International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 2, No. 3, November 2013, pp. 99~105
ISSN: 2089-4864

FPGA Implementation of Park-Miller Algorithm to Generate Sequence of 32-Bit Pseudo Random Key for Encryption and Decryption of Plain Text

Bharatesh N, Rohith S
Department of Electronics and Communication Engineering, Nagarjuna College of Engineering and Technology

Article Info

Article history:

Received Jul 21, 2013
Revised Sep 20, 2013
Accepted Oct 12, 2013

Keyword:

Decryption
Encryption
Logistic map
Pseudo-Random Bit Generator
Cryptographic security

ABSTRACT

There are many problems arises in randomized algorithms whose solutions are fundamentally based on assumptions that pure random numbers exist, so pseudo-random number generators can imitate randomness sufficiently well for most applications. The proposed scheme is a FPGA implementation of Park-Miller Algorithm for generating sequence of Pseudo-Random keys. The properties like High speed, low power and flexibility of designed PRNG (Pseudo Random Number Generator) makes any digital circuit faster and smaller. The algorithm uses a PRNG Module, it contains 32-bit Booth Multiplier, 32-bit Floating point divider and a FSM module. After generating a sequence of 32-bit Pseudo-Random numbers we have used these numbers as a key to Encrypt 128-bit plain text to become a cipher text and by using the same key to decrypt the encrypted data to get original Plain text. The Programming is done in Verilog-HDL, successfully synthesized and implemented in XILINX Spartan 3E FPGA kit.

Corresponding Author:

Rohith S
Departement of Electronics and Communication Engineering,
Nagarjuna College of Engineering and Technology
Venkatagiri Kote, Devanahalli (T) Bangalore, Karnataka, India
Email: rohithvjp2006@gmail.com

1. INTRODUCTION

Encryption is used to securely transmit data in open networks. Each type of data has its own features, therefore different techniques should be used to protect confidential data from unauthorized access. The Pseudo-Random numbers are plays very important role in the communication system for protection of information from unauthorized access. The efficiency of the PRNG based cryptosystem is mainly based on the random key generated by its PRNG [1]-[5].

Depending on the application, three types of number sequences might prove as the “random numbers”. A numbers generated by a truly random process is most desirable. This type of sequence is called a random-number sequence, and the key problem is deciding whether or not the generating process is random. A more practical sequence is the pseudo-random sequence, a series of numbers generated by a deterministic process that is intended merely to imitate a random sequence but which, of course, does not correctly obey such things as the laws of large numbers. A quasi-random sequence is a series of numbers that makes no guaranty at being random but that has important predefine statistical properties of random sequences.

PRNGs are efficient, meaning that in a short time, they can produce many numbers and deterministic, means that if the starting point in the sequence is known a given sequence of numbers can be regenerate at a later time [1]. Efficiency is a nice characteristic, if our application needs many numbers, and determinism is handy if we need to replay the same sequence of numbers again at a later stage. PRNGs are typically periodic, which means that the sequence will sequentially repeat itself. While periodicity is

hardly ever a desirable characteristic, modern PRNGs have a period that is so long that it can be ignored for most practical purposes.

Pseudorandom sequences [1] have been widely used in various fields, navigation, including communications, radar technology, cipher technologies, remote control, measurements, and industrial automation [2]. For example, pseudorandom sequences have been used in error-correcting codes [3], spread spectrum communication [4], [5], and system identification and parameter measurements [6], [7]. Other example applications are found in surface characterization and 3D scene modeling [8]. The design of a general purpose pseudorandom sequence generator has matured and has already been commercialized [9]-[11].

Pseudorandom sequences are series of 1's and 0's that lack any definite pattern and look statistically independent and uniformly distributed. The sequences are deterministic, but existence of noise properties similar to randomness [12]. In general, a pseudorandom sequence generator is usually made up of shift registers with feedback. By linearly combining elements from the shift register and feeding them back to the input of the generator, we can obtain a sequence of much longer repeat length using the same number of delay elements in the shift register. Therefore, these blocks are also referred to as a linear feedback shift register (LFSR) [13], [14]. The length of the shift register, the number of taps and their positions in the LFSR are important to generate pseudorandom sequences with desirable auto-correlation properties [15].

The outputs from above said PRNGs suffer from some problems they include; Periods are shorter than expected for some seed states. Generated numbers have lack of uniformity of distribution, Correlation of successive values, Output sequence is poor dimensional distribution, The distances between where some values occur are distributed differently from those in a random sequence distribution.

