54. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

Information Technology and Electrical Engineering - Devices and Systems, Materials and Technologies for the Future

Faculty of Electrical Engineering and Information Technology

Startseite / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=14089

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum

Herausgeber: Der Rektor der Technischen Universität Ilmenau
Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c.
Peter Scharff

Redaktion: Referat Marketing
Andrea Schneider

Fakultät für Elektrotechnik und Informationstechnik
Univ.-Prof. Dr.-Ing. Frank Berger

Redaktionsschluss: 17. August 2009

Technische Realisierung (USB-Flash-Ausgabe):
Institut für Medientechnik an der TU Ilmenau
Dipl.-Ing. Christian Weigel
Dipl.-Ing. Helge Drumm

Technische Realisierung (Online-Ausgabe):
Universitätsbibliothek Ilmenau
ilmedia
Postfach 10 05 65
98684 Ilmenau

Verlag:
Verlag ISLE, Betriebsstätte des ISLE e.V.
Werner-von-Siemens-Str. 16
98693 Ilmenau

ISBN (USB-Flash-Ausgabe): 978-3-938843-45-1
ISBN (Druckausgabe der Kurzfassungen): 978-3-938843-44-4

Startseite / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=14089

OPTIMIZED FIPS 140 STATISTICAL TESTS IP CORE EMBEDDED IN FLASH BASED ACTEL FPGA

Miloš Drutarovský, Michal Varchola, Ondrej Vancák

Department of Electronics and Multimedia Communications
Technical University of Košice
Komenského 13, 04120, Košice, Slovakia
Milos.Drutarovsky@tuke.sk

ABSTRACT

We present the FPGA-optimized implementation of the statistical tests for evaluating sufficient quality of the embedded True Random Number Generator output. Tests are fully compatible the FIPS 140 standard and are implemented in the Actel Fusion FPGA. Hardware can run up to 109 MHz clock frequency and can process one random bit per one clock period. Whole implementation consumes 1345 logic tiles including interface to ARM7TDMI soft-core embedded processor.

Index Terms— FIPS 140, FPGA, Actel Fusion

1. INTRODUCTION

We present the FPGA-optimized implementation of the statistical tests for evaluating sufficient quality of the embedded True Random Number Generator (TRNG) output. Role of TRNGs is crucial in the various modern embedded cryptographic applications. Random Numbers (RNs) are required e.g. as session keys, and therefore they should meet strict requirements – they should be unpredictable, uniformly distributed on their range and independent [1]. Although designers make a heavy effort to implement the TRNG in a robust way, their designs are potentially vulnerable to technology ageing, variation of operating conditions, or an attack in a hostile environment. Each of mentioned disorders could cause degradation of the RNs' statistical properties and thus weaken otherwise secure system. The most straightforward way how to detect such a breakdown is to implement a set of basic statistical tests working in real-time as a supplement to the TRNG. Requirements for online tests are listed in [2] – the online test should detect non-tolerable statistical weaknesses, should run fast and consumes only little hardware resources.

A necessity of the fast on-chip randomness testing arises within our research work on the secure TRNGs for FPGAs. That is why we have developed a specialized IP-core that performs basic statistical randomness tests. Such hardware implementations appear in literature rarely; as an example we can mention [3] where authors used Virtex II FPGA.

We adopted the Federal Information Processing Standard (FIPS) 140 [4] among published statistical tests, which consists of 4 following tests – Monobit, Poker, Runs, and Long run. Required length of the sequence is 20,000 bits, that implies small demand on hardware resources. Nevertheless, the small-area implementation of these tests is not trivial.

Paper is organized as follows: Section 2 describes FIPS 140 standard. The implementation Platform and the architecture of implemented system is introduced in the sections 3 and 4 respectively. Section 5 deals with analysis and synthesis of the particular tests. Section 6 provides implementation results. Conclusion is given in the last section.

2. DESCRIPTION OF FIPS 140 TEST SUITE

The first version of FIPS 140 standard [5] was introduced in 1994. Particular part of it deals with statistical randomness tests that we are interested in. In 2001 was introduced the second version of the document [4] with slightly changed treshold regions for the Runs test.

Here is description of the tests that are included:

- T1: The Monobit Test
  1. Count the number of ones in the $20,000$ bit stream. Denote this quantity by X.
  2. The test is passed if $9,725 < X < 10,275$
- T2: The Poker Test
  1. Divide the $20,000$ bit stream into $5,000$ contiguous 4 bit segments. Count and store the number of occurrences of each of the 16 possible 4 bit values. Denote $f(i)$ as the number of each 4 bit value i where $0 \leq i \leq 15$

This work has been done in the frame of the Slovak scientific project VEGA 1/4054/07, and project KEGA 3/5238/07of Slovak Ministry of Education.

