Fibonacci Numeral Systems and Fibonacci Computers for Mission-Critical Computer Applications

Alexey Stakhov[1]

[1]*International Club of the Golden Section (Canada)*

Abstract. In the 70 and 80-years of the past century, the new positional numeral systems called Fibonacci *p*-codes, as new informational and arithmetical foundations for mission-critical computer applications, was considered as one of the most perspective directions of Soviet computer science in this field. The main advantage of this direction was to improve the information reliability of computer and measuring systems. This direction has been patented abroad widely (more than 60 patents of US, Japan, England, France, Germany, Canada and other countries). Theoretical basis of this direction has been described in author's book "Introduction to the Algorithmic Measurement Theory" (1977). Unfortunately, this book wasn't be translated into English and, therefore, the Soviet scientific achievements in this field were virtually unknown for Western and world experts in computer science. Under author's leadership, a number of interesting engineering developments have been performed. The purpose of this article is to state the history and the main scientific and engineering achievements in this field, as one of the important direction in the improvement of informational reliability of computers for mission-critical applications.

Key words: *Informational reliability of computers, binary system, Fibonacci numbers, Fibonacci numeral systems, Fibonacci computers*

1. Introduction

1.1. Evolution of numeral systems. A history of numeral systems [1] highlights several stages of its development: *initial stage of counting*, *alphabetical*, *non-positional*, and *positional* numeral systems. The initial stage is characterized by imaging of countable sets by using body parts, in particular, fingers and toes, rope knots, etc. As it is highlighted in the article [1], *"in spite of extreme primitiveness of such method of numbers representation, it played a crucial role in the development of the concept of number."*

Why number theory and theoretical arithmetic didn't give attention to numeral systems, which they deserved? The whole thing consists in the "tradition". In the Ancient Greek mathematics, which reached a high level of development, for the first time mathematics was divided at the "higher" mathematics, which included geometry and number theory, and "logistics", which was considered in Ancient Greece as applied science on arithmetic techniques, geometric measuring and constructions. Already since Plato's time, "logistics" was considered as the unimportant, applied discipline, which wasn't be included in scope of education of the philosopher and scientist. Dismissive attitude towards the "school" arithmetic and its problems (starting since Plato), and the lack of a sufficiently serious need for the creation of new numeral systems in computing practice, which in the last centuries was satisfied entirely with the decimal system, and in the last decades with the binary system (in computer science), and may explain the fact why number theory neglected numeral systems, and why this field of number theory not moved forward in comparison with the period of its origin.

1.2. Decimal system. Each person on the globe, who graduated at least four classes of elementary school, knows at least two useful things: he knows how to read and write and how to use the decimal system to perform the simplest arithmetic operations. And this system seems so simple and "elementary" that many of us will relate with large mistrust to the assertion that the **decimal system is one of the greatest mathematical discoveries in the history of mathematics.** In order to convince the reader of this, let us turn to the opinion of the "authorities".

Pierre-Simon Laplace (1749-1827), French mathematician, a member of the Paris Academy of Sciences, an honorary foreign member of the St. Petersburg Academy of Sciences:

"The idea to express all numbers with 9 characters, giving them other than the values by the form, also the value by the place, so simple that it is because of this simplicity, it is difficult to understand how it is amazing. As it was not easy to come to this method, we see on the example of the greatest geniuses of Greek science Archimedes and Apollonius, from which this idea has remained hidden. "

M.V. Ostrogradsky (1801-1862), Russian mathematician, a member of the St. Petersburg Academy of Sciences and many foreign academies:

"We think that after the invention of written language, the use of so-called decimal system was the greatest discovery. We want to say that the agreement, by which we can express all the useful numbers with the twelve symbols and their endings, is one of the most remarkable creations of human genius. "

Jules Tannery (1848-1910), French mathematician, a member of the Paris Academy of Sciences:

"With regard to the contemporary written numeral system, which uses nine significant figures, and zero, and where the relative value of figures is determined by a special rule, this system was introduced in India in an era that is not defined precisely, but apparently, after the Christian era. The invention of this system is one of the most important events in the history of science, and in spite of the habit to use the decimal system; we cannot be surprised with the wonderful simplicity of its mechanism. "

1.3. Brief history of the binary system. Discussing the history of the binary system, we should note that the ancient Indian writer and mathematician **Pingala** presented the first known description of the binary system in c. 200 BC. A long time before Pingala's method (c. 2800 BC) a full set of 8 trigrams and 64 hexagrams (similar to 3-bit and 6-bit binary numerals) was represented in the *Book of Changes* of the ancient Chinese.

The Ancient Egyptians used original method of numbers multiplication and division, based on *doubling method* [2]; in essence, this method coincides with the methods of multiplication and division, used in modern computers.

The modern binary system was fully documented in the 17th century in the article *Explication de l'Arithmétique Binaire* written by German polymath **Gottfried Wilhelm von Leibniz** (1646 -1716). Leibniz's system used 0 and 1 in a manner similar to the modern binary system.

The binary arithmetic, developed by Leibnitz, is much similar to arithmetic in other numeral systems. Summation, subtraction, multiplication, and division can be performed over binary numerals. All arithmetical operations in the binary system are

reduced to binary summation. The summation of two single-digit binary numbers is relatively simple:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10 \ (\textit{carry of } 1)$$

In 1854, British mathematician **George Boole** (1815 - 1864) published an important article detailing a system of logic that became known as *Boolean algebra*. His logic system became the instrumental tool for the implementation of the binary system in electronic circuits.

In 1937, **Claude Shannon** (1916 - 2001) presented master's thesis at MIT that initiated the implementation of Boolean algebra and binary arithmetic into electronic circuits. Entitled *"A Symbolic Analysis of Relay and Switching Circuits,"* Shannon's thesis became essential step in practical implementation of digital circuit design.

However, the greatest contribution to the implementation of the binary system into computer science was made by John von Neumann and his colleagues at the Princeton Institute for Advanced Studies. In 1946, **John von Neumann**, **Goldstein** and **Berks** in their *Preliminary Discussion on Logic Designing the Electronic Computing Device* gave a decisive preference for the use of the binary system in electronic computers. A justification of the use of the binary system in electronic computers was one of the most important among "*von Neumann's principles"* [3]. In that time, this proposal was absolutely right and well-founded decision, because the binary system was the best suited to binary nature of the electronic components and Boolean logic. Besides, we should take into account the fact that at that time other alternative numeral systems in science simply did not exist. The choice was very small: *decimal, ternary* or *binary* systems. Preference was given to the *binary system*.

1.4. The "*Trojan horse"* of the binary system. However, along with the binary system its *"Trojan horse"* (according to the apt saying of Russian academician **Hetagurov** [4]) was introduced into computer technology. We are talking on **zero redundancy** of the binary system. **Zero redundancy** means that all binary codeword's are "allowed," what makes impossible detecting any errors and faults in electronic systems that inevitably (with more or less likely) may appear in electronic systems under

the influence of various external and internal factors (radiation, electromagnetic feedback, tire noise power, “Row Hammer” effect etc.). Thus, the next not very optimistic conclusion follows from this consideration. **Mankind became hostage of the classical binary system**, which is the basis of modern microprocessors and information technology. Therefore, the further development of microprocessor and information technology, based on the binary system, should be declared inadmissible for certain areas of applications, in particular, for **mission-critical applications.** The binary system can not serve as the informational and arithmetical basis for specialized computer and measuring systems (space system, fast transport, complex technological objects, nuclear systems and so on), as well as nano-electronic systems, where problems of reliability, noise immunity, stability, survivability of computer and measuring systems come to the fore.

Problems of reliability, noise immunity, stability, survivability have to stand in the center of designing systems for mission-critical computer applications.

Unfortunately, microelectronics was forced to adopt all technical solutions of the classical computer technology, along with the binary system. In this field, the "Trojan horse" of the binary system (**zero redundancy**) has moved to modern microprocessors and microcontrollers.

Currently, the binary system together with its "Trojan horse" begins to take its firm positions in nano-electronics [5] what can lead to unpredictable consequences for the further development of information technology for mission-critical applications.

1.5. The first author’s publications on the redundant positional numeral systems. For the first time, the studies on the redundant methods of positional representation of numbers were performed by the author in the early 70s of the 20th century in the Taganrog Radio Engineering Institute (Russia) (1971-1977). The first author’s article in this field [6] was published in 1974. At the same time, other author’s articles [7-9] on the redundant positional numeral systems (Fibonacci *p*-codes and codes of the “golden” *p*-proportions) had been published. The fundamentals of the Fibonacci *p*-codes and codes of the golden *p*-proportions are stated in author’s articles and books [5-17].

1.6. Fibonacci patenting. In 1976 the author worked as Visiting-Professor of the Vienna University of Technology (Austria). At the final stage of the stay in Austria, the author made the speech *"Algorithmic measurement theory and foundations of computer arithmetic's"* at the joint meeting of the Austrian Cybernetics and Computer Societies. The speech aroused great interest of Austrian scientists. In this connection, the Soviet Ambassador in Austria **Efremov** had written to the Soviet State Committee on Science and Technology the letter, which contained the following proposal:

"Taking into consideration the interest of the Austrian scientists in Prof. Stakhov invention in the field of new numeral system, based on Fibonacci numbers, it would be appropriate speeding up the process of patenting invention by Prof. Stakhov abroad what will preserve a priority of Soviet science in this computer field and possibly will give economic benefit. "

The proposal of the USSR Ambassador in Austria got approval at the state level; as a result, since 1976, the widespread patenting of Stakhov's inventions on "Fibonacci computers" was launched in all the leading countries-producers of computer technology (USA, Japan. Britain, France, Germany, Canada and other countries).

The main purpose of patenting was protecting the priority of Soviet computer science in this field. New computer arithmetic was the subject of patenting. However, in accordance with the patent laws of most countries, it is impossible to get a patent on the mathematical invention, in particular, computer arithmetic based on Fibonacci numbers.

Therefore, there appeared the idea of indirect protection of new computer arithmetic by using computer devices, which implement this arithmetic. Registers, counters, devices for summation, subtraction, multiplication, division and other operating devices of "Fibonacci computer" belong to such devices. Thus it is desirable to design such an original operating device, which could be *pioneering invention* in the field and which would allow implementing all other operating devices. As a result of these reasoning's, there appeared the idea of multi-section invention formula, where the first point describes the pioneering "Fibonacci" invention.

What should have been a pioneering "Fibonacci" invention? The device for bringing (or reducing) of the Fibonacci code to the so-called MINIMAL FORM (see below) was this invention. The invention had no analogues and the prototype, and was

recognized in the Soviet Union and then in other countries as the *pioneering invention*. Then, other operating devices of Fibonacci computer have been developed (such as counters and adders). It was also new devices of computer technology, but they were not pioneering inventions, because they are based on the device for bringing of the Fibonacci code to the MINIMAL FORM.

What are the results of this unprecedented patenting? On Soviet inventions in the field of "Fibonacci computers," more than 60 patents of USA, Japan, England, France, FRG, Canada, Poland and DDR were issued. This shows that the idea of the Fibonacci computer was completely new and original, and these patents are official legal documents confirming the priority of Soviet science (and the author of this article) in a new direction in the field of computer science.

