IMPLEMENTATION OF FPGA-BASED PSEUDO-RANDOM WORDS GENERATOR

Volodymyr Opanasenko, Stanislaw Zavyalov, and Olexander Sofiyuk

Institute of Cybernetics of the National Academy of Science of Ukraine,
40, Glushkov Avenue, Kyiv, 03187, Ukraine.
Authors' e-mail*: opanasenkoincyb@gmail.com, radionix13@gmail.com, otsof@yandex.ua*

Submitted on 12.11.2020

Abstract **– A hardware implementation of pseudo-random bit generator based on FPGA chips, which use the principle of reconfigurability that allows the modernization of their algorithms and on-line replacement of the internal structure (reconfiguration) in the process of functioning have been considered in the paper. Available DSP blocks embedded into the structure of FPGA chips allow efficient hardware implementation of the pseudorandom bit generator through the implementation of the basic operations of multiplication with accumulation on the gate level. Using CAD ISE 14.02 Foundation and VHDL language three types of pseudo-random bit generators have been implemented on Spartan series chip 6SLX4CSG225-3, for which time and hardware expenses are represented. Using the simulating system ModelSim SE 10.1c, timing diagrams of simulation for these structures have been obtained.**

Index Terms**: pseudorandom bit generator, simulation, CAD, DSP, FPGA**

I. INTRODUCTION

Now, in connection with the intensive development of mobile communication systems with code channel distribution, the problem of technical modernization of devices that implement algorithms for generating pseudo-random words (PRW) [1, 3], using a modern element base - the chips with programmable logic, has been raised.

FPGAs are increasingly used in the world to create modern control systems, high-performance data processing, digital signal processing, telecommunications support and others [4, 5, 8].

PRW generation (sampling) is performed by pseudo-random numbers of sensors. The number of pseudo-random numbers is in a fairly wide range: from tens of thousands for simple tasks, to hundreds of thousands or more for complex systems. Therefore, an important problem is to ensure high speed.

Sensors with a given distribution law (for example, normal, exponential and others) are usually implemented programmatically, their work is based on the conversion of a sequence of pseudo-random numbers with a uniform distribution in the interval [0, 1] in PRN with a given distribution law. Therefore, the quality and efficiency of the formation procedures largely depend on the properties of the sensor of evenly distributed pseudo-random numbers [2, 3].

Today, there are a large number of algorithms for forming pseudo-random words, which have their own advantages and disadvantages and are used in various applications.

The most widespread in practice are linear congruent methods [2, 3, 9] of generating pseudo-random numbers with uniform distribution and formation on their basis of PRW of a given length, which have given properties. In General, the algorithm of such sensors is implemented using a recurrent relationship:

$$x_{n+1} = \sum_{i=0}^{j} a_i x_{n-1} + c (\mathrm{mod}\, M), \quad (1)$$

where: $a_0, a_1, ..., a_j$, $c > 1, M > 1$, and the obtained numbers x_1, x_2, … x_j are integers. Module M means: the number $A = \sum_{i=0}^{j} a_i x_{n-1} + c$ is divisible to M; the obtained integer q and integer remainder x_{n-1} are presented as:

$$A = qM + x_{n+1}; 0 \le x_{n-1} \le M - 1.$$

Since x_{n+1} – the number that is between 0 and M, it must still be divided into M to get a number that is between 0 and 1:

$$R_{n+1} = \frac{x_{n+1}}{M}.$$

Sequences obtained using linear congruent methods are repeated periodically. This is because numbers x can only take values $0, 1, 2, ..., (M-1)$. The maximum length of the sequence period cannot exceed $M = 2m$, so take, as a rule $m = N$, where N – the number of significant digits to represent integers.

From relation (1) we can obtain various modifications of the linear algorithms of pseudo-random number sensors.

The mixed congruent method of generating pseudo-random numbers proposed by Lemer is obtained from (1) by $a_1 = a_2 = ... = a_j = 0$ and assuming $a_0 > 0, c > 0$. Then:

$$x_{n+1} = a x_n + c (\mathrm{mod}\, M). \quad (2)$$

You can improve the algorithm that implements the multiplicative congruent method.

To do this, in (1) we substitute $c = a_1 = a_2 = ... = a_j = 0$ and accept $a_0 > 0$. In this case:

$$x_{n+1} = a x_n + c.$$

The quality of numbers that are calculated by this algorithm is worse than in algorithm (2), but the program that implements it is simpler and allows you to generate numbers with higher performance. This is important when experimenting with simulation models, because the run time is reduced.

