Implementation of pseudo random noise generator in FPGA for Free Space Optics BER testing

Nikolay Kolev[1] and Tzvetan Mitzev[2]

Abstract – **The way to determine the quality of communication link is to measure its Bit Error Ratio (BER). BER most often is measured by the so called brute force method. It is sending random bits through the system and calculating BER. The devices which measure BER are called Bit Error Rate Testers. They consist of pseudo random noise generator and comparing scheme. This report presents the creation of FPGA based system for testing BER in Free Space Optics Systems. The paper offers block and principle realisation of PRNG algorithm. There is synthesized HDL code for Spartan 6 FPGA chip. Our generator is tested in real conditions with oscilloscope. There is shown computer simulation and practical results.**

Keywords – Bit Error Rate, Free Space Optics, Pseudo Random Noise Generator, FPGA

I. Introduction

One of the most important parameter, for quality assessment in telecommunication systems is BER. It is a measure of the percentage of bits that a system does not transmit or receive correctly. The experiments connected with BER give us the information about the maximal bandwidth and quality of the FSO communication link. The device which measure BER is called Bit Error Rate Tester (BERT). There are a lot of companies that make BERTs such as Tektronix, Agilent, Rohde & Schwarz. Because of price of these commercial systems we decide to build our own BERT. The main block of BERT is the pseudo random noise generator which is explained in this work.

Random number generators are mainly used in telecommunications, cryptographic algorithms and authentication protocols. There are many different ways to generate pseudo random bit sequence. Most popular are ADC method , shift register method and software method [1].

ADC method can be achieved by feeding sinusoidal signal to the ADC, while at the output of the IC there is a random digitized signal. With ADC can be achieved hi speeds, but fast ADC's are expensive. In Software method we are restricted form computer's communication ports. We choose to work with shift register method because of the low price and hi speed of FPGA. In this method we have a couple of D type flip flops and one XOR logic element connected in series. The clock generator moves the logic levels through the scheme. In this generator the output is the last D type flip flop. In software generators we increment fast one or couple registers, it depends of the case. The random bits or bytes are obtained by reading registers in definite period of time. We decide to work with shift register method because of simple hardware implementation. This technique has good statistical properties and leads to very efficient hardware implementations.

FPGA are reconfigurable silicon chips which can be programmed in various different hardware configurations [2]. In this way we obtain completely different hardware implementations. The specific thing in this IC's is the ability to perform parallel processes unlike the MCU's. Each processing task is assigned to a different section of the chip, and can function autonomously without any impact from other logic blocks. The performance of single part of the application is not affected when we increase the number of processes. The FPGA's have a couple of advantages which are connected with better performance, low cost and good reliability. Due to the parallelism of these chips their computing power is almost equal to the Digital Signal Processor's computing power. The response times of each I/O pin is faster.

In this work FPGA implementation of PRNG is presented. This report is addition of our paper which presents Experimental setup for BER measuring of Free Space Optical System [3]. Due to the needs of increasing the throughput of our system we replace the existing discrete components based generator with FPGA chip. The advantages of new system are: easy reprogramming, high frequencies (500+ MHz), minimizing the physical dimensions.

II. Schematic design

The block scheme of generator is shown in fig.1. The operating principle is explained in the next paragraph.

The principle scheme of the pseudo random noise generator can be seen on fig.2. It possible to consist of several D type flip flops connected in series, one XOR logic gate and one inverter. In shown case there are four D type flip flops which Q outputs are connected to D inputs of next flip flop. The two inputs of the logic gate XOR are connected to the last Q output of the scheme, through inverter, and between two middle flip flops.

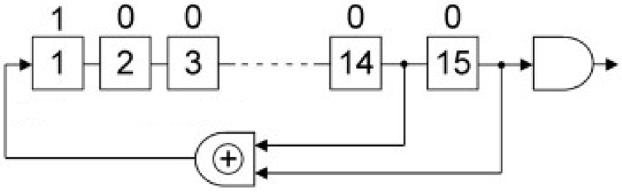

Fig.1. Block scheme of random number generator

[1] Nikolai Kolev is with the Faculty of Telecommunications at Technical University, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, E-mail: kolev@tu-sofia.bg.

