Робота присвячена розробці та дослідженню нетрадиційного високоефективного розрядного кортежно-табличного логіко-оборотного методу, на основі якого створюються прецизійні моделі обчислювальних перетворювачів інформації, представленої у вигляді однополярних двійково-кодових операндів з позиційно-впорядкованої формою запису.

Сучасні моделі перетворювачів, побудовані традиційними методами, не несуть, як правило, обчислювального навантаження і є узгоджуючими компонентами, що забезпечують необхідну форму подання інформації, як на вході, так і на виході обчислювального пристрою. При цьому вони мають ряд обмежень, виконання яких вимагає апаратурною підтримки, що призводить до збільшення ваги і габаритів, погіршення надійності та енерго-часових показників, підвищенню вартості.

Тому розробка нового нетрадиційного методу, що перетворює різного виду позиційно-впорядковані двійково-кодові операнди в певні значення кодової комбінації і, навпаки, використовуючи одні й ті ж табличні дані відповідності (попередньо обчислені), є актуальною задачею.

Метод включає: формування таблиць відповідності на базі формальної логіки; визначення значень коригуючих констант, використовуючи операцію XOR; ліквідацію інформаційної надмірності завдяки кортежній декомпозиції та синтез компонентів моделі обчислювального перетворювача інформації. Сукупність процедур забезпечує багатофункціональність, високу швидкодію і надійність, зменшення енергоспоживання при збереженні прецизійності результатів.

Верифікація запропонованої логіко-математичної моделі для створення ефективного методу, який перетворює різного виду двійково-кодові операнди, підтверджена розрахунками коригувальних констант, наведеними в таблицях, та експериментом. Експеримент проведено на розробленій фізичній моделі з єдиним числовим блоком пам'яті, яка перетворює двійковий код в код Грея і навпаки.

Запропоновані оригінальні багатофункціональні обчислювальні перетворювачі дозволяють з меншими енерго-часовими та апаратурними витратами вирішувати локальні завдання управління в комп'ютерно-інтегрованих системах спеціального призначення для управління високошвидкісними технологічними процесами або автономними фізичними об'єктами

Ключові слова: компоненти комп'ютерно-інтегрованих систем, кодоперетворення, двійковий-кодові операнди, розрядний кортежно-табличний логіко-оборотний метод

UDC 681.5:519.24
DOI: 10.15587/1729-4061.2018.142975

BITWISE METHOD FOR THE BINARY-CODED OPERANDS CONVERSION BASED ON MATHEMATICAL LOGIC

A. Lukashenko
PhD, Senior Researcher*
E-mail: ineks-kiev@ukr.net
D. Harder
Junior Researcher*
E-mail: lukash_d@ukr.net
V. Lukashenko
PhD, Junior Researcher*
E. Fedorov
Doctor of Technical Sciences,
Associate Professor, Head of Department
Department of Computer Sciences
Donetsk National Technical University
Shybankova sq., 2, Pokrovsk, Ukraine, 85300
E-mail: yevhen.fedorov@donntu.edu.ua
V. Lukashenko
Doctor of Technical Sciences,
Professor, Head of Department**
E-mail: ckc@chdtu.edu.ua
T. Utkina
PhD, Associate Professor**
E-mail: t.utkina@chdtu.edu.ua
S. Mitsenko
PhD, Senior Lecturer**
E-mail: s.mitsenko@chdtu.edu.ua
K. Rudakov
PhD, Senior Lecturer**
E-mail: k.rudakov@chdtu.edu.ua
*E. O. Paton Electric Welding
Kazymyra Malevycha str., 11, Kyiv, Ukraine, 03680
**Department of Robotics
and Specialized Computer Systems
Cherkasy State Technological University
Shevchenka blvd., 460, Cherkasy, Ukraine, 18006

1. Introduction

At present, computer-integrated system (CIS) for special purposes are widely used in space technology, gyroscopic platforms, radar installations, robotics, resource-saving technological complexes, etc., which operate in real time. The development of computational tools puts forward a number of tasks related to the need to meet the ever-increasing

requirements for CIS components in terms of performance speed, representation accuracy of calculation results, reliability, power consumption, weight, dimensions, and cost at the same time.

One of the main CIS components are the computational converters of binary-coded information.

Binary-coded operand converters with different systems of calculus are often the aligning components both at the input and output of a computing device. Such models of converters, built using traditional methods, are not typically computationally loaded. At the same time, they have a number of constraints meeting which requires hardware support, which leads to an increase in weight and dimensions, deterioration of reliability and energy-time indicators, and increases the cost. For example, the interference-resistant unweighted Gray code is not applicable for computational operations, which is why there is a need to convert it into a binary weighted code. That increases hardware, energy, and time costs, resulting in a clear decrease in the CIS structure parameters in general and deterioration of its technical-economic indicators.

Computational converters of binary-coded systems that operating under extreme conditions are mostly built on the basis of classical tabular methods (CTM) or tabular-algorithmic methods (TAM). The CTM tables contain two sets: values for the input codes (argument) $\{x_i\}$ and values for the output codes (function) $\{y_i\}$, while the TAM tables include $\{x_i\}$ and the corresponding values for corrective constants $\{\Delta_i\}$.

The procedure of conversion implementation comes down to constructing a read only memory (ROM), consisting of a decoder for CTM, or a combinational circuit for the TAM address scheme and numeric memory blocks that store, respectively, the sets $\{y_i\}$ or $\{\Delta_i\}$.

The advantage of ROM fabricated based on CTM on a single chip is their small weight and dimensions, high performance speed and high reliability, and their low cost at industrial production.

It is known that the volume of ROM depends on the bit size of the operands, for example, for $n \geq 32$ bits, ROM volume reaches ≥1,011 bits. Despite advances in micro- and nanotechnologies, when manufacturing large integrated circuits (LIC) with a high degree of integration, there increases the percentage of defective crystals per plate and, as a consequence, the cost of ROM LIC grows. At the same time, there is an increase in the number of hidden technological defects, which reduces the reliability of the product.

