Розглянуто теоретичні основи CRC кодів на основі математичного апарату лінійних послідовнісних схем (ЛПС). Проведено аналіз кортежно-паралельного способу обчислення CRC, розглянуті його апаратна реалізація за допомогою багатовходових ЛПС та програмна реалізація по таблицям пошуку. Запропоновані символьно-паралельний і символьно-кортежно-паралельний способи обчислення CRC, а також недвійкові коди Хемінга і Абрамсона

Ключові слова: CRC коди, контрольна сума, лінійна послідовнісна схема, таблиці пошуку

Рассмотрены теоретические основы CRC кодов на основе математического аппарата линейных последовательностных схем (ЛПС). Проведен анализ кортежно-параллельного способа вычисления CRC, рассмотрены его аппаратная реализация с помощью многовходовых ЛПС и программная реализация по таблицам поиска. Предложены символьно-параллельный и символьно-кортежно-параллельный способы вычисления CRC, а также недвоичные коди Хемминга и Абрамсона

Ключевые слова: CRC коды, контрольная сумма, линейная последовательностная схема, таблицы поиска

UDC 681.32
DOI: 10.15587/1729-4061.2016.85603

THE THEORY OF PARALLEL CRC CODES BASED ON AUTOMATON MODELS

V. Semerenko
PhD, Associate Professor
Department of Computer Technique
Vinnytsia National Technical University
Khmelnytske highway, 95,
Vinnytsia, Ukraine, 21021
E-mail: VPSemerenko@ukr.net

1. Introduction

CRC-check belongs to the most commonly used methods of check in different information systems. From the time of its appearance in 1961 [1], this method has gained wide acceptance in the systems for data transmission, storage and compression. CRC is the method for detecting errors, which has very simple software and hardware implementation. But this advantage manifests itself only at the serial entering of input data, which was characteristic in the early years of development of information systems.

However, in contemporary high-speed data transmission, especially in the multichannel communication systems, large delays occur in the realization of procedures of searching for errors with the aid of traditional CRC. A basic direction of modern studies for solving the indicated problem is the parallelization of computations. At the engineering level, it was possible to partially accelerate the CRC computations. But the attempts to apply parallel processing to the traditional CRC cannot provide for the effective solution of the problem as a whole. On the other hand, new methods of CRC computation proposed over recent years require a strict mathematical substantiation of their ability to search for the assigned type of errors. That is why it is absolutely relevant to simultaneously solve the problems on the provision of high performance speed of CRC codes and on the detection of the maximum number of various data distortions by them.

2. Literature review and problem statement

CRC codes are a variety of the binary cyclic codes [2]. If for the traditional, that is, serial, CRC, it was sufficient to have a general theoretical base of cyclic codes, then for the high-speed parallel CRC there appeared a pressing need in adequate theory. Several directions of studies have been outlined over recent years.

In the hardware implementation of parallel CRC, high productivity of computations is achieved most frequently due to the use of the known parallel architectures (for example, pipelinability [3]). For accelerating the computations, data stream M is frequently split into the h-bit tuples (blocks), whose all bits are processed simultaneously, and the entire structure operates in the pipelining [4]. An increase in the performance speed may be also achieved through the optimization of structure of the field programmable gate array (FPGA), on which the coders and decoders of these encoders are realized [5].

A useful property of parallel CRC is the possibility of rapid replacement of the generator polynomial of code, that is, reprogrammability either with the aid of FPGA [5], or with the help of additional gates [6].

A basic method for accelerating the computations in the CRC software implementation is decreasing the size of the lookup tables [7, 8].

The use of special forms of polynomials in the hardware [6, 9] and software [8] implementation is one of the interesting directions for the acceleration of computations.

However, researchers frequently ignore the need for a strict mathematical substantiation of the approaches proposed by them. For example, authors in [6, 9] propose special OZO-polynomials. They are conveniently calculated, but they cannot be applied in practice because of the short period of code generation.

It is equally important to check swiftly and to have a guarantee for the detection of the assigned type of errors. A lack of comprehensive approach to provide, with the aid of the CRC method, for both high speed operation and the detection of maximum number of errors in data streams, presents a problem in this area of information technologies, not resolved as yet.

A key factor for the solution of this problem is the optimum choice of mathematical tools.