The *"Device for reduction of p-Fibonacci codes to the minimal form"* was pioneering invention in the list of technical solutions submitted for patenting. This means that 12 pioneering patents of different countries [18-30] has been issued for this invention.

In June 1989, on the initiative of the President of the Academy of Sciences of Ukraine Boris Paton, this direction was discussed at a special meeting of the Presidium of the Academy of Sciences of Ukraine.

Unfortunately, the so-called "Gorbachev's perestroika" and the collapse of the Soviet Union in 1991 caused irreparable blow on this scientific direction. Since 1989, governmental funding of this direction was sharply reduced and then completely stopped. But this does not mean that the conception of "Fibonacci computers" has been obsolete. On the contrary, at the present stage of the development of computer technology, this conception has become even more relevant in the modern field of computer technology for mission-critical applications, especially for microprocessor and nano-electronic technology. This circumstance became the main motive for returning to these ideas [31-36]. Note that for the first time the theoretical justifications of these ideas was stated in author's 1977-, 1984-books [16, 17]).

In conclusion, it should be noted that along with the Soviet studies on "Fibonacci Arithmetic" and "Fibonacci computers", in the same period, such studies have been performed in the United States (University of Maryland) [37-42].

Studies of the American, Soviet and also Ukrainian and Canadian researchers in this field are confirmation of the fact that, since 70s years of the 20th century, the notions of "Fibonacci code," “Fibonacci arithmetic” and “Fibonacci computer" became widely known both in the American and Soviet, Ukrainian and Canadian scientific and technical literature.

2. Fibonacci numbers

2.1. Fibonacci’s recurrence relation. By solving the *“rabbit reproduction”* problem (Liber Abaci, 1202), Fibonacci derived the following important recurrence relation:

$$F_n = F_{n-1} + F_{n-2}. \tag{1}$$

Note that the concrete values of the numerical sequences, generated by the recurrence relation (1), depend on the initial values (the *seeds*) of the sequence F_1 and F_2. Here it is assumed that

$$F_1 = F_2 = 1. \tag{2}$$

For this case the recurrence formula (1) with initial conditions (2) "generates" the following numerical sequence:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots. \tag{3}$$

In mathematics, this numerical sequence (3) is called, as a rule, *Fibonacci numbers*. Fibonacci numbers (3) have a variety of remarkable mathematical properties (see below).

2.2. Variations on the Fibonacci theme. Variations on a given theme are a well known genre in music. A distinctive feature of such musical compositions is that they begin, in most cases, with one simple essential musical theme, which thereafter undergoes considerable changes in tempo, mood and nature. If we follow this example of a musical piece and select a simple mathematical subject, such as the Fibonacci sequence of numbers, we can consider it together with its numerous variations.

The sum of the first *n* consecutive Fibonacci numbers. Fibonacci numbers have a range of delightful mathematical properties which for many centuries have

stimulated the mathematical imagination. For example, calculate the sum of the first n Fibonacci numbers beginning with the simplest sums as follows:

$$\boxed{\begin{array}{l} 1+1=2=\mathbf{3}-1 \\ 1+1+2=4=\mathbf{5}-1 \\ 1+1+2+3=7=\mathbf{8}-1 \\ 1+1+2+3+5=12=\mathbf{13}-1 \end{array}}. \tag{4}$$

If in these sums (4) we consider the numbers marked by bold type: **3, 5, 8, 13**, …, then it is easy to see, that they are Fibonacci numbers! We can then write the sums (4) using the following general form:

$$F_1+F_2+...+F_n=F_{n+2}-1. \tag{5}$$

The sum of consecutive Fibonacci numbers with odd indices. We start with the simplest sums:

$$\boxed{\begin{array}{l} 1+2=\mathbf{3} \\ 1+2+5=\mathbf{8} \\ 1+2+5+13=\mathbf{21} \\ 1+2+5+13+34=\mathbf{55} \end{array}}. \tag{6}$$

The following general formula is derived from (6):

$$F_1+F_3+F_5+...+F_{2n-1}=F_{2n}. \tag{7}$$

The sum of consecutive Fibonacci numbers with even indices is

$$F_2+F_4+F_6+...+F_{2n}=F_{2n+1}-1. \tag{8}$$

The sum of the squares of the *n* consecutive Fibonacci numbers. Next we calculate the sum of the squares of the n sequential Fibonacci numbers:

$$F_1^2+F_2^2+...+F_n^2. \tag{9}$$

We start from an analysis of the simplest sums of the kind (9):

$$\boxed{\begin{array}{l} 1^2+1^2=2=\mathbf{1}\times\mathbf{2} \\ 1^2+1^2+2^2=6=\mathbf{2}\times\mathbf{3} \\ 1^2+1^2+2^2+3^2=15=\mathbf{3}\times\mathbf{5} \\ 1^2+1^2+2^2+3^2+5^2=40=\mathbf{5}\times\mathbf{8} \end{array}}. \tag{10}$$

This analysis of (10) leads us to the following general formula:

$$F_1^2 + F_2^2 + ... + F_n^2 = F_n F_{n+1}. \tag{11}$$

The sum of the squares of two adjacent Fibonacci numbers. Next let us consider the sum of the squares of two adjacent Fibonacci numbers:

$$F_{n-1}^2 + F_n^2. \tag{12}$$

We start from an analysis of the simplest sums of the kind (12):

$$\boxed{\begin{array}{l} 1^2 + 1^2 = 1 + 1 = \mathbf{2} \\ 1^2 + 2^2 = 1 + 4 = \mathbf{5} \\ 2^2 + 3^2 = 4 + 9 = \mathbf{13} \\ 3^2 + 5^2 = 9 + 25 = \mathbf{34} \end{array}}. \tag{13}$$

The analysis of (13) leads us to the following general formula:

$$F_{n-1}^2 + F_n^2 = F_{2n-1}. \tag{14}$$

Cassini's formula for Fibonacci numbers:

$$F_n^2 - F_{n-1} F_{n+1} = (-1)^{n+1}. \tag{15}$$

This wonderful formula evokes an aesthetic if not spiritual thrill when one realizes that it is valid for any value of n (where n is any integer from $-\infty$ up to $+\infty$). Thus this endless alternation of + 1 and -1 in the expression (15) evokes a genuine aesthetic feeling of rhythm and harmony.

3. Fibonacci *p*-codes and their peculiarities

3.1. Pascal's Triangle and Fibonacci numbers. In the book [43], the famous American mathematician and populariser of mathematics **George Polya** (1887 - 1985) has found a surprising connection between *Fibonacci numbers* and *"diagonal sums" of Pascal's triangle* (see Table 1)

Table 1. Diagonal sums of Pascal's Triangle

1	1								
1	1	1							
2	1	2	1						
3	1	3	3	1					
5	1	4	6	4	1				
8	1	5	10	10	5	1			
13	1	6	15	20	15	6	1		
21	1	7	21	35	35	21	7	1	
34	1	8	28	56	70	56	28	8	1
		55	89	144	233	377	610	987	1597

If we calculate the sums of binomial coefficients standing on the diagonals: 1=1, 1=1, 2=1+1, 3=1+2, 5=1+3+1, 8=1+4+3, 13=1+5+6+1, 21=1+6+10+1, …, we get the Fibonacci sequence, that is, Pascal's triangle is a "generator" of the Fibonacci numbers!

At first glance, it seems that finding this connection is so simple and so "elementary" what it hardly worthy the attention of mathematicians. However, this mathematical result, which, as say, "lay on the surface," for several centuries was the "big secret" for **Blaise Pascal** (1623–1662) and other mathematicians who studied *Fibonacci numbers* and *Pascal's triangle*. However, the surprisingly simple mathematical connection between Fibonacci numbers and Pascal triangle opens the way to the deep union of two important mathematical theories, the *theory of Fibonacci numbers* [44 - 46] and *combinatorics* and this union can become a fruitful source for new mathematical ideas and generalizations.

3.2. Fibonacci *p*-numbers. The development of *Polya's idea* [43] led the author in the book [16] to the discovery of the surprising generalized recurrence relation, which "generates" an infinite number of the new recurrence sequences, *Fibonacci p-numbers* (p=0,1,2,3,…) given by the following recurrence relation:

$$F_p(i) = F_p(i-1) + F_p(i-p-1) \quad \text{for} \quad i > p+1 \tag{16}$$

at the following initial terms:

$$F_p(1) = F_p(2) = \ldots = F_p(p+1) = 1. \tag{17}$$

Here the numbers p=0,1,2,3,… correspond to the different inclinations of diagonals in Pascal's Triangle (see Table 1).

Note that the recurrence relation (16) at the seeds (17) generates many remarkable numerical sequences, in particular, the *binary sequence* for the case p=0:

$$1,2,4,8,16,32,64,\ldots,2^{n-1},\ldots, \tag{18}$$

and the *Fibonacci sequence* (3) for the case p=1.

3.3. The "golden" *p*-proportions. A study of the limit of the ratios of two adjacent *Fibonacci p-numbers* [16, 33] led us to the following algebraic equation:

$$x^{p+1} - x^p - 1 = 0, \tag{19}$$

which is a generalisation of the "golden" algebraic equation $x^2 - x - 1 = 0$ with the positive root $\Phi = \frac{1+\sqrt{5}}{2}$ (the golden ratio).

In general, the positive roots Φ_p of the equation (19) are new mathematical constants called the *golden p-proportions* (p=0,1,2,3,…) that are a generalization of the classical golden proportion $\Phi = \frac{1+\sqrt{5}}{2}$.

Table 2. Partial cases of Φ_p

p	0	1	2	3	4
Φ_p	2	1.618	1.4656	1.3802	1.3247

Thus, we have every right to claim that *Pascal's triangle* is a "universal generator" for new recurrent sequences $F_p(n)$, which are a generalization of the classical Fibonacci numbers 1,1,2,3,5,8,13,… and new mathematical constants Φ_p (see Table 2), which are a generalization of the classical golden ratio. These mathematical concepts underlie the "*theory of the Fibonacci p-numbers and golden p-proportions,*" [6 - 17], which is a generalization of the classical *"theory of Fibonacci numbers and golden ratio*" [44 - 46].

3.4. Definition of the Fibonacci *p*-codes. It is proved in [16] that for the given integer $p \in \{0,1,2,3,...\}$, any positive integer N can be represented as the following sum:

$$N = a_n F_p(n) + a_{n-1} F_p(n-1) + \ldots + a_i F_p(i) + \ldots + a_n F_p(1), \tag{20}$$

where $a_i \in \{0,1\}$ is a bit of the i-th digit of the positional representation (20); n is a number of bits of the code (20).; $F_p(i)(i=1,2,3,...,n)$ is the *Fibonacci p-number,* the weight of the i-th digit of the positional representation (20).

The sum (20) is called *Fibonacci p-code* [16].

In the Fibonacci *p*-code (20), the weights of digits are linked with the recurrence relation (16), which "generates" *Fibonacci p-numbers* starting from the seeds (17).

The abridged notation of the sum (20) looks as follows:

$$N = a_n a_{n-1} ... a_i ... a_1 . \tag{21}$$

The abridged notation (21) is called *Fibonacci representation* of positive integer *N*.

3.5. Partial cases of the Fibonacci *p*-codes. Note that the sum (20) includes an infinite number of different binary (0,1) positional representations, because every $p\,(p=0,1,2,3,...)$ "generates" its own positional representation of the type (20).