The task on compressing the volume of tables is resolved by applying tabular-algorithmic methods, at which ROM records corrective constants, most often the results of calculations for part of the source data set, using the arithmetic and logical operations. This is not acceptable for special purpose systems that working in real time, due to the increased time on time-consuming operations, such as multiplication, division, etc. when processing code combinations.

Despite the large number of papers on methods for converting the binary-coded operands [1–21], there is no a unified theory for the unconventional method of bitwise code conversion with different number systems. It is desirable that it should contain simple ratios that would link the values for binary-coded operands not only for different types of number systems, but also the unipolar codes with a positional form of notation.

The search for new principles of conversion has led to the development of the unconventional bitwise tuple-tabular logical-reverse (BTTLR) method of hardware implementation, based on the proposed mathematical logic (ML). It makes it possible to directly convert the value of input information in the form of a multi-bit binary coded combination in a single number system into the corresponding value of output in the form of a binary code of another number system. A limitation in the application of this method is the execution of positioning, orderliness, and a unipolar form of the operand notation.

Thus, a BTTLR method of hardware implementation is promising for further development and introduction of new principles for the binary-code conversion designed for special purposes CIS that covers a rather broad scope of industry and science.

2. Literature review and problem statement

Traditional methods to convert multi-bit binary-coded operands do not always meet the high requirements put to the modern components of special-purpose CIS due to the large number of additional discrete elements or considerable time cost. For example, the implementation of converters based on software algorithm is distinguished by high flexibility, but they are characterized by a slow information processing rate and the complexity of computation [1].

Authors of paper [2] examine the transformation of information using the optimal choice of the generating polynomial. However, obtaining a result using it includes long arithmetic operations: multiplication, exponentiation, etc.

The disadvantage of known specialized digital conversion systems, which are manufactured by firms Fourier Systems Ltd. (Israel) [3], PascoInc. (United States) [4], Phywe GmbH (Germany) [5], is their high cost.

In addition, there are information converters for the industrial automation systems [6], which require additional equipment to align signals from sensors with computational devices, which leads to the bulkiness of the converter and increases the cost of the complex in general.

The designed hardware-software platform [7] only partially elininates the above shortcomings, but requires additional expenditures for the development of proprietary software.

The high-performance methods for converting the binary-coded special-purpose operands include tabular methods [8]. However, when processing multi-bit information, they require a significant volume of ROM, for example, a volume of the numeric memory block for operands with a bit size of $n=32$ is $32\cdot(2^{32}-1)\approx 10^{11}$ bits. This reduces the percentage of yield of suitable crystals per plate and, as a consequence, increases the cost of the product.

Several important results were reported in papers that relate to the structural organization [9], generation [10], and determining the best binary cyclic codes [11], however, their disadvantage is the slow performance of the information conversion process.

Paper [12] shows that the most widely used codes in computational equipment are the binary-decimal codes that satisfy the condition for uniqueness, they have the properties of additivity, orderliness, balance, parity, additionality. However, the code «8, 4, 2, 1» hinders the formation of transfer from a lower tetrad to the higher tetrad. The code with a «redundant 3» lacks the property of weighting. The code «2, 4, 2, 1» with an artificial order of weights lacks the

mutually unambiguous correspondence between decimal digits and their binary codes.

Also of interest are the weighted codes in the binary numeral system [13] whose main advantage is the universality of number representations, while the drawback is performing arithmetic operations only at full-bit representation of multibit operands. That reduces the information conversion speed, which is not always acceptable for the special purpose CIS components.

The sign-bit number system [14] has a number of advantages: the possibility of organizing digital information processing from higher bits to the lower ones; the extensibility of computation bit size. The main disadvantage is slow performance due to the local distribution of carries when performing arithmetic operations.

A significant improvement in performance is provided by a parallel implementation of basic arithmetic operations and a reduction in the amount of memory for function tabulations when using code combinations in the system of residual classes (SRC). However, the obstacle to widespread application of SRC for the specialized computing tools is the complicated division algorithms (when a result is not an integer) and the difficulty in locating an overflow [13].

Work [15] fails to provide a description of the method for an error-free recovery of the original coded information at a reverse operation. Despite the large number of papers on the conversion of codes, there are no any grouped set of different kinds of binary-coded operands, which are the most needed ones for special-purpose systems. For instance, there are no descriptions of conversion of the non-weighted noise-resistant code with a positional notation or a unipolar noise-like code directly into a binary code of the value for a transcendent function and vice versa. Therefore, development of the method for converting the codes of different number systems using a common methodological and information basis is a promising task.

3. The aim and objectives of the study

The aim of this study is to develop a bitwise tuple-tabular logical-reverse method for converting multibit binary-coded operands with a positionally-ordered notation based on mathematical logic for the computer-integrated systems for special purposes.

To accomplish the aim, the following tasks have been set:

– to generate a set of modern positionally-ordered binary-coded (POBC) operands for the special purpose CIS components;

– to define a formalized logical-mathematical model that would describe the direct and inverse conversion of POBC multi-bit operands using corrective constants for different number systems;

– to implement the principles of forming the output binary-coded operands based on the adjustment of input codes;

– to build a multifunctional model for converting a noise-resistant code with a positional notation into multi-bit binary-coded combinations of values for functions $\sin(x)$, $\mathrm{tg}(x)$, $\mathrm{th}(x)$ and a natural binary code based on BTTLR method;

– to verify the proposed BTTLR method based on the designed physical model that would convert the Gray code into a weighted binary code and vice versa, utilizing the same volume of the memory numeric block.

4. Defining the class of positionally-ordered binary-coded operands for the special purpose converters

The complexity of tasks on managing objects within CIS, and especially in the special-purpose systems, is explained by the increased requirements to the basic technical-economic indicators of their components (such as performance speed, accuracy, reliability, power consumption, weight, dimensions, and cost).

We should note the peculiarity in the design and manufacture of special purpose converters. That implies that one knows in advance the input set of the converted operands in a single number system and the corresponding output set of the same or a completely different number system of the converted binary-coded operands [16–18].

A customer often demands that high performance speed, high accuracy, high reliability, good weight-and-size indices, low cost, small energy consumption and reversibility should be simultaneously ensured [19–21].