In recent years, new methods of describing CRC – z-transform [10] and automaton theory [4–6] have been explored. Encoders and decoders of the CRC code are represented in the form finite automaton on the basis of the so-called F matrix. The matrix representation of automaton in the Galois fields is the basis of linear systems, proposed in [11] for the first time for the CRC codes.

Despite certain positive aspects (replacement of slow operations of dividing the polynomials into more rapid operations of computing the internal states of linear systems), this theory does not make it possible to give a strict and vivid interpretation of the both specified problems (high speed and check) in a complex. A theory of linear finite-state machine (LFSM), which is further examined in paper, demonstrates better possibilities in this regard.

3. The aim and tasks of the study

The aim of present work is to devise highly productive methods for the CRC computation with high detecting capability based on the mathematical tools of LFSM, and to solve relevant tasks of practical application of CRC.

To achieve the set aim, the following tasks are to be solved:

– to demonstrate the essence of the CRC codes from the positions of LFSM theory in the binary and non-binary Galois fields;

– to propose different methods for the parallel computation of the CRC codes with the aid of bit and symbolic LFSM;

– to demonstrate the peculiarities of highly productive CRC hardware and software implementation;

– to propose parallel CRC codes and to substantiate their detectivity capabilities based on LFSM theory.

4. CRC computation methods

Let us first note that there are two known interpretations of the abbreviation CRC: *Cyclic Redundancy Code* and *Cyclic Redundancy Check* [2].

Assume that data stream M of w bits length and pre-computed check parameter Σ_s, representing the compressed representation of M, is transmitted by certain data transmission channel.

By the first interpretation of CRC, parameter Σ_s is considered as the check word of (n-k) length of the shortened cyclic code (k≥w) and the evidence of error-free transfer is the equality to zero of the calculated error syndrome Σ_r of this code. By the second interpretation of CRC, for the obtained data stream M, on the side of the receiver, checksum Σ_r is calculated and it is compared to the checksum Σ_s, calculated on the side of the transmitter.

In both cases, the method for computing check sum and check word may be identical, but the duration of computation can differ: k or n cycles in the first case and k cycles in the second case. And one more important difference in the first interpretation of CRC: capability not only to detect errors, but to correct them as well.

Historically, the first method of CRC computation was polynomial: sequence M was considered as the row of binary coefficients of a certain polynomial f(x), which was divided into the assigned generator polynomial g(x), and the remainder of this division was the required CRC [1]. Naturally, the sequence of performing w operations of division requires too much time.

Usually they distinguish the methods of CRC computation depending on the means of realization of the computational algorithm – software or hardware (with the aid of the linear feedback shift register (LFSR)). In this case, the essence of the algorithm for computations has remained practically constant over many decades: the bit-wise division of polynomial.

Let us examine the methods for CRC computation depending on the following two factors:

– the method of arrival of data stream M;

– the ratio of arrival rate of data stream M and productivity of the means of CRC computation.

Therefore, it is possible to distinguish between four methods of the CRC computation:

1) serial;
2) tuple-parallel;
3) symbolic-parallel;
4) symbolic-tuple-parallel.

The easiest method is the bit-wise arrival of input data and the equality between their arrival speed and performance efficiency of the means of data processing. In other words, duration t_{sh} of bit-wise shift in LFSR of encoder and decoder (duration of cycle step in the software cycles) is equal to period t_b of arrival of the next bit of data stream M:

$$t_{sh} \approx t_b.$$

In this case, general duration of time T_1 of the CRC serial computation will be determined by length w of data stream M:

$$T_1 = w\,t_b.$$

It is this foundation that the traditional approach to the CRC computation (Fig. 1) is based on.

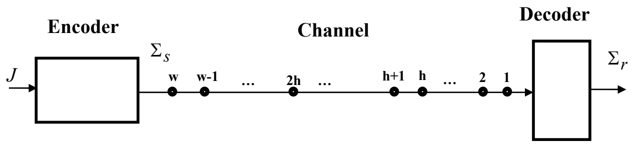

Fig. 1. Serial method of the CRC computation

In practice, however, the speed of receipt of input data can by many times exceed the operating speed of the software-hardware tools of code conversion, which creates a large reserve for the acceleration of CRC computation. Provided the condition is satisfied

$$h = \frac{t_{sh}}{t_b},$$

then it is possible to process simultaneously the h-bit tuple of input information before arrival of the next portion of data (Fig. 2). Converter in Fig. 2 is the h-bit shift register, which converts serial binary code of h bits at the input into the parallel code (h-bit tuple) at the output. General duration of time T_2 of the tuple-parallel CRC computation will amount to:

$$T_2 = \frac{w}{h} t_b.$$

Parallel CRC on the basis by fragment processing of data has already been proposed by some authors [4, 11].