The numbers $c, M, a_0, a_1, ..., a_j, x_0$ are called sensor parameters. x_0 is the initial value of the number from which the sample generation begins. The quality of the sample generation depends on the sensor parameters, so they cannot be selected at random. The rules for selecting the parameters of linear sensors are considered in [2].

II. THE ALGORITHMS OF PRS FORMATION

Algorithm 1.

PRS is formed by a pseudo-random sequence generator (PRSG) according to the following formula:

$$X_{i+1} = \left[A \times X_i + B_{i+1} \right], \quad (3)$$

where: $B_{i+1} = B_i + 1$ (when overflowing B_i the information in the lower categories is not distorted); X_i – current n –bit word PRS; N – the number of words PRS.

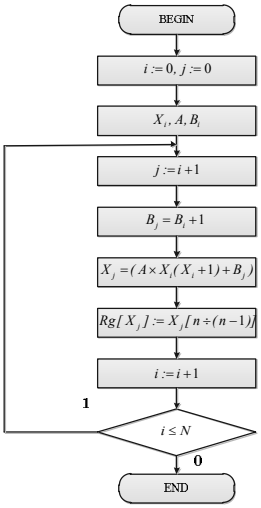

Fig. 1. Block diagram of the calculation of the pseudo-random sequence according to algorithm 1

Since the result X_{i+1} will be $2n$ –bit, we take only n the lower digits of the result.

The value of n –bit words A, X_0, B_0 is a constant.

The block diagram of sequence formation is shown in Fig. 1.

Algorithm 2.

PRS is formed by a pseudo-random sequence generator by (3).

Since the result X_{i+1} will be $2n$ –bit, we take only n bits (for odd i – only n lower bits, for even i – only n medium bits) of the result $(i/2 = z + w)$, where: z – whole, w – fractional part of the division result).

The block diagram of sequence formation according to algorithm 2 is shown in Fig. 2.

Algorithm 3.

PRS is formed by a pseudo-random sequence generator to the following formula.

$$X_{i+1} = [A \times X_i (X_i + 1) + B_{i+1}], \quad (4)$$

where: $B_{i+1} = B_i + 1$ (when overflowing B_i the information in the lower categories is not distorted); X_i – current 16 –bits PRS word; $(A \times X_i)$ – (only the n lower digits of the multiplication result are taken); $(A \times X_i \times X_i)$ – (only the n medium digits of the multiplication result are taken);

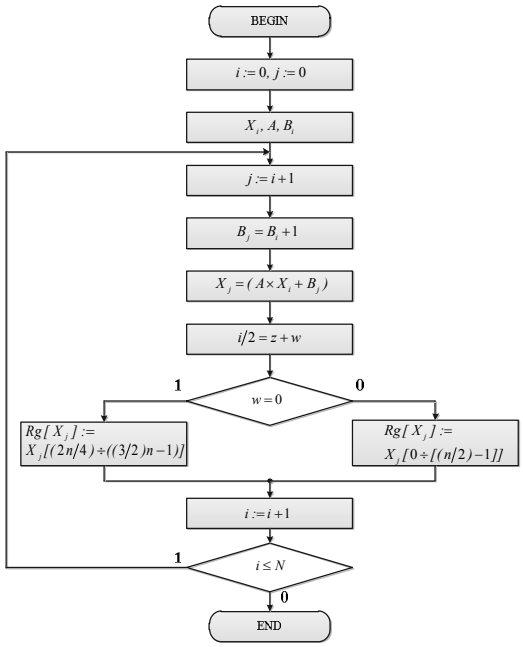

Fig. 2. Block diagram of the calculation of the pseudo-random sequence according to algorithm 2

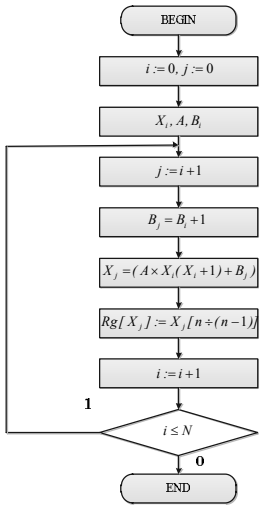

Fig. 3. Block diagram of the calculation of the pseudo-random sequence according to algorithm 3

Since the result X_{i+1} will be $2n$ –bits (when the amount is overflowed, the information in the lower digits is not distorted), we take only the n lower digits of the result. The block diagram of sequence formation according to algorithm 3 is shown in Fig. 3.