Conversion reversibility is characterized by that in the application of a reverse operation, the source information is restored without errors [22–25].

Fig. 1 shows a set of binary-coded operands for the direct and inverse conversion, which are most in demand in the special purpose CIS.

The principal attributes of the models of computational information converters for special purposes are positioning, orderliness, representation of the binary-coded operands in the form (0, +1), codes with a positional unipolar notation, codes from different number systems.

An analysis of the set of existing binary-coded operands (Fig. 1) confirms the variety of their notation systems. Some of them do not contain computational operations and are applied to match the input information source and the computational unit of a special-purpose CIS.

The problem is that there are no simple correlations linking the values for different types of the positionally-ordered binary-coded operands regardless of number systems and the form of notation.

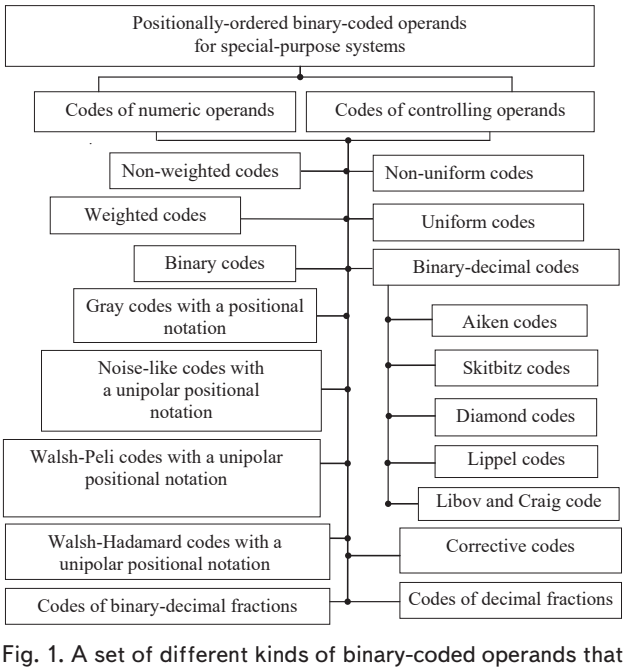

Fig. 1. A set of different kinds of binary-coded operands that are employed in special-purpose systems

To this end, it is proposed to reproduce a unified formalized model to describe the procedure of converting codes of different kinds based on mathematical logic.

5. Formalized logic-mathematical model for converting binary-coded operands of various types

A *logic-mathematical model* (LMM) for the formalized description of the procedure to convert the positionally-ordered input operands into respective output binary-coded operands is constructed using corrective constants derived from the *XOR* operation within Zhegalkin algebra.

A special feature of the formal logic is that one defines the location (position) of unity «1» and zero «0».

Let the positionally-ordered converted set X and its corresponding predefined converted set Y of binary-coded sequences be recorded in the following form:

$$\left.\begin{aligned} X &= (x_{n-1}\, x_{n-2} \dots x_{n-i} \dots x_1\, x_0), \\ Y &= (y_{n-1}\, y_{n-2} \dots y_{n-i} \dots y_1\, y_0). \end{aligned}\right\} \quad (1)$$

A bitwise difference between the code sequence that is converted and the code sequence that has been converted, when using the properties of the Zhegalkin algebra's *XOR* operation, takes the form:

$$\begin{aligned} &\Delta_{\oplus} = X \oplus Y = \\ &= (x_{n-1} \oplus y_{n-1}) \dots (x_{n-i} \oplus y_{n-i}) \dots (x_1 \oplus y_1)(x_0 \oplus y_0), \end{aligned} \quad (2)$$

where

$$\begin{aligned} &(x_{n-1} \oplus y_{n-1}) = \Delta_{n-1}; \dots (x_{n-i} \oplus y_{n-i}) = \\ &= \Delta_{n-1}; \dots (x_1 \oplus y_1) = \Delta_1;\ (x_0 \oplus y_0) = \Delta_0. \end{aligned}$$

A positionally-ordered code of the difference is a corrective constant $\Delta_{\oplus}$ that is represented in the following form:

$$\Delta_{\oplus} = (\Delta_{n-1}\, \Delta_{n-2} \dots \Delta_{n-i} \dots \Delta_1\, \Delta_0). \quad (3)$$

Code sequence (3) maintains positionality and orderliness:

$$0 < \Delta_0 < \Delta_1 < \Delta_2 < \dots < \Delta_{n-i} < \dots < \Delta_{n-2} < \Delta_{n-1},$$

$$0 > \Delta_1 > \Delta_2 > \dots > \Delta_{n-i} > \dots > \Delta_{n-2} > \Delta_{n-1}. \quad (4)$$

The positionally-ordered sequence of the converted code (direct code) is calculated using LMM, recorded in the following form:

$$\begin{aligned} &Y = X \oplus \Delta_{\oplus} = \\ &= (x_{n-1} \oplus \Delta_{n-1}) \dots (x_{n-i} \oplus \Delta_{n-i}) \dots (x_1 \oplus \Delta_1) \dots (x_0 \oplus \Delta_0). \end{aligned} \quad (5)$$

The positionally-ordered sequence of the inverse code (inverse transform) based on the properties of the *XOR* operation is calculated from formula:

$$\begin{aligned} &X = \Delta_{\oplus} \oplus Y = \\ &= (\Delta_{n-1} \oplus y_{n-1}) \dots (\Delta_{n-i} \oplus y_{n-i}) \dots (\Delta_1 \oplus y_1) \dots (\Delta_0 \oplus y_0). \end{aligned} \quad (6)$$

It follows from formulae (3), (5) and (6) that the representation of the set of direct and inverse codes employs the same set of codes of corrective constants, that is, the same volume of tables.

Investigating LMM for direct and inverse transforms has showed that their characteristic features are:

– the absence of lengthy arithmetic operations, which improves performance speed;

– the lack of additivity eliminates the expectation to represent operands only in full-bit form for further conversion, which makes it possible to process operands from high-order bits to low-order bits and vice versa. That speeds up the process of information processing;

– the scalability of bit size, which improves the accuracy of information representation;

– the possibility of parallel information processing, which reduces the time of delay to derive a result;

– the possibility to represent direct and inverse coded operations at the same volume of tables of corrective constants, which reduces the overall memory volume compared with classical tabular methods (CTM).