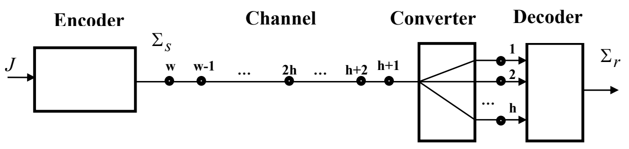

Fig. 2. Tuple-parallel method of the CRC computation

In the contemporary multichannel data-transmission systems, the parallel transmission of data is realized: bits of one byte or word (2, 4, 8 bytes) arrive simultaneously. Each byte or word may be interpreted as one h-digit symbol (h=8,16,32,64).

In the symbolic-parallel CRC computation, m symbols enter alternately (Fig. 3), and the symbolic-tuple-parallel method is a simultaneous processing of z symbolic tuples (Fig. 4). The converter in Fig. 4 is intended for converting the serial symbolic code into the parallel symbolic code.

If period t_b of symbols arrival is equal to the duration of data processing in the encoder and decoder, then total duration of time T_3 of the CRC symbolic-parallel computation will amount to.

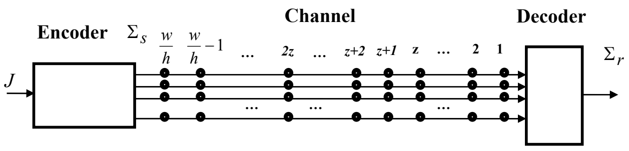

Fig. 3. Symbolic-parallel method of the CRC computation

$$T_3 = m\,t_b = \frac{w}{h} t_b.$$

If a symbol's bit capacity is equal to length h of tuple at the tuple-parallel method of CRC computation, then the duration of time T_3 will be nearly equal to the duration of time T_2. (Let us note that here we are abstracted from the details of technical implementation and we examine only mathematical time). The symbolic-tuple-parallel method of CRC computation makes it possible to reach minimum time T_4 of the computations:

$$T_4 = \frac{w}{zh} t_b.$$

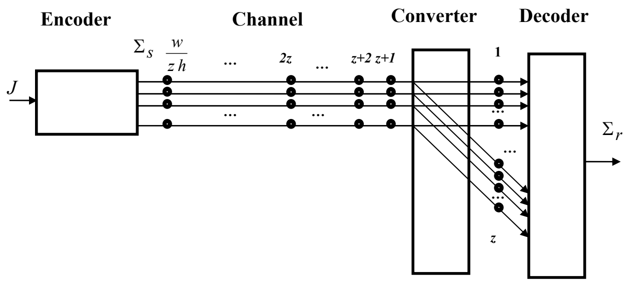

Fig. 4. Symbolic-tuple-parallel method of CRC computation

All examined methods of CRC computation can be realized on the uniform mathematical tools – the LFSM theory. In all variants of the computations, which are represented in Fig. 1–4, different variants of hardware implementation of LFSM as the encoders and decoders.

5. Theoretical basis of the CRC-codes

In order to comprehend the essence of CRC-codes, it is necessary to return to basics of the cyclic codes. Traditional methods for the representation of cyclic codes (matrix, polynomial, algebraic) have played an important role in the formation of this class of the error correcting codes. Practice, however, presents new problems and new approaches are required for their solution.

A number of authors proposed to use for the CRC-codes the theory of linear systems [11] while others focused attention on the theory of finite automaton [4]. It is expedient to combine these directions. Then we shall obtain a new type of the finite automaton, which one may call the linear automaton or, to be more precise, by LFSM.

According to [12, 13] LFSM with l inputs, m outputs and r memory elements in the discrete time clock t over GF(2) (binary LFSM) is defined by the transition (state) function

$$S(t+1) = A \times S(t) + B \times U(t),\ \ GF(2) \tag{1}$$

and output function

$$Y(t) = C \times S(t) + D \times U(t),\ GF(2), \tag{2}$$

where

$$A = |a_{ij}|_{r\times r},\ B = |b_{ij}|_{r\times l},\ C = |c_{ij}|_{m\times r},\ D = |d_{ij}|_{m\times l}$$

are the characteristic LFSM matrices; $S(t) = |s_i|_r$ is the word of state; $U(t) = |u_i|_l$ is the input word; $Y(t) = |y_i|_m$ is the output word.

