Lightweight FPGA Implementation of FIPS140-2 Online Statistical Tests

İhsan Çiçek, Mustafa Parlak, & Çetin Kaya Koç
University of California Santa Barbara

CryptArchi 2016

Outline

- RNG Failures and True Randomness
- Our Contributions
- Power Aware Design of FIPS140-2 Tests
  - Low Power Tests: Monobit, Runs, Long Run
  - High Power Test: Poker Test
- Poker Test without a multiplier
- Implementation and Measurement Results

True or Deterministic Randomness: That is The problem

- RNG is (probably) the most important part of a cryptographic system
- A failure in the RNG is a catastrophic failure for the security
- Cryptography without **<u>TRUE</u>** randomness invites disaster

$\longrightarrow$

1996	- Mozilla SSL RNG failure
2007	- NSA's Dual EC DRBG backdoor revealed - MS-Win Insecure RNG
2008	- Debian/SSL Fiasco - MiFare Classic Hacked
2010	- PHP sessions hijacked due to weak RNG - Sony PS3 Hacked
2012	- Brazilian voting machine fiasco due to weak RNG - Juniper ScreenOS backdoor due to weak RNG

True Randomness Caveats

- Noticeably more TRNG side channel attacks are being reported than the previous years
- Statistical quality assessment of an RNG output is highly crucial to ensure security
- Federal and International standards (FIPS140-2, ISO18031) mandate the run-time monitoring of RNGs
- Continuous RNG monitoring means increased power consumption -- a burden for lightweight embedded systems
- Power-efficient design of run-time RNG monitoring is desired

Contributions

1. **Power-aware design**: Design is partitioned into low, and high power consuming test blocks. In the run-time low power tests enabled first, if results are ok, then high power tests are enabled.

2. **A novel poker test design** which avoids the use of a power, and area hungry multiplier.

3. **Use of a multi-bit status bus** to avoid creating a single point of attack advantage for an adversary.

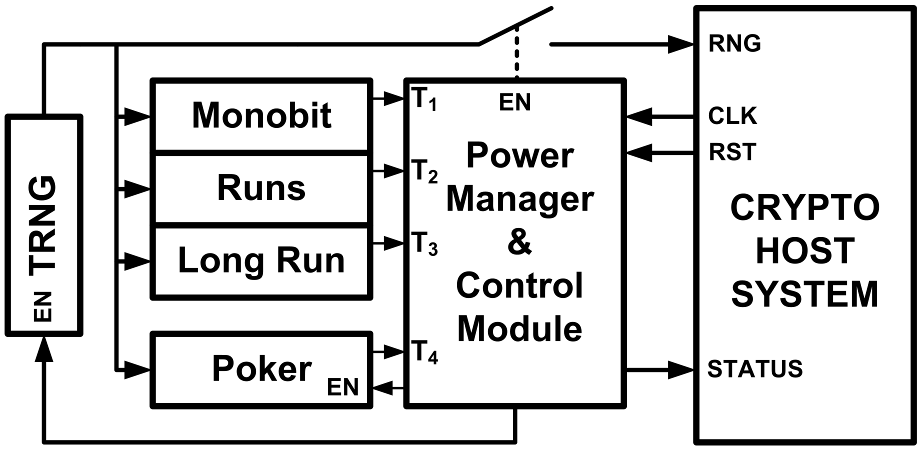
4. **A technology agnostic energy efficiency metric** for making fair comparisons with other designs in the literature.