Desire to compress the volume of tables that store information to generate the results of representation of the output binary-coded operands has led to the implementation of appropriate principles aimed to condense information.

6. Principles of forming the output binary-coded operands based on the input code correction

It is known that the method is characterized by the combination of techniques for applying the principles and means of hardware implementation [22–25].

The principle of formal logic underlies the design of the bit tuple-tabular logic-reverse (BTTLR) method for the hardware implementation of high-speed conversion of multi-bit binary-codes operands by the components of special-purpose CIS.

At the stage of preliminary preparation, we perform the procedure to level the bit size of the operands. We insert «0» after comma (after a lower bit), or before a comma, we insert «0» before a higher bit.

Since the emergence of «1» and «0» in the binary-codes operands X and Y at a bit size $n \geq 8$ are equally likely, there is a possibility to use the array of input information X with further adjustment.

The principle of determining the difference between the positionally-ordered binary-coded operands X and Y is built based on the logical *XOR* operation.

The principle of forming the output set of respective multi-bit unipolar positionally-ordered binary-coded operands implies the adjustment of the input set of independent variables using low-bit constants.

Search for compacting the set of corrective constants has led to the application of the *principle of decomposition of the structured multi-bit input and output codes* based on the low-bit tuples followed by determining the difference between them.

Examples of tuple decomposition of the sets of input X binary-coded operands with corresponding numbers and the positionally-ordered subsets of low-bit tuples r_x in the general form are given in Table 1.

Examples of tuple decomposition of the sets of the corresponding input X binary-coded operands with corresponding numbers and the positionally-ordered subsets of low-bit tuples r_x in the general form are given in Table 2.

Table 3 gives, in the general form, the relational model of binary-coded operands of the respective number and the subsets of tuples of corrective constantлs Δ calculated from formula (2).

Table 1

Relational model of tuples and domains of the converted code sequence X

No. of set of operands X	Subset of tuples of higher bits X	Subset of tuples of the i-th bits X	Subset of tuples of lower bits X
X_1	$x_{n-1,1}\, x_{n-2,1}\, x_{n-3,1}\, x_{n-4,1}$	... $x_{i,1}$ ...	$x_{3,1}\, x_{2,1}\, x_{1,1}\, x_{0,1}$
X_2	$x_{n-1,2}\, x_{n-2,2}\, x_{n-3,2}\, x_{n-4,2}$	... $x_{i,2}$ ...	$x_{3,2}\, x_{2,2}\, x_{1,2}\, x_{0,2}$
...	...	...	...
X_k	$x_{n-1,k}\, x_{n-2,k}\, x_{n-3,k}\, x_{n-4,k}$	... $x_{i,3}$ ...	$x_{3,k}\, x_{2,k}\, x_{1,k}\, x_{0,k}$
X_r	r_{1x}	r_{ix}	r_{mx}

Table 2

Relational model of tuples and domains of the code sequence Y that has been converted

No. of set of operands Y	Subset of tuples of higher bits Y	Subset of tuples of the i-th bits Y	Subset of tuples of lower bits Y
Y_1	$y_{n-1,1}\, y_{n-2,1}\, y_{n-3,1}\, y_{n-4,1}$	... $y_{i,1}$ ...	$y_{3,1}\, y_{2,1}\, y_{1,1}\, y_{0,1}$
Y_2	$y_{n-1,2}\, y_{n-2,2}\, y_{n-3,2}\, y_{n-4,2}$	... $y_{i,2}$ ...	$y_{3,2}\, y_{2,2}\, y_{1,2}\, y_{0,2}$
...	...	...	...
Y_k	$y_{n-1,k}\, y_{n-2,k}\, y_{n-3,k}\, y_{n-4,k}$	... $y_{i,3}$ ...	$y_{3,k}\, y_{2,k}\, y_{1,k}\, y_{0,k}$
Y_r	r_{1y}	r_{iy}	r_{my}

Table 3

Relational model of code sequences of the tuples of corrective constants Δ and domains

No. of set of constants Δ	Subset of tuples of higher bits Δ	Subset of tuples of the i-th bits Δ	Subset of tuples of lower bits Δ
Δ_1	$\Delta_{n-1,1}\, \Delta_{n-2,1}\, \Delta_{n-3,1}\, \Delta_{n-4,1}$	... $\Delta_{i,1}$ ...	$\Delta_{3,1}\, \Delta_{2,1}\, \Delta_{1,1}\, \Delta_{0,1}$
Δ_2	$\Delta_{n-1,2}\, \Delta_{n-2,2}\, \Delta_{n-3,2}\, \Delta_{n-4,2}$	... $\Delta_{i,2}$ ...	$\Delta_{3,2}\, \Delta_{2,2}\, \Delta_{1,2}\, \Delta_{0,2}$
...	...	...	...
Δ_k	$\Delta_{n-1,k}\, \Delta_{n-2,k}\, \Delta_{n-3,k}\, \Delta_{n-4,k}$	... $\Delta_{i,3}$ ...	$\Delta_{3,k}\, \Delta_{2,k}\, \Delta_{1,k}\, \Delta_{0,k}$
Δ_r	$r_{1\Delta}$	$r_{i\Delta}$	r_{my}

The principle of forming the output code combination Y is based on the application of the logical operation of concatenation.

The result of concatenation is the following form of the multi-bit positionally-ordered binary-coded combination:

$$Y = r_{1y} \ldots r_{iy} \ldots r_{my}.$$

The totality of employment of the specified principles has allowed us to form directly (bypassing the cost of the procedure for information adjustment of different number systems) a set of multi-bit polyfunctional binary-coded operands.