Let $p=0$. For this case, Fibonacci (*p*=0)-numbers $F_0(i)$ coincide with the "binary" numbers, ie, $F_0(i)=2^{i-1}$ and therefore the sum (20) takes the form of the classic binary code:

$$N = a_n 2^{n-1} + a_{n-1} 2^{n-2} + ... + a_i 2^{i-1} + ... + a_1 2^0 . \tag{22}$$

Let $p=1$. For this case, Fibonacci (*p*=1)-numbers $F_1(i)$ coincide with the classic Fibonacci numbers, ie, $F_1(i)=F_i$ and for this case the sum (20) takes the following form:

$$N = a_n F_n + a_{n-1} F_{n-1} + ... + a_i F_i + ... + a_1 F_1 . \tag{23}$$

Recall that the weight of *i*-th digits F_i in the Fibonacci code (23) are classic Fibonacci numbers (3) generated by the Fibonacci recurrence relation (1) at the seeds (2).

Let now $p=\infty$. For this case all Fibonacci ($p=\infty$)-numbers, given by (16), (17), are equal to 1 identically, ie, for any i we have: $F_p(i)=1$. For this case, the sum (20) takes the following form, called *"unitary code"*:

$$N = \underbrace{1+1+...+1}_{N} . \tag{24}$$

Thus, the Fibonacci p-codes (20) is a wider generalization of the "binary code" (22) (the case $p=1$). The classic Fibonacci code (23) ($p=1$) and "unitary code" (24)

$(p=\infty)$ are partial cases of the Fibonacci p-codes (20). It follows from this consideration the following important conclusions:

1) **The first conclusion** follows from the fact that the Fibonacci *p*-codes are a generalization of the "unitary code" (24) $(p=\infty)$. But, by its form the "unitary code" (24) coincides with *Euclidean definition* of positive integers and is beginning of *number theory,* one of fundamental theories of mathematics. We can hypothesize from this fact that the Fibonacci *p*-codes (20) can be viewed as a new definition of positive integers, which implies the idea of new number theory, based on Fibonacci *p*-numbers.
2) **The second conclusion** follows from the fact that Fibonacci *p*-codes are a generalization of the "binary system" (22) ($p=0$). But we should not forget that the binary system (22) is the basis of modern computers! But then the following question arises. Because the Fibonacci *p*-codes (20) are new method of the binary (0,1) positional representation of numbers, this means that we will come to a new class of computers, *Fibonacci computers* as a new direction in the development of computer technology!
3) Because the *Fibonacci p-codes* (20) are based on *the Fibonacci p-numbers*, which follow from the *Pascal Triangle*, this means that the *Fibonacci p-codes* (20) are new unique mathematical object, which unites three fundamental areas of modern science: *number theory*, which is the basis of mathematics, *combinatorics,* based on the *Pascal Triangle*, and finally, *computer science*, based on the *binary system* (22).

4. Properties of Fibonacci representations

4.1. A range of number representation in the Fibonacci *p*-codes. Consider the set of the *n*-digit binary words. A number of them is equal to 2^n. For the classical binary code (22) ($p = 0$) the mapping of the *n*-digit binary words onto the set of positive integers has the following peculiarities:

(a) *Uniqueness of mapping.* This means that for the infinite *n* there is one-to-one correspondence between positive integers and sum (22), that is, each positive integer *N* has the only representation in the form (21).

(b) For the given n, by using the binary code (22) we can represent all positive integers in the range from 0 to $2^n - 1$, that is, the range of number representation is equal to 2^n .

The minimal number 0 and the maximal number $2^n - 1$ have the following binary representations in the binary code (22), respectively:

$$0 = 0\,0 \ldots 0$$

$$2^n - 1 = 1\,1 \ldots 1$$

For the Fibonacci p-code (20) the mapping of the n-digit binary words onto positive integers has distinct peculiarities for the case $p > 0$.

Let n =5. Then for the case p =1 the mapping of the 5-digit Fibonacci (p=1)-code (23) onto positive integers have the form, represented in Tables 3.

Table 3. Mapping of the Fibonacci (p=1)-code onto positive integers

CC	5	3	2	1	1	N	CC	5	3	2	1	1	N
A_0	**0**	**0**	**0**	**0**	**0**	**0**	$\boldsymbol{A_{16}}$	**1**	**0**	**0**	**0**	**0**	**5**
A_1	0	0	0	0	1	1	A_{17}	1	0	0	0	1	6
A_2	**0**	**0**	**0**	**1**	**0**	**1**	$\boldsymbol{A_{18}}$	**1**	**0**	**0**	**1**	**0**	**6**
A_3	0	0	0	1	1	2	A_{19}	1	0	0	1	1	7
A_4	**0**	**0**	**1**	**0**	**0**	**2**	$\boldsymbol{A_{20}}$	**1**	**0**	**1**	**0**	**0**	**7**
A_5	0	0	1	0	1	3	A_{21}	1	0	1	0	1	8
A_6	0	0	1	1	0	3	A_{22}	1	0	1	1	0	8
A_7	0	0	1	1	1	4	A_{23}	1	0	1	1	1	9
A_8	**0**	**1**	**0**	**0**	**0**	**3**	A_{24}	1	1	0	0	0	8
A_9	0	1	0	0	1	4	A_{25}	1	1	0	0	1	9
A_{10}	**0**	**1**	**0**	**1**	**0**	**4**	A_{26}	1	1	0	1	0	9
A_{11}	0	1	0	1	1	5	A_{27}	1	1	0	1	1	10
A_{12}	0	1	1	0	0	5	A_{28}	1	1	1	0	0	10
A_{13}	0	1	1	0	1	6	A_{29}	1	1	1	0	1	11

A_{14}	0	1	1	1	0	6	A_{30}	1	1	1	1	0	11
A_{15}	0	1	1	1	1	7	A_{31}	1	1	1	1	1	12

The analysis of Table 3 allows finding the following peculiarities of the binary representations of natural numbers in the Fibonacci p-codes (20). By using the 5-digit Fibonacci (p=1)-code (Table 2) we can represent 13 integers in the range from 0 to 12, inclusively. Note that the number 13 is the Fibonacci 1-number with the index 7, i.e. $F_1(7) = F_7 = 13$. The result of this consideration are partial case of the following general theorem [33].

Theorem 1. *For the given integers $n \geq 0$ and $p \geq 0$ by using the n-digit Fibonacci p-code we can represent $F_p(n+p+1)$ integers in the range from the minimal number 0 to the maximal number $F_p(n+p) - 1$, inclusively.*

Note that for the case p=0 $F_0(n+1) = 2^n$ and Theorem 1 is reduced to the well-known theorem about the number representation range, equal to 2^n for the n-digit binary code (22) .

4.2. Plurality of number representation. *Plurality* of Fibonacci representation in the form (21) is the next peculiarity of the Fibonacci p-codes (20) for the cases $p > 0$. By excepting the minimal number 0 and the maximal number $F_p(n+p) - 1$, the rest positive integers from the range $[0, F_p(n+p) - 1]$ have more than one Fibonacci representation in the form (6). This means that all integers in the range $[1, F_p(n+p) - 2]$ have multiple representations in the Fibonacci p-codes (20) for the cases $p > 0$.

Consider now the mapping of positive integers onto the 5-digit binary code combinations A in accordance with Table 3 (p = 1).

Table 4. Mapping of positive integers onto binary code words for the Fibonacci (p=1)-code

$p = 1$
0 $= \{A_0\}$
1 $= \{A_1, A_2\}$
2 $= \{A_3, A_4\}$
3 $= \{A_5, A_6, A_8\}$

4 = $\{A_7, A_9, A_{10}\}$
5 = $\{A_{11}, A_{12}, A_{16}\}$
6 = $\{A_{13}, A_{14}, A_{17}, A_{18}\}$
7 = $\{A_{15}, A_{19}, A_{20}\}$
8 = $\{A_{21}, A_{22}, A_{24}\}$
9 = $\{A_{23}, A_{25}, A_{26}\}$
10 = $\{A_{27}, A_{28}\}$
11 = $\{A_{29}, A_{30}\}$
12 = $\{A_{31}\}$

Note that for the arbitrary p the minimal and the maximal numbers have the only Fibonacci representations in the n-digit Fibonacci p-codes (20):

$$N_{min} = 0 = \underbrace{00...0}_{n} \tag{25}$$

$$N_{\max} = F_p(n+p) - 1 = \underbrace{11...1}_{n}. \tag{26}$$

4.3. "Convolution" and "devolution" of the Fibonacci digits. The different Fibonacci representations (21) of one and the same positive integer N in the Fibonacci p-codes (20) for the cases p>0 may be obtained each from other by means of the peculiar code transformations called *"convolution"* and "*devolution"* of the binary digits. These code transformations are performed in scope of one Fibonacci representations (21) and follow from the basic recurrence relation (16), which connects the adjacent digit weights of the Fibonacci p-code (20). The general idea of these code transformations is described in [33].

It is most simply to demonstrate the "convolution" and "devolution" of the Fibonacci representation (33) for the case p=1. In this case these operations are performed over the three adjacent digits, namely over the lth , $(l\text{-}1)^{\text{th}}$ and $(l\text{-}2)^{\text{th}}$ digits. Consider the fulfillment of these operations for the Fibonacci (p=1)-code:

(a) *Convolution*

$$7 = \begin{cases} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{cases} \tag{27}$$

(b) *Devolution*

$$5 = \begin{cases} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{cases} \tag{28}$$

A procedure, which consists in the fulfillment of all possible "convolutions" or "devolutions" in the Fibonacci representation (21), is named a "*code convolution*" or "*code devolution*", respectively. It is easy to prove that the fulfillment of the "code convolution" or "code devolution" will result in the so-called "*convolute*" or "*devolute*" Fibonacci representation of positive integer N.

For the case p=1, the *"convolute"* and *"devolute"* Fibonacci representations of the integer N have peculiar indications. In particular, *in the "convolute" Fibonacci representation two bits of 1 together do not meet* and *in the "devolute" Fibonacci representation two bits of 0 together do not meet, starting from the highest 1 of the Fibonacci representation (21).*

A rule of the "convolution-devolution inversion" is of great importance for technical applications. This rule consists of the following: *the "convolution" of the initial Fibonacci representation is equivalent to the "devolution" of the inverse Fibonacci representation and conversely.* By using this rule, we can fulfill bringing to the "convolute" form for the example (27) as follows:

(a) Inversion of the initial Fibonacci representation 0 1 1 1 1:

1 0 0 0 0

(b) "Code devolution" of the inverse Fibonacci representation:

0 1 0 1 1

(c) Inversion of the obtained Fibonacci representation:

1 0 1 0 0 (the "convolute" form of the Fibonacci representation 0 1 1 1 1).