The dimensionalities of LFSM matrices and parameters of cyclic (n, k)-code Ω are connected through coefficient r, which for the code is equal to the number of the check bits of check word at systematic coding.

The simplest hardware implementation of r-bit LSC LFSM is LFSR, which consists of r flip-flops and certain quantity of gates XOR.

Different types of LFSM and corresponding characteristic LFSM matrices are possible [13]. Binary LFSM with one input and one output can be recursive (for systematic encoding) and non-recursive (for unsystematic encoding). In the direct (inversable) LFSM, the direction of data shift in the flip-flops coincides (oppositely) with the numeration of flip-flops. Depending on the interrelation between flip-flops and gates XOR, there are LFSM of different types. The following ones are the most common of them:

– LFSM of type 1 (Galois type according to terminology [13], two-input gates XOR in them are located between the flip-flops);

– LFSM of type 2 (F type according to terminology [6, 11]);

– LFSM of type 3 (Fibonacci type, according to terminology [13], one multi-input gate XOR in them is located before one flip-flop).

To cyclic (n, k)-code Ω with the generator polynomial

$$g(x) = g_0 + g_1 x + \cdots + g_{r-1} x^{r-1} + g_r x^r,\ \ GF(q), \tag{3}$$

correspond to the recursive LFSM of type 1 with the characteristic matrices

$$A=\begin{bmatrix} 0 & 0 & 0 & \dots & g_0 \\ 1 & 0 & 0 & \dots & g_1 \\ 0 & 1 & 0 & \dots & g_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 & g_{r-1} \end{bmatrix},\quad B=\begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix},$$

$$C=\begin{bmatrix} 0 & \dots & 0 & 0 & 1 \end{bmatrix},\ D=\begin{bmatrix} 0 \end{bmatrix}, \qquad (4)$$

or recursive LFSM of type 2 with the characteristic matrices

$$A=\begin{bmatrix} g_0 & 1 & 0 & \dots & 0 \\ g_1 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ g_{r-2} & 0 & 0 & \dots & 1 \\ g_{r-1} & 0 & 0 & \dots & 0 \end{bmatrix},\quad B=\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \dots \\ g_{r-1} \end{bmatrix},$$

$$C=\begin{bmatrix} 0 & \dots & 0 & 0 & 1 \end{bmatrix},\ D=\begin{bmatrix} 0 \end{bmatrix}, \qquad (5)$$

or recursive LFSM of type 3 with the characteristic matrices

$$A=\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \\ g_0 & g_1 & g_2 & \dots & g_{r-1} \end{bmatrix},\ B=\begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 1 \end{bmatrix},$$

$$C=\begin{bmatrix} 1 & 0 & \dots & 0 & 0 \end{bmatrix},\ D=\begin{bmatrix} 0 \end{bmatrix}. \qquad (6)$$

The entries of the last column of matrix A from (4), the entries of matrix B and of the first column of matrix A from (5), and entries of the last row of matrix A from (6), are the constant coefficients of the generator polynomial (3).

The automaton representation of cyclic codes has two models: automaton-analytical and automaton-graphical [13]. Automaton-analytical model, examined above, based on the characteristic LFSM matrices, is used for developing the algorithms of encoding and decoding. Automaton-graphical model is convenient for the visual representation of the essence of the code transformations procedures.

As an automaton-graphical model, it is possible to select the state diagram of LFSM as automaton. For r-dimensional LFSM above field GF(2), this state diagram is the directed graph $G_{FA}(V_{FA}, E_{FA})$, in which 2^r vertices from the set of vertices V_{FA} correspond to 2^r internal states of automaton, while zero and unity edges from the set of edges E_{FA} show the directions of transitions between internal states. The separate vertices of graph G_{FA} with the aid of zero edges are combined in the zero cycles (ZC). We shall distinguish trivial ZC, which consist of one vertex, and full ZC, which consist of n vertices.

The states of LFSM correspond to vertices of graph G_{FA} therefore they create their ZC of the same structure.

The LFSM under consideration over the Galois field GF(2) are applied at the serial arrival of input data and processing at each time cycle of only one input bit. But if the input data arrive byte-by-byte or word-by-word (that is, the groups of input bits enter simultaneously), then for such methods of the CRC computation (symbolic-parallel and symbolic-tuple-parallel, according to our terminology), LFSM over the non-binary Galois fields are required.