Volume of the set of corrective constants $\Delta_{k,\omega}$ for each tuple at the same length of bit size r is calculated from formula:

$$V_r = r \cdot \left(2^r - 1\right) \text{ bits.} \qquad (7)$$

At the parallel conversion of n-bit codes, the volume of tables of corrective constants is derived from formula:

$$V_n = \left[\left(n^2/r\right) \cdot \left(2^r - 1\right)\right] \text{ bits.}$$

When compared to the classical tabular method, the volume of stored tables reduces owing to that $r \ll n$ by approximately:

$$V_{CTM}/V_{BTTLR} = n \cdot \left(2^n - 1\right) \Big/ \left(n^2/r\right) \cdot \left(2^r - 1\right) \approx$$
$$\approx \left(r/n\right) \cdot \left(2^{n-r}\right) \text{ times.}$$

Verification of the information compression principles is confirmed by the specific examples [22, 24–27] shown in Tables 4, 5, and the histogram in Fig. 2.

The relational model of decomposition of values for the positionally-ordered non-weighted noise-resistant Gray code and the corresponding values for the measured binary code of functions Y_s=sin(x), Y_{tg}=tg(x), Y_{th}=th(x) for n=8 are given in Table 4.

Table 4

Relational model of values for the Gray code and output binary codes of functions

No. of entry	Decomposition values for the input noise-resistant (INR) code		Decomposition values for respective functions based on tuples in the binary number system					
			Y_s=sin(x)		Y_{tg}=tg(x)		Y_{th}=th(x)	
	INR_1	INR_2	Y_{1s}	Y_{2s}	Y_{1tg}	Y_{2tg}	Y_{1th}	Y_{2th}
1	0001	1000	.0011	0010	.0010	0010	.0011	0010
2	0001	1001	.0011	0100	.0011	0100	.0011	0110
3	0001	1011	.0011	0110	.0011	0110	.0011	1000
4	0001	1010	.0011	1010	.0011	1010	.0011	1010
5	0001	1110	.0011	1110	.0011	1100	.0011	1110
6	0001	1111	.0011	1100	.0011	1110	.0100	0000
7	0001	1101	.0100	0000	.0100	0000	.0100	0100
8	0001	1100	.0100	0100	.0100	0010	.0100	0110
9	0001	0100	.0100	0110	.0100	0100	.0100	0100
10	0001	0101	.0100	1000	.0100	0000	.0100	1100
11	0001	0111	.0100	1010	.0100	1100	.0100	1100

Note: INR_1; INR_2 are the values for the higher and lowe tuples, respectively, of the noise-resistant Gray code

The relational model of decomposition of values for the non-weighted positionally-ordered noise-resistant Gray code and the corresponding tuples of values for the corrective constants Δ are given in Table 5.

An analysis of values in the tuples of corrective constants for functions sin(x), tg(x), th(x) reveals the multitude of the same values, indicating information redundancy.

In order to visualize results of the tuple decomposition of structured corrective constants derived from formula (2), we constructed a histogram shown in Fig. 2.

Table 5

Relational model of values for the Gray code and values for the corrective constants

No. of entry	Decomposition values for the input noise-resistant (INR) code		Decomposition values for the corrective constants based on tuples of the binary-coded combinations for respective functions					
			sin(x)		tg(x)		th(x)	
	INR_1	INR_2	Δ_{1s}	Δ_{2s}	Δ_{1tg}	Δ_{2tg}	Δ_{1th}	Δ_{2th}
1	0001	1000	0010	1010	0011	1010	0010	1010
2	0001	1001	0010	1101	0010	1101	0010	1111
3	0001	1011	0010	1101	0010	1101	0010	0011
4	0001	1010	0010	0000	0010	0000	0010	0000
5	0001	1110	0010	0000	0010	0010	0010	0000
6	0001	1111	0010	0011	0010	0001	0101	1111
7	0001	1101	0101	1101	0101	1101	0101	1001
8	0001	1100	0101	1000	0101	1110	0101	1010
9	0001	0100	0101	0010	0101	0000	0101	0000
10	0001	0101	0101	1101	0101	0101	0101	1001
11	0001	0111	0101	1101	0101	1011	0101	1011

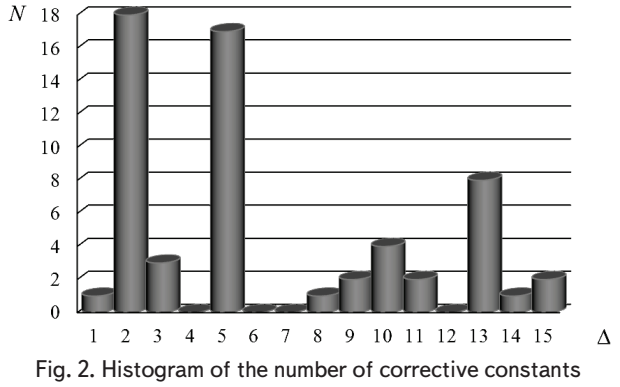

Fig. 2. Histogram of the number of corrective constants for the direct and reverse conversion of a noise-resistant code. Note: *N* is the number of identical values of corrective constants

Fig. 2 shows the redundancy (multitude of identical values) of constants (some are repeated 17, 18 times), which makes it possible to reduce the total volume of tables. In addition, owing to the *XOR* operation, the number of corrective constants for the direct and reverse conversion of codes according to formulae (5) and (6) is maintained.

During hardware implementation, the memory's numeric block (NB) records one constant (from the multitude of the same) and one volume for corrective constants for the direct and reverse conversion, which significantly reduces the total amount of memory's NB.

It should be noted that with a substantial reduction in the volume of memory's NB the information content is preserved.

The proposed method has been developed and protected by patents for invention: «Multifunctional tabular-logic co-processor»; «Co-processor for computing values for the «direct» and «inverse» functions; «Converter of binary code into unipolar inverse codes, and vice versa» [22, 24, 25].

They differ from those already known, given the multifunctionality, by low hardware cost for the implementation of tables and by the improved reliability, by the increased time between failures, by high-speed performance owing to the use of the logical operations «AND», «OR», *XOR* only.

7. Multifunctional model of the noise-resistant code conversion into a set of operands for function values

One means of hardware implementation [22, 24–27], based on the BTTLR method, is a multifunctional model (Fig. 3) for converting the noise-resistant non-weighted Gray code into the respective binary function codes Y_s=sin(x), Y_{tg}=tg(x), Y_{th}=th(x) and a binary code.

