Fibonacci Counter based on Zeckendorf's Theorem (Boolean Realization)

Alexey Stakhov [1], Alexey Borisenko [2], Svetlana Matsenko [2]
[1]International Club of the Golden Section (Canada); E-Mail: goldenmuseum@rogers.com
[2]Sumy State University (Ukraine); 5352008@ukr.net
[2]Sumy State University (Ukraine); s.matsenko@mail.ru

Abstract

The purpose of this article is to present the research results on the development of the original noise-immune Fibonacci counter based on Zeckendorf's Theorem. The main peculiarity of the counter consists in the fact that we use only the so-called minimal forms of the Fibonacci code for error detection what increases noise immunity and informational reliability. The Fibonacci counter demonstrates significant benefits in comparison with the known Fibonacci counters based on the "convolutions" and "devolutions," both in speed and error detection ability. The Fibonacci counter can be the important step for designing noise-immune Fibonacci microcontrollers and microprocessors.

Keywords*:* Fibonacci numbers, Zeckendorf's Theorem, Fibonacci code, Fibonacci counter,Fibonacci computers

MSC: 68M15, 68M20, 68P30, 11T71, 94B60.

1. Introduction

A theory of Fibonacci numbers in modern mathematics began to develop rapidly starting since 60th years of the 20 century. In 1961 he Soviet mathematician Nikolai Vorobyov published the book "Fibonacci Numbers" [1]. This book has been reprinted many times and translated into many languages. In 1963 a group of the American mathematicians created the mathematical Fibonacci Association and began to publish *The Fibonacci Quarterly*. The book "Fibonacci and Lucas Numbers," published by the American mathematician Verner Hoggatt in 1969 [2], until now is one of the best books in this field. In 1992 a group of Slavic Fibonacci scientists (Ukraine, Russia, Belarus) created the Slavic Golden Group, which played a great role in the development of this research area in Ukraine, Russia, Belarus and other countries. In 1993 the Group was reorganized into the International Club of the Golden Section. At the initiative of the Club, in 2005 the Academy of Trinitarism (Russia) has organized the Institute of the Golden Section, which united all Slavic scholars in the field. In 2010, according to the initiative of the Club, the International Congress on the Mathematics of Harmony has been held in the Odessa National University (Ukraine).

In 1939 the Belgian medical doctor, army officer and amateur of mathematics Eduardo Zeckendorf (1901-1983) had proved in 1939 an interesting theorem, known as **Zeckendorf's Theorem** [3].

Although Zeckendorf did not have any relation to computers, his theorem is a very important mathematical result of fundamental importance for computer technology. This theorem is concerning to the representation of positive integers as a sum of Fibonacci numbers. Thus, Eduardo Zeckendorf predicted by his famous theorem a principally new way in the development of computer technology – **Fibonacci computers**, which can develop in modern science starting since 60th years of 20 c. [4-19].

The purpose of this article is to describe a new computer device, the original Fibonacci counter, based on Zeckendorf's Theorem. The Fibonacci counter demonstrates significant benefits in comparison with the known Fibonacci counters [9-12], both in speed and error detection ability what improves noise immunity and informational reliability of computers. The Fibonacci counter, based on Zeckendorf's Theorem, is an important step in designing noise-immune Fibonacci microcontrollers and microprocessors.

2. Mathematical foundations

2.1. The simplest mathematical identities for the Fibonacci numbers

Fibonacci numbers are numerical sequence, generated with the following recurrence relation:

$$F_i = F_{i-1} + F_{i-2}; \quad F_1 = F_2 = 1 \tag{1}$$

We emphasize that in mathematics [1,2] the **Fibonacci numbers** are called a recurrence numerical sequence, which begins with two 1's, and the next elements of the Fibonacci sequence are calculated according to the recurrence relation (1), that is,

$$F_n : 1,1,2,3,5,8,13,21,34, \tag{2}$$

We will use here the following simplest mathematical properties of the Fibonacci numbers [1, 2]:

$$F_1 + F_2 + \ldots + F_n = F_{n+2} - 1 \tag{3}$$

$$F_1 + F_3 + F_5 + \ldots + F_{2k-1} = F_{2k} \tag{4}$$

$$F_2 + F_4 + F_6 + \ldots + F_{2k} = F_{2k+1} - 1 \tag{5}$$

2.2. Fibonacci code

Fibonacci code is the following positional representation of natural numbers:

$$N = a_n F_n + a_{n-1} F_{n-1} + \ldots + a_i F_i + \ldots + a_1 F_1 \tag{6}$$

where $a_i \in \{0,1\}$ is a binary numeral, $F_i \ (i = 1, 2, 3, \ldots, n)$ is the weight of the *i*-th digit.

The Fibonacci code is similar to the classical binary code:

$$N = a_n 2^{n-1} + a_{n-1} 2^{n-2} + \ldots + a_i 2^{i-1} + \ldots + a_1 2^0 \qquad (7)$$

where $a_i \in \{0,1\}$ is a binary numeral, $2^{i-1}\left(i = 1, 2, 3, \ldots, n\right)$ is the weight of the *i*-th digit.

The following mathematical property of binary numbers

$$2^0 + 2^1 + 2^3 + \ldots + 2^{n-1} = 2^n - 1 \qquad (8)$$

is very important for designing binary counters. According to (8) the transition of the binary combination $\underbrace{011...11}_{n}$ to the next binary combination $\underbrace{100...00}_{n}$ is performed with carry-over of the bit 1 from the preceding digits to the next digit. The problem transferring bits from the junior to senior digits is a major problem at designing classical binary counters.

Comparing the property (8) for the binary numbers to the similar properties (3)-(5) for the Fibonacci numbers, we see that the Fibonacci counter has a much greater opportunity to create effective algorithms for Fibonacci counters. This idea underlies the Fibonacci counter presented in this article.

The brief notation of the *n*-bit Fibonacci code (6) consists of the *n* bits, starting from the higher bit a_n, and represents the positive integer *N* in the form:

$$N = a_n a_{n-1} \ldots a_i \ldots a_2 a_1 \qquad (9)$$

The brief notation (9) is called **Fibonacci representation** of positive integer *N*.