Park-Miller algorithm is used to generate 32-bit sequences of key [16]. The Park-Miller algorithm satisfies the following three criteria to generate good pseudo random numbers: sequence is sufficiently Random, Sequence full period, Efficient Implementation with 32-bit arithmetic.

The generated sequence of 32-bit sequence keys are plays a important role in the case of stream cipher to encrypt and decrypt plain text. In cryptography, a stream cipher is asymmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (key stream). In a stream cipher each plaintext digit is encrypted one at a time with the corresponding digit of the key stream, to give a digit of the cipher text stream. Finally the generated sequence of 32-bit pseudo random keys and by same sequence of keys used to encrypt and decrypt plain text of 128-bit.simulation of proposed scheme is by using verilog-HDL and MATLAB.The implementation with the help of XILINX Spartan 3E FPGA kit.

The organization of this paper is as follows. Section II describes Algorithm design and architecture of the proposed PRNG, Section III describe the design of Encryption and Decryption block The implementation results and synthesis report discussed in section IV. Finally, the conclusion is presented in the last section.

2. PARK-MILLER ALGORITHM

Park-Miller algorithm [16] is based on the linear congruential form: $X+1 = A\ X \bmod (2^{31}-1)$. Where the constant value "A" chosen as 16,807.

```
Const
A=168007;          (2 < A < M-1)
M=2147483647;      (2^31-1)
q=127773;          (M div A)
r=2836;            (M mod A)
Var
hi: =in_seed div q;

lo: =in_seed mod q;

test: =a*lo –r*hi;
if test > 0 then
Out_seed:= test;
random: =out_seed / m;
else
Out_seed:= test +m;
random: =out_seed / m;
end;
```

Steps of the algorithm are as follows:

Initialize the input in_seed and parameters,

1. a = 16807 m= 2,147,483,647, q = 127773, r = 2836.
2. Compute the value of hi = in_seed div q and lo = in_seed mod q.
3. Then compute the corresponding test value.
4. If test > 0, save test as new out_seed, otherwise save test + m.
5. Output the new out_seed.
6. Iterate, and let the out_seed be the new in_seed.

As this algorithm is designed for computer system, the limitation of the precision of the operation system is considered. It is designed for any system whose precision is 32 bits or larger. In order to avoid overflow error, the seed adds a maximum allowable number m in the 32 bits system at before the output in the event that the value of the seed is not positive, as is illustrated in above algorithm [16].

The below Figure 1 shows the flow chart of Park Miller algorithm. Here first initialize 'm' (2^{31}-1), 'a' (16807), 'q' (127773), 'r' (2836), En=0 and In_seed is a user choice 32-bit value. After initializing these values for first random generation En = 0 at that time, the variables hi and lo have to be calculated as In_seed div q and In_seed mod q respectively. Then another variable Test as a*lo – r*hi, if this value is greater than zero then out_seed becomes Test, else out_seed becomes Test + m. finally random output will calculated as out_seed divided by m. at this stage first random output is generated. For the next pseudorandom number generation, En = 1 now out_seed becomes In_seed or hi and lo will calculated as out_seed div q and out_seed mod q there after same steps will continue to generate sequence of pseudorandom number generation.