Assume that in each clock cycle a group of h input bits (h=8, 16, 32, 64) is entered. Then it is necessary to determine Galois field GF(q), where $q=p^h$. In practice, they usually use the case p=2, therefore, for computing the elements of field GF(q), we shall use LFSM over field GF(2), which will be named as binary. This LFSM is determined with the aid of formulas (1) and (2). Matrix A of binary LFSM contains coefficients of the primitive polynomial

$$g_b(x)=1+\lambda_1 x+\lambda_2 x^2+\dots+\lambda_{h-2}x^{h-2}+\lambda_{h-1}x^{h-1}+x^h,$$

what is used for the construction of the field GF(2).

The zero element α^0 of field GF(2^h) will be corresponded to by the word of zero state S(0) of binary LFSM. The remaining elements of field GF(2^h) can be calculated by formula:

$$\alpha^i=A\times\alpha^{i-1},\quad i=1,2,\dots,2^h-1.$$

Now, over Galois field GF(q), it is possible to determine LFSM (let us name it symbolic), which is defined by transition (state) function

$$S(t+1)=A_q\times S(t)+B_q\times U(t),\ \ GF(q), \qquad (7)$$

and output function

$$Y(t)=C_q\times S(t)+D_q\times U(t),\ \ GF(q). \qquad (8)$$

In order to obtain characteristic matrices A_q, B_q, C_q, D_q of symbolic LFSM, it is necessary at first to define the generator polynomial of CRC-code.

Finally, upon definition of the Galois field GF(q), it is possible to perform the operations of encoding and decoding in relation to input symbols for the purpose of the CRC computation.

The advantage of LFSM theory in comparison with the theory of linear systems is its deeper and more comprehensive development, which makes it possible to effectively use it for the ground of different aspects of CRC.

6. Encoding the CRC-codes

The stage of encoding is necessary only at the interpretation of CRC as the shortened cyclic (n, k)-code. In this case, data stream M of length w can be considered as the w-bit information word J (w≤k). After word J, r-bit check word Ψ is transmitted, calculated on the side of transmitter according to the algorithm of systematic encoding (n=k+r) [14].

Without loss of generality, from now on we shall examine cyclic (n, k)-code of full length, that is, w=k.

Words J and Ψ can be considered as a single code word P, with which to perform on the side of receiver its usual decoding, that is, to calculate at n-th cycle the syndrome of error Serr. Zero syndrome Serr will testify to the error-free transfer.

As already mentioned, CRC in the form of check word Ψ makes it possible both to detect errors and to correct some of them. Additional charges are the price for this possibility: complication of the CRC computation and certain increase in the duration of its computation (not exceeding r cycles). Under what conditions is it possible to minimize these expenditures? An answer to this question is easy to obtain within the framework of theory of LFSM.

From the positions of theory of LFSM, cyclic (n, k)-code Ω is a set of all sequences of length n, which transfer LFSM from certain initial state $S_{beg}(t)$ back again to state $S_{beg}(t)$. Each such sequence is the code word P of code Ω. As the state $S_{beg}(t)$, we shall from now on examine the zero state S(0).

It is known from the theory of LFSM [12] that under the action of the assigned k-bit information word J, LFSM will pass from initial state S(0) to a certain state S(k), which it is determined from equation

$$S(k) = A^k \times S(0) + L_k \times J, \quad GF(2),$$

where

$$L_k = \left[\, A^{k-1} \times B, \quad A^{k-2} \times B, \quad \ldots \quad , \quad A \times B, \quad B \right],$$

A, B are the characteristic matrices of LFSM.

For obtaining code word P, it is necessary to determine the check word Ψ of length r, which will convert LFSM from the state S(k) back to the state S(0). Word Ψ is determined from equation

$$S(n) = A^r \times S(k) + L_r \times \Psi, \quad GF(2), \tag{9}$$

where

$$L_r = \left[\, A^{r-1} \times B, \quad A^{r-2} \times B, \quad \ldots \quad , \quad A \times B, \quad B \right].$$

Since S(n)=S(0), equality (9) may be written down as

$$L_r \times \Psi = A^r \times S(k). \tag{10}$$

In equation (10), the only unknown is the word Ψ, that is, CRC. Complexity and duration of computing Ψ depends on the structure of matrices A and L_r. By selecting different variants of characteristic matrices A and B, it is possible to obtain the correspondent matrix L_r. The only requirement to matrix L_r is that its rank must be equal to r, which ensures r-controllability of corresponding LFSM and strong connectedness of graph models.