Design of FIPS140-2 Tests

A simple FSM based module controls the operation of the online monitor

<u>**Control Module Operation**</u>

1. Enable RNG, and start acquisition
2. Enable low power tests
3. If no failure is detected, then enable high power tests
4. If no failure is detected, then enable transfer.
5. If any failure is detected at any stage, send alarm to host, and halt until reset



We use a. 4-bit bus to signal pass/fail status

Only 1 out of 16 states represents pass, other 15 represent failure

Design of FIPS140-2 Tests

Monobit Test (Tests the distribution of 0/1s)
N = **20-kbit**,
Confidence Interval **(9725, 10275),** Implemented using 14-bit counter

Runs Test (Tests the distribution of m-bit runs)

0/1 Runs m-bit	Confidence Interval
1	**(2315, 2685)**
2	**(1114, 1386)**
3	**(527, 723)**
4	**(240, 384)**
5	**(103, 209)**
6	**(103, 209)**

Implemented using 12 counters.

Long Run Test (Is the Longest run of 0/1s < 26 ?),
Implemented using a 5-bit counter

Design of FIPS140-2 Tests

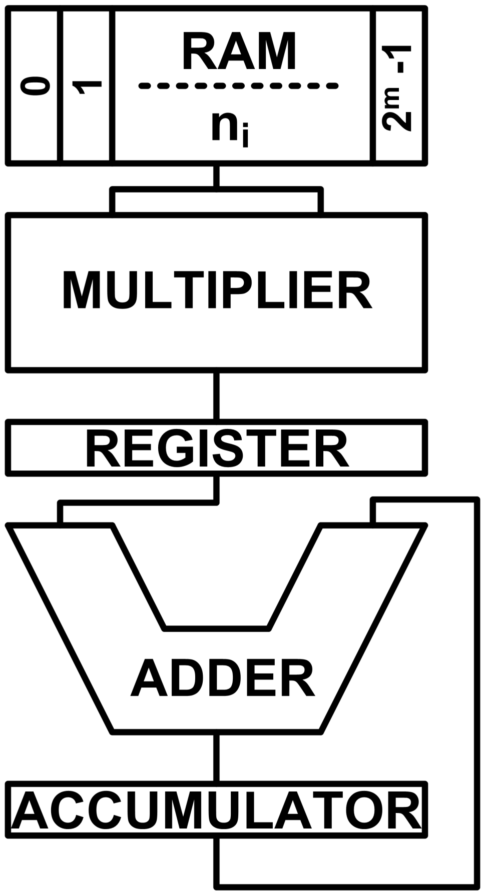
Poker Test (Tests the distribution of m-bit blocks)
N = **20-kbit**, **m = 4**, test metric **X**, defined by

$$X = \frac{m \times 2^m}{N} \times \sum_{i=0}^{2^m-1} n_i^2 - \frac{N}{m}$$

where $\mathbf{n_i}$ = occurrences of **m-bit** integer **i**.

Hardware implementation is more complex when compared to other tests, and requires more resources

Traditional designs use a multiplier which cause increased power consumption. After N-bits ($\mathbf{n_i}$ values are finalized), $\mathbf{n_i}^2$s are calculated and accumulated



Design of FIPS140-2 Tests

Can we calculate X without using a multiplier?

Acquisition time can be used more effectively if $\mathbf{n_i}$ and $\mathbf{n_i^2}$ can be calculated during acquisition

It may also be required (and possible) to eliminate the multiplier

Poker Test Without Multiplier

The basic idea is to calculate $\mathbf{n_i}$ and $\mathbf{n_i^2}$ during acquisition and accumulate thereafter

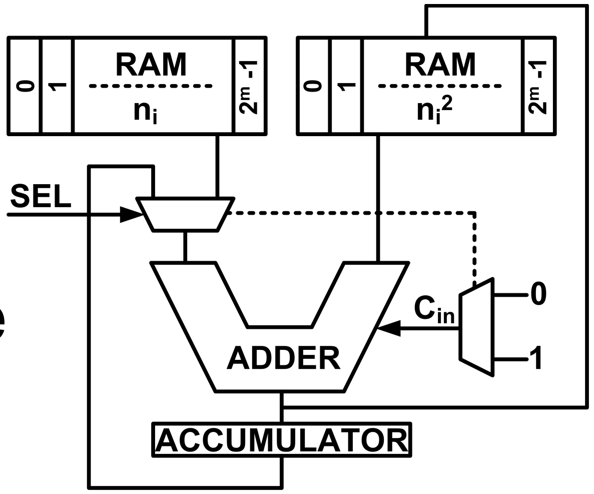
- Two 2^m word Block RAMs are used to store $\mathbf{n_i}$, and $\mathbf{n_i^2}$ s