Consider now peculiarities of the "convolution" and "devolution" for the lowest digits of the Fibonacci (p=1)-code. As is well-known for the case p = 1 the weights of the two lowest digits of the Fibonacci (p=1)-code is equal to 1 identically, that is, $F_1 = F_2$

= 1. And then the operations of the "devolution" and "convolution" for these digits are performed as follows:

$$10 = 01 \text{ ("devolution") and } 01 = 10 \text{ ("convolution").}$$

4.4. The radix of the Fibonacci *p*-code. For the case $p = 0$ the radix of the binary system (22) is calculated as the ratio of the adjacent digit weights, that is,

$$\frac{2^k}{2^{k-1}} = 2.$$

Apply this principle to the Fibonacci p-code (20) and consider the ratio

$$\frac{F_p(k)}{F_p(k-1)} \ . \tag{29}$$

A limit of the ratio (29) is named *a radix of the Fibonacci p-code* (20). It is easy to prove that

$$\lim_{k\to\infty} \frac{F_p(k)}{F_p(k-1)} = \Phi_p \ , \tag{30}$$

where Φ_p is the golden p-proportion.

This means that the radix of the Fibonacci p-code (20) for the case $p>0$ is the irrational number Φ_p.

4.5. A minimal form of the Fibonacci *p*-code. The following theorem is of great importance for a theory of the Fibonacci p-codes [33].

Theorem 2. *For the given integers $p\geq 0$ and $n\geq p+1$ the arbitrary positive integer N can be represented in the following unique form:*

$$N = F_p(n) + R_1, \tag{31}$$

where

$$0 \leq R_1 < F_p(n-p). \tag{32}$$

Note that for the case $p=0$ we have:

$$F_0(n) = 2^{n-1}$$

and then the expressions (31) and (32) take the following well-known form (for the "binary" arithmetic):

$$N = 2^{n-1} + R_1, \qquad 0 \le R_1 < 2^{n-1}. \tag{33}$$

If we represent the integer N according to the formula (31) and then represent all remainders $R_1, R_2, ..., R_k$, arising as result of this representation, according to the same formula (31) up to the obtaining the remainder equal to 0, we get a peculiar representation of the integer N in the Fibonacci p-code (20). Its peculiarity consists of the fact that in the Fibonacci representation of the positive integer N given by (21) no less than p bits of 0 follow after every bit $a_l = 1$ from the left to the right, that is,

$$a_{l-1} = a_{l-2} = ... = a_{l-p} = 0.$$

Such Fibonacci representation of the integer N is called **MINIMAL FORM** or *minimal Fibonacci representation of the integer N in the Fibonacci p-code* (20). This name reflects the fact that for the case $p = 1$ the **MINIMAL FORM** of the integer N has a minimal number of the bits of 1 in the Fibonacci representations (21) of the Fibonacci (p=1)-code (22) among all Fibonacci representations (21) of the same integer N.

For example, by using the above algorithm, we can obtain the following **MINIMAL FORMS** of the positive integer 25 for the Fibonacci 1- and 2-codes (Table 5).

Table 5. MINIMAL FORMS of the Fibonacci 1- and 2-codes

	$F_1(i)$	55	34	21	13	8	5	3	2	1	1
$p = 1$	25 =	0	0	**1**	**0**	0	0	**1**	**0**	**1**	**0**
	$F_2(i)$	19	13	9	6	4	3	2	1	1	1
$p = 2$	25 =	**1**	**0**	**0**	**1**	**0**	**0**	0	0	0	0

A peculiarity of the Fibonacci representations of the number 25 given by Table 5 consists in the following. For the case p=1 the bit of 0 follows after every bit of 1 from the left to the right in the Fibonacci representation of the positive integer 25; for the case p=2 two bits of 0 follow after every bit of 1 from the left to the right in the Fibonacci representation of the same positive integer 25.

Corollary from Theorem 2. *For a given p (p=0, 1, 2, 3, …) every positive integer N has the only minimal form in the Fibonacci p-code (20).*

This means that there is one-to-one mapping of positive integers onto the MINIMAL FORMS of the Fibonacci p-code (20).

Note that for the case $p=0$ (the classical binary code) every positive integer N has the only binary representation in the form (21). This means that every binary representation (21) is its "minimal form".

The following theorem is proved in [33].

Theorem 3. *For a given integer $p \geq 0$ in the minimal form of the n-digit Fibonacci p-code we can represent $F_p(n+1)$ positive integers in the range from 0 to $F_p(n+1)-1$, inclusively.*

For the case $p=1$ the MINIMAL FORM of the Fibonacci ($p=1$)-code has very simple indication: in the MINIMAL FORM two bits of 1 together do not meet. But the "convolute" Fibonacci representation, considered above, has the same property. This means that for the Fibonacci ($p=1$)-code the "convolute" form coincides with the MINIMAL FORM and the bringing of the Fibonacci representation for the Fibonacci ($p=1$)-code (22) to its MINIMAL FORM can be performed by using "convolutions". The example (27) demonstrates a process of the bringing of the Fibonacci representation for the Fibonacci ($p=1$)-code (23) to the MINIMAL FORM. Also the notion of the MAXIMAL FORM is very important for the Fibonacci ($p=1$)-code. The MAXIMAL FORM can be obtained from the initial Fibonacci representation (21) by using "devolutions" and the MAXIMAL FORM coincides with the "devolute" form. The example (28) demonstrates a process of the bringing of the minimal Fibonacci representation 10000 to the maximal Fibonacci representation 01011. Note that the operations of the bringing of the Fibonacci representations to their MINIMAL or MAXIMAL FORMS are the most important operations of the Fibonacci arithmetic.

4.6. Code redundancy of the Fibonacci *p*-codes. For the case $p=0$ the Fibonacci ($p=0$)-code (classical binary code) is non-redundant. But for the case $p>0$ all Fibonacci p-codes are redundant. And their code redundancy shows itself in *plurality* of the Fibonacci representations of one and the same integer N. Theorems 1 and 2 allow calculating the code redundancy of the Fibonacci p-codes for the cases $p>0$ in comparison with the classical binary code ($p = 0$), which is non-redundant code.

We can calculate the relative code redundancy r by the following formula [47]:

$$r = \frac{n-m}{m} = \frac{n}{m} - 1, \tag{34}$$

where n and m are the code length of the redundant and non-redundant codes, respectively. Note that the code redundancy definition, given by (34), characterizes a relative increasing of the code length of the redundant code in the comparison to the non-redundant code for the representation of one and the same number range.

Theorems 1 and 2 determines ranges of number representations in Fibonacci p-codes for two cases: (a) when we use all possible Fibonacci representations (Theorem 1) and (b) when we use only MINIMAL FORMS of Fibonacci representations (Theorem 2). For the case (a) we can represent $F_p(n+p+1)$ positive integers, for the case (b) we can represent $F_p(n+1)$ positive integers.

For the code representation of numbers in the number range $F_p(n+p+1)$ or $F_p(n+1)$, it is necessary to use either $m_1 \approx \log_2 F_p(n+p+1)$ or $m_2 \approx \log_2 F_p(n+1)$ binary digits of the non-redundant code, respectively. Using (34), we can obtain the following formulas for the calculation of the relative code redundancy of the Fibonacci p-code (20):

$$r_1 = \frac{n}{\log_2 F_p(n+p+1)} - 1 \tag{35}$$

$$r_2 = \frac{n}{\log_2 F_p(n+1)} - 1 \tag{36}$$

The simplest redundant Fibonacci p-code (20) is the code, corresponding to the case p=1. We can calculate the limiting value of the relative redundancy for this code. For this case the formulas (35), (36) take the following forms, respectively:

$$r_1 = \frac{n}{\log_2 F_{n+2}} - 1 \tag{37}$$

$$r_2 = \frac{n}{\log_2 F_{n+1}} - 1, \tag{38}$$

where F_{n+2}, F_{n+1} are the classical Fibonacci numbers.

We can represent the Fibonacci numbers F_{n+2} and F_{n+1} by using Binet formulas [17]:

$$F_n = \begin{cases} \dfrac{\Phi^n + \Phi^{-n}}{\sqrt{5}} & for \quad n = 2k+1; \\ \dfrac{\Phi^n - \Phi^{-n}}{\sqrt{5}} & for \quad n = 2k \end{cases} \tag{39}$$

For a large n we can write Binet formulas (39) in the following approximate form:

$$F_n \approx \frac{\Phi^n}{\sqrt{5}} \tag{40}$$

Using (40) and substituting the approximate values for F_{n+2} and F_{n+1} given by (40), into the formulas (37) and (38), we can obtain the following formulas:

$$r_1 = \frac{n}{(n+2)\log_2 \Phi - \log_2 \sqrt{5}} - 1; \tag{41}$$

$$r_2 = \frac{n}{(n+1)\log_2 \Phi - \log_2 \sqrt{5}} - 1. \tag{42}$$

If we aim $n \to \infty$ in the expressions (41) and (42), we can see that they coincide for the case $n \to \infty$. Here the limiting value of the relative code redundancy for the Fibonacci (p=1)-code (23) is determined by the following expression:

$$r = \frac{1}{\log_2 \Phi} - 1 = 0.44. \tag{43}$$

Thus, the limiting value of the relative code redundancy of the Fibonacci (p=1)-code is a constant value equal to 0.44 (44%).

5. Fibonacci arithmetic

5.1. Comparison of numbers in the Fibonacci *p*-codes. As it is shown above, Fibonacci p-codes (20) are a new class of positional numeral systems. Fibonacci p-codes are similar to the binary system and are its generalization. Therefore, all the well-known properties of the positional numeral systems, in particular, of the binary system, can be used to create the Fibonacci arithmetic, although the difference consists of the fact that we should take into consideration fundamental features of the Fibonacci p-codes, in particular, such as the *plurality* of number representation and *MINIMAL FORM*.

Let us begin from the comparison of numbers in the Fibonacci p-codes. A comparison of Fibonacci representations A and B is carried out in the Fibonacci p-codes similarly to the classical binary code, if before the comparison we reduce the compared

Fibonacci representations to the MINIMAL FORM. This property (simplicity of number comparison) is one of the important arithmetical advantages of the Fibonacci p-codes.

For example we need to compare two Fibonacci representations A = 00111101101 and B = 00111110110, represented in the Fibonacci (p=1)-code (22). The comparison of numbers is performed in two steps:

1. To bring the compared Fibonacci representations to the MINIMAL FORM:

$$A = 01010010010 \text{ and } B = 01010100000 \qquad (44)$$

2. Compare digit-by-digit the MINIMAL FORMS (44), starting since the highest digit until obtaining first pair of the different bits:

$$A = 01010\mathbf{0}10010$$

$$B = 01010\mathbf{1}00000$$

We can see that the first non-coincident pair of the compared Fibonacci representations A and B contains the bit of 0 in the MINIMAL FORM of the first Fibonacci representation A and the bit of 1 in the MINIMAL FORM of the second Fibonacci representation B. This means that $B>A$.

5.2. Fibonacci counters. Algorithm of the Fibonacci summing counter is based on the following rule:

1) Before adding of bit 1 to LSB, the initial Fibonacci representation, corresponding to the number N, is reduced to such a form that the value of LSB will be equal to 0.

2) After that, we add the bit 1 to LSB what leads to increasing of number in the Fibonacci counter to the value N+1.

3) We repeat point 1 and 2 until overflowing Fibonacci counter.