It is proven in [13] that matrix L_r for characteristic matrices (4) takes the form:

$$L_r = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 1 & 0 \\ \ldots & \ldots & \ldots & \ldots & \cdots \\ 0 & 1 & \ldots & 0 & 0 \\ 1 & 0 & \ldots & 0 & 0 \end{bmatrix}. \tag{11}$$

As a result, systematic encoding of CRC (n, k)-code with the aid of recursive LFSM of type 1, which is assigned by characteristic matrices (4), can be performed either in k or in n cycles. But in the first case, for finding word Ψ, it will be required to solve a system of r linear equations over field GF(2).

It is also proven in [13] that the systematic encoding of CRC (n, k)-code with the aid of recursive LFSM of type 3, assigned by characteristic matrices (6), can be conducted during k or n cycles after sending information word J to its input. But, in contrast to LFSM of type 1, in both cases it is necessary to solve the system of r linear equations.

The particularity of LFSM of type 2 is the equality of matrices A^r and L_r. Then equation (10) can be written down as

$$\Psi = S(k). \tag{12}$$

Therefore, according to the state S(k), calculated in the k-th cycle in (12), it is possible to immediately obtain the components of word Ψ. Thus, with the aid of LFSM of type 2, it is possible to obtain the check word of the code most rapidly.

Maximum gain in time is not more than r cycles. In practice, however, data streams have sufficiently large length (n>>r, w>>r), therefore this gain will be insignificant.

The main problem in the CRC computation consists in the acceleration of processing of the entire w-digit data stream M independent of the interpretation of CRC. The algorithm itself of the accelerated CRC computation will be the same both on the side of transmitter and on the side of receiver.

7. Tuple-parallel method of the CRC computation

We shall from now on consider the value of h tuple to be equal to dimensionality r of LFSM because this variant is the most convenient for practical realization.

Let there be r-controlled LFSM, which is in certain state S(i). The property of r-controllability guarantees the existence of input sequence u_{ij} of length not exceeding r, which provides for the possibility of transfer of LFSM in r cycles from state S(i) into any assigned state S(j). We shall name an iteration the time interval, necessary for the transfer from state S(i) to state S(j).

In the automaton-graphical model of r-controlled LFSM (in graph G_{FA}) there is a directed path from r zero and unity edges from vertex v_i that corresponds to state S(i) to vertex v_j, corresponding to state S(j).

Therefore, it is possible to pass from certain initial vertex v_{beg} of graph G_{FA} to the assigned final vertex v_{end} by indicating only each r-th vertex of the path. In the automaton-analytical model of LFSM, which is examined below, only each its r-th state will be computed.

Let there be k-bit information word J. Let us separate word J into r-bit tuples u(i) $(u(i) \in J, i = 0 \div \left]\frac{w}{r} - 1\right[$ where $]x[$ is the rounding to the nearest whole towards the larger side). We shall assume that symbols enter communication channel from the left side. If length of the first left tuple u(0) will be less than r, then it is possible to complement it from the left to r by insignificant zeros, which will increase length k of the information word to k''.

Theorem 1. A tuple-parallel method of the CRC computation as the checksum of information word J of length k'' with the aid of r-input recursive LFSM of type 1, assigned by characteristic matrices (4), can be performed in $\frac{k''}{r}$ single-cycle iterations.

Proof. It is known from the theory of LFSM [12] that if the r-input LFSM in the i-th cycle was found in state S(i), then, after applying to its inputs r-bit tuple u(j), it will pass into state

$$S(i+r) = A^r \times S(i) + L_r \times u(i), \quad GF(2). \tag{13}$$

Since LFSM with characteristic matrices (4) has identity matrix L_r (11), then equation (13) can be written down as

$$S(i\times r)=A^{r}\times S(i)+u'(i),\ \ GF(2), \qquad (14)$$

where $u'(i)$ is the tuple u(i) with the mutual transposition between the high-order digits and low-order digits:

$$u_{j}(i)=u_{r-j+1}(i),\ \ u_{j}(i)\in u'(i),\ \ u_{r-j+1}(i)\in u(i).$$

Expression (14) differs from transition function (1) of single-input LFSM by the presence of matrix A^r instead of matrix A. Matrix A^r can be prepared in advance; which is why state S(i+r) can be computed in one automaton cycle. Therefore, final state $S(n'')$, that is, CRC, will be reached in $\frac{k''}{r}$ automaton cycles of the LFSM operation.