The algorithm for generating the output values for the respective operands based on model (Fig. 3) after traditional preparatory operations implies the following:

1. Record the Gray code into the register for code inputs.
2. Recognize the respective Gray code combination by using a matching address scheme.
3. Read the tuples of corrective constants from memory's NB, which were recorded previously.
4. Corrective constants appear at the computational register inputs by the corrective feedback.
5. Triggers of the register change their state, under the action of unities from the corrective constant, to the opposite state.
6. The generated corresponding function code appears at the register outputs and at the inputs of the gate block.
7. The function code is received at the gate block outlet under the influence of the generated pulse.
8. The corresponding function's value appears by the reverse information link at the inputs of the converter inside the crystal, which are connected to the external bus «input/output».

It should be noted that the conversion algorithm has the same structure of operations when representing the code values for functions Y_s=sin(x), Y_{tg}=tg(x), Y_{th}=th(x) and a binary code.

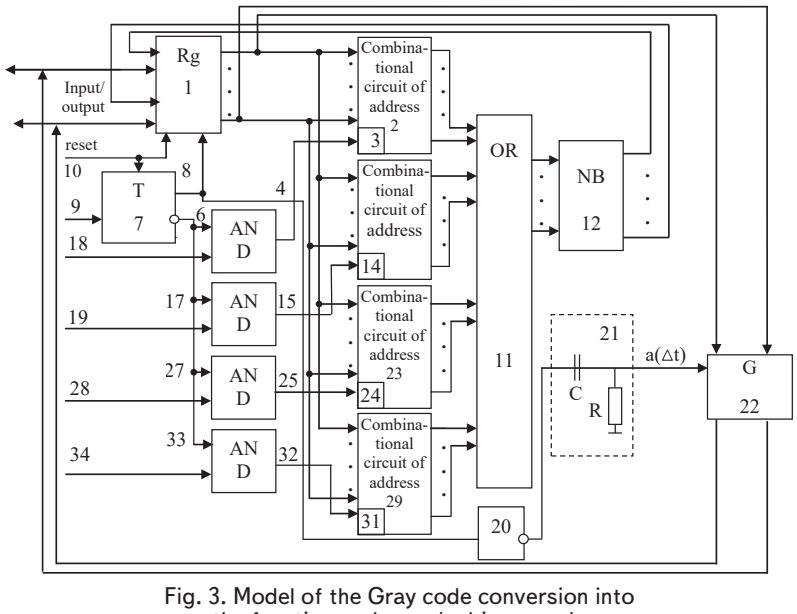

Fig. 3. Model of the Gray code conversion into the function codes and a binary code

In Fig. 3, symbols and digits Rg 1, «Combinational circuit of address» 2, 13, 23, 29, «NB» 12, «T» 7, «G» 22 denote the input/output register, combinational circuits of address, memory's NB, a trigger, a block of gates; digits 9, 10, 18, 19, 28, 34 are the external control buses: «trigger start», «reset», «run to represent functions $Y_s=\sin(x)$, $Y_{tg}=tg(x)$, $Y_{th}=th(x)$ and a binary code, respectively.

A special feature of the model (Fig. 3) is the uniformity of topologies for 4 elements «AND» and 4 combinational circuits of address. In addition, 4 conversion procedures utilize a single memory's NB that stores constants, and a single register that performs the functions of receiving an input code and adjusting it to the output code. It facilitates the work of the builder, reduces the number of errors when designing topologies. A two-fold decrease in the number of registers and the use of a single memory's NB reduce hardware costs, lower power consumption, and improve reliability.

9. Discussion of results of examining a logic-reverse operation when restoring the converted code

We have experimentally confirmed verification of the model for converting the Gray code with a positionally-ordered notation into a binary code, and vice versa, conversely, at the designed bench shown in Fig. 4.

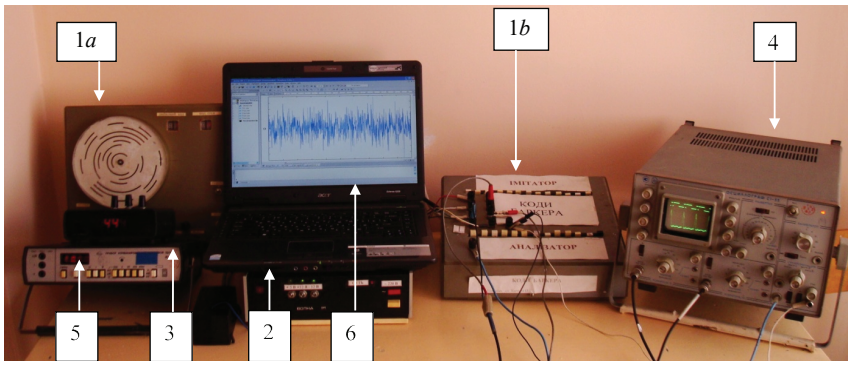

Fig. 4. Multifunctional educational-research bench

In Fig. 4:

Examined blocks:

1*a* – converters of the Gray code into a binary code and vice versa (these include: a coded disk, engine, control and visualization system, power supply);

1*b* – Barker code (it includes: a code generator, analytical unit that generates correlation functions).

2 – power supply unit (U_1=5 V; U_2=+12 V; U_3=−12 V).

3 – generator of noise signals.

4 – oscilloscope S1-55.

5 – combined digital device Shch4300.

6 – Acer Extensa 5220 laptop (connected to the generator of noise signals).

Experimental tests have confirmed that values for a binary code (direct conversion) and for the Gray code (reverse conversion) are generated using the same volume of memory's NB.

The experiment has confirmed a 100-% recovery verification of the input code applying the developed physical model, which implements the method proposed.

Thus, this study into the proposed method and model has confirmed the convergence between theoretical and practical results.

The drawback is the application of the proposed method only for the special-purpose control systems components.

In the future, it is of interest to investigate the effect of length (uniform and non-uniform) of tuples' bits on the binary-coded operations.

The in-depth research into these issues should be based on the predefined technical characteristics of the designed CIS.

10. Conclusions

1. We have analyzed and formed a class of modern positionally-ordered binary codes that are most effectively implemented based on the proposed method for the components of special-purpose CIS. That made it possible to expand the scientific and technical base for designing computational information converters.