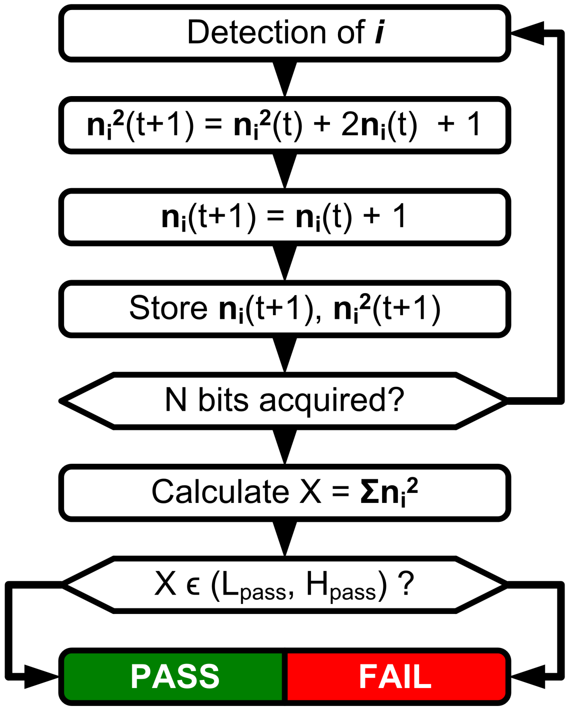
- On detection of integer **i**, $\mathbf{n_i}$, and $\mathbf{n_i^2}$ are calculated using

$$n_{i,t+1} = n_{i,t} + 1 \Longrightarrow n_{i,t+1}^2 = (n_{i,t}+1)^2, \\ = n_{i,t}^2 + 2n_{i,t} + 1.$$