We emphasize that Fibonacci code (6) is similar to the binary code (7), because the Fibonacci representation (9) consists only of the bits 0 and 1. This means that the Fibonacci counter and Fibonacci microprocessors and microcontrollers are similar to the classical binary counters, microprocessors and microcontrollers, because they use the same binary elements and binary logic what is important for microelectronic technology.

2.3. Zeckendorf's theorem

Let us prove the following theorem.

Theorem 1. Arbitrary positive integer *N* has one and the only representation in the form:

$$N = F_i + r \qquad (10)$$

where $F_i\left(i = 2, 3, 4, \ldots\right)$ is some Fibonacci number, *r* is some nonnegative integer, which satisfies to the conditions:

$$0 \le r < F_{i-1} \qquad (11)$$

Proof. Consider the sequence of Fibonacci numbers

$$\{1, 1, 2, 3, 5, 8, 13, \ldots, F_i, F_{i+1}, \ldots\} \qquad (12)$$

Since the numerical sequence (12) is a strictly increasing one, starting from the lowest Fibonacci number $F_2 = 1$, this means that we can always choose in the sequence (12) one and the only couple of the adjacent Fibonacci numbers F_i, F_{i+1}, which are connected with the positive integer N by the following inequality:

$$F_i \le N < F_{i+1} \tag{13}$$

Consider the difference

$$r = N - F_i \tag{14}$$

whence follows:

$$N = F_i + r \tag{15}$$

Subtracting the Fibonacci number F_i from each member of the inequality (13), we get:

$$0 \le N - F_i < F_{i+1} - F_i \tag{16}$$

It follows from the recurrence relation (1) that

$$F_{i+1} - F_i = F_{i-1} \tag{17}$$

If we take into consideration the formulas (14) and (17), then the inequality (16) can be written as follows:

$$0 \le r < F_{i-1}.$$

Theorem is proved.

Theorem 1 can be used for the proof of **Zeckendorf's theorem**. As is known, Zeckendorf proved that each positive integer can be represented as the unique sum of non adjacent Fibonacci numbers, as exemplified below:

38=34+3+1; 39=34+5; 40=34+5+1; 41=34+5+2; 42=34+8. (18)

Zeckendorf's theorem is described in the book [2] and in the numerous articles published in *The Fibonacci Quarterly*.

Theorem 1 gives the following simple algorithm for the representation of the positive integer $N < F_{n+1}$ in the *n*-bit Fibonacci code (6). Let us represent the weights of the *n*-bit Fibonacci code (6), starting from the higher digit *n*,as follows:

$$\{F_n, F_{n-1}, F_{n-2}, ..., F_{i+1}, F_i, F_{i-1}, ..., F_3, F_2, F_1\} \tag{19}$$

The first step consists in a comparison of the number $N < F_{n+1}$ with the highest Fibonacci number F_n. If $F_n \le N < F_{n+1}$, this means that the value of the *n*-th bit of the Fibonacci representation (9) is equal to $a_n = 1$. After that we should calculate the difference $r_1 = N - F_n$. According to (11) this difference $r_1 < F_{n-1}$. This means that the binary numeral a_{n-1}, which follows after $a_n = 1$, automatic should be

$a_{n-1}=0$. By continuing a comparison of the difference r_1 to the next Fibonacci numbers $F_{n-1},F_{n-2},...,\boxed{F_{i+1},F_i},...$, we found the couple of the adjacent Fibonacci numbers F_{i+1},F_i, which are connected with the difference r_1 by the following inequality:

$$F_i \le r_1 < F_{i+1} \tag{20}$$

This means that the value of the bit $a_i=1$. After that we should calculate the difference $r_2=r_1-F_i$. According to (11) this difference $0\le r_2<F_{i-1}$.This means that the bit a_{i-1}, which follows after $a_i=1$, automatic should be $a_{i-1}=0$.

This conclusion has a general character, that is, the Fibonacci representation (9), obtained by using this algorithm, has the following unusual property: after each bit $a_i=1\ (i=n,n-1,...,3,2)$ from the left to the right automatic follows the bit $a_{i-1}=0$. Such Fibonacci representation (9) is unique and called a **minimal form.**

These arguments are another proof of Zeckendorf's Theorem: **each positive integer can be represented as the unique sum of non adjacent Fibonacci numbers.**

3. Algorithm of the Fibonacci counter for the minimal form

3.1. An example of the Fibonacci counting for the minimal form

Table 1 demonstrates the example of the "Fibonacci counting" for the case of the 8-bit minimal form:

$$N=a_8a_7a_6a_5a_4a_3a_2a_1 \tag{21}$$

The analysis of Table 1 allows formulating the following general peculiarities of the "Fibonacci counting," used only minimal forms:

1. First of all, we make one important remark. The lowest bit a_1 of every minimal form is equal to 0 identically, that is, the minimal form (21) looks as follows:

$$N=a_8a_7a_6a_5a_4a_3a_2 0 \tag{22}$$

This property is valid for the arbitrary *n*-bit minimal form, that is, in the general case we have:

$$N=a_na_{n-1}...a_ia_{i-1}...a_3a_2 0 \tag{23}$$

In this case we can exclude from consideration the lowest bit $a_1=0$ from the Fibonacci code (6) and consider the "truncated" Fibonacci code with the digit weights $F_2=1,F_3=2,F_4=3,F_5=5,F_6=8,...,F_n$, that is,

$$N=a_nF_n+a_{n-1}F_{n-1}+...+a_iF_i+...+a_3F_3+a_2F_2 \tag{24}$$

Table 1. Sequential change of the minimal forms

n	8	7	6	5	4	3	2	1	n	8	7	6	5	4	3	2	1
N/F_n	**21**	**13**	**8**	**5**	**3**	**2**	**1**	**1**	N/F_n	**21**	**13**	**8**	**5**	**3**	**2**	**1**	**1**
0	0	0	0	0	0	0	0	0	**13**	0	**1**	**0**	**0**	**0**	**0**	**0**	**0**
1	0	0	0	0	0	0	1	0	**14**	0	1	0	0	0	0	1	0
2	0	0	0	0	0	1	0	0	**15**	0	1	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0	**16**	0	1	0	0	1	0	0	0
4	0	0	0	0	**1**	**0**	**1**	**0**	**17**	0	1	0	0	1	0	1	0
5	0	0	0	**1**	**0**	**0**	**0**	**0**	**18**	0	1	0	1	0	0	0	0
6	0	0	0	1	0	0	1	0	**19**	0	1	0	1	0	0	1	0
7	0	0	0	**1**	**0**	**1**	**0**	**0**	**20**	0	**1**	**0**	**1**	**0**	**1**	**0**	**0**
8	0	0	**1**	**0**	**0**	**0**	**0**	**0**	**21**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
9	0	0	1	0	0	0	1	0	**22**	1	0	0	0	0	0	1	0
10	0	0	1	0	0	1	0	0	**23**	1	0	0	0	0	1	0	0
11	0	0	1	0	1	0	0	0	**24**	1	0	0	0	1	0	0	0
12	0	0	**1**	**0**	**1**	**0**	**1**	**0**	**25**	1	0	0	0	1	0	1	0