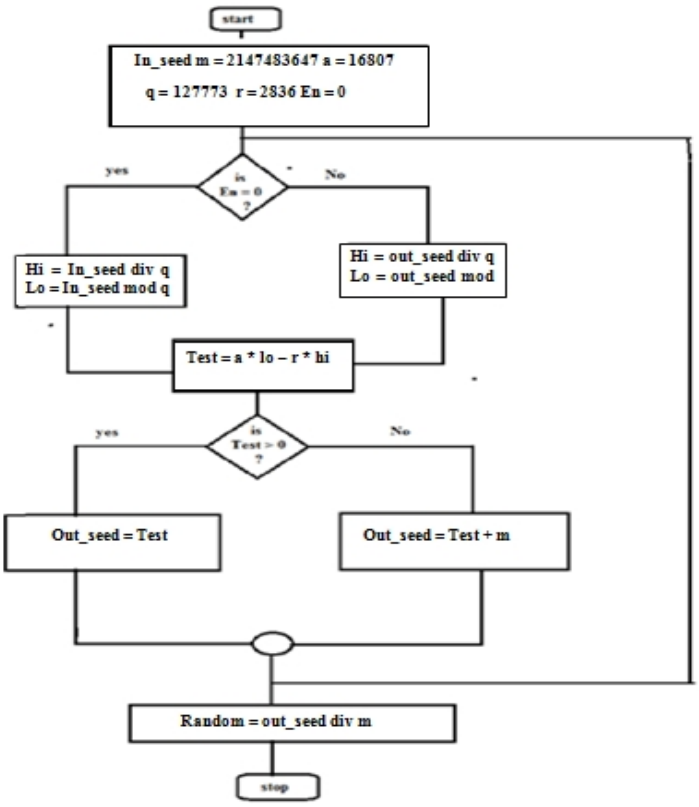

Figure 1. Flow Chart

The entire general block diagram is shown in Figure 2. It consists of PRNG module, Encryption Block, Decryption Block and slave register. These internal modules described in Figure 3 and Figure 5. PRNG block generates sequence of 32 bit pseudo random keys with the help of state machine, divider and multiplier. Here we used an AHB bus protocol for FPGA Implementation of proposed PRNG.

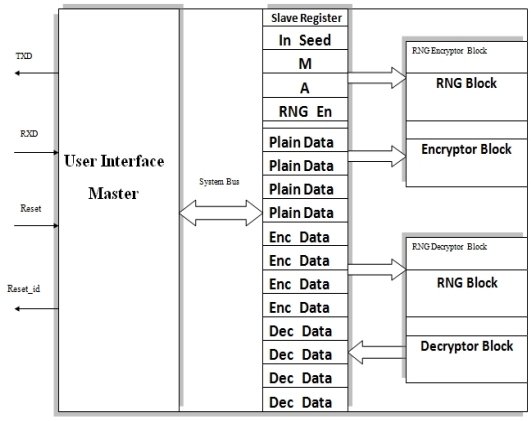

Figure 2. General Block Diagram

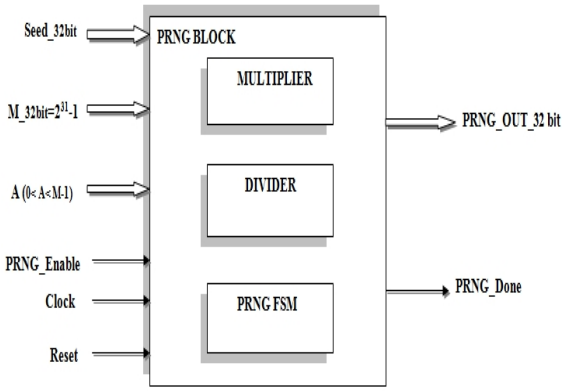

Figure 3. PRNG Block

The PRNG module will use this 32-bit input in_seed number to generate its first 32-bit output out_seed when 'Reset' is activated. The second random number generation is different from the first in the input data. The second output out_seed is generated using the first output out_seed as its in_seed number and the subsequent outputs are generated using their immediate previous outputs as in_seed number.

The PRNG module shown in Figure 3 is composed mainly of a state machine, 32 bit booth multiplier and 32 bit floating point divider. The state diagram of the PRNG is presented in Figure 4.