- Carry-in of the adder is used for **+1**

- After N-bits are acquired, **X** is calculated, and a decision is made

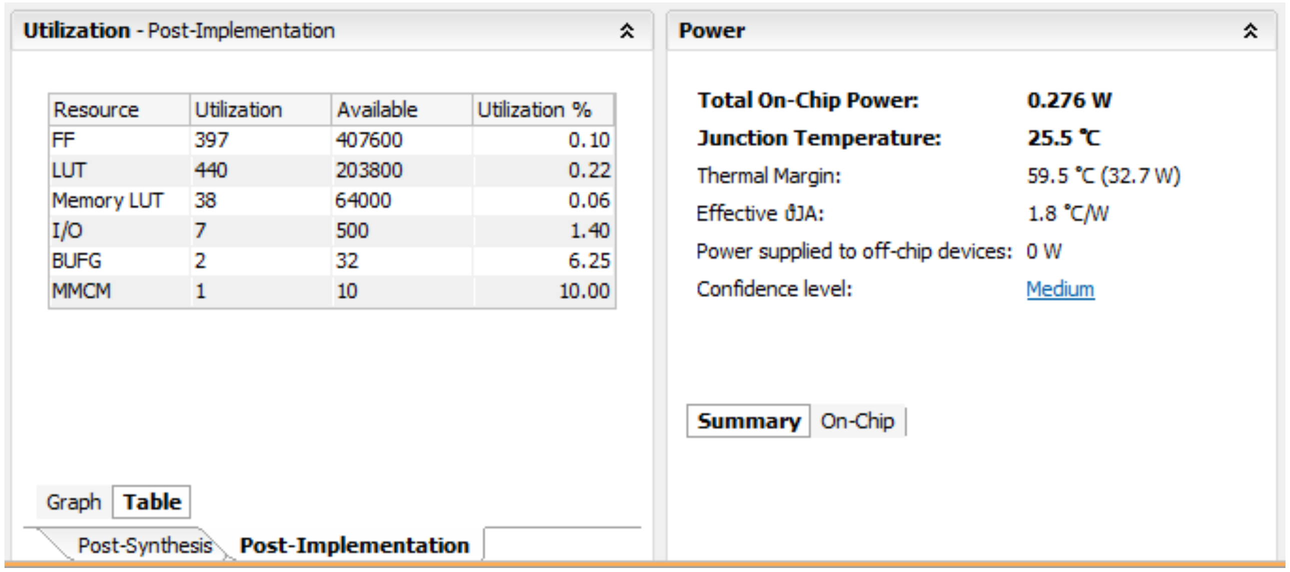



Detection of *i*
$\mathbf{n_i^2}(t+1) = \mathbf{n_i^2}(t) + 2\mathbf{n_i}(t) + 1$
$\mathbf{n_i}(t+1) = \mathbf{n_i}(t) + 1$
Store $\mathbf{n_i}(t+1)$, $\mathbf{n_i^2}(t+1)$
N bits acquired?
Calculate X = $\mathbf{\Sigma n_i^2}$
X ϵ (L_{pass}, H_{pass}) ?
PASS
FAIL

FPGA Implementation

The design was implemented on KC705 Board that hosts a Xilinx Kintex-7 XC7K325TFFG900-2. Board allows current and voltage measurements through Power Management Bus (PMBUS).

Single Instance Implementation Results

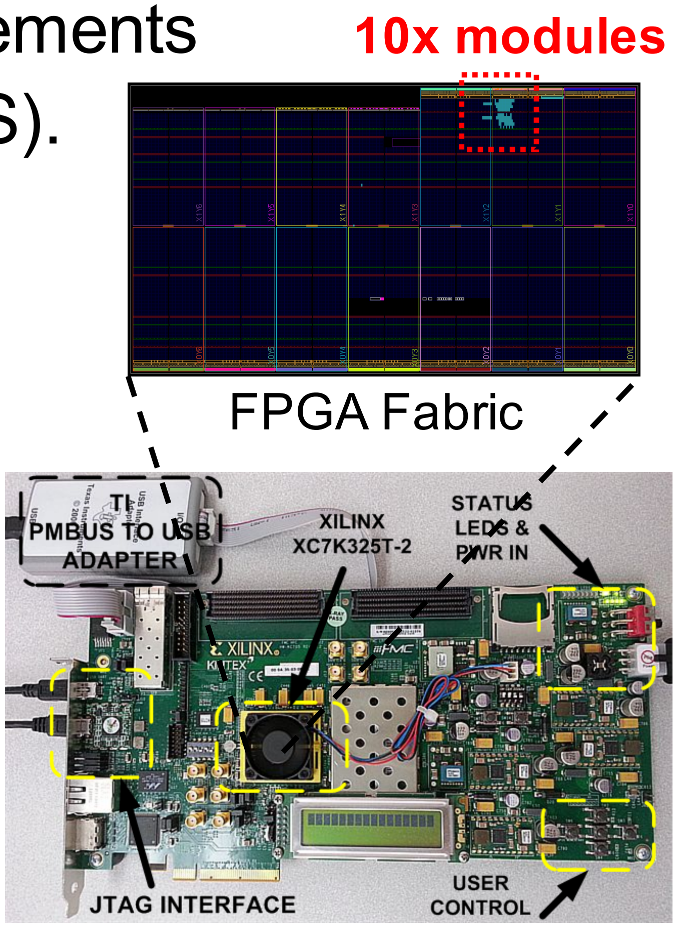


Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
FF	397	407600	0.10
LUT	440	203800	0.22
Memory LUT	38	64000	0.06
I/O	7	500	1.40
BUFG	2	32	6.25
MMCM	1	10	10.00