2. Evaluate following:

$$X = (16/5000) \cdot (\sum_{i=0}^{15} [f(i)]^2) - 5000 \quad (1)$$

3. The test is passed if $2.16 < X < 46.17$

- T3: The Runs Test
  1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros, which is part of the $20,000$ bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (≥ 1) in the sample stream should be counted and stored.
  2. The test is passed if the number of runs that occur (of lengths 1 through 6) is each within the corresponding interval specified in Table 1. This must hold for both the zeros and ones; that is, all 12 counts must lie in the specified interval. For the purpose of this test, runs of greater than 6 are considered to be of length 6.

- T4: The Long Run Test
  1. A long run is defined to be a run of length 26 or more (of either zeros or ones).
  2. On the sample of $20,000$ bits, the test is passed if there are NO long runs.

3. IMPLEMENTATION PLATFORM

We chose the recent Flash based Actel Fusion FPGA M7AFS600 due to benefits listed in [6]. Available soft-core ARM7TDMI compatible processor is used for interfacing the tests. Available Phase Locked Loops (PLLs) are necessary for implementation of a PLL-based TRNG [7], used as source of RNs under test. Drawback of Fusion family is unavailability of dedicated multipliers which would be helpful for squaring needed by the Poker test. Proposed IP core is written in the VHDL and special emphasis was taken on the speed-and-area effective implementation of all tests under Flash FPGA constraints.

Table 1. FIPS 140 - the Runs Test tresholds, according [4]

Length of Run	Required Interval
1	2,315 - 2,685
2	1,114 - 1,386
3	527 - 723
4	240 - 384
5	103 - 209
6+	103 - 209

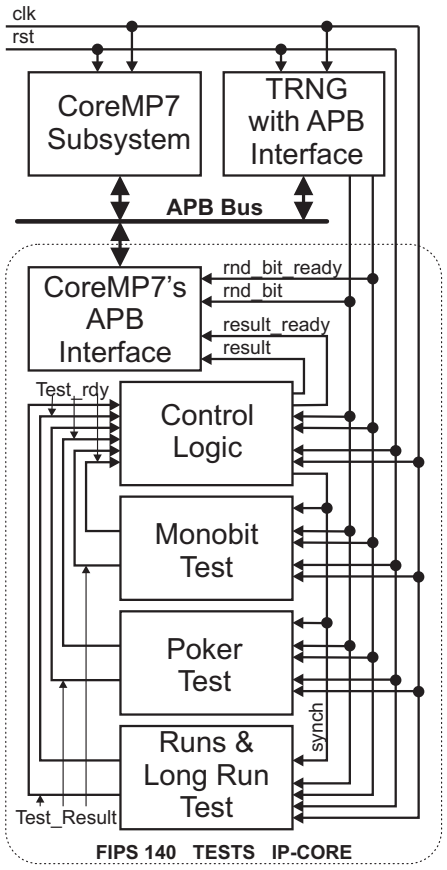

Fig. 1. Architecture of the specialized FIPS 140 IP-Core and its connection to the CoreMP7 subsystem and the TRNG

4. ARCHITECTURE OF IMPLEMENTED SYSTEM

Architecture of the entire system is shown in the Fig. 1. There are three main components: the CoreMP7 subsystem, the TRNG with APB interface and the FIPS 140 Tests IP-core with the APB interface. All these components communicate by the Advanced Peripheral Bus (APB). There is direct interconnection between TRNG and FIPS 140 Tests IP-core that provides random bits for tests.

FIPS 140 tests IP-core consists of APB interface, Control Logic and instances of particular tests. The APB interface translates results of the tests to the status register of the peripheral, that is possible to read by software running in the CoreMP7. The Control Logic distribute the $synch$ signal which indicates end of the 20,000 bits packet and is used for synchronizing and controlling all instances of tests. Control Logic block also merges results of all tests into one result.

5. SYNTHESIS OF PARTICULAR TESTS

Each test from the suite is different from hardware implementation point of view and the specific approach of the effective synthesis is needed. This section deals with each test separately.

5.1. Monobit test

The Monobit test is the easiest one within entire suite. There is the only one register necessary that increments its content by one when input random bit has $'1'$ value. Content of the register is compared with reference threshold values when $synch$ signal is active.