Let us demonstrate this algorithm in the following **Example 1.**

Example 1. Algorithm of the Fibonacci summing counter

$$
\begin{aligned}
&000000+1=0000\boxed{\mathbf{01}}=000010=1\\
&000010+1=000\boxed{\mathbf{011}}=000100=2\\
&000100+1=0001\boxed{\mathbf{01}}=000110=3\\
&00\boxed{\mathbf{011}}0+1=0010\boxed{\mathbf{01}}=001010=4\\
&001010+1=001\boxed{\mathbf{011}}=001100=5\\
&0\boxed{\mathbf{011}}00+1=0100\boxed{\mathbf{01}}=010010=6\\
&010010+1=010\boxed{\mathbf{011}}=010100=7\\
&010100+1=0101\boxed{\mathbf{01}}=010110=8\\
&01\boxed{\mathbf{011}}0+1=\boxed{\mathbf{011}}0\boxed{\mathbf{01}}=100010=9\\
&100010+1=100\boxed{\mathbf{011}}=100100=10\\
&100100+1=1001\boxed{\mathbf{01}}=100110=11\\
&10\boxed{\mathbf{011}}0+1=1010\boxed{\mathbf{01}}=101010=12\\
&101010+1=101\boxed{\mathbf{011}}=1\boxed{\mathbf{011}}00=\boxed{\mathbf{110}}000=000000
\end{aligned}
\tag{45}
$$

Here we single out in bold in parentheses those situations where we can perform "convolutions" in Fibonacci representations. Consider, for example, the situation of the transition of the Fibonacci representation of the number 8 to the Fibonacci representation of the number 9. In this case, while recording of bit 1 to LSB of Fibonacci representation, we carry out convolution of bits 1 from the 2nd and 3rd digits in the 4th digit of Fibonacci representation ($011 \to 100$) what results to the Fibonacci representation of the number 9 ($\boxed{\mathbf{011}}00\boxed{\mathbf{01}} \to \mathbf{100}00\mathbf{10} = \mathbf{9}$).Note that the bottom row of the table (45) corresponds to the overflow of Fibonacci counter.

Subtraction of 1's in the Fibonacci code (22) is carried out by subtracting of bit 1 from the LSB of the Fibonacci representation of the number *N*, where the value of the LSB is equal to 1. Consider the **Example 2** of the functioning of the Fibonacci subtracting counter.

Example 2. Algorithm of the Fibonacci subtracting counter

$$
\begin{aligned}
&1111-1=11\boxed{\mathbf{10}}=1101=6\\
&1101-1=1\boxed{\mathbf{100}}=1011=5\\
&1011-1=10\boxed{\mathbf{10}}=1001=4\\
&\boxed{\mathbf{100}}1-1=0110=0101=3\\
&0101-1=0\boxed{\mathbf{100}}=0011=2\\
&0011-1=00\boxed{\mathbf{10}}=0001=1\\
&0001-1=0000=0
\end{aligned}
\tag{46}
$$

Here we single out in bold in parentheses those situations, when we can realize "devolutions" in Fibonacci representations. Thus, a feature of the Fibonacci subtracting counter consist of the fact that in any situation the transition from the Fibonacci representation of the number N to the Fibonacci representation of the number N-1 is performed during sequential operations of the "devolutions."

The above algorithms of summing and subtracting Fibonacci counters show that these counters the prerequisites for the construction of high-speed Fibonacci counters (without the use of complex schemes of group transfer). That is, this simple examples shows certain advantages of the Fibonacci code (23) in comparison to the classical binary code (22).

5.3. Fibonacci summation and subtraction. It is well known, the classical binary summation is based on the following elementary identity for the binary numbers:

$$2^k + 2^k = 2^{k+1} \tag{47}$$

where $2^k, 2^{k+1}$ are the weights of the k-th and (k+1)-th digits of the binary code (22), respectively .

The "deduction" of the Fibonacci p-summation rule begins from the analysis of the sum:

$$F_p(k) + F_p(k) \tag{48}$$

where $F_p(k)$ is the weight of the k-th digit of the Fibonacci p-code (20).

Let p = 1. For this case we have

$$F_p(k) = F_k\,, \tag{49}$$

where F_k is the classical Fibonacci numbers: 1,1,2,3,5,8,13,…, given by the recurrence relation (1). By using (1), we can represent the sum (48) as follows:

$$\text{(a) } F_k + F_k = F_k + F_{k\text{-}1} + F_{k\text{-}2} \tag{50}$$

$$\text{(b) } F_k + F_k = F_{k+1} + F_{k-2} \tag{51}$$

It follows from (50), (51) the following table for the Fibonacci summation (p=1) of the two binary digits $a_k + b_k$ with the same index k.

Table 6. Fibonacci summation for the case *p*=1

0	+	0	=	0	
0	+	1	=	1	
1	+	0	=	1	
1	+	1	=	1 1 1	(a)
1	+	1	=	1 0 0 1	(b)

It follows from Table 6 that the rules of the Fibonacci summation (p=1) coincide with the binary summation for the cases: 0+0=0, 0+1=1, 1+0=1; but for the case 1+1, the rules of the Fibonacci summation (p=1) don't coincide with the classic binary summation. For the case 1+1, the rules of the Fibonacci summation (p=1) are reduced to the following:

Rule 1. At the summation of the bits of 1 of the k-th digit of the summand Fibonacci representations, there arises the carry-over of two bits of 1 from the k-th digit at the other two digits.

Rule 2. There are two methods of carry-over formation. For the method (a) the bit of 1 is written to the k-th digit of the intermediate sum and the two carry-over's of 1 arise to the next two lower digits, namely to the (k-1)-th and (k-2)-th digits.

The method (b) assumes another rule of the Fibonacci summation (p=1) of the summand Fibonacci representations. The bit of 0 is written to the k-th digit of the intermediate sum and the two carry-over's of 1 arise to the other digits, namely to the (k+1)-th and (k-2)-th digits.

The summation of the multi-digit numbers in the Fibonacci code (p=1) is fulfilled in accordance with the Table 6. However, we should follow the following rules:

Rule 3. Before the summation, the summand Fibonacci representations are reduced to the MINIMAL FORM.

Rule 4. In accordance with Table 6 it is necessary to form multi-digit intermediate sum and multi-digit carry-over.

Rule 5. The multi-digit intermediate sum is reduced to the MINIMAL FORM and then is summarized with the multi-digit carry-over.

Rule 6. The summation process continues in accordance with the rules 4, 5 until obtaining the multi-digit carry-over equal to 0. The last intermediate sum, reduced to the MINIMAL FORM, is the result of Fibonacci summation.

For the above Fibonacci summation we need to add the following additional rule.

Rule 7. Consider the case when we have two binary 1's in the kth digits of the summand Fibonacci representations. It follows from the property of the MINIMAL FORM that the bits of the (k+1)-th and (k-1)-th digits of both summand Fibonacci representations are always equal to 0. It is clear that for this case the intermediate sums, arising at the addition of the (k+1)-th and (k-1)-th digits of the both summand Fibonacci representations, are always equal to 0. This means that we can place one of the carry-over's, arising at the summation of the k-th significant digits (1+1), at once to the (k-1)-th digit of the intermediate sum (for the (a)-method of the Fibonacci summation) or to the (k+1)-th digit of the intermediate sum (for the (b)-method of the Fibonacci summation).

We can demonstrate the above summation rules by the following example.

Example 3. Fibonacci summation of multi-digit Fibonacci representations

Sum the Fibonacci representations 31 = 10011011 and 22 = 01011010, represented in the Fibonacci code (23):

1. To bring the Fibonacci representations to the MINIMAL FORM*:*

$$31 = 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0$$

$$22 = 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0.$$

2. To form of the multi-digit intermediate sum S_1 and multi-digit carry-over C_1 in accordance with the method (a) of Table 6:

$$\begin{array}{l} 31 = 10100100 \\ \underline{22 = 10000010} \\ S_1 = 11100110 \\ C_1 = 00100000 \end{array}$$

3. To bring the intermediate sum S_1 to the MINIMAL FORM:

$$S_1 = 100101000$$

4. To sum S_1 and C_1:

$$\begin{array}{l} S_1 = 100101000 \\ \underline{C_1 = 000100000} \\ S_2 = 100111000 \\ C_2 = 000001000 \end{array}$$

5. To bring the intermediate sum S_2 to the MINIMAL FORM:

$$S_2 = 101001000$$

6. To sum S_2 and C_2:

$$\begin{array}{l} S_2 = 101001000 \\ \underline{C_2 = 000001000} \\ S_3 = 101001100 \\ C_3 = 000000010 \end{array}$$

7. To bring the intermediate sum S_3 to the MINIMAL FORM:

$$S_3 = 101010000$$

The summation is over because carry-over $C_4 = 0$.

As it is well-known, the method of the «direct» number subtraction in the classical binary arithmetic (p=0) is based on the following property of the binary numbers:

$$2^{n+k} - 2^n = 2^{n+k-1} + 2^{n+k-2} + \ldots + 2^n. \qquad (52)$$

Write now the similar identity for the Fibonacci (p=1)-numbers:

$$F_{n+k} - F_n = F_{n+k-2} + F_{n+k-3} + \ldots + F_{n-1}. \qquad (53)$$

Using the identity (53) and Fibonacci recurrence relation (1), we can construct the following Fibonacci subtraction table:

Table 7. Fibonacci subtraction table for the case *p*=1

0	-	0	=	0
1	-	1	=	0
1	-	0	=	0 1 1
1 0	-	1	=	0 1

1 0 0 - 1 = 1 1
1 0 0 0 - 1 = 1 1 1

The direct Fibonacci subtraction of multi-digit numbers uses the following rules:

Rule 1. Before subtraction, the subtracted Fibonacci representations are reduced to the MINIMAL FORM.

Rule 2. The subtracted MINIMAL FORMS are compared by their value according to the comparison rule of Fibonacci representations and then in accordance with Table 7, the smaller Fibonacci representation is subtracted from the bigger Fibonacci representation.

Note that these arithmetic algorithms are new technical solutions. They were developed by the author in the early 70s of the 20th century and these algorithms have been used as a basis of comparison devices, addition and subtraction devices, as well as summing and subtracting Fibonacci counters, patented abroad [17-30].

There are other rules of arithmetical operations in the Fibonacci code (23). In the book [33], the rules of Fibonacci subtraction, based on the use of the concepts of additional and inverse Fibonacci codes, are developed. In the same book [33] the original methods of the Fibonacci summation and subtraction, based on the so-called *"basic micro-operations*," are suggested.

5.4. Fibonacci multiplication and division.

5.4.1. Fibonacci multiplication. The analysis of the *Ancient Egyptian multiplication* [2] leads us to the following method of the *Fibonacci multiplication.*

Consider now the product $P = A \times B$, where the numbers A and B are represented in the Fibonacci p-code (20). Using the representation of the multiplier B in the Fibonacci p-code (20), we can write the product $P = A \times B$ in the following form:

$$P = A \times b_n F_p(n) + A \times b_{n-1} F_p(n-1) + \ldots + A \times b_i F_p(i) + \ldots + A \times b_1 F_p(1), \qquad (54)$$

where $F_p(i)$ is the Fibonacci p-number.

The following algorithm of the *Fibonacci p-multiplication* follows from the expression (54). The multiplication is reduced to the summation of the partial products of the kind $A \times b_i F_p(i)$. They are formed from the multiplier A according to the special

procedure that reminds the *Ancient Egyptian multiplication* [2]. Demonstrate now the "Fibonacci multiplication" for the case of the simplest Fibonacci p-code (p=1).

Example 4. Find the following product: 41×305.