Example 1. For the cyclic (15,11)-code with the generator polynomial $g(x)=1+x^3+x^4$ and LFSM of type 1, to calculate CRC for information word:

$$J=\begin{bmatrix}1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0\end{bmatrix}. \qquad (15)$$

For the assigned g(x), characteristic matrices of LFSM and its initial state are equal to:

$$A=\begin{bmatrix}0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&1\end{bmatrix},\ A^{4}=\begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\1&1&1&0\end{bmatrix},\ B=\begin{bmatrix}1\\0\\0\\0\end{bmatrix},\ S(0)=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}.$$

We shall form 3 tuples for the assigned word J (with regard to the transposition of digits):

$$u(1)=\begin{bmatrix}0\\1\\1\\0\end{bmatrix};\ u(2)=\begin{bmatrix}0\\0\\1\\1\end{bmatrix};\ u(3)=\begin{bmatrix}0\\1\\0\\1\end{bmatrix}.$$

In accordance with (14), we shall alternately form states S(3), S(7), and S(11).

$$S(3)=A^{4}\times S(0)+u(1)=\begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\1&1&1&0\end{bmatrix}\times\begin{bmatrix}0\\0\\0\\0\end{bmatrix}+\begin{bmatrix}0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}0\\1\\1\\0\end{bmatrix},$$

$$S(7)=A^{4}\times S(3)+u(2)=\begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\1&1&1&0\end{bmatrix}\times\begin{bmatrix}0\\1\\1\\0\end{bmatrix}+\begin{bmatrix}0\\0\\1\\1\end{bmatrix}=\begin{bmatrix}0\\0\\0\\1\end{bmatrix},$$

$$S(11)=A^{4}\times S(7)+u(3)=\begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\1&1&1&0\end{bmatrix}\times\begin{bmatrix}0\\0\\0\\1\end{bmatrix}+\begin{bmatrix}0\\1\\0\\1\end{bmatrix}=\begin{bmatrix}1\\0\\1\\1\end{bmatrix}.$$

The computed state S(11) for the assigned type of LFSM appears to be CRC as the checksum, it took three iterations and three automaton cycles for its computation. For obtaining CRC as the check word for the information word (15) of the cyclic code, one additional iteration is necessary:

$$S(15)=A^{4}\times S(11)=\begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\1&1&1&0\end{bmatrix}\times\begin{bmatrix}1\\0\\1\\1\end{bmatrix}=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}. \qquad (16)$$

As we already mentioned in the previous chapter, it would be possible to find the check word for LFSM of type 1 without computations (16), but in order to achieve this, it would be necessary to solve a system of four equations above field GF(2).

Theorem 2. A tuple-parallel method of the CRC computation as the checksum of information word J of length k'' with the aid of r-input recursive LFSM of type 2, assigned by characteristic matrices (5), can be performed in $\frac{k''}{r}$ double-cycle iterations.

Proof. Similar to the preceding case, word Ψ and state S(k) are connected by equation (13).

Let us examine in more detail the process of forming the matrices A^r and L_r. It is easy to note that matrix A^i contains the first columns of matrices A, A^2,..., A^{i-1}, respectively, in the i-th place, in the (i −1)-th place,..., in the 2nd place (i=2÷r). On the other hand, the product A^iB is equal to the value of the first column of matrix A^{i+1}. As a result, for the matrices of form (5), the L_r and A^r matrices will be equal. Therefore, equation (13) can be written down as

$$S(i+r)=A^{r}\times S(i)+A^{r}\times u(i)=A^{r}\times(S(i)+u(i)),$$
$$GF(2). \qquad (17)$$

Expression (17) can be computed in 2 automaton cycles only:

1) computation of the intermediate state:

$$S''(i)=S(i)+u(i),\ GF(2),$$

2) computation of the desired state:

$$S(i+r)=A^{r}\times S''(i),\ \ GF(2).$$

Therefore, the final state, that is, CRC will be reached in $\frac{k''}{r}$ iterations, each duration for two cycles of the LFSM operation.