[2] Tsvetan Mitsev is with the Faculty of Telecommunications at Technical University of Sofia. E-mail: mitzev@tu-sofia.bg

The four D type flip flops form 4 stages shift register. At every tick of the clock signal the logic level on every input is shifted one step left. In this situation the output of the XOR gate is a source of pseudo random bit sequence. The function of inverter is to start the generator, because in the beginning at every inputs and outputs there are only zeroes. The XOR gate need different levels to produce ones at his outputs, that's why there is inverter at the one of two inputs of the XOR gate [4].

The period of repetition depends on number of D type flip flops. In particular case it can be calculated by formula 1, where "n" is number of D type flip flops.

$$2^n - 1 \qquad (1)$$

In the shown scheme in fig.2 Feedback polynomial is (2), the period of repetition is 15 because 2^4 -1 = 15

$$x^4 + x^3 + 1 \qquad (2)$$

For example if the number of bits are 16 then the period of repetition will be 65536, because 2^{16}-1 = 65535. If the numbers of bits are 19, the period of repetition is 524287 and so on. So we can connect the necessary number of D type flip flops for desired period of repetition.

For implementation of the generation algorithm it is necessary to produce confirmation code for the FPGA chip. This code makes individual connections between the different parts in the chip. For code configuration tool we used Xilinx ISE Web Pack tool which is freely accessed in Xilinx web site [5]. For top layer of our project we used schematic design fig.2.

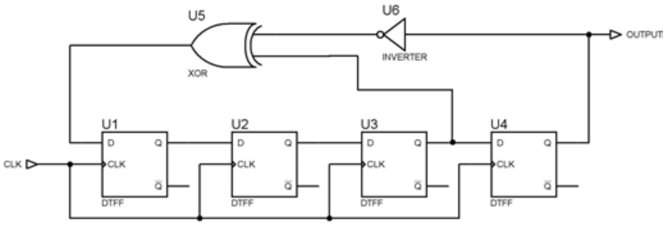

Fig.2. Principle realization of PRNG

The HDL code is too big to be shown in paper, that's why we will show a essential part of it in listing 1. In the beginning of code there is definition of standard libraries like ieee.std, ieee.numeric and so on. After that we have definition of inputs and outputs of the chip. At the next lines there is the configuration of the divider of the clock generator. After that we have defining inputs and outputs of the logic elements D flip flop, inverter and XOR gates of the scheme. At the end of the code there is shown interconnections between the logic elements and I/O pins.

Listing1 HDL Code for Spartan 6 board.

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;

entity tesALTYS is
   port ( clk  : in    std_logic;
          Led5 : out   std_logic;
          Led6 : out   std_logic;
          Led7 : out   std_logic);
end tesALTYS;
…………..
end component;
   attribute CLKFXDV_DIVIDE of DCM_CLKGEN : component is "2";
   attribute CLKFX_DIVIDE of DCM_CLKGEN : component is "1";
   attribute CLKFX_MD_MAX of DCM_CLKGEN : component is "0.000";
   attribute CLKFX_MULTIPLY of DCM_CLKGEN : component is "4";
   attribute CLKIN_PERIOD of DCM_CLKGEN : component is "0.0";
   attribute SPREAD_SPECTRUM of DCM_CLKGEN : component is "NONE";
   attribute STARTUP_WAIT of DCM_CLKGEN : component is "FALSE";
   attribute BOX_TYPE of DCM_CLKGEN : component is "BLACK_BOX";
component FD
      generic( INIT : bit :=  '0');
      port ( C : in    std_logic;
             D : in    std_logic;
             Q : out   std_logic);
   end component;
   attribute BOX_TYPE of FD : component is "BLACK_BOX";

   component INV
      port ( I : in    std_logic;
             O : out   std_logic);
   end component;
   attribute BOX_TYPE of INV : component is "BLACK_BOX";

   component XOR2
      port ( I0 : in    std_logic;
             I1 : in    std_logic;
             O  : out   std_logic);
   end component;
…………..
begin
   Led5 <= Led5_DUMMY;
   Led6 <= Led6_DUMMY;
   Led7 <= Led7_DUMMY;
   XLXI_7 : DCM_CLKGEN
…………
…………
XLXI_8 : FD
      port map (C=>XLXN_51,
                D=>XLXN_5,
                Q=>XLXN_4);

   XLXI_9 : INV
      port map (I=>XLXN_4,
                O=>XLXN_5);