III. IMPLEMENTATION OF PRW FORMATION ALGORITHMS

Initial data for the development of the algorithm for the formation of PRW.

The bit size of pseudo-random words is n. Numerosity of PRW which is required to receive – N.

Frame encryption requires the N words PRW, then we will determine:

$$\langle X_1, B_1 \rangle, \langle X_2, B_2 \rangle, \ldots, \langle X_N, B_N \rangle.$$

Algorithm 1.

We will use (1).

This algorithm for FPGA-based implementation has the following functional diagram, which is shown in Fig. 4 (where: CnB_i – counter that implements the increment; $Rg(X_i)$ – register for storing n–bit values X_i ; $Rg(A)$ – register for storing n–bits constant $Rg(A)$).

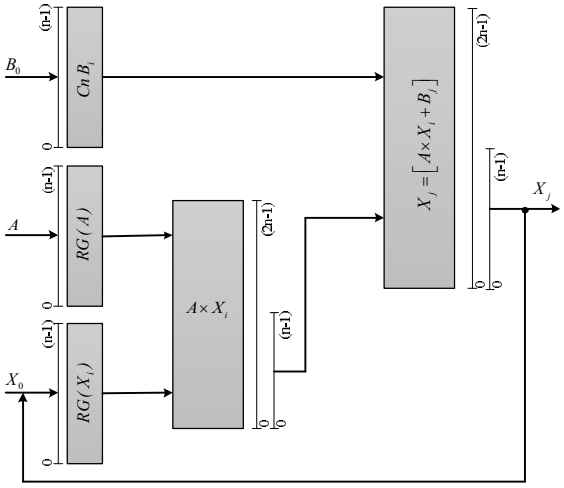

Fig. 4. Functional diagram of the pseudo-random word generator, which implements the algorithm 1

Algorithm 2.

We will use (3).

The functional diagram of the pseudo-random word generator based on the proposed algorithm is shown in Fig. 5:

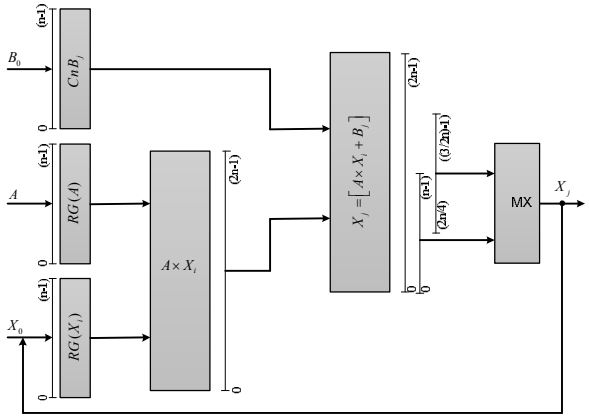

Fig. 5. Functional diagram of the pseudo-random word generator, which implements the algorithm 2

where: MX is a multiplexer that transmits the n lower bits of the result X_j to the output for odd i , or n medium bits for even i).

Algorithm 3.

We will use (4).

The functional diagram of the pseudo-random word generator based on the proposed algorithm is shown in Fig. 6.

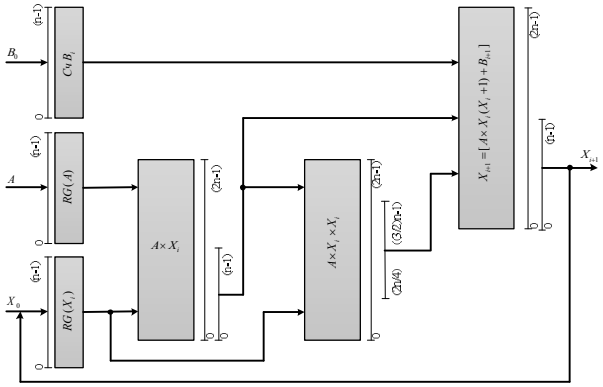

Fig. 6. Functional diagram of the pseudo-random word generator, which implements the algorithm 3

IV. HARDWARE IMPLEMENTATION OF ALGORITHMS FOR FORMING PSEUDO-RANDOM WORDS.

Consider the example of developing a pseudo-random number generator by describing in VHDL using the ISE Foundation package, its modeling using the ModelSim system [7] based on the crystal 6SLX4CSG225-3 series Spartan6 [10] for bitwise pseudo-random words – n bits and the number of PRW to be obtained – N .