2. A logic-mathematical model has been proposed, whose special feature is: the presence of simple correlations linking values for different types of positionally-ordered binary-coded operands regardless of number systems; the ease of hardware implementation; the absence of lengthy arithmetic operations. In combination, this accelerates the design process of computational information converters.

3. We have implemented the principles of generation of output binary-coded operands based on the adjustment of input codes, which formed the base for the development of a bitwise tuple-tabular logic-reverse conversion methods, specifically:

– the principle of formal logic that ensures the formalization of the conversion process;

– the principle of determining the difference between the input and output multi-bit codes using the properties of the logical *XOR* operation. That provides for a decrease in the volume of tables of corrective constants by not less than 2 times, and improves performance speed;

– the principle of forming an output set of respective multi-bit unipolar positionally-ordered binary-coded operands using the adjustment of the input set of operands. That has made it possible to reduce the number of registers by 2 times, to decrease power consumption, and to improve reliability;

– the principle of decomposition of the structured multi-bit input and output codes for low-bit tuples and with the subsequent determining the difference between them, which has allowed us to identify a large number of identical values for tuples;

– the principle of generation of the output code combination based on applying the logical operation of concatenation, which has made it possible to speed up the process of forming the input value for the respective code.

4. We have developed a multifunctional model for converting the Gray code values with a positional notation into the codes of values for transcendental functions and a natural binary code. A special feature of the model is:

– the conversion algorithm has a uniform structure of operations, ensuring high regularity of implementation when representing many functions;

– the uniformity of topologies for 4 elements «AND», 4 combinational circuits of address facilitate the work of a builder, reducing the number of errors in design;

– it reduces the number of registers by 2 times and utilizes one numeric memory block for 4 conversion procedures, resulting in lower power consumption and increased reliability.

5. A special feature of the developed bitwise tuple-tabular logic-reverse method for converting various kinds of binary-coded multi-bit operands with a positionally-ordered notation is:

– the formalization of the process of their conversion, owing to the employment of formal logic. This expands the scope of their application and simplifies the process of converting binary-coded multi-bit operands;

– high performance owing to the high degree of parallelism in the information processing;

– the possibility of a tuple build-up that provides increased accuracy;

– the restoration of the input code without an error during logic-reverse procedure using the same volume of tables of corrective constants.

6. Based on the developed structural, functional, and fundamental electrical circuits, we have constructed a physical model to investigate the code conversion procedure, which is built into a multifunctional educational-research bench. Verification of the theoretical conclusions has been confirmed by the results of the experimental study.

References

1. Nguyen G. D. Fast CRCs // IEEE Transactions on Computers. 2009. Vol. 58, Issue 10. P. 1321–1331. doi: https://doi.org/10.1109/tc.2009.83
2. Ahmad A., Hayat L. Selection of Polynomials for Cyclic Redundancy Check for the use of High Speed Embedded – An Algorithmic Procedure // WSEAS Transactions on Computers. 2011. Vol. 10, Issue 1. P. 16–20.
3. PASCO. URL: http://pasco.com/
4. PHYWE – Cobra4 Wireless. URL: https://www.phywe.com/en/cobra4-wireless-link.html
5. L-CARD. URL: http://www.lcard.ru
6. Semerenko V. P. Theory and practice of crc codes: new results based on automaton models // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 4, Issue 9 (76). P. 38–48. doi: https://doi.org/10.15587/1729-4061.2015.47860
7. Galuza A. A., Kolenov I. V., Belyaeva A. I. Software and hardware platform for developing laboratory experiment automation systems // Eastern-European Journal of Enterprise Technologies. 2013. Vol. 5, Issue 9 (65). P. 11–16. URL: http://journals.uran.ua/eejet/article/view/18446/16193
8. Sarwate D. V. Computation of cyclic redundancy checks via table look-up // Communications of the ACM. 1988. Vol. 31, Issue 8. P. 1008–1013. doi: https://doi.org/10.1145/63030.63037
9. Semerenko V. P. Estimation of the correcting capability of cyclic codes based on their automation models // Eastern-European Journal of Enterprise Technologies. 2015. Vol. 2, Issue 9 (74). P. 16–24. doi: https://doi.org/10.15587/1729-4061.2015.39947
10. Krishna K. V. An Optimization Technique for CRC Generation // International Journal of Computer Trends and Technology (IJCTT). 2013. Vol. 4, Issue 9. P. 3260–3265.
11. Baicheva T. Determination of the Best CRC Codes with up to 10-Bit Redundancy // IEEE Transactions on Communications. 2008. Vol. 56, Issue 8. P. 1214–1220. doi: https://doi.org/10.1109/tcomm.2008.070033
12. Osinin I. P., Knyaz'kov V. S. Organizaciya parallel'no-konveyernogo SBIS-processora dlya pryamogo modulyarnogo preobrazovaniya chisel na osnove arifmetiki razryadnyh srezov // Izvestiya vysshih uchebnyh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki. 2014. Issue 3 (31). P. 5–13.
13. Korneychuk V. I., Tarasenko V. P. Osnovy komp'yuternoy arifmetiki. Kyiv: Korniychuk, 2003. 176 p.
14. Sergeev A. M. Ob osobennostyah predstavleniya chisel pri znakorazryadnom kodirovanii i vychislitel'niy eksperiment s nimi // Informacionno-upravyayushchie sistemy. 2006. Issue 3. P. 56–58.
15. Riznyk O., Balych B., Yurchak I. A synthesis of barker sequences is by means of numerical bundles // 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM). 2017. doi: https://doi.org/10.1109/cadsm.2017.7916090
16. Semerenko V. The Theory of Parallel CRC Codes Based on Automaton Models // Eastern-European Journal of Enterprise Technologies. 2016. Vol. 6, Issue 9 (84). P. 45–55. doi: https://doi.org/10.15587/1729-4061.2016.85603
17. Semerenko V. P. Paralelni tsyklichni kody // Visnyk VPI. 2014. Issue 6. P. 65–72.
18. Grymel M., Furber S. B. A Novel Programmable Parallel CRC Circuit // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2011. Vol. 19, Issue 10. P. 1898–1902. doi: https://doi.org/10.1109/tvlsi.2010.2058872
19. Gross T. R., Joppi N. P., Hennessy J. L. A Retrospective on "MIPS": A Microprocessor Architecture // IEEE Computer Society. 2016. Vol. 36, Issue 4. P. 73–76.
20. Arhitektura CPU. URL: https://old.computerra.ru/2005/609/233266/
21. Baykov V. D., Smolov V. B. Specializirovannye processory: iteracionnye algoritmy i struktury. Moscow: Radio i svyaz', 1985. 288 p.
22. Peretvoriuvach dviykovoho kodu v odnopoliarni oborotni kody i navpaky: Pat. No. 107544 UA / Zubko I. A., Lukashenko V. A., Lukashenko A. H., Lukashenko V. M., Lukashenko D. A. No. a201401392; declareted: 12.02.2014; published: 12.01.2015, Bul. No. 1.
23. Vysokonadiynyi bahatofunktsionalnyi obchysliuvach dlia spetsializovanykh lazernykh tekhnolohichnykh kompleksiv / Lukashenko A. H., Lukashenko D. A., Lukashenko V. A., Lukashenko V. M. // Visnyk ChDTU. 2011. Issue 1. P. 67–70.