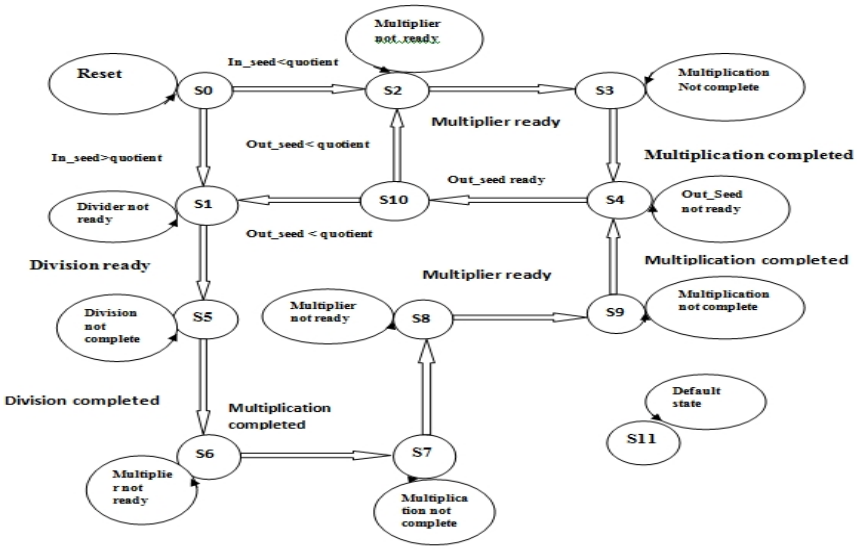

Figure 4. State Diagram

The default state is 11 and the PRNG will be activated when the 'Reset' signal is high, which changes the state to 0. From state 0, the input in_seed is compared against quotient 127773. If the input in_seed is less than 127773, hi =in_ seed div q will be 0 and lo = in_seed mod q will be the input in_ seed. So from Figure 2, only a multiplication out_seed = a * lo is needed and a shorter period of producing the random number can be achieved. The PRNG then goes to state 1 and the multiplication is taken from state 3 to state 4. On the other hand, if the input in_seed is greater than 127773, one division in_seed/q and two multiplications a*lo and r*hi are needed. So it goes to state 2 and follows the state path. From state 5 to state 9, these computations are taken. As soon as the operation completes, the status register is updated. When they converge at state 10, a random number is generated and this number is taken as the input to restart the flow at either state 1 or 2. As part of the PRNG module, the booth multiplier is implemented using repeated shift and addition algorithm. That is, shift the second multiplicand to correspond with each 1 in the first multiplicand and add to the result.

The floating point divider is designed using the repeated shift and subtraction algorithm which is the reverse of the multiplication by shifting and adding. Basically, division process can be summarized as follows: shift the divisor to correspond with each portion of the dividend, subtract divisor from that portion of the dividend and concatenate 0 or 1 to the result based on the result of the subtraction.

One good feature of the Park-Miller algorithm is that "m**"** is more than 2 billion and is therefore beneficial for series simulations. This algorithm this algorithm has passed a wide range of statistical tests and is known as one of the best [16].

3. ENCRYPTION AND DECRYPTION BLOCK

The generated sequence of 32-bit sequence keys are plays a important role in the case of stream cipher to encrypt and decrypt plain text. In cryptography, a stream cipher is asymmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (key stream). In astream cipher each plaintext digit is encrypted one at a time with the corresponding digit of the key stream, to give a digit of the cipher text stream. An alternative name is a state cipher, as the encryption of each digit is dependent on the current state

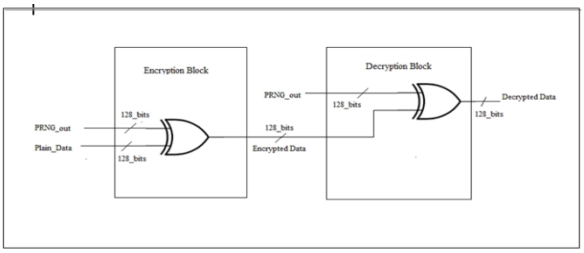

Figure 5. Encryption and Decryption of Plain Text

The design Block diagram of Encryption and Decryption is shown in Figure 5. Here we used a xor gates to encrypt and decrypt the plain data. Whatever the pseudo random keys generated from the designed PRNG we made that 32-bit sequence into sequence of 128 bit and after that we did bitwise xor operation with a Plain data the result obtained is a cipher text ie, encrypted data of 128 bit which cannot be readable so that we can transfer information. Then at the receiver side by using same 128 bit of PRNG key to Decrypt Encrypted data by the same bitwise xor operation.

4. IMPLEMENTATION AND RESULT

The Programming is done in Verilog-Hardware description language with behavioral level. To validate randomness of the PRNG sequence MATLAB Software used. The proposed module is successfully synthesized and implemented on XILINX Spartan 3E FPGA kit.