The stages of development involve verification of the project by the simulation method, in the process of which the inputs of the logical model of the designed device are fed input effects in the form of virtual signals (test-bench) generated by the developer, i.e., using the stand described below.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY PVS_1_TB IS
END PVS_1_TB;

ARCHITECTURE behavior OF PVS_1_TB IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT PVS_1
   PORT(
      CLK  : IN  std_logic;
      A    : IN  std_logic_vector(15 downto 0);
      X0   : IN  std_logic_vector(15 downto 0);
      B0   : IN  std_logic_vector(15 downto 0);
      X1   : OUT  std_logic_vector(15 downto 0);
      X2   : OUT  std_logic_vector(15 downto 0);
      X3   : OUT  std_logic_vector(15 downto 0);
      X4   : OUT  std_logic_vector(15 downto 0);
```

```
      X5     : OUT  std_logic_vector(15 downto 0)
    );
  END COMPONENT;

  --Inputs
  signal CLK  : std_logic := '0';
  signal A   : std_logic_vector(15 downto 0) := (others => '0');
  signal X0     : std_logic_vector(15 downto 0) := (others => '0');
  signal B0     : std_logic_vector(15 downto 0) := (others => '0');

   --Outputs
  signal X1     : std_logic_vector(15 downto 0);
  signal X2     : std_logic_vector(15 downto 0);
  signal X3     : std_logic_vector(15 downto 0);
  signal X4     : std_logic_vector(15 downto 0);
  signal X5     : std_logic_vector(15 downto 0);

  -- Clock period definitions
  constant CLK_period : time := 10 ns;

BEGIN
   -- Instantiate the Unit Under Test (UUT)
  uut: PVS_1 PORT MAP (
       CLK => CLK,
       A  => A,
       X0    => X0,
       B0    => B0,
       X1    => X1,
       X2    => X2,
       X3    => X3,
       X4    => X4,
       X5    => X5
      );

   tb : PROCESS
  BEGIN
    CLK <= '1'; wait for 12.5 ns;
    CLK <= '0'; wait for 12.5 ns;
  END PROCESS;

   tb1 : PROCESS
  BEGIN
    A <= X"1357"; wait;
  END PROCESS;

   tb2 : PROCESS
  BEGIN
    X0 <= X"2468"; wait;
  END PROCESS;

   tb3 : PROCESS
  BEGIN
    B0 <= X"ABCD"; wait;
  END PROCESS;