   XLXI_10 : FD
      port map (C=>XLXN_4,
                D=>XLXN_7,
                Q=>XLXN_6);
XLXI_11 : INV
      port map (I=>XLXN_6,
                O=>XLXN_7);

   XLXI_12 : FD
      port map (C=>XLXN_6,
                D=>XLXN_10,
                Q=>XLXN_9);

   XLXI_13 : INV
      port map (I=>XLXN_9,
                O=>XLXN_10);

   XLXI_14 : FD
      port map (C=>XLXN_9,
                D=>XLXN_12,
                Q=>XLXN_11);

   XLXI_15 : INV
      port map (I=>XLXN_11,
                O=>XLXN_12);

   XLXI_16 : FD
      port map (C=>XLXN_11,
                D=>XLXN_14,
                Q=>XLXN_13);

   XLXI_17 : INV
      port map (I=>XLXN_13,
                O=>XLXN_14);

   XLXI_18 : FD
      port map (C=>XLXN_13,
                D=>XLXN_16,
                Q=>XLXN_15);
```

For our development we used evaluation board Altys Spartan 6 from DIGILENT (fig.3) [6] . It is very powerful board with rich peripheral devices such as LAN interface unit, a couple of switch units and micro switches, LED's, USB, UART, HDMI, external RAM memory. We choose to work with ALTYS because of it high operating frequencies 500+MHz and existence of Gigabit Ethernet.

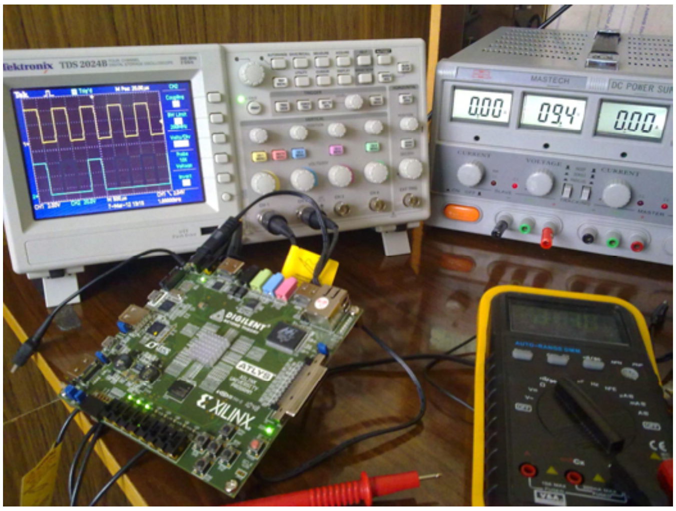

Fig.3 ALTYS Spartan 6 Board

III. Experimental Setup

The block scheme for measuring BER in FSO atmosphere channel can be seen at fig.4. It consists of Rx and Tx optical drivers, FSO atmosphere channel, counter device, BER measuring scheme and pseudo random noise generator. The information is passed through atmosphere channels forward and backward. At the end of the backward channel there is BER measuring scheme, which compare signals from the generator and atmosphere channel. If signals are different the scheme formed logical ones. Otherwise the scheme form logic zeroes, which mean no error [3] .

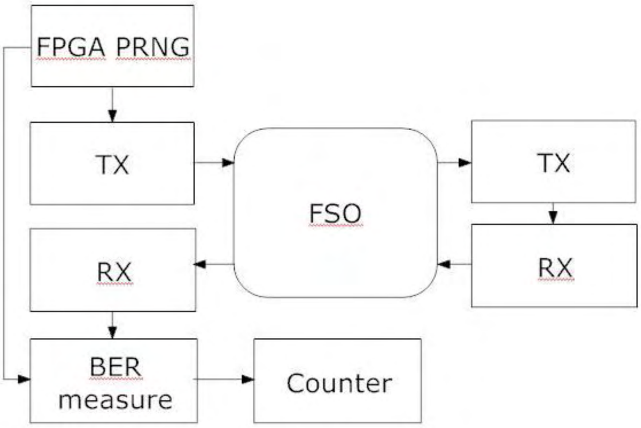

Fig.4 FSO system