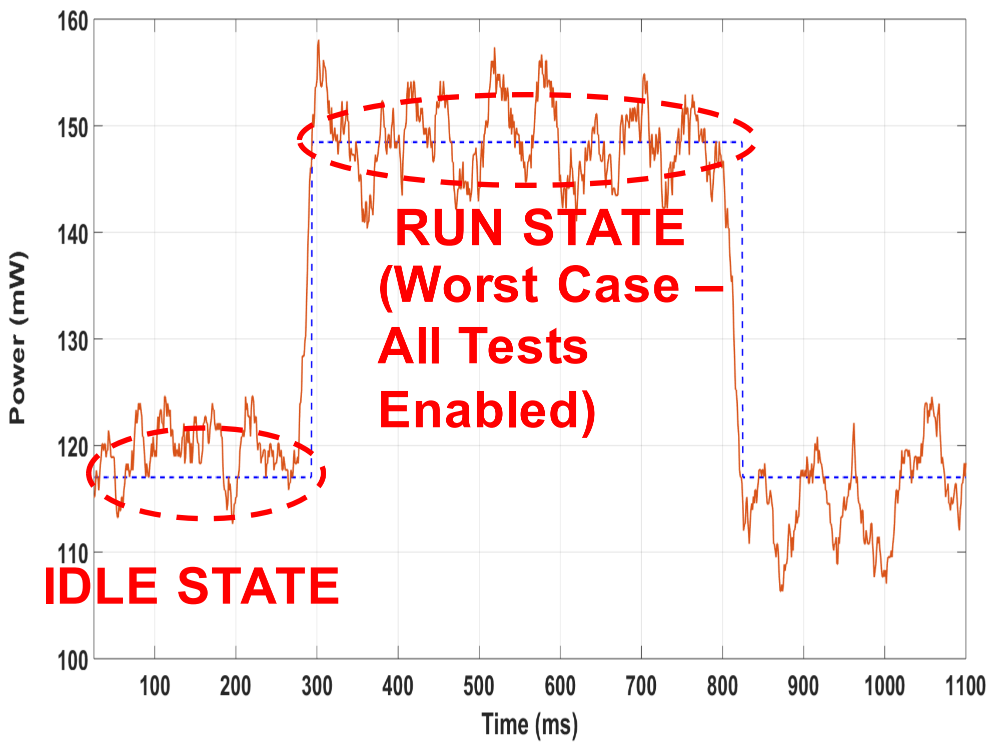
Power

Total On-Chip Power:	**0.276 W**
Junction Temperature:	**25.5 ℃**
Thermal Margin:	59.5 ℃ (32.7 W)
Effective ϑJA:	1.8 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium



Measurement Results

- Module has a very small power footprint, in order to make observable measurements, we implemented **10x** instances on FPGA.
- Difference of the power consumption between IDLE and RUN states (Worst case forced - all tests enabled) are used to make measurements.
- We used an LFSR based RNG to minimize error/influence on measurements.



- For f_{clk} = 200MHz, P = 3.15mW

Performance Comparison

There are various FIPS140-2 designs implemented using different approaches. We propose the cost of testing 1-bit in terms of energy as an efficiency metric for fair comparison with the literature.

$$\eta = \frac{Power \times Time}{Total\ number\ of\ bits}$$

Platform	**Device**	**LUTs (%)**	$\mathbf{f_{MaxClk}}$ **(MHz)**	η **(pj/bits)**
$Virtex-2^*$	XC2V1000-6	626 (6%)	134.7	N/A
$Virtex-5^*$	XC5VLX50T-3	482 (1%)	189.4	N/A
Kintex – 7	**XC7K325T-2**	**465 (0.23%)**	**399.8**	**15.75@200MHz**

*R. Santoro, O. Sentieys, and S. Roy, "On-line monitoring of random number generators for embedded security," in Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on, May 2009, pp. 3050–3053.