2. The following characteristic property is valid for arbitrary minimal form: after each bit 1 from the left to the right the bit 0 follows, i.e., two bits of 1 do not occur near. This property is a reflection of Zeckendorf's Theorem in the Fibonacci representations (22), (23). Taking into consideration this fundamental property, Table 1 can be represented as is shown in Table 2.

Table 2. The "truncated" Fibonacci code

n	8	7	6	5	4	3	2	n	8	7	6	5	4	3	2

$N/$ F_n	**21**	**13**	**8**	**5**	**3**	**2**	**1**	$N/$ F_n	**21**	**13**	**8**	**5**	**3**	**2**	**1**
0	0	0	0	0	0	0	0	**13**	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	**14**	0	1	0	0	0	0	1
2	0	0	0	0	0	1	0	**15**	0	1	0	0	0	1	0
3	0	0	0	0	1	0	0	**16**	0	1	0	0	1	0	0
4	0	0	0		1	0	1	**17**	0	1	0	0	1	0	1
5	0	0	0	1	0	0	0	**18**	0	1	0	1	0	0	0
6	0	0	0	1	0	0	1	**19**	0	1	0	1	0	0	1
7	0	0	0	1	0	1	0	**20**	0	1	0	1	0	1	0
8	0	0	1	0	0	0	0	**21**	1	0	0	0	0	0	0
9	0	0	1	0	0	0	1	**22**	1	0	0	0	0	0	1
10	0	0	1	0	0	1	0	**23**	1	0	0	0	0	1	0
11	0	0	1	0	1	0	0	**24**	1	0	0	0	1	0	0
12	0	0	1	0	1	0	1	**25**	1	0	0	0	1	0	1

3.2. General operational rules of the Fibonacci counter for the minimal form

Analysis of Table 2 allows establishing the following general operational rules of the Fibonacci counting for the minimal form. As follows from Table 2, the transition from the Fibonacci code (24) of the number *N* to the Fibonacci code of the next number *N*+1 is done by writing the bit 1 into the corresponding digit of the next Fibonacci code of the number *N*+1. At the same time, we must follow the following rules.

The rule 1. The next bit of 1 is written directly into the lowest digit of the "truncated" Fibonacci code (24) (that is, to the digit with the number 2) only for the condition when the bits of the first two digits (with the numbers 2 and 3) in the preceding Fibonacci code of the number *N*are equal to 0 (the cases, corresponding to the Fibonacci codes of the integers 0,5,8,11,13,16,19,21,24 in Table 2).

The rule 2. Let the Fibonacci representation of the integer *N* has the following form:

n	$n-1$	...	$2k+3$	$2k+2$	$2k+1$	$2k$	$2k-1$	$2k-2$	$2k-3$	...	4	3	2
a_n	a_{n-1}	...	a_{2k+3}	**0**	**0**	1	0	1	0	...	1	0	1

We can see that in this Fibonacci representation we have a specific group of bits, starting from the 2nd and ending with the (2*k*+2) th bits. This group of bits consists of two parts:

1) The group of the digits from the 2nd to $(2k)$st digits. Here, the 2nd digit is equal to 1, but the others digits are alternating bits 1 and 0; herewith the bits 1 are in all digits with the even numbers: $2,4,6,...,2k-2,2k$.

2) The group of the two digits with the numbers $(2k+1)$ and $(2k+2)$, here with the bits 0 are in the both digits (see in bold). Note that the remaining group of bits with the numbers from (2*k* +3) to *n* satisfies to the condition of the minimal form (two bits of 1 do not occur near).

Then, the change of the Fibonacci representation of *N* on the next Fibonacci representation of *N*+1 (when the next bit 1 enters the input of the counter) is realized according to the following **Rule 2:** the bit of 1 is recorded into $(2k+1)$th digit, and all the remaining digits to the right are nulled; herewith the Fibonacci representation of *N*+1 takes the following form:

n	$n-1$	...	$2k+3$	$2k+2$	$2k+1$	$2k$	$2k-1$	$2k-2$	$2k-3$	...	4	3	2
a_n	a_{n-1}	...	a_{k+3}	0	1	0	0	0	0	...	0	0	0

Let's prove that this Fibonacci representation corresponds to the integer $N+1$. This follows directly from the formula (5), that is, the new Fibonacci representation, in fact, is the Fibonacci representation (minimal form) of the integer $N+1$.

The rule 3. Let the Fibonacci representation of the integer *N* has the following form:

n	$n-1$	...	$2k+2$	$2k+1$	$2k$	$2k-1$	$2k-2$	$2k-3$	$2k-4$	...	4	3	2
a_n	a_{n-1}	...	a_{2k+2}	**0**	**0**	1	0	1	0	...	0	1	0

We can see that in this Fibonacci representation we have a specific group of the digits, starting from the 2nd and ending with the $(2k+1)$th digits. This group of the digits consists of two parts:

1) The group of the digits from the 2nd to $(2k-1)$st digits. Here, the 2nd digit is equal to0, but the others digits are the alternating bits 1 and 0; herewith the bits 1 are in all the digits with the odd numbers: $3,5,7,...,2k-3,2k-1$.

2) The group of the two digits with the numbers 2*k* and $(2k+1)$, here with the numerals 0 are in the both digits (see in bold). Note that the remaining group of the digits with the numbers from $(2k+2)$ to n satisfies to the condition of the minimal form (two bits of 1 do not occur near).