5.2. Poker test

The Poker test described by equation (1) can be hardly implemented in hardware. That is why we derived compatible equation using integers [8]:

$$1563175 < \sum_{i=0}^{15} [f(i)]^2 < 1576929 \qquad (2)$$

As we need to count an appearance of each group of four bits we are interested what is the maximum value of $[f(i)]$ for passing the test in order to allocate appropriate number of bits for registering. We found if the count of at least one combination of bits exceeds 428 the test would not pass [8]. That why we need $16 \cdot 9$-bit register field. The computation of the square and sum takes basically some time and so we use two such register field; one is used for storing counts of the bit groups and the second is used for performing computation on it. Two times $16 \cdot 9$-bits would consumes quite a lot of FPGA resources and so we have decided to implement the register field into the RAM. The data-flow architecture of poker test is shown in the Fig. 2.

First of all, a group of 4-bits is collected by the shift register. After that, active RAM & incrementer block fetch value from particular register in RAM, increment it and store again to the RAM. Shift register is collecting next 4 bit group meanwhile. Both operations, collecting bits and incrementing particular register in RAM takes 4 clock cycles and so we can process one random bit per one clock period.

After storing 5,000 4-bit groups Block A and Block B are switched - content of Block A will be processed and counts of 4-bit groups will be stored in Block B.

We use the serial multiplier for performing square operation in order to save logic resources. The lower speed does not cause complication because we have enough time while 20,000 bits are stored in Block B. We used Ripple Carry Adder because of the same reason for the addition operation. Finally, we test if the result fits into interval shown in (2).

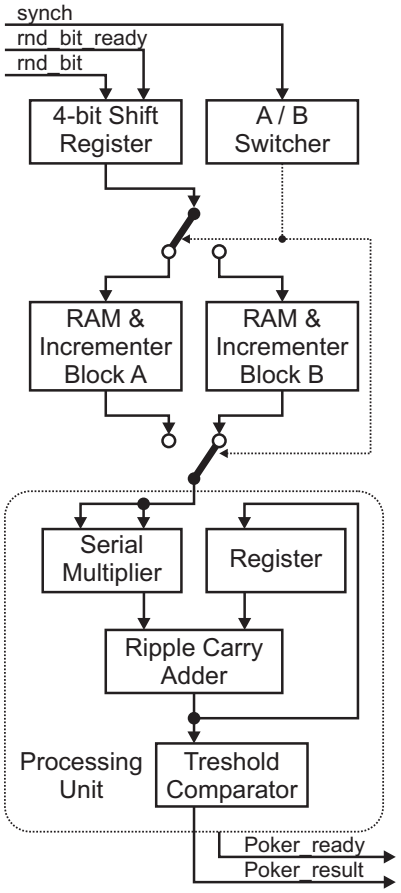

Fig. 2. The data-flow architecture of the Poker test where RAM & Incrementer Blocks A and B and the Processing unit are synthesized as FSMs

5.3. Runs and Long run test

The Runs and Long run test are merged to the same instance due to similar nature of both tests. The test is synthesized as single FSM. The FSM transaction diagram is shown in the Fig 3.

The FSM consists of following states: idle, compare, zX, oX; where X in zX and oX states means number of zeros or ones respectively of particular run. The FSM falls into idle state during reset. After that random numbers are entered to the FSM in the sequence. When the value of the first random bit is $'0'$, the FSM falls to the z1 state. When the value of the second random bit is $'0'$, the FSM falls to the z2 state. When the value of the second random bit is $'1'$, the FSM falls to the o1 state and the register counting appearance of the run of single $'0'$ is incremented, etc. When the state is changed from the right side of the diagram to the left (or vice versa) – that means the run of several same consecutive bits is over – the appropriate register is incremented. In the case of this test, register field is stored in the logic in order of possibility of incrementing it in the single clock period. The FSM falls to compare state when $synch$ signal is active (at the end of the 20,000 bit se-