END;
```

The simulation results (time diagrams) of the proposed random word generators for the corresponding algorithms are shown in Fig. 7-9.

The obtained data will be used in the process of word encryption, for transmission over the radar line, as well as in the process of decrypting words after receiving parcels at the facility.

As a result of the implementation of PRW generators by three algorithms, the following characteristics are obtained, given in Table.

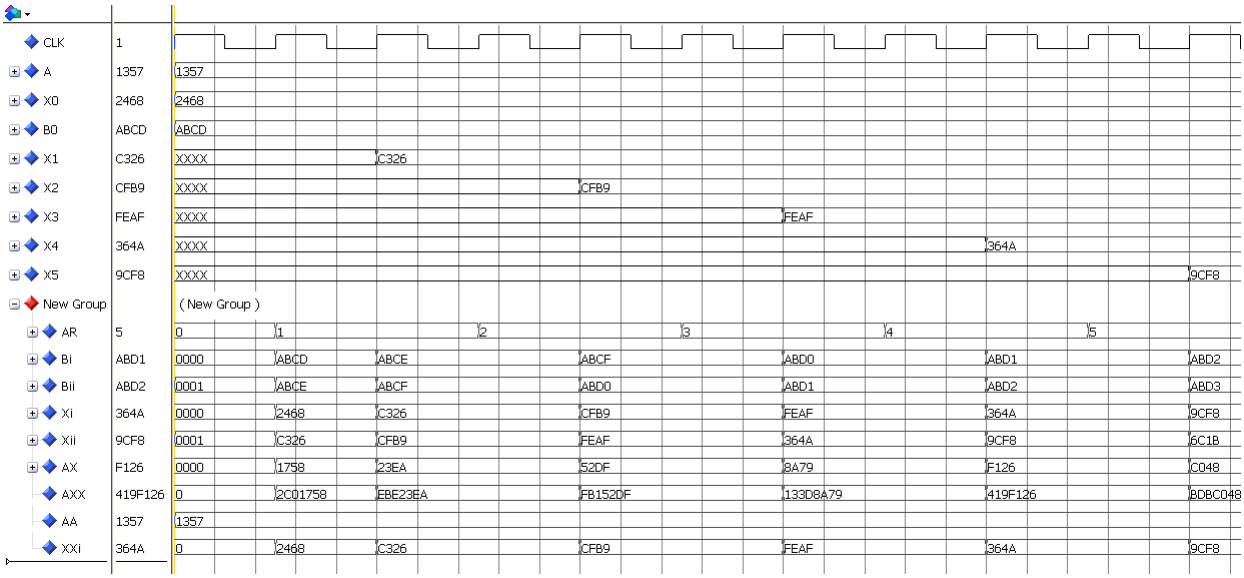

Fig. 7. Time diagram of the PVA generator according to algorithm 1

Table

Estimates of hardware and time costs in the implementation of PRW generators

Algorithm Type	T_{CLK}, нс	DSP		Numerosity Tg			Numerosity LUTs		
		Used	Available	Used	Available	%	Used	Available	%
Algorithm 1	5,773	1	8	103	4800	2	108	2400	4
Algorithm 2	10,750	2	8	103	4800	2	92	2400	3
Algorithm 3	6,224	2	8	87	4800	1	45	2400	1

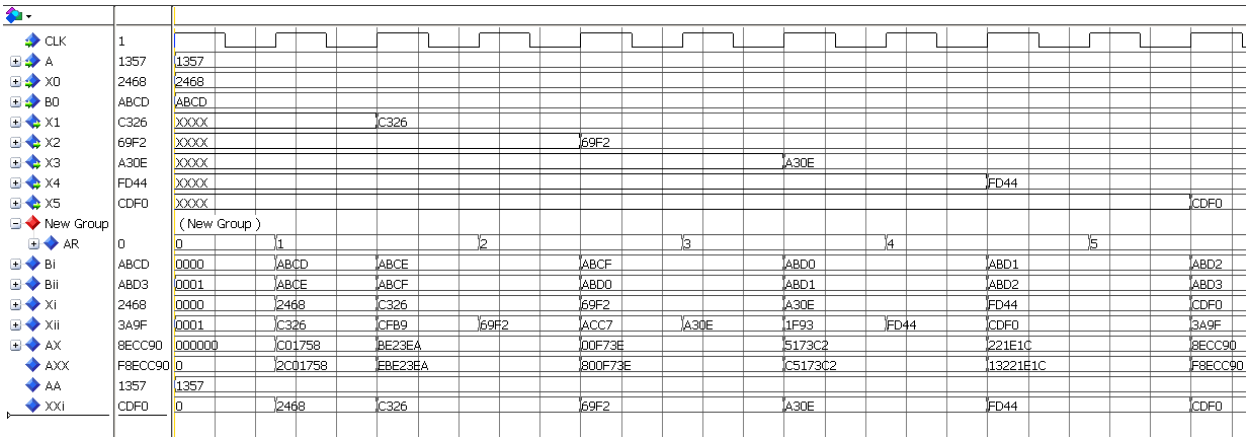

Fig. 8. Time diagram of the PVA generator according to algorithm 2

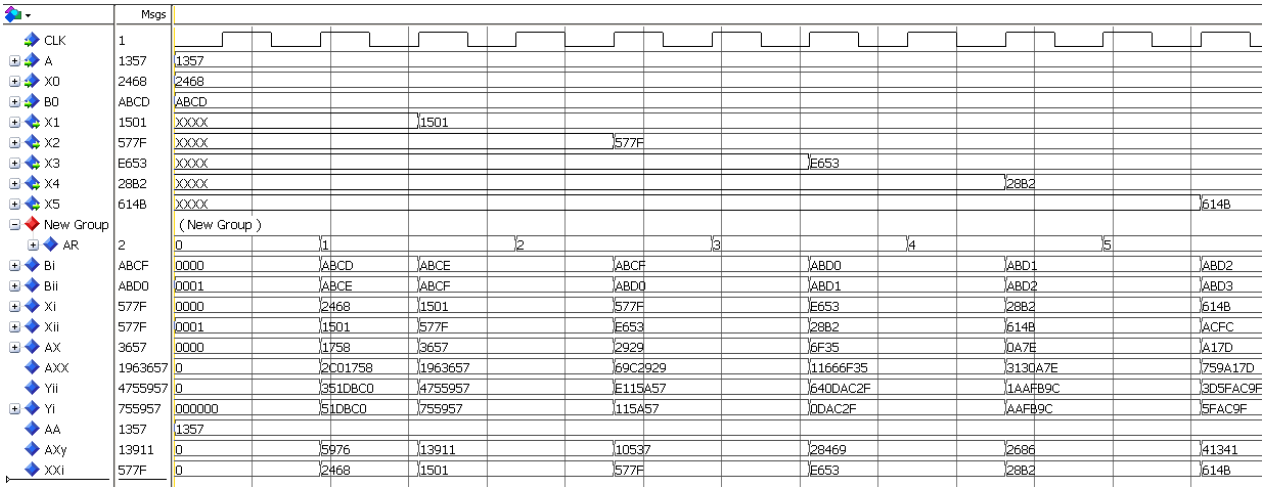

Fig. 9. Time diagram of the PVA generator according to algorithm 3

V. CONCLUSIONS

FPGA-based generator was developed for the implementation of pseudo-random word generation algorithms using the Xilinx ISE computer-aided design (CAD) system ISE 14.02 Foundation by Xilinx and its modeling was performed using the ModelSim SE 10.1c system. The advantage of the developed devices over the existing analogues is the use of the principle of reconfigurability to build high-performance computer tools, which provides opportunities to upgrade algorithms and quickly replace their structure (reconfiguration) during operation.

REFERENCES

[1] Knuth, Donald E. Seminumerical Algorithms. *The Art of Computer Programming*. (vol. 2). Third edition. Boston: Addison-Wesley, 1998. P. 764.

[2] V.V. Korchinsky, K.M. Filkin, "On the choice of the primary sensor for the simulation tasks". *Modeling and information technology*, vol. 42, 2007. pp. 81-90. (In Russian)

[3] A.A. Lavandsky, "Quality assessment of pseudo-random number generators by argest reproduction error distribution law". Bulletin of Khmelnytsky National University, no. 1, 2014, pp. 113-116. (In Russian)

[4] A.V. Palagin, and V.N. Opanasenko, *Reconfigurable computing systems*. Kiev, Prosvita Publ., 2006. 295 p. (In Russian).