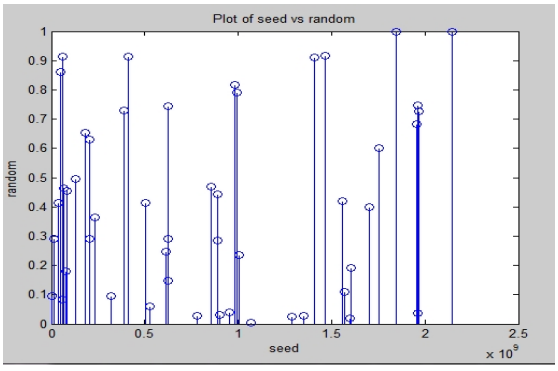

Figure 6. Plot of first 50 PRN sequence with seed input 140000

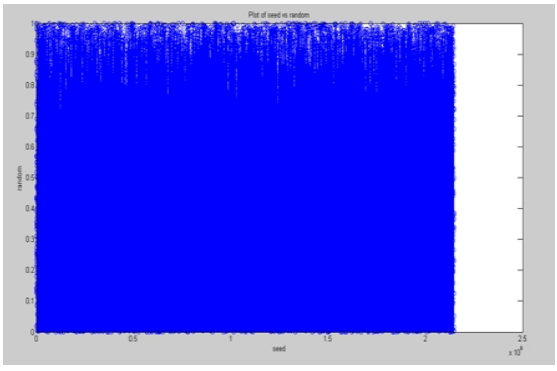

Figure 7. Mat lab first 20000 PRN's with input seed 140000

The first 50 and 20000 Pseudo random numbers generated as discussed in section II for an arbitrarily chosen inputs in_seed 140000, a is 16807, m is 2147483647 resulting plots are shown in Figure 6 and in Figure 7 respectively. The plot shown in Figure 6 random in nature. To verify the functionality of behavioral description of the algorithm simulation is carried out for same inputs in_seed, m and obtained Modelsim results is perfectly matching with MATLAB results

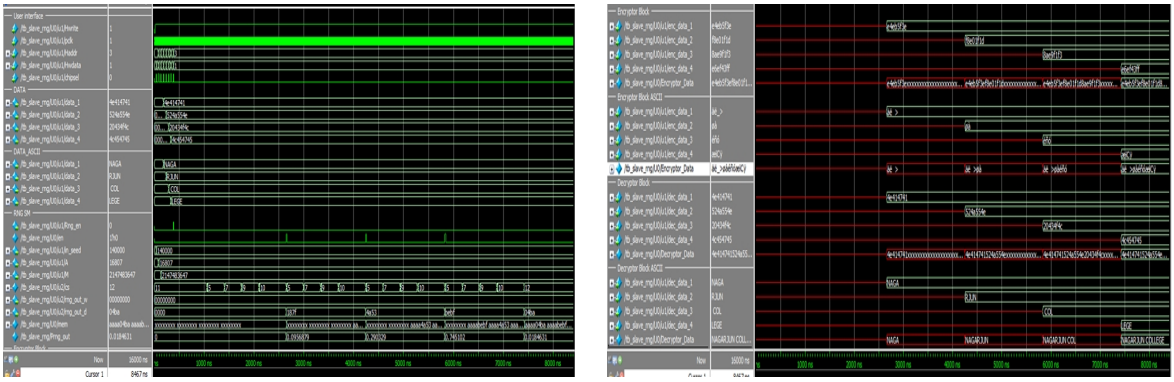

Figure 8. Modelsim simulation result of generated random numbers and Encryption - Decryption of Plain Text.

The pseudo random numbers generated from Park – Miller algorithm are used to encrypt the textual message. Generated sequence is converted into 32 bit length. The obtained 32 bit length of keys XOR ed with plain textual message converted into ASCII digits. The group of 4 ASCII digit form a frame of length 32 bit. In this paper "NAGARJUN COLLEGE" is chosen as a plain text and encrypted with the 32 bit random numbers. The obtained 32 bit to become a cipher text which is converted back to text. To decrypt the cipher text same sequence of keys are used and reverse process used to get back an original plain text

Table 1. Device Utilization Report of PRNG

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1161	18224	6%
Number of Slice LUTs	1497	9112	16%
Number of fully used LUT-FF pairs	796	1862	42%
Number of bonded IOBs	148	232	63%
Number of BUFG/BUFGCTRLs	5	16	31%

The Xilinx FPGA RTL schematic and synthesis report of proposed Pseudo Random number Generator is shown in Figure 9, Figure 10 and in Table 1. Maximum operating frequency is 191.131MHz.