Theorem 3. The tuple-parallel method of the CRC computation as the checksum of information word J of length k'' with the aid of r-input recursive LFSM of type 3, assigned by characteristic matrices (6), can be performed in $\frac{k''}{r}$ three-cycle iterations.

Proof. For computing the equation (13), in a general case, three cycles are necessary:

a) $S'=A^{r}\times S(i),\ \ GF(2),$

b) $S''=L_{r}\times u(i),\ \ GF(2),$

c) $S(i+r)=S'+S'',\ \ GF(2).$

Then CRC will be calculated in $\frac{k''}{r}$ iterations, each duration three automaton cycles of the LFSM operation. But if we simultaneously compute intermediate values S' and S'', then $\frac{k''}{r}$ double-cycle iterations are required.

Thus, for the tuple-parallel method of the CRC computation with the aid of r-input LFSM of type 1, there will be required two times less time in comparison with r-input LFSM of type 2 or three times less time than for the LFSM of type 3.

The CRC value, calculated with the aid of LFSM of different types for one and the same information word J, will be identical.

8. Symbolic-parallel method of the CRC computation

A symbolic-parallel method of the CRC computation can be used when the input data enter in the form of symbols, which consist of h bits (h=2÷64). Usually, such symbolic data are converted into binary format and then the checksum is calculated in the traditional binary form.

Let us examine of CRC computation in the symbolic format over Galois field GF(2^h) according to (7) and (8). As already mentioned, solving of these tasks is possible with the aid of two LFSM: binary LFSM for computing the elements of field GF(2^h) and symbolic LFSM for the formation of CRC over field GF(2^h).

For the computations over field GF(2^h), it is necessary to select the appropriate generator polynomial. In the binary Galois field, the best choice for the computation of CRC is the generator polynomials of the Hamming codes and the Abramson codes [2]. Analogous approach can be applied also to the non-binary Galois fields.

It is known that in the non-binary Galois fields, the codes with primitive generator polynomial $g_s(x)$ have maximum length and make it possible to correct single symbolic errors; therefore, by analogy with field GF(2), we shall name them the non-binary Hamming codes. Let us note that [14] gives another definition for the non-binary Hamming codes.

In order to detect the maximum number of errors, let us multiply polynomial $g_s(x)$ by term $(\alpha^0+\alpha^0x)$:

$$g_a(x) = g_s(x)(\alpha^0 + \alpha^0 x),\ GF(2^h), \tag{18}$$

where α^0 is the element of field GF(2^h).

As a result of multiplication by term $(\alpha^0+\alpha^0x)$, the number of ZC of the corresponding length increases by 2h times in the graphical model of the code, which considerably improves the correcting and detecting capability of the code. Taking into account similar approach to the formation of generator polynomial of the Abramson code, we shall name a code with polynomial (18) over field GF(2^h) the non-binary Abramson code.

Example 2. Let us assume that two-bit symbols are transmitted by the channel, that is, 2 bits of information are transmitted in parallel. Let us examine the CRC computation for such input stream. The computations will be carried out with the aid of symbolic LFSM over Galois field GF(4).

We shall consider field GF(4) as the extension field GF(2^2) of field GF(2). Let us determine the elements of field GF(2^2) with the aid of primitive generator polynomial $g(x) = 1 + x + x^2$ of the binary field. For the specified generator polynomial, characteristic matrices of binary LFSM of type 1 take the form:

$$A_h = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix},\ B_h = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Let us determine the states of binary LFSM according to (1):

$$S(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix};\ S(1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix};$$

$$S(2) = A \times S(1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix};\ S(3) = A \times S(2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The obtained results can be presented in the form of the elements of field GF(2^2): $0,\ \alpha^0,\ \alpha^1,\ \alpha^2$ or 0, 1, 2, 3.

Let us select the primitive generator polynomial over field GF(4)

$$g_s(x) = \alpha^1 + \alpha^0 x + \alpha^0 x^2,\ GF(2^2), \tag{19}$$

to which such characteristic matrices of symbolic LFSM of type 1 correspond:

$$A_s = \begin{bmatrix} 0 & \alpha^1 \\ \alpha^0 & \alpha^0 \end{bmatrix},\ B_s = \begin{bmatrix} \alpha^0 \\ 0 \end{bmatrix}.$$

The property of the primitiveness of polynomial can be confirmed analytically: if in the (n