Solution:

1. Construct the table consisting of the three columns: *F, G* and *P* (see Table 8).
2. Insert the Fibonacci 1-sequence (the classical Fibonacci numbers) 1, 1, 2, 3, 5, 8, 13, 21, 34 to the *F*-column of Table 8.
3. Insert the generalized Fibonacci 1-sequence: 305, 305, 610, 915, 1525, 2440, 3965, 6505, 10370, which is formed in the *G*-column from the first multiplier 305 according to the "Fibonacci recurrence relation."
4. Mark by the inclined line (/) and fat all the *F*-numbers that give the second multiplier in the sum (41 = 34 + 5 + 2).
5. Mark by fat all the *G*-numbers corresponding to the marked *F*-numbers and rewrite them to the *P*-column.
6. Summarizing all the *P*-numbers, we obtain the product: 41×305 = 12 505.

Table 8. Fibonacci multiplication for the case *p*=1

F	*G*	*P*
1	305	
1	305	
/2	**610**	→ **610**
3	915	
/5	**1 525**	→ **1 525**
8	2 440	
13	3 965	
21	6 505	
/34	**10 370**	→ **10 370**
41=34+5+2	**41×305**	= **12 505**

The above Fibonacci multiplication algorithm is easily generalized for the case of the Fibonacci p-codes (20) for arbitrary p.

5.4.2. Fibonacci division. Consider the **Example 5** of the Fibonacci division for the case $p=1$.

Example 5. Divide the number 481 (the dividend) by the number 13 (the divisor) in the Fibonacci code (9) ($p=1$).

The first stage

1. Construct the table consisting of three columns: *F, G* and *D* (see Table 9).
2. Insert the Fibonacci 1-sequence (the classical Fibonacci numbers) 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 to the *F*-column of Table 9.
3. Insert the generalized Fibonacci 1-sequence: 13, 13, 26, 39, 65, 104, 169, 273, 442, 615, formed from the divisor 13 according to the "Fibonacci recursive relation," to the *G*-column.
4. Compare sequentially every *G*-number with the dividend 481, inscribed into the *D*-column, and fix the result of comparison (≤ or >) until when we obtain the first comparison result of the kind (>): 615>481.
5. Mark with bold type and the incline line (/) the Fibonacci number 34, which corresponds to preceding G-number 442, and mark this number with bold type too. Calculate the difference: $R_1 = 481 - 442 = 39$.

Table 9. The first stage of the Fibonacci division

F	*G*	*D*
1	13	≤481
1	13	≤481
2	26	≤481
3	39	≤481
5	65	≤481
8	104	≤481
13	169	≤481
21	273	≤481
/34	**442**	≤481
55	615	>481

$R_1 = 481 - 442 = 39$

The second stage

The second stage of the Fibonacci 1-division is a repetition of the first stage but we use instead the dividend 481 the difference $R_1 = 39$ (see Table 10).

Table 10. The second stage of the Fibonacci division

F	*G*	*D*
1	13	≤39
1	13	≤39
2	26	≤39
/3	**39**	≤39
5	65	>39
$R_2 = 39 - 39 = 0$		

Because the second difference $R_2 = 39 - 39 = 0$, this means the Fibonacci 1-division is over. The result of division is equal to the sum of all the marked *F*-numbers, obtained on all stages (see Tables 9, 10), that is:

$$481 : 13 = 34 + 3 = 37$$

5.5. The device for bringing of Fibonacci representations to MINIMAL FORM

5.5.1. Boolean realization. The “convolution” and “devolution” devices play an important role in the technical realization of arithmetical operations over the Fibonacci representations. They can be designed on the base of the binary register having special logical circuits to perform “convolutions” and “devolutions”. Each digit of the register contains a binary flip-flop" (trigger) and logical elements. The operations of “convolution” (011 = 100) and “devolution” (100 = 011) can be performed by means of the inversion of the "flip-flops" (triggers).

One of the possible variants of the “convolution” register or “the device for reduuction of the Fibonacci code to the minimal form” is shown in Fig. 1.

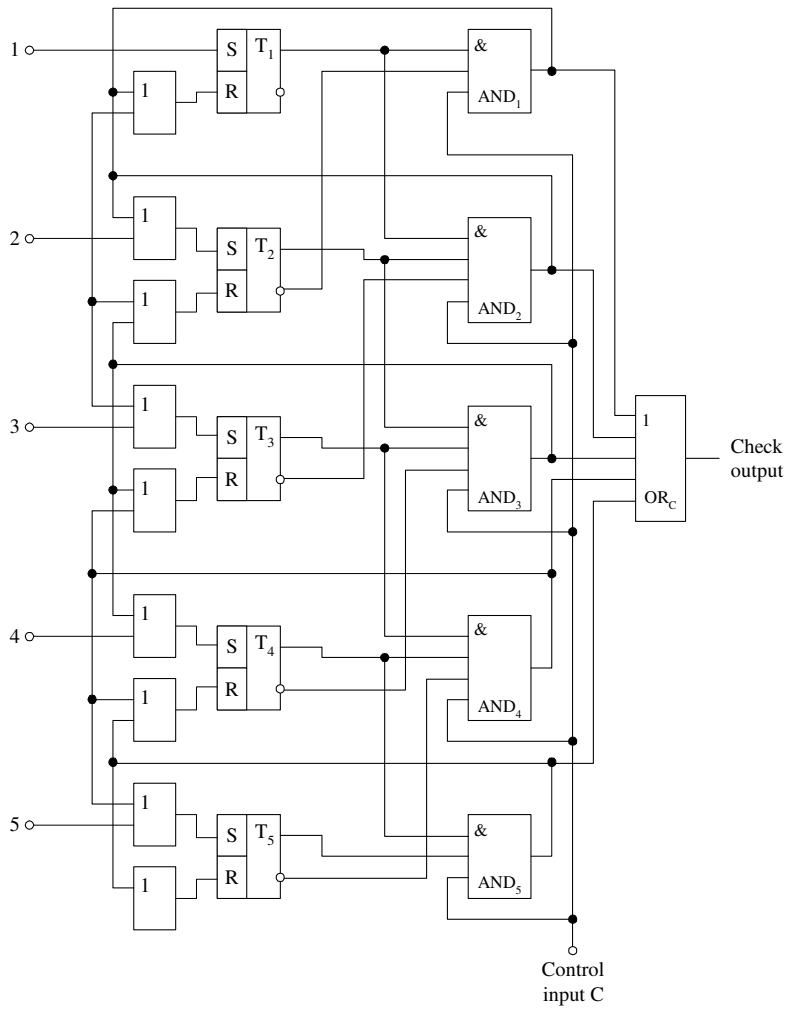

Figure 1. The device for reduction of the Fibonacci code to the MINIMAL FORM

The device in Fig.1 consists of the five *R*-*S*-triggers and the logical elements *AND*, *OR,* which are used to perform the “convolutions”. The “convolution” is performed by using the logical gates AND_1 - AND_5 and corresponding logical gate *OR,* standing before the *R*- and *S*-inputs of the triggers. The logical gate AND_1 performs the “convolution” of the 1-st digit to the 2-d digit. Its two inputs are connected with the direct output of the trigger T_1 and the inverse output of the trigger T_2. The 3-d input is connected with the synchronization input *C*. The logical gate AND_1 analyzes the states Q_1 and Q_2 of the triggers T_1 and T_2. If $Q_1 = 1$ and $Q_2 = 0$, this means that the convolution condition is satisfied for the 1-st and 2-d digits. The synchronization signal $C = 1$ causes

the appearance of the logical 1 at the output of the gate AND_1 . The latter causes switching the triggers T_1 and T_2. This results to the "convolution" (01 = 10).

The logical gate AND_k of the k-th digit (k=2, 3, 4, 5) performs the "convolution" of the (k-1)-th and k-th digits to the (k+1)-th digit. Its three inputs are connected with the direct outputs of the triggers T_{k-1} and T_k and the inverse output of the trigger T_{k+1}. The 4-th input is connected with the synchronization input C. The logical gate AND_k analyzes the states Q_{k-1}, Q_k, and Q_{k+1} of the triggers T_{k-1}, T_k, and T_{k+1}. If Q_{k-1} = 1, Q_k,= 1, and Q_{k+1} = 0, this means that the "convolution" condition is satisfied. The synchronization signal C = 1 results switching triggers T_{k-1}, T_k, and T_{k+1}. The "convolution" of the corresponding digits (011 = 100) is over.

Notice that all the gates AND_1 - AND_5 are connected through the common logical gate OR_c with the check output of the "convolution" register.

The device for bringing of the Fibonacci code to the MINIMAL FORM in Fig. 1 operates in the following manner. The input code information is sent to the information inputs 1 - 5 of the "convolution" register and enters the S-inputs of the triggers through the corresponding logical gates OR. Let the initial slate of the convolution register be in the following state:

5 4 3 2 1

0 1 0 1 1

It is clear that the "convolution" condition is satisfied only for the 1-st, 2-d and 3-d digits. The first synchronization signal C = 1 results in the passage of the "convolution" register to the following state:

5 4 3 2 1

0 1 1 0 0.

Here the "convolution" condition is satisfied for the 3-d, 4-s and 5-th digits. The next synchronization signal C = 1 results in the passage of the "convolution" register to the following state:

5 4 3 2 1

1 0 0 0 0.

The “convolution” is over.

5.5.2. The “convolution” register as self-checking device. The outputs of the logical gates AND_1 - AND_5 of the “convolution” registers in Fig. 1 are connected with the register check output through the common gate *OR*. This output plays an important role as the check output of the “convolution” register.

It follows from the functioning principle of the “convolution” register that the logical 1 appears on the check output only for two situations:

(1) The binary code word, written into the “convolution” register, is not MINIMAL FORM. This means that the “convolution” condition is satisfied at least for one triple of the adjacent triggers of the “convolution” register. This causes the appearance of the logical 1 at the output of the corresponding gate *AND*. Hence in this case the appearance of the logical 1 at the check output of the “convolution” register indicates the fact that the “convolution” process is not over. This means that we have a possibility to indicate the termination of the “convolution” process by means of observing the check output of the “convolution” register.

(2) The appearance of the constant logical 1 at the check output is an indication of the fault in the “convolution” register. Hence the “**convolution” register is a natural self-checking device** what is important for the improvement of informational reliability of Fibonacci computer.

There are other variants of implementation of such device, described in the book [33].

5.5.3. Device for checking MINIMAL FORM. Fig. 2 shows a device for checking MINIMAL FORM. The device consists of the *n* logical gates *AND*. Their outputs are connected with the inputs of the common logic gate *OR*. If the initial Fibonacci representation has a violation of the MINIMAL FORM, that is, has two adjacent bits of 1 or the bit 1 in the lower digit, there appears the logical 1 at least at one input of the logical gate *AND*. It results in the appearance of the logical 1 at the output of the common gate *OR* and this logical 1 is the error indication.