24. Bahatofunktsionalnyi tablychno-lohichnyi spivprotsesor: Pat. No. 111459 UA / Lukashenko V. A., Lukashenko D. A., Zubko I. A., Lukashenko A. H., Lukashenko V. M. No. a201509351; declareted: 28.09.2015; published: 25.04.2016, Bul. No. 8.
25. Spivprotsesor dlia obchyslennia znachen «priamykh» ta «obernenykh» funktsiy: Pat. No. 111808 UA / Lukashenko A. H., Lukashenko V. M., Lukashenko D. A., Lukashenko V. A., Zubko I. A., Rudakov K. S. No. a201510690; declareted: 02.11.2015; published: 10.06.2016, Bul. No. 11.
26. Tablychno-lohichnyi peretvoriuvach kodiv: Pat. No. 89784 UA / Chychuzhko M. V., Lukashenko V. A., Lukashenko D. A., Zubko I. A., Lukashenko V. M., Lukashenko A. H. No. u201315042; declareted: 23.12.2013; published: 25.04.2014, Bul. No. 8.
27. Lukashenko V. A. Udoskonalenyi tablychno-alhorytmichnyi metod i modeli aparaturnoi realizatsiyi pretsyziynykh obchysliuvachiv spetsialnoho pryznachennia: avtoref. dys. ... kand. tekhn. nauk. Cherkasy: ChDTU, 2016. 20 p.

Визначення показника для оцінки ефективності системних операцій є найважливішим етапом для оптимізації технологічних процесів будь-якого підприємства. Цей крок зумовлює сталий режим функціонування всіх його системних процесів.

Те, що всі без винятку технологічні процеси повинні бути оптимізовані з використанням узгодженого критерію оптимізації, є аксіомою. При цьому така можливість забезпечується тільки в одному випадку – якщо у всіх функціональних системах в якості критерію оптимізації використовується формула ефективності. Саме такий підхід забезпечує максимізацію фінансових можливостей власника підприємства.

Проблема полягає в тому, щоб серед безлічі оціночних показників, однакових за формальними ознаками, ідентифікувати таку структуру, яка відповідає структурі оригінальної формули ефективності.

Для практичного вирішення цього завдання в даний час визначені класи еталонних моделей операцій, кожен з яких має свою функціональну спрямованість. При цьому найбільш розробленими є класи еталонних моделей простих операцій.

По відношенню до класів еталонних моделей операцій з розподіленими параметрами, задача вирішена для операційних процесів з однаковою тривалістю в часі, а також для процесів, з різною тривалістю ресурсовіддачі по виходу.

У пропонованій роботі визначається обмежений клас моделей операцій з розподіленими параметрами різної тривалості по входу. Створення такого класу операцій є досить складним завданням, оскільки необхідно враховувати фактор часу і забезпечувати можливість зіставлення операційних процесів різної тривалості.

Для вирішення цього завдання, на першому етапі, були сформовані глобальні моделі простих операцій різної тривалості з визначеною рейтингової ефективністю. На наступному етапі, шляхом композиції, формувалися еталонні моделі операцій з розподіленими параметрами щодо виходу різної тривалості.

Розвиток методу верифікації, за рахунок визначення класу операцій з розподіленими параметрами по входу різної тривалості у часі, істотно підвищує надійність і достовірність результатів верифікації оціночного показника. Процедуру верифікації необхідно проводити для критерію якщо передбачається його використовувати в якості показника ефективності

Ключові слова: верифікація оціночного показника, операція з розподіленими параметрами, клас операцій, метод верифікації

UDC 007.5
DOI: 10.15587/1729-4061.2018.142212

DEVELOPMENT OF TEST OPERATIONS OF DIFFERENT DURATION IN TERMS OF INPUT FOR THE VERIFICATION OF EFFICIENCY FORMULA

I. Lutsenko
Doctor of Technical Sciences, Professor
Department of Information and Control Systems*
E-mail: delo-do@i.ua
O. Fomovskaya
PhD, Associate Professor, Head of Department
Department of Electronic Devices*
O. Serdiuk
PhD, Senior Lecturer
Department of automation,
computer science and technology**
M. Baranovskaya
PhD, Associate Professor
Department of Electromechanics**
V. Fomovskyi
Lanzhou Jiaotong University
Anning West road, 88,
Lanzhou, P. R. China, 730070
*Kremenchuk Mykhailo Ostrohradskyi
National University
Pershotravneva str., 20,
Kremenchuk, Ukraine, 39600
**Kryvyi Rih National University
Vitaliya Matusevycha str., 11,
Kryvyi Rih, Ukraine, 50027

1. Introduction

A basis of the principle of operation of any enterprise is the execution of system operations. The task of system management is to determine parameters of operations, which provide results of system functioning maximally coordinated with the purpose of its owner [1, 2]. In turn, the aim of an owner of an enterprise is to maximize its investment opportunities [3].