We plan to combine our FSO system fig.5 and FPGA based PRNG for measuring BER in real conditions. We want to bind the results for BER with atmospheric effects such as fog, rain, snow. For reading the intensity of meteorological effects we will use meteorological station.

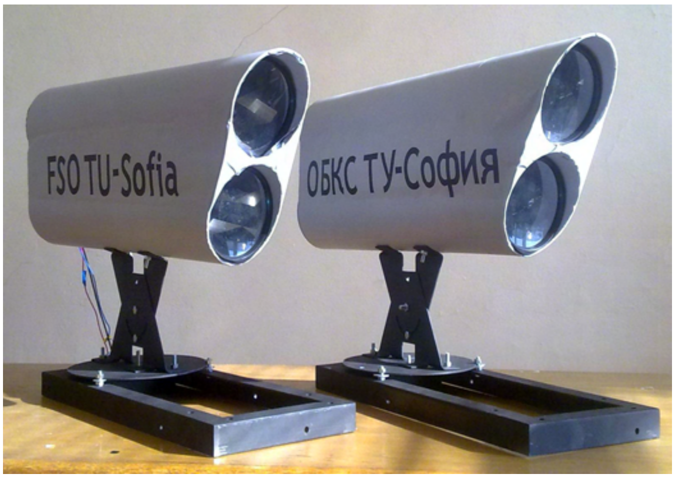

Fig.5. Scheme for measuring BER in atmosphere channel

IV. Simulation

Fig.6 and Fig.7 shows computer simulations and real oscillogram. In the upper part of the figures there is a PRNG signal, while in the bottom of the figures there is a clock signal [7].

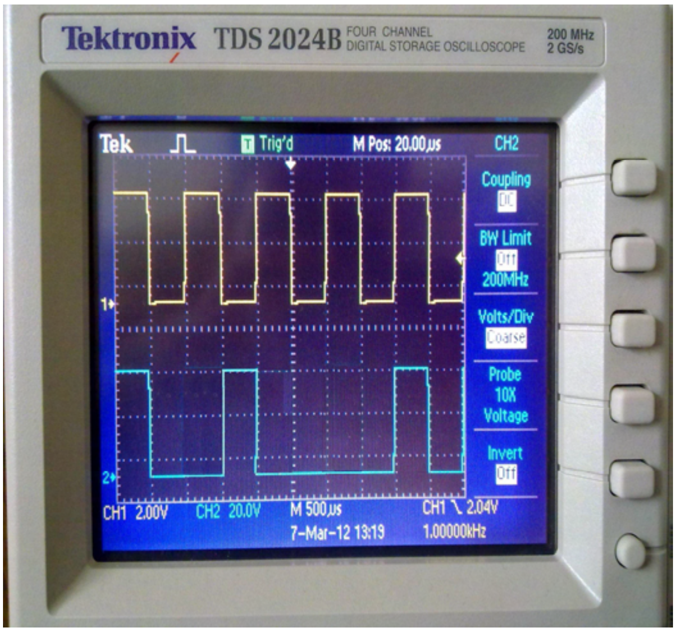

Fig.7. Oscillogram of PRNG signal

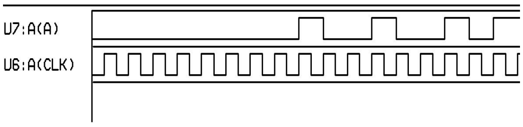

Fig.6. Computer simulation of PRNG signal

V. Conclusion

The described algorithm can be used for learning the dependence of the level of bit error in Free Space Optics systems [8]. This system is easy reprogramming, it can generate pseudo random bit sequence at frequency of 500Mhz, it have small physical dimensions. In future we plan to realize all blocks in fig. 3 with FPGA.

Research results presented in this publication are funded by Internal competition of TU-Sofia-2011, contr. Nb112pd033-7.

References

[1] Beker H., P.Fred,The Protection of Communications. Wiley-Interscience. p. 212. 1982.

[2] Pong P. Chu FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version (Feb 4, 2008)

[3] Kolev N., K. Dimitrov Y. Velchev T. Mitsev Experimental Setup for BER Measuring of Free Space Optical System , pp.214-217, Nis, Serbia , 2011.

[4] Anil K. Maini Defence Digital Electronics Principles, Devices and Applications
Research and Development Organization (DRDO), India

[5] www.xilinx.com

[6] http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,836&Prod=ATLYS

[7] Hartmann A. K. A practical guide to computer simulations World Scientific Publishing Company; Pap/Cdr edition (March 30, 2009)

[8] Stamatios V. Kartalopoulos Optical Bit Error Rate: An Estimation Methodology Wiley-IEEE Press 2004