Then, the change of the Fibonacci representation of *N* on the next Fibonacci representation of $N+1$ (when the next 1 enters the input of the counter) is realized according to the following **Rule 3**: the bit of 1 is recorded into the $(2k)$th digit, but all the remaining digits to the right are nulled; here with the new Fibonacci representation of *N*+1 has the following form:

n	$n-1$	...	$2k+2$	$2k+1$	$2k$	$2k-1$	$2k-2$	$2k-3$	$2k-4$	...	4	3	2
a_n	a_{n-1}	...	a_{k+2}	0	1	0	0	0	0	...	0	0	0

Let's prove now that this Fibonacci representation corresponds to the integer $N+1$. It is clear that the weight of the $(2k)$th digit is equal to F_{2k}. On the other hand, according to (4) the sum of all the Fibonacci numbers with the even indexes $1,3,5,...,2k-1$ is equal to F_{2k}. Let's recall that we consider the "truncated" Fibonacci code (24), where the 1st digit is absent. Taking into consideration this fact, we can rewrite the identity (4) as follows:

$$F_3+F_5+...+F_{2k-1}=F_{2k}-F_1=F_{2k}-1 \tag{25}$$

Thus, the numerical equivalent of the nulled digits is equal to $F_{2k}-1$, whereas the numerical equivalent of the $(2k)$th digit is equal F_{2k}. It follows from this consideration that the new Fibonacci representation, in fact, is the minimal form of the number $N+1$.

A new operational algorithm of the Fibonacci counter shows that the transition from the Fibonacci representation of the number N into the Fibonacci representation of the next number $N+1$ is carried out per one clock cycle, which includes a record of the bit of 1 into the appropriate digit, and nulling the groups of the digits. This creates prerequisites for the creation of the high-speed Fibonacci counter. But most importantly, this counter uses only the minimal forms for the representation of numbers, these forms are the "allowed" Fibonacci representations and the main "checking" forms of the Fibonacci code.

In order to ease a description of the structural circuit of the Fibonacci counter, we are changing renumbering the digits in the "truncated" Fibonacci code (24) and will use the Fibonacci code, in which the numeration of digits begins with the 1-st digit:

$$N=a_nF_{n+1}+a_{n-1}F_n+...+a_{i-1}F_i+...+a_2F_3+a_1F_2 \tag{26}$$

We will use this numeration below in the Fibonacci counter.

4. Boolean realization of the Fibonacci counter for the minimal form

4.1. A general description

The analysis of the above Fibonacci counter algorithm shows that we should use 5 operational blocks for the realization of the Fibonacci counter: **Register, Disposition Block, Block for Analysis, Block for Error Checking, and Block for Zero Installation**. These blocks allow designing a high-speed and noise-immune Fibonacci counter. Its noise immunity is achieved due to the presence of the "forbidden" states of the counter and high speed is achieved as a result of the absence of the previous carries, needed in the binary counters, and also the micro operations of convolutions and devolutions, used in the known Fibonacci counters [9-11]. These properties give some benefits compared with classical binary counters and the known Fibonacci counters [9-11].

Let us consider the operation of the Fibonacci counter on the example of the 5-bit Fibonacci counter (Fig.1).