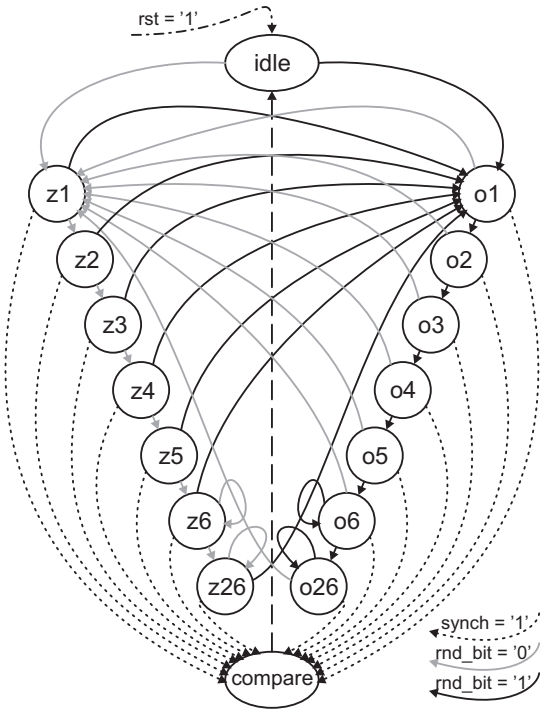

Fig. 3. The state diagram of the FSM of the Runs and Long run test; registered value of particular runs is incremented when transaction from the left side to right one appears (or vice versa)

quence) and all registers are tested if its values fit into treshold regions listed in the Table 1.

6. IMPLEMENTATION RESULTS

The VHDL code was synthesized by the Synplify (Actel Version) and the implementation was done by the Designer. Both of them are distributed within the Libero 8.5 that is Actel's Integrated Design Environment. Synplify provided the best results (resources consumption and speed) when it was asked for the 110 MHz resulting frequency. Designer was configured for multiple passes (in order to find better starting position of the placer) with the high-effort place and route algorithm in order to achieve the best results. Results are given in the Table 2

Table 2. Implementation results showing resources consumption and maximum achieved clock frequency

instance	area (tiles)	M7AFS600 percentage	RAM blocks	speed (MHz)
whole	1345	9,7%	2	109
monobit	91	0,7%	0	250
poker	561	4,1%	2	130
runs	560	4,1%	0	125

7. CONCLUSION

We have proposed hardware implementation of the FIPS 140 statistical randomness test suite. Proposed specialized IP-core can be used as a component of the each cryptographic system where detection of the TRNG malfunction is critical. Our design can process one random bit per single clock period. Maximum achieved clock period was 109 MHz for entire system that means we can perform testing of random data with 109 Mbps bit-rate. Whole solution consumes 1345 tiles that is less than 10% of the popular M7AFS600 FPGA. An alarm can be reported to the CoreMP7 processor which does not need to run FIPS 140 tests anymore and save its computation time for another tasks. Authors in [3] achieved 113MHz maximum frequency that is comparable with our results but they used high-performance Virtex II FPGA by Xilinx while our Fusion is cost-effective Flash FPGA.

8. REFERENCES

[1] W. Schindler, *Cryptographic Engineering*, chapter Random Number Generators for Cryptographic Applications, Springer, 2009, ISBN: 978-0-387-71816-3.

[2] W. Schindler, "Efficient online tests for true random number generators," in *Proceedinds of Cryptographic Hardware and Embedded Systems - CHES 2001, May 14-16, Paris, France*. 2001, vol. 2162 of *LNCS*, pp. 103–117, Springer.

[3] A. Hasegawa, S. J. Kim, K. Umeno, and N. Kitamachi, "Ip core of statistical test suite of fips 140-2," *Design & Reuse*, Online. http://www.design-reuse.com/articles/7946/ip-core-of-statistical-test-suite-of-fips-140-2.html.

[4] U.S. Department of Commerce / National Institute of Standards and Technology, *Federal Information Processing Standards Publication FIPS PUB 140-2*, May 2001.

[5] U.S. Department of Commerce / National Institute of Standards and Technology, *Federal Information Processing Standards Publication FIPS PUB 140-1*, January 1994.

[6] M. Drutarovsky and M. Varchola, "Cryptographic system on a chip based on actel ARM7 soft-core with embedded true random number generator," in *Proceedinds of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Bratislava, Slovakia*, April 16-18 2008, pp. 164–169.

[7] V. Fischer and M. Drutarovsky, "True random number generator embedded in reconfigurable

hardware," in *Proceedinds of Cryptographic Hardware and Embedded Systems - CHES 2002, August 13-15, Redwood Shores, CA, USA*. 2002, vol. 2523 of *LNCS*, pp. 415–430, Springer.

[8] O. Vancak, "Implementation of the statistical randomness tests for cryptographic applications in fpga devices," M.S. thesis, Technical University of Kosice, Park Komenskeho 13, May 2009, In Slovak.