[5] Available at http://www.xilinx.com/products/design-tools/ise-design-suite.html.

[6] ModelSim. ASIC and FPGA design / Available at http: // www.mentor.com/products/fv/modelsim/

[7] Available at http://www.xilinx.com/products/design-tools/ise-design-suite.html.

[8] Random Number Generator Results. Available at http://www.cacert.at/cgi-bin/rngresults.

[9] Spartan-6 Family Overview. Product Specification DS160 (v2.0), October 25, 2011. Xilinx, Inc. 11 p.

[10] Spartan-6 FPGA DSP48A1 Slice. User Guide, UG389 (v1.2) May 29, 2014. Xilinx, Inc. 46 p.

V. Opanasenko was born in Uzbekistan in 1957. He received an Engineer Degree in Radio-electronic Engineering from the Kazan Aircraft Institute (Kazan, Russia) in 1979 and PhD degree in Computer Systems in 1987 and Dr.Sc degree in Computer Systems at V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine in 2007.

From 1979 to 1982 he was an engineer in research and production association "Cybernetics" of the Academy of Sciences of Uzbekistan. After graduating from full-time graduate school (1982-1985) at V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine from 1985 to the present he has been working at the Microprocessor Engineering department, has passed the way from junior researcher to leading researcher.

Fields of research are FPGA-based reconfigurable computer system, modeling of computer system. Author of more than 150 publications.

S. Zavyalov was born in city of Polevskoy, Sverdlovsk region, Russia, in 1964. He received the Engineer Degree in Radio-electronic Engineering at Kyiv Higher Military Aviation Engineering School in 1987 and Ph.D. degree. From 1985 to the present he has been working as a director of "Radionix" Limited Liability Company.

Fields of research are radar, digital signal processing. He is the author of more than 27 scientific publications, including scientific articles and patents for inventions.

Sofiyuk O. was born in Cherepashintsi village, Vinnytsia region, Ukraine, in 1949. He received an Engineer Degree in Radio-electronic Engineering at Ryazan Radio Engineering Institute (Ryazan, Russia) in 1972. Since 2004 he has been working as a researcher at the Microprocessor Engineering Department of V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine.

Fields of research are design theory of the FPGA-based problem-oriented devices and systems. He is the author of more than 30 publications.