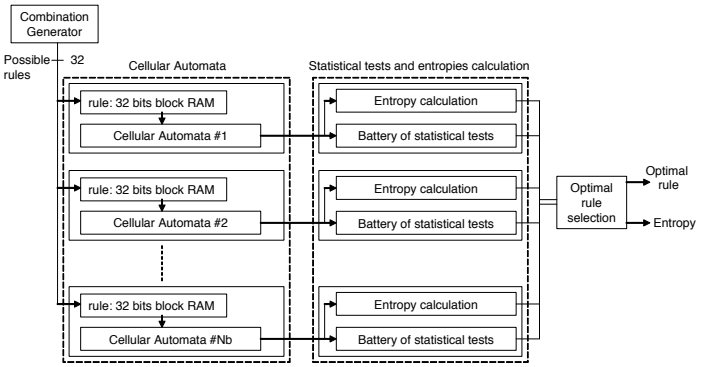


FIG.: Global architecture.

Results

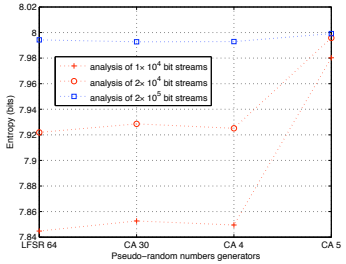


FIG.: Entropy results of some random number generators.

PRNG	Entropy (bits)
LFSR64	7.9942
CA 30	7.9927
CA 639_{16}	7.9929
CA $AAD32197_{16}$	7.9991

TAB.: Entropy of the four selected pseudo-random number generators when a 2×10^5 bits stream is analyzed.

Conclusion on optimal 5-neighbor

- Converges more quickly to the highest entropy
- Close to the maximal entropy $H_{max} = 8$
- Selected for a given initial state
- Cannot be applied for other seeds $\Rightarrow$ adaptive optimal 5-neighbor search

Conclusion

Hardware statistical tests

- Efficient implementation
- Allowing real-time bit stream randomness evaluation without storing the RNG output
- Warn against RNG weakness
- TRNG depend on their implementation and environment
  - Entropy source can vary with time
  - Statistical tests must be written according to the entropy source (e.g. jitter)
  - Each TRNG can have its own statistical test to detect attacks

Future work : testing RNG by more stringent statistical tests

- Approximate entropy test [Ruk00]
- Autocorrelation test [Ais99]
- Currently implementing the *gcd* test, the *gorilla* test and the *birthday spacing* test described in [MT02]

Conclusion

Finding cellular automata exhibiting optimal randomness

- Exhaustive search is allowed by using
  - Hardware statistical test
  - Entropy measure
- FPGA circuits boost the search speed
- Several rules can be analyzed at the same time
- Increasing the number of neighbors improves entropy
- Optimal rule is found for a given seed
- For other seeds, the bit stream generated can exhibit randomness failure
- An adaptive CA rule architecture based on hardware statistical tests will be proposed

[Ais99] Application notes and interpretation of the scheme (ais).
Technical report, Bundesamt für Sicherheit in der Informationstechnik, 1999.

[FIP99] FIPS.
Security requirements for cryptographic modules, FIPS PUB 140-2, 1999.

[Ila01] Andrew Ilachinski.
Cellular Automata : A Discrete Universe.
World Scientific Publishing Company, 2001.

[Knu97] Donald Ervin Knuth.
The art of computer programming, volume 2 : seminumerical algorithms.
Addison-Wesley, 3rd edition, 1997.

[Mar96] George Marsaglia.
Diehard : A battery of tests of randomness.
Technical report, Florida State University, Tallahassee, FL, USA, 1996.

[Mau91] Ueli M. Maurer.
A universal statistical test for random bit generators.
In *CRYPTO '90 : Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 409–420, London, UK, 1991. Springer-Verlag.

[MT02] George Marsaglia and Wai Wan Tsang.
Some difficult-to-pass tests of randomness.
7(3) :1–8, 2002.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot.
Handbook of Applied Cryptography.
CRC Press, Inc., Boca Raton, FL, USA, 1996.

[RSN+01] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and D.L. Banks.
A statistical test suite for random and pseudorandom number generators for statistical applications.
NIST Special Publication in Computer Security, pages 800–22, 2001.

[Ruk00] Andrew L. Rukhin.
Approximate entropy for testing randomness.
Journal of Applied Probability, 37(1), 2000.

[SFDF06] Martin Simka, Viktor Fischer, Milos Drutarovsky, and Jacques Fayolle.
Model of a true random number generator aimed at cryptographic applications.
In *In Proceedings of IEEE International Symposium on Circuits and Systems*, 2006.

[STCS02] Barry Shackleford, Motoo Tanaka, Richard J. Carter, and Greg Snider.
High-performance cellular automata random number generators for embedded probabilistic computing systems.
In *EH '02 : Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02)*, page 191. IEEE Computer Society, 2002.

[TSP00] Marco Tomassini, Moshe Sipper, and Mathieu Perrenoud.
On the generation of high-quality random numbers by two-dimensional cellular automata.
IEEE Trans. Comput., 49(10) :1146–1151, 2000.