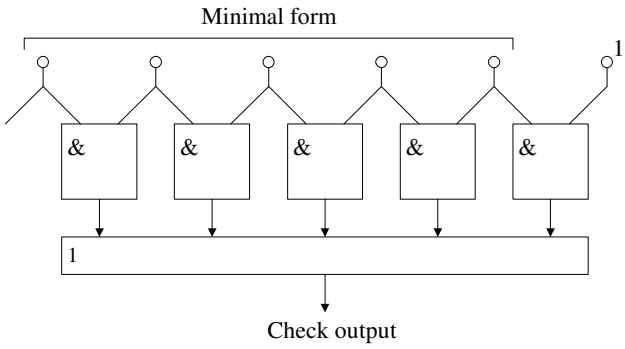

Figure 2. The logical circuit for checking the minimal form

6. Conception of Fibonacci computer

6.1. The main principles of Fibonacci computer

We can formulate the following main principles of the Fibonacci computer:

1. We use in Fibonacci computer the redundant Fibonacci *p*-codes. However, **Fibonacci (*p*=1) –code**, having the least code redundancy, is the **most suitable** in terms of hardware cost.

2. We use the **MINIMAL FORM** of the Fibonacci (*p*=1)–code for error detection at all stages of data transformation, including arithmetic operations, data transfer and storage.

Fibonacci computer has a number of important advantages in comparison with classic binary computers. The most important of these is the high ability to detect errors in functional units of Fibonacci computer.

6.2. Error detection

6.2.1. "Soft" and "hard" errors. As it is well-known, all errors, arising in functional devices of computer, can be divided into two groups: 1) the *"soft errors"* that result from random effects on electronic elements and the *"hard errors"* that result from constant failures of electronic elements. Both types of errors are dangerous and may lead to "false" data on the computer.

As for the *"hard errors"*, they can be detected by the register for bringing Fibonacci code to the MINIMAL FORM. This register is an important device of all arithmetic units and thanks to this device all Fibonacci arithmetic devices become self-checking devices. This is the first important advantage of the Fibonacci computer.

6.2.2. Formula for error-detecting ability of "soft errors". Let us consider now the error detection in such an important computer unit as an electronic memory. Error-detecting ability of the Fibonacci *p*-codes is determined by the relationship between the *allowed* and *forbidden* code combinations. Note that the set of the *allowed* combinations of the *n*-bit Fibonacci *p*-code coincides with the set of the *n*-bit MINIMAL FORMS. This means that the number of the *n*-bit *allowed* combinations is equal to $N_1 = F_p(n+1)$. Then, the number of the f*orbidden* code combinations is equal to $N_2 = 2^n - F_p(n+1)$, where 2^n is the number of the possible *n*-bit binary combinations. Then the **ERROR-DETECTING ABILITY** S_d of the Fibonacci *p*-code (for the case of the MINIMAL FORM), is determined as follows:

$$S_d(p) = \frac{N_2}{2^n} = 1 - \frac{F_p(n+1)}{2^n}. \tag{55}$$

For example, for the case of the 24-bit Fibonacci *p*-codes (*p*=1 and *p*=2) we have the following numerical values for the range of representations, respectively:

$$\boldsymbol{p = 1}: F_1(25) = 62215,\ 2^{24} = 16777216 \tag{56}$$

$$\boldsymbol{p = 2}: F_1(25) = 6450,\ 2^{24} = 16777216. \tag{57}$$

By using (55) - (57), we can calculate the ERROR-DETECTING ABILITY of the 24-bit Fibonacci 1- and 2-codes, respectively:

$$\begin{aligned} S_d(p=1) &= 0.9963(99.63\%) \\ S_d(p=2) &= 0.9996(99.96\%) \end{aligned} \tag{58}$$

6.2.3. Fibonacci Parity Code. As seen from the example (58), the potential error detecting ability of the Fibonacci *p*-codes is enough high. In order to improve the potential error-detection ability of the Fibonacci code, we can use the so-called Fibonacci Parity Code (FPC) by adding a PARITY BIT a_{par} to the MINIMAL FORM (MF) of the Fibonacci code:

$$\underbrace{a_n a_{n-1} \ldots a_i \ldots a_2 a_1}_{MF} \underbrace{a_{par}}_{PB} \tag{59}$$

The FIBONACCI PARITY CODE (59) significantly improves the error-detecting ability of the Fibonacci *p*-code. In this case, the main feature of the FPC is ensuring the 100% detection of all odd-bit errors, in particular, the single-bit errors. It is easy to prove [33] that the POTENTIAL ERROR-DETECTING ABILITY of the FPC (59) is calculated by the formula:

$$S_d\left(FPC\right)=1-\frac{F_p\left(n+1\right)}{2^{n+1}}. \tag{60}$$

For example, for the case of the 24-bit Fibonacci *p*-codes (p=1 and p=2) we have the following data ranges for the FIBONACCI-PARITY CODES (p=1 and p=2), respectively:

$$F_1(25)=62215,\ 2^{25}=33554432 \tag{61}$$

$$F_2(25)=6450,\ 2^{25}=33554432. \tag{62}$$

By using (60) for the cases (61), (62), we can calculate the ERROR-DETECTING ABILITY of the FIBONACCI-PARITY CODES (p=1 and p=2), respectively, as follows:

$$S_d\left(FPC, p=1\right)=0.9981459\left(99.8\%\right) \tag{63}$$

$$S_d\left(FPC, p=2\right)=0.9998078\left(99.98\%\right). \tag{64}$$

This means that the FIBONACCI-PARITY CODE can provide the continuous detection of errors in Fibonacci microcontroller or microprocessor at various stages of storage, transmission and processing of data with the error detection coefficient equal to 99.8 - 99.98%.

6.2.4. Energy consumption and power dissipation in ROM. It is known that for certain types of electronic memory (EM) there is some asymmetry between bits 1 and 0 at their storage in different kinds of electronic memory, for example, ROM. In particular, the recording of the bit 1 and its reading from the ROM requires more energy consumption, than for the bit 0. From this point of view, the MINIMAL FORM of the Fibonacci representations is an optimal binary representations from the point of view of **energy consumption and power dissipation**, because the bits 1 are separated always with bits 0 (in general, in the MINIMAL FORM, two bits 1 are separated by no less then *p* bits 0).

It is clear, in the array of the MINIMAL FORMS, the bits 1 and 0 are distributed non-uniformly; in this case, always the number of the bits 0 exceeds the number of the bits 1. This creates "comfortable" conditions for electronic memory, in particular, ROM, from the point of view of the **energy consumption and power dissipation**. The ROM's with fusible links or electrically programmable ROM's are examples of such kind of electronic memory. For such ROM's only the bits 1 determine energy consumption and power dissipation.

In order to estimate the decrease of energy consumption in ROM, when the data is stored in the MINIMAL FORM of the Fibonacci *p*-codes (20), we consider the following reasoning's. When we store in ROM the numbers, represented with the *m*-bit binary code, the maximal energy consumption appears at recording and reading of the following binary code combination:

$$N_{\max}(m) = \underbrace{11...1}_{m}. \tag{65}$$

If the storage of one bit of 1 demands the energy consumption P_1, then we can express the maximal energy consumption $P_{\max}(m)$ for the storage of the code combination (65) as follows:

$$P_{\max}(m) = mP_1 . \tag{66}$$

With the *m*-bit binary code we can represent 2^m positive integers, that is, the range of the number representation is equal:

$$D = 2^m . \tag{67}$$

If we will represent the positive integers in the MINIMAL FORM of the Fibonacci *p*-code (20), then, due its code redundancy, for the storage of the same range of positive integers, given by (67), we need to increase the number of digits of the Fibonacci *p*-code (29) proportionally to its relative code redundancy r. In this case, the number of the Fibonacci's bits *n* for the storage of the number range (67) is equal approximately to:

$$n = m(1+r). \tag{68}$$

The formula (68) shows that the number of the bits n in the Fibonacci *p*-code (20) increases by $(1+r)$ times in comparison with the number of the bits *m* for the binary system, necessary for the representation of the same number range (67). In this case, the *n*-bit Fibonacci

representation of the maximal number $N_{\max}$ in the MINIMAL FORM of the Fibonacci p-code (20) looks as follows:

$$N_{\max} = 1\underbrace{00...0}_{p}1\underbrace{00...0}_{p}...1\underbrace{00...0}_{p}...1\underbrace{00...0}_{p}. \tag{69}$$

In particular, for the case p=1 we have:

$$N_{\max} = 1010...10...10. \tag{70}$$

We can see that the Fibonacci representation (69) consists of the k groups of the kind $1\underbrace{00..0}_{p}$, which contains one bit of 1 and p bits of 0, where k is the number of the bits of 1 in the representation (69). By using (68) and (70), we can express the number of k as follows:

$$k = \frac{n}{p+1} = \frac{m(1+R)}{p+1}. \tag{71}$$

It follows from (71) that the maximal energy consumption for the storage of the n-bit Fibonacci representation is determined by the expression:

$$P_{\max}^{f} = kP_1 = \frac{m(1+r)}{p+1}P_1. \tag{72}$$

Consider now the ratio:

$$\beta = \frac{P_{\max}}{P_{\max}^{f}} = \frac{p+1}{1+R}. \tag{73}$$

Because R<1 and $p \geq 1$, then the coefficient β describes the decrease of the energy consumption in ROM, if we use the Fibonacci p-codes (20).

Table 11 sets forth the values of β for the cases p=1, 2.

Table 11. The values of β

p	1	2
R	$\approx 0.33(33\%)$	$\approx 0.5(50\%)$
β	1.5	2

Thus, for the case p=2 the improvement in energy consumption is about 2 times, despite the fact that the number of digits in Fibonacci 2-code increases by about 1.5 times compared with the classic binary system. It follows from Table 11 that the gain in the energy consumption in the ROM increases with increasing p. This result could have great significance for the future of nano-

computers and microelectronics, where decreasing energy consumption and optimizing power dissipation is becoming one of the central problems.

6.2.5. Fibonacci *p*-codes as self-synchronization codes. A distinction of the bits 1 and 0 in sequentially transmitted binary data relates to a serious technical problem, called *problem of synchronization*. To solve this problem, special synchronization signals, clock-signals, need to be used. One of the effective ways for improving of the synchronization of transmitted data (without special clock-signals of synchronization) is to use self-synchronisation codes (see diagram B in Fig.3).

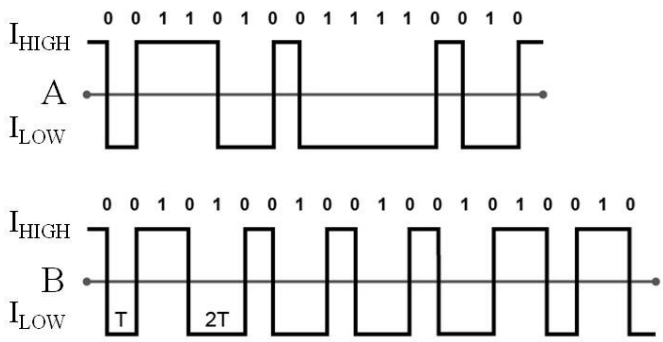

Figure 3. Fibonacci code as self-synchronization code

The diagrams A, B represent a widespread method transmitting binary data. Switching two electrical currents performs the transmission of bits: the high level current I_{high} and the low level current I_{low}. The switching from one level to another is performed only for the bit 0; the switching is not performed for the bit of 1. The diagram A shows the electrical signal, which is generated during the transmission of the classic binary code combination without self-synchronization mechanism. The greatest difficulty in the transmission of the signal arises when the transmitted binary sequence contains a long "packet," consisting of bits 1.