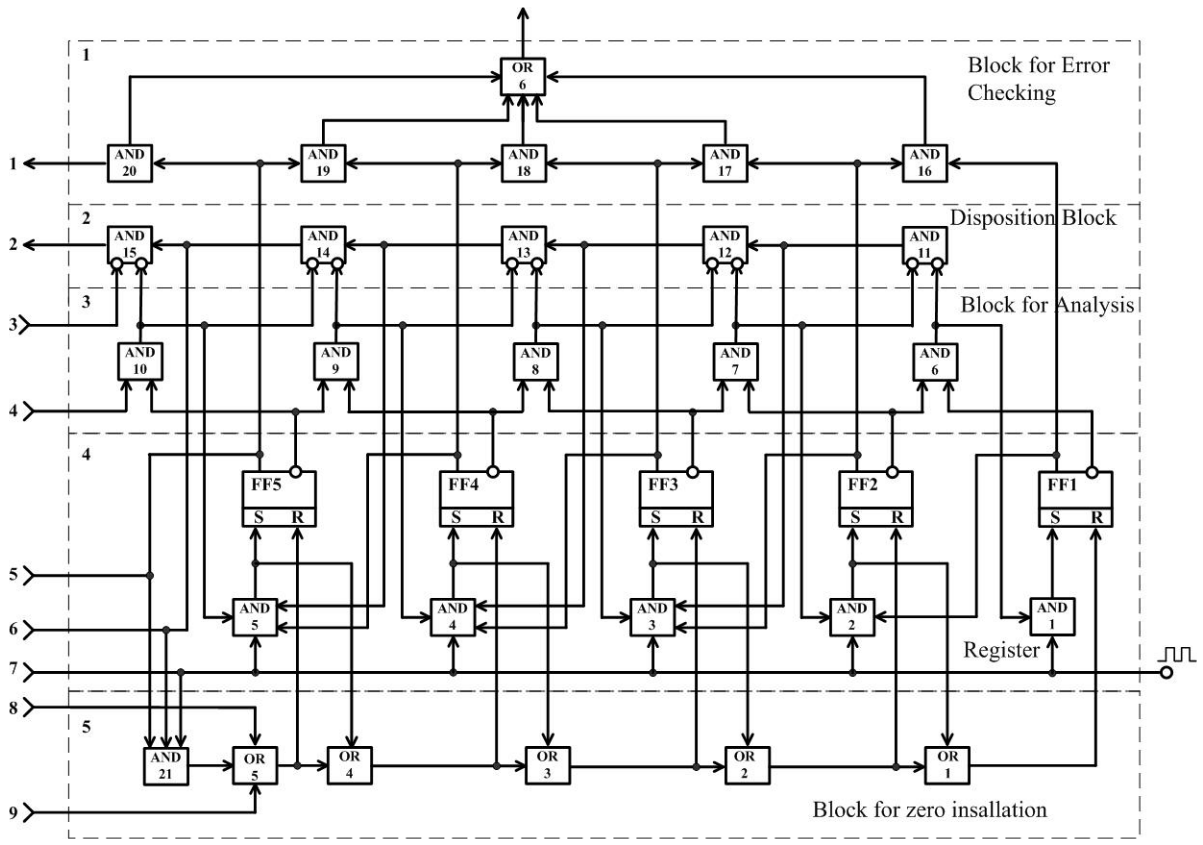

Figure 1. *The Fibonacci counter for the minimal form (Boolean realization)*

The Fibonacci counter in Fig.1 consists of the **Register** (the 5 RS-flip-flops FF1-FF5 with the logical circuits AND1-AND5), **Block for Analysis** (the logical circuits AND6-AND10), **Disposition Block** (the logical circuits AND11-AND15), **Block for Error Checking** (the logical circuits AND16-AND20 and OR6), **Block for Zero Installation**(the logical circuits AND21 and OR1-OR5). All these blocks have a regular structure and can be of unlimited length. Accordingly, the Fibonacci counters can have an unlimited length. To increase the length of the Fibonacci counter, for example, on one digit (the sixth digit), it is only necessary to link corresponding outputs of the sixth digit to the inputs 1-9 by analogy as it was done previously in Fig. 1 at the connection of the fourth digit with the fifth digit. Herewith, we should put the logical circuit AND21 to the input of the sixth flip-flop. By analogy, we can design the seventh, eighth digits and so on up to *n*-th digit of the Fibonacci counter.

4.2. TheRegister

The Register (the 5 RS-flip-flops FF1-FF5 with the logical circuits AND1-AND5) is needed to memorize the counter states. For the counter in Fig.1, we use the RS-flip-flops FF1-FF5, as the most simple, although we may use also other kinds of flip-flops, which perform this function, for example, JK-flip-flops. The outputs of the RS-flip-flops FF1-FF5 are linked to the inputs of the logical circuits AND6-AND10 of the **Block for Analysis**, but the inputs of the RS-flip-flopsFF1-FF5are linked with the **Block for Zero Installation** and the bus clock

4.3. Block for Analysis, Block for Error Checking, Disposition Block, and Block for Zero Installation

The Block for Analysis controls the outputs of the pairs of the neighboring flip-flops of the **Register**, starting from FF1 and FF2, which can be in three "allowed" states 00, 01 or 10. Let us recall, that the state 11 for the two adjacent flip-flops is "forbidden," in accordance with the algorithm of the Fibonacci counter. In order to identify the "forbidden" states, we introduced into the Fibonacci counter the **Block for Error Checking**, which signals about the error state of the Fibonacci counter.

To analyze the location of the bits 1's in the Fibonacci counter, we introduced the **Disposition Block**. The **Disposition Block** together with the **Block for Analysis** control that the flip-flops of the **Register** go to the next correct state of the counter.

The **Block for Zero Installation** sets all flip-flops of the **Register** into zero state for the two cases. The first case, when any flip-flop of the **Register** is switched over into the state 1; herewith, all flip-flops on the left should be switched over into state 0. For the second case, the **Block for Zero Installation** sets all the flip-flops of the **Register** into zero state after the end of the counting cycle.

A structure in Figure 1is quite simple and it is easily to trace that the transition from the minimal form of the number N to the minimal form of the number $N+1$is performed strictly in accordance to Table 2.

5. Functioning of the Fibonacci counter for the minimal form

5.1.The basic steps

The Initial state. The Fibonacci counter in Fig.1 operates as follows. First, the Fibonacci counter is in the initial state, when all flip-flops FF1-FF5 of the *Register* are installed in the zero state. For this case, the logical signals of 0 from the direct outputs of the flip-flops FF1-FF4, according to Fig.1, enter to the inputs of the logical circuits AND2-AND5, respectively. Note that the inverse outputs of the flip-flops FF1-FF2 are connected with the inputs of the logical circuit AND6, and the inverse outputs of the flip-flops FF2-FF3 are connected with the inputs of the logical circuit AND7 and so on up to the logical circuits AND10, as is shown on Fig.1, this means that on the outputs of the logical circuits AND6-AND10 are formed the signals of 1. The logical signals of 1 from the outputs of the logical circuits AND6-AND10 enter to the corresponding inputs of the logical circuits AND1-AND5, respectively.

The First step. According to Fig.1, the first clock pulse enters to the inputs of the logical circuits AND1-AND5. However, if the Fibonacci counter is in the zero state 00000, only the flip-flop FF1 through the logical circuit AND1 can be switched over into the state of 1 because all the rest logical circuits

AND2-AND5 are blocked by the 0-signals, entered from the direct outputs of the flip-flops FF2-FF5. As a result, the Fibonacci counter is installed into the state 00001, corresponding to the integer 1.

The Second step. The transition of the flip-flop FF1 into the state 1 leads to the following situation. The signal 0 appears on the output of the logical circuit AND6 and then on the input of the logical circuit AND1. This signal forbids the passage of the next clock pulse to the S-input of the flip-flop FF1. The logical circuit AND2, on the contrary, is open for the passing of the next clock pulse to the S-input of the flip-flop FF2. This means that the next (second) clock pulse switches over the flip-flop FF2 into the state of 1, while the second clock pulse, through the logical circuit OR1, switches over the flip-flop FF1 to the state 0. As a result, we get a new state of the Fibonacci counter 00010, corresponding to the integer 2.

The Third step. Let us consider the situation with the logical circuit AND3 after the second step. This logical circuit has 4 inputs. Because the Fibonacci counter is in the state 00010, this means that the signal of 1 from the direct output of the flip-flop FF2 enters to the first input of the logical circuit AND3. Because the flip-flops FF3 and FF4 are in the state 0, this means that the signal 1 from the output of the logical circuit AND8 enters to the second input of the logical circuit AND3. Finally, let us consider the situation with the logical circuits AND6, AND7 and AND11. Because the Fibonacci counter is in the state 00010, this means that the input signals of the logical circuits AND6, AND7 are equal 0 and the input signal of the logical circuit AND11 is equal 1. Thus, the signal 1 enters to the third input of the logical circuit AND3. This means that the third clock pulse, through the logical circuit AND3, enters to the S-input of the flip-flop FF3 and switches over it into the state of 1. Besides, this logical 1, through the logical circuits OR2 and then OR1, enters to the R-inputs of the flip-flops FF2 and FF1 and switches over them into the state of 0. As a result, we get the new state of the Fibonacci counter 00100, corresponding to the integer 3.

The Fourth step. When the Fibonacci counter is in the state 00100, the logical signals of 1 appear on the inverse outputs of the flip-flops FF2 and FF1. These logical signals lead to the appearance of the signal of 1 on the output of the logical circuit AND6. On the other hand, the signal of 1 from the output of the logic circuit AND6 enters to the input the logical circuit AND11 of the Disposition Block. This leads to the appearance of the logical 0 at the output of the logical circuit AND11. The signal of 0 from the output of the logical circuit AND11 leads to the appearance of the signal of 0 on the outputs of the logical circuits AND12, AND13, AND14. These signals of 0 enter to the inputs of the logical circuits AND3, AND4, AND5. Finally, the signal 0 on the inverse output of the flip-flop FF3, which is in state of 1, causes the logic signal 0 on the output of the logical circuit AND7 and on the input of the logical circuit AND2. This means that the next (fourth) clock pulse does not change the states of the flip-flops FF2-FF5. However, the flip-flop FF1 is switched over into the state of 1 with the fourth clock signal. As a result, the counter turns into the state 00101 what corresponds to the integer 4.

The Fifth step. Let us now consider, how is performed the transition from the number $4 = 00101$ to the next number 5 = 01000. For the situation 00101 the flip-flops FF1 and FF3 are in state of 1. This means that the signals 0 appear on the outputs of the logical circuits AND6-AND8. These signals enter to the inputs of the logic circuits AND1-AND3. The signals 0 on the outputs of the logical circuits AND7, AND8 are a cause of the appearance of the signal of 1 on the output of the logical circuit AND12, and then this signal enters to the input of the logical circuit AND4. In addition, the signal of 1 enters to another input of the logical circuit AND4 from the direct output of the flip-flop FF3, which is in the state of 1. Also, the signal of 1 enters to the third input of the logical circuit AND4 from the output of the logical circuit AND9. As a result, the next (fifth) clock pulse sets flip-flop FF4in the state of 1, and then, through the logical circuits OR3-OR1, sets the flip-flops FF3,FF2, FF1into the state of 0.As a result, the Fibonacci counter turns in to the state 01000,corresponding to the number 5=01000.

The Sixth step. As the flip-flops FF1, FF2, FF3 are in the state 0, then the signals of 0 enter to the corresponding inputs of the logical circuits AND2-AND4 from the direct outputs of these flip-flops. Besides, the signals of 1 enter to the inputs of the logical circuits AND6, AND7 from the inverse outputs of the flip-flops FF1, FF2, FF3, which are in the state of 0. The signals of 1enter to the inverse inputs of the logical circuit AND11 and they are a cause of the appearance of the signal of 0 on the output of the logical circuit AND11. This signal of 0 is a cause of the appearance of the signal of 0 on the output of the logical circuit AND12 and then AND13. The signal of 0 enters to the input of the logical circuit AND5 from the output of the logical circuit AND13. Because the flip-flops FF1, FF2 are in the state 1, this is a cause of the appearance the signal of 1 on the output of the logical circuit AND6. This signal enters to the input of the logical circuit AND1. Thus, only the flip-flop FF1 can switch over with the clock pulse. The next (sixth) clocks signal switches over the flip-flop FF1 into the state of 1 and does not change the states of the flip-flops FF2-FF5. As a result, the Fibonacci counter turns into the state 01001, corresponding to the number 6=01001.

The Seventh step. Further, by analogy, the next (seventh) clock pulse turns the Fibonacci counter into the state 01010, corresponding to the number 7=01010.

The Eighth step. For this situation, there appear the signal 1 on the inputs of the logical circuit AND5 and the next (eights) clock pulse turns the Fibonacci counter into the state 10000, corresponding to the number 8=10000.

The Last steps. This process will continue up the transition of the Fibonacci counter into the state 10101, corresponding to the maximal number 12=10101, which can be represented with the 5-bit Fibonacci code in the minimal form. For this case, two signals of 1 enter to the inputs of the logical circuit AND21 from the direct input of the flip-flop FF5 and the logical circuit AND14 of the *Disposition Block*. The next clock pulse, through the logical circuits AND21 and OR5-OR1, turns the flip-flops FF1 - FF5 into the initial state 00000. After that, the counter is ready for a new counting cycle.

5.2. Increasing the Fibonacci counter capacity

As mentioned above, the additional inputs and outputs of the counter 1 – 9 are intended for un limited increasing the Fibonacci counter capacity on the left and on the right. For that, we can use the standard set of logical circuits for one digit of the counter. This standard structure consists of one two-input logic circuit OR of the *Block for zero installation*, one flip-flop and one logical circuit AND of the *Register*, one two-input logical circuit AND of the *Block for Analysis*, one two-input logical circuit AND of the *Disposition Block*, and one two-digit logical circuit of the *Block for Error Checking*

5.3. Functioningof the Block for Error Checking

The *Block for Error Checking* is, in fact, an external block regarding to the Fibonacci counter and does not affect the counting algorithm. Its task is to identify errors in the counter. The random appearance of two adjacent bits of 1 in the flip-flops of the *Register* lead to the fact that the "error signal" of 1 appears on one of the logical circuits AND of the *Block for Error Checking*. This "error signal" passes through the logical circuit OR6 to the output of the *Block for Error Checking.* For this case, the Fibonacci counter can start a new counting cycle or, if there is a duplicate counter, can be switched over on the duplicate counter.

6. The main peculiarities of the Fibonacci counter for the minimal form

6.1. A range of number representation for the minimal form

Let us consider the *n*-bit minimal forms (9) of the Fibonacci code (6). It is clear that we can represent all integer numbers from the minimal number $N_{\min} = 0 = \underset{n}{00...0}.$ to the maximal number $N_{\max}$.

In order to calculate the maximal number $N_{\max}$, which can be represented by using the *n*-bit minimal forms of the Fibonacci code (6), we study carefully Table 1.

Let us consider the adjacent Fibonacci representations of the numbers, taken from Table 1: **4=1010** and **5=10000**, **7=10100** and **8=100000**, **12=101010** and **13=1000000** (see in bold in Table 1) and so on. Note that the numbers 5,8 and 13 are the Fibonacci numbers $F_5 = 5, F_6 = 8, F_7 = 13$. It easy to see that the number **4=1010** is the maximal number, which can be represented by the 4-bit minimal form, the number **7=10100** is the maximal number, which can be represented by the 5-bit minimal form, and the number **12=101010** is the maximal number, which can be represented by the 6-bit minimal form. All of them are calculated according to the formulas: **4=5-1, 7=8-1, 12=13-1**

These arguments lead us to the following theorem, which can be proved by the induction on *n*.

Theorem 2. The maximal number $N_{\max}$, which can be represented by using *n*-bit minimal forms of the Fibonacci code (6), is equal to $F_{n+1} - 1$; in this case the range of number representation from $N_{\min} = 0 = \underset{n}{00...0}$ to $N_{\max} = F_{n+1} - 1$ is equal to F_{n+1}, where F_{n+1} is Fibonacci number.

6.2. A redundancy of the Fibonacci code for the minimal form

Let $n = m + k$ is a number of bits of redundant code, where m is a number of data bits, and k is a number of redundant bits. Then, the *relative redundancy* of the code can be calculated according to the formula [15,16]:

$$R=\frac{k}{n}=\frac{n-m}{n}=1-\frac{m}{n} \tag{27}$$

We now calculate the **relative redundancy** of the Fibonacci code(6) for the minimal form, by using the formula(27).According to Theorem 2, by using the n-bit minimal form of the Fibonacci code (6),we can represent F_{n+1} integers in the range from $N_{\min}=0=\underbrace{00...0}_{n}$ to $N_{\max}=F_{n+1}-1$. It is clear that we need $m\approx\log_2 F_{n+1}$ bits to represent this number range in the classical binary code (7). Substituting this expression into (27), we get the following formula for the calculation of the **relative redundancy** of the Fibonacci code (6) for the minimal form:

$$R=1-\frac{m}{n}=1-\frac{\log_2 F_{n+1}}{n} \tag{28}$$

To simplify the formula (28), we can use the so-called **Binet's formula** for the Fibonacci numbers [1,2]:

$$F_{n+1}=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}}{\sqrt{5}}\approx\frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}}{\sqrt{5}}=\frac{\Phi^{n+1}}{\sqrt{5}} \tag{29}$$

where $\Phi=\frac{1+\sqrt{5}}{2}\approx 1.618$ is the golden ratio [1, 2].

By using (29), we can represent the formulas (28), as follows:

$$R=1-\frac{n\log_2\Phi+\log_2\Phi-\frac{1}{2}\log_2 5}{n} \tag{30}$$

With increasing n the expression (30) converges to the following:

$$R=1-\log_2\Phi\approx 0.306 \tag{31}$$

It is important to emphasize that the relative code redundancy (31) for the Fibonacci code (6) does not depend on the length of the Fibonacci representation and in the limit is equal to 0.306 (30.6%).

6.3. Error detection ability of the Fibonacci code

We draw the following considerations, which are of general character and applicable for arbitrary redundant code [17]. If we use the minimal form for the number representation, then a number of the "allowed"n-bit Fibonacci representations $\{a_n a_{n-1}...a_i a_{i-1}...a_2 a_1\}$ is equal to the number of all the n-bit minimal forms F_{n+1}, while the number of all possible n-bit binary representations $\{a_n a_{n-1}...a_i a_{i-1}...a_2 a_1\}$ is equal to 2^n, then the number of the "for bidden" n-bit Fibonacci representations is equal to the difference 2^n-F_{n+1}.

Errors are detected in all cases when the "allowed" Fibonacci representations (minimal forms) pass into the "forbidden" Fibonacci representation. If the "allowed" Fibonacci representation passes into the other "allowed" Fibonacci representation, this is the case of "undetectable error."

A number of all the possible transitions is equal to $F_{n+1} \times 2^n$; here, a number of the "detectable" transitions is equal to $F_{n+1} \times \left(2^n - F_{n+1}\right)$. A ratio

$$S_d = \frac{F_{n+1} \times \left(2^n - F_{n+1}\right)}{F_{n+1} \times 2^n} = 1 - \frac{F_{n+1}}{2^n} \tag{32}$$

characterizes the **error detection ability** of the Fibonacci code (6).

The numerical values of the coefficient S_d , given by (32), for the various values of *n*, are presented in Table 3.

Table 3. Error detection ability of the Fibonacci code for the minimal form

n	2^n	F_{n+1}	$S_d\%$
10	1 024	89	91.31
16	65 536	1 597	97.56
24	16 777 216	75 025	99.55

By comparing the classical error-correcting codes [1, 2] with the Fibonacci code (6), we should not forget that the Fibonacci code (6) is, first of all, a numeral system. By using Fibonacci code, we can design noise-immune Fibonacci computers and microprocessors, which allow performing all the data transformations and providing checking these transformations. If we use the Fibonacci code (6) in computers, we don't need to convert the non-redundant code (the binary code) into the redundant code and conversely. This automatically solves encoding-decoding problem and leads to the simplification of computational structures.

Thus, the Fibonacci code (6) exceeds the classical error-correcting codes [1, 2], first of all, in qualitative relation (because the classical error-correcting codes don't are numeral systems). On the other hand, the Fibonacci code (6) is redundant binary numeral system what allows detecting errors in computers. This means that the Fibonacci code (6) exceeds the classical binary numeral system regarding error detection ability.

6.4. The main advantages of the Fibonacci counter for the minimal form

The main peculiarity of the Fibonacci counter in Fig.1 compared with the well-known Fibonacci counters [9] consists in the fact that the counter functions only in the minimal form of the Fibonacci code (6). All other forms of number representation in the Fibonacci code (6) are "forbidden," and their occurrence is a signal of errors, which may occur in the Fibonacci counter under the influence of various internal and external effects. As can be seen from Table 3, the coefficient of error detection increases with an increase of the Fibonacci counter capacity. For example, the 24-bit Fibonacci counter has a coefficient of error detection equal 99.55%.Recall that a potential error detection coefficient for the code with parity check, used widely in computers today, is equal only 50%. On the other hand, the device for error detection for the Fibonacci code (see Fig.1) is simpler than the similar device for the code with parity check.

The Fibonacci counter in Fig.1 has a fairly high speed. This is achieved due to the fact that the carry-overs from the digit to digit are absent. This improves a speed in comparison with the known binary error-correcting counters and the known Fibonacci counters [9-12], based on the "convolutions" and "devolutions." A time delay in this device only occurs in the chain of the logic circuits AND in the Disposition Block and the Block for Zero Installation (see Fig.1). However, this delay occurs within one clock cycle in parallel with switching over one flip-flop, but not during several cycles, as it usually occurs in the noise-immune binary counters. Regarding a speed, this counter is comparable to the classical binary counter with ripple-through carry, but it differs from it with high noise immunity. This counting device was realized as a computer model in the program Electronics Workbench and its efficiency has been confirmed.

As follows from Fig.1, the proposed counter has a high homogeneity of logical elements and connections between them what is important for microelectronics and nanoelectronics. This is another advantage of the Fibonacci counter in Fig.1.

Thus, the proposed counter has high noise immunity and informational reliability (see Table 3), it has sufficiently big speed and regular structure. Therefore, it makes sense to recommend this counter for the use in digital devices, which require high-performance, noise immunity and reliability, such as devices for frequency measurement and control systems for critical applications.

7. Instead conclusions: a significance of Zeckendorf's Theorem for future development of computing and microelectronic technology

As we mentioned in the Introduction, in 1939 the Belgian medical doctor, army officer and amateur of mathematics Eduardo Zeckendorf (1901-1983) had proved in 1939 the interesting theorem, known in Fibonacci numbers theory [1, 2] as **Zeckendorf's Theorem** [3]. This theorem is the mathematical result of great importance for computer technology. This theorem concerns to the representation of positive integers as a sum of Fibonacci numbers. Thus, Eduardo Zeckendorf predicted in his famous theorem a principally new way in the development of computer technology – **Fibonacci computers**, which begun to develop in modern science starting since 60th years of 20 c. [4-19].

The Fibonacci counter in Fig.1 is only the first step for designing the reliable Fibonacci computers based on **Zeckendorf's Theorem** and on the conception of the **minimal form**, based on this theorem.

The next step in the development of the Fibonacci computers is to design all operational devices of the Fibonacci computers, in particular, ALU, memory and microchips, on the base of the minimal form. In the

Fibonacci computers, ALU, memory and microchips, all the input and output data should be represented in the minimal form what provides the continuous checking of errors on all stages of data transformations in Fibonacci computers (arithmetical operations, storage and data transmission). A further increasing of reliability is connected with the use of the so-called Fibonacci *p*-codes [6, 8, 12] and codes of the golden *p*-proportions [12-15].

Description of the invention is beyond the scope of this article.

References

1. Vorobyov N.N. *Fibonacci Numbers*, first ed. 1961; Moscow: Nauka, Russia, 1978; p.192.
2. Hoggatt V. E. Jr. *Fibonacci and Lucas Numbers*, Boston: Houghton Mifflin, 1969.
3. Zeckendorf's theorem. From Wikipedia, the free encyclopediahttp://en.wikipedia.org/wiki/Zeckendorf's_theorem
4. Stakhov A.P. Redundant Binary Positional Number Systems. *Homogeneous digital computing and integral structures.* Taganrog Radio Institute, Russia, 1974, 5-41.
5. Newcomb R. Fibonacci Numbers as a Computer Base. *Conference Proceedings of the Second Inter-American Conference on Systems and Informatics*, Mexico City, 1974.
6. Stakhov A.P. An Use of Natural Redundancy of the Fibonacci Number Systems for Checking Computing Structures. *Automation and Computer Engineering*, Russia, 1975, No 6, 80-87.
7. Monteiro P. and R.W. Newcomb. Minimal and maximal Fibonacci Representations: Boolean Generation. *The Fibonacci Quarterly*, 1976, Vol.14, No 1.
8. Stakhov A.P. *Introduction into algorithmic measurement theory*, Moscow: Soviet Radio, Russia, 1977.
9. Borisenko A.A., Stakhov A.P., Matsenko S.M. About one method of the constructions Fibonacci counters. *Bulletin of Sumy State University*, Sumy, Ukraine, 2012, 3, 165-170.
10. Borisenko A.A., Stakhov A.P. About one method counting in the Fibonacci code. *Bulletin of Sumy State University*, Sumy, Ukraine, 2011, 3, 141-149.
11. Borisenko A.A., Stakhov A.P. The noise-immynity Borisenko-Stakhov counter. The patent for invention № 104 939. *Sumy State University*, Sumy, Ukraine, 2014, 6 p.
12. Wishnjakov, Y.M. Principles of Design and Research of Counting Devices for the Fibonacci p-codes. PhD thesis, Taganrog Radio Engineering Institute, Russian, 1977.
13. Hoang V.D.A Class of Arithmetic Burst-Error-Correcting Codes for the Fibonacci Computer. PhD thesis, University Maryland, 1979.
14. Stakhov A.P. Codes of the Golden Proportion. Moscow: Radio and Communications, Russia, 1984.
15. Stakhov A.P. *Noise-immune codes: Fibonacci computer*, Moscow: Znanie, series “Radio electronics and Communications, Russia,No.6, 1989.
16. Stakhov A.P. By the Principle of the Golden Ratio: a New Way for the Development of Computing Technology. *Bulletin of the Ukrainian Academy of Sciences*, Ukraine,1990, No.1, 30-36.

17. Stakhov A.P. By the Principle of the Golden Ratio: a New Way for the Development of Computing Technology. Bulletin of the Ukrainian Academy of Sciences, Ukraine, 1990, No.2, 14-25.
18. Stakhov A.P. Brousentsov's Ternary Principle, Bergman's Number System and Ternary Mirror-symmetrical Arithmetic. The Computer Journal2002, Vol.45, No.2, 221-236.
19. Stakhov A.P. The Mathematics of Harmony. From Euclid to Contemporary Mathematics and Computer Science. New Jersey, London, Singapore, Hong Kong: World Scientific 2009.
20. Kharkevich A.A. Fight with Noises, Moscow: Fizmatgiz, Russia, 1963.
21. Peterson W.W. Weldon E.J. Error-Correcting Codes. Cambridge, Massachusetts, and London: The MIT Press, 1972.