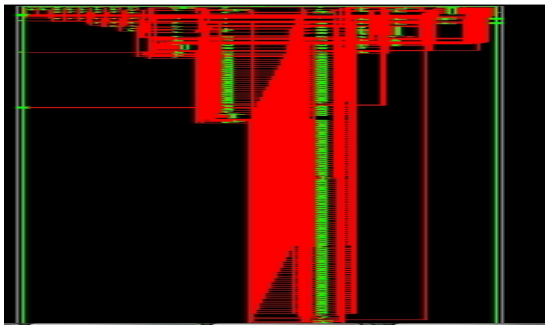

Figure 9. XILINX RTL Schematic of PRNG

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 5.232ns (Maximum Frequency: 191.134MHz)
   Minimum input arrival time before clock: 5.640ns
   Maximum output required time after clock: 5.622ns
   Maximum combinational path delay: No path found
```

Figure 10. Timing Report of PRNG

5. CONCLUSION

In today's world, all the e-commerce transaction happens over the internet. The data that is transferred over the internet needs to be secured before the transmission process happens. For these purposes, the data needs to encrypted while transmission and decrypted at receiving end with suitable key. The main input to these algorithms is the keys that are used for the encryption and decryption purposes. The keys need to be as random as possible in order to avoid any hacking of the data. on the other hand the Implementation, Speed of the encryption process is major challenge. The simple algorithm may be work very fast but may not secure the data. At the same time complex algorithm may provide better security but processing time may be large. This leads to Hardware description of the algorithm.

In this paper one such Hardware description of Park-Miller algorithm to generate the random number is discussed. The implementation of this algorithm has been done using the Verilog. The design provides a random number on each clock cycle. The random number generated are able to meet the standards set by the NIST and hence can be directly used for Cryptographic algorithm keys. This PRNG is based on simple algorithm and it can be deployed on general FPGA development board for design verification. The design can be implemented completely in digital circuit and does not require external components.

REFERENCES

[1] ZG Xiao. Pseudo-Random Sequence and Its Applications, Beijing, China: Nat. Defence Ind., 1985.
[2] L Xu, X Li. Dual-Channel Pseudorandom Sequence Generator with Precise Time Delay between Its Two Channels. *IEEE Trans. Instrum. Meas.*, 2008; 57(12): 2880-2884.
[3] CH Yen, BF Wu. An Error-Correcting Stream Cipher Design with State-Hopping Architecture. *J. Chin. Inst. Eng.*, 2005; 28(1): 9-16.
[4] XG Wangetal. Spread-Spectrum Communication Using Binary Spatiotemporal Chaotic Codes. *Phys. Lett. A*. 2005; 334(1): 30-36.
[5] HJ Kim, et al. PN Sequence Generation from 2-D Array of Shift Registers. *ETRI J*., 2005; 27(3): 273-279.
[6] T Johnsen, et al. *Simultaneous Use of Multiple Pseudo Random Noise Codes in Multistatic CW Radar*. Proc. IEEE Nat. Radar Conf., 2004: 266-270.
[7] DK Rollins, et al. A Quantitative Measure to Evaluate Competing Designs for Non-linear Dynamic Process Identification. *Can. J. Chem. Eng.*, 2006; 84(4): 459-468.
[8] HJW Spoelder, et al. Some Aspects of Pseudo Random Binary Array-Based Surface Characterization. *IEEE Trans. Instrum. Meas.*, 2000; 49(6): 1331-1336.
[9] R Shaltiel, C Umans. *Simple Extractors for All Minentropies and a New Pseudorandom Generator*. Proc. Annu. Symp. Found. Compute. Sci., 2001: 648-657.
[10] AH Tan, KR Godfrey. *The Generation of Binary and Near-Binary Pseudorandom Signals: An Overview*. Proc. IEEE Instrum. Meas. Technol. Conf., 2001; 2: 766-771.
[11] J Szczepanski, et al. Biometric Random Number Generators. *Comput. Secur.*, 2004; 23(1): 77-84.
[12] P Alfke. Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators. *Application Note, Xilinx Corp.*, 1995.
[13] Gold Code Generator Reference Design, Altera Application Note 295. 2003.
[14] F Principe, et al. *Rapid Acquisition of Gold Codes and Related Sequences Using Iterative Message Passing on Redundant Graphical Models*. Proc. Int. Conf. Military Commun. 2006.
[15] XD Lin, KH Chang. Optimal PN Sequence Design for Quasi synchronous CDMA Communication Systems. *IEEE Trans. Comm.*, 1997; 45(2): 221-226.
[16] SK Park, KW Miller. Random number generators: Good ones are hard to find. *Communications of the ACM*. 32(10): 1192-1201.