The problem of distinction of bits 1 and 0 in this method is greatly simplified if we impose certain restrictions on the length of the "packets," consisting of the consecutive bits of 1. The simplest solution is to use the codes, in which two bits of 1 do not appear together. This restriction is a basic property for MINIMAL FORMS of all the Fibonacci *p*-codes (20). For the first time, American engineer **W.H. Kautz** paid attention to such use of the Fibonacci code for synchronization control [50].

The diagram B shows the example of the formation of the digital signal by using simplest Fibonacci code (p=1). Denote by T the period of the original code sequence. Analysis of the digital signal, generated from the MINIMAL FORM of the Fibonacci code (p=1), shows that this digital signal contains only two pulse durations T and $2T$. This digital signal is called *dual-frequency signal*. It is easy to form clock-signals from the dual-frequency signal. This fact is a confirmation of the self-synchronization property of the Fibonacci p-codes (p=1,2,3,…).

Conclusion

1. The most significant advantage of the Fibonacci numeral systems in comparison to other methods of introduction of code redundancy in computer systems is that they are a generalization of a classical binary system and retain all arithmetical and technical advantages of the binary system (simple rules of arithmetical operations, simplicity of number comparison, iterative or uniform arithmetic structures).
2. In the well-known methods of introducing code redundancy into computer systems (e.g., error-correcting codes) we always assume the presence at least two codes: the *source code* to represent the numbers and *error-correcting code*; this creates the so-called *encoding-decoding problem*. A complexity of encoder and decoder, and the need to take into account the errors in these devices in many cases can negate the benefits of error-correcting codes. In the Fibonacci numeral systems, the *encoding-decoding problem* does not arise, because the Fibonacci representations are used simultaneously to *represent numbers, perform arithmetic operations and to detect errors*. Besides error detection with high detection coefficient, Fibonacci numeral systems have other useful properties, such, as a solution to the *self-synchronization problem* in serial code sequence. The *self-synchronization property* allows creating high-performance serial data transmission systems.
3. Another additional feature is the improvement of *energy consumption and power dissipation.* It is shown that when we store Fibonacci representations in MINIMAL FORM, a gain in energy consumption for the worst case is equal to

1.5 or 2 for Fibonacci (p=1)- and (p=2)- codes, respectively. Property of MINIMAL FORM (every bit of 1 is surrounded at the left and the right by at least p bits of 0) is useful to improve *power dissipation*. The solution of this problem is particularly important for certain types of micro-electronic and nano-electronic memory.

4. The totality of the above advantages of the Fibonacci numeral systems (*high coefficient of error detection, properties of self-synchronization, improvement of energy consumption and power dissipation*) gives us a right to single out the Fibonacci numeral systems as a unique multi-functional method of introducing code redundancy, which exceeds all well-known methods, in particular, error-correcting codes, first of all in qualitative relation.
5. **Proposed areas of applications**:

5.1 **. SPACE:** spacecraft control and onboard systems, satellites etc.

5.2 **. ENERGY:** nuclear power stations and other energy facilities.

5.3 **. TRANSPORT:** aircraft, trains, subway and cars.

5.4 **. AUTOMATION:** banks, factories and public services.

5.5 **. MEDICAL:** computerised medical equipment.

5.6 **. COMMUNICATION:** telephone, Internet.

5.7 **. SCIENCE:** research computers, super computers.

5.8 **. MEDIA:** TV, radio broadcasting.

5.9 **. ROBOTICS:** control systems for robots and robotic tools

5.10 **NANO:** a basis for the creation of nano-computers and electronics where requirements of reliability and survivability are vital.

References

1. Bashmakova IG, Yushkevich, AP. (1951). The origin of the numeral systems. - Encyclopaedia of Elementary Mathematics, Volume 1 "Arithmetic". Moscow: Gostehidat, 1951. - p.7-74 (Rssian).
2. *Ancient Egyptian multiplication*. From Wikipedia, the free encyclopaedia https://en.wikipedia.org/wiki/Ancient_Egyptian_multiplication

3. *Von Neumann architecture*. From Wikipedia, the free encyclopaedia https://en.wikipedia.org/wiki/Von_Neumann_architecture
4. Hetagurov J.A. (2009). Ensuring the national security of real-time systems, *BC / NW* 2009; №2 (15): 11.1
5. Yanushkevich Svetlana N., Lyshevski Sergey E., Shmerko Vlad P. *Computer Arithmetics for Nanoelectronics*. CRC PRESS: Boca Raton, Ann Arbor, London, Tokyo.
6. Stakhov, A.P. (1974). Redundant binary positional numeral systems. *In the book "Homogenous digital computer and integrated structures."* Taganrog: Publishing House "Taganrog Radio University", No 2, 3-40 (Russian).
7. Stakhov, A.P. (1975). A use of natural redundancy of the Fibonacci number systems for computer systems control. *Automation and Computer Systems*, 1975, No 6 , 80-87 (Russian).
8. Stakhov, A.P. (1978). Digital Metrology on the base of the Fibonacci codes and Golden Proportion Codes. *In the book "Contemporary Problems of Metrology."* Moscow: Publishing House of Moscow Machine-building Institute, 1978, 51-65 (Russian).
9. Stakhov, A.P. (1978). Fibonacci and "Golden" Ratio Codes. *In the book "Fault-tolerant Systems and Diagnostic FTSD-78,"* Gdansk.
10. Stakhov, A.P. (1980). The golden mean in the digital technology. *Automation and Computer Systems*, 1980, No 1, 27-33 (Russian).
11. Stakhov, A.P. (1988). Algorithmic measurement theory and fundamentals of computer arithmetic. *Measurement. Control. Automation*, 1988, No 2, 3-21 (Russian).
12. Stakhov, A.P. (1989). The Golden Section in the Measurement Theory. *Computers & Mathematics with Applications,* 1989, 17 (4-6), 613-638.
13. Stakhov, A.P. (1990). The Golden Proportion Principle: Perspective Way of Computer Progress. *Bulletin of the National Academy of Sciences of Ukraine*, 1990, No 1-2, 31-48 (Ukrainian).

14. Stakhov, A.P. (1994). Algorithmic Measurement Theory: a General Approach to Number Systems and Computer Arithmetic. *Control Systems and Computers*, 1994, No 4-5, 10-31 (Russian).
15. Collection of articles (1989) *Noise-tolerant codes. Fibonacci computer*. Moscow: Publishing House "Nauka" (Russian).
16. Stakhov, AP. (1977). *Introduction into algorithmic measurement theory*. Moscow: Publishing House "Soviet Radio" (Russian).
17. Stakhov, A.P. (1984). *Codes of the Golden Proportion*. Moscow: Publishing House "Radio and Communication" (Russian).
18. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of USA No 4187500.
19. Device for reduction of *p*-Fibonacci codes to the minimal form. Patent certificate of USA No 4290051
20. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of England No 1543302.
21. Device for reduction of *p*-Fibonacci codes to the minimal form. Patent certificate of England No 2050011.
22. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of Germany No 2732008.
23. Device for reduction of *p*-Fibonacci codes to the minimal form. Patent certificate of Germany No 2921053.
24. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of Japan No 1118407.
25. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificates of France No 7722036, No 2359460.
26. Device for reduction of *p*-Fibonacci codes to the minimal form. Patent certificates of France No 7917216, No 2460367.
27. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of Canada No 1134510.
28. Device for reduction of *p*-Fibonacci codes to the minimal form. Patent certificate of Canada N1132263.

29. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of Poland No 108086.
30. Reduction method of *p*-Fibonacci code to the minimal form and device for its realization. Patent certificate of DDR No 150514
31. Stakhov AP. (2002). Brousentsov's ternary principle, Bergman's number system and ternary mirror-symmetrical arithmetic. *The Computer Journal*, 2002, Vol. 45, No. 2: 222-236.
32. Stakhov A. (2006). Fibonacci matrices, a generalization of the "Cassini formula", and a new coding theory. *Chaos, Solitons & Fractals*, 2006, Vol. 30, Issue 1, 56-66
33. Stakhov A.P. (2009). *The Mathematics of Harmony. From Euclid to Contemporary Mathematics and Computer Science*. Assisted by Scott Olsen. New Jersey, London, Singapure, Beijing, Shanghay, Hong Kong, Taipei, Chennai: World Scientific.
34. Borisenko, A.A., Stakhov, A.P. (2011). A method for counting in the Fibonacci code. *Bulletin of Sumy State University*, 2011, №3, - s.141-149.
35. Stakhov, A., Borisenko, A., Matsenko, S. (2014). Fibonacci counter based on Zeckendorf's Theorem (Boolean Realization). *Visual Mathematics*, 2014, Issue 16, No 2 http://elib.mi.sanu.ac.rs/files/journals/vm/57/vmn57p1-19.pdf
36. Stakhov A.P. (2015). The "Golden" Number Theory and New Properties of Natural Numbers. *British Journal of Mathematics and Computer Science*, 2015, Vol. 11, Issue 06.
37. Monteiro P. and R.W. Newcomb (University of Maryland) (1976). Minimal and maximal Fibonacci Representations: Boolean Generation. *The Fibonacci Quarterly*, 1976, Vol.14, No 1.
38. Licomendes, P. and Newcomb, R. (University of Maryland) (1984). Multilevel Fibonacci Conversion and Addition. *The Fibonacci Quarterly*, 1984, Vol. 22, No 3.
39. Ligomenides, P. and Newcomb, R. (1981). Equivalence of some Binary, Ternary, and Quaternary Fibonacci Computers. *Proceeding of the Eleventh International Symposium on Muliple-Valued Logic*, Norman, Oklahoma.

40. Ligomenides, P. and Newcomb, R. (1981). Complement Representations in the Fibonacci Computer. *Proceedings of the Fifth Symposium on Computer Arithmetic,* Ann Arbor, Michigan.
41. Newcomb, R. (1974). Fibonacci Numbers as a Computer Base. *Conference Proceedings of the Second Inter-American Conference on Systems and Informatics*, Mexico City.
42. Hoang V.D. (1979). A Class of Arithmetic Burst-Error-Correcting Codes for the Fibonacci Computer. PhD Dissertation, University Maryland.
43. Polya, George. (1962). *Mathematical Discovery*. Publishing House "John Wiley & Sons".
44. Vorobyov, N.N. (1969). *Fibonacci Numbers.* Third edition. Moscow: Nauka, 1969, (the first edition, 1961) (Russian).
45. Hoggatt, V. E., Jr. (1969). *Fibonacci and Lucas Numbers*. Boston, MA: Houghton Mifflin.
46. Vajda, S. (1989). *Fibonacci & Lucas Numbers, and the Golden Section. Theory and Applications*. Ellis Harwood Limited.
47. Kharkevitch, A.A.(1963). *A struggle against noises.* Moscow: Publishing House "Fizmatgiz" (Russian).
48. Pospelov, D.A. (1970). *Arithmetical foundations of computers*. Moscow: Publishing House "Vyshaja shkola" (Russian).
49. Kasatkin, V.N. (1989). *News on numeral systems*. Kiev: Publishing House "Vyscha shkola" (Russian).
50. Kautz, W.H. (1965). Fibonacci codes for synchronization control. – *IEEE Trans. Inform. Theory*, 1965, Vol.11, No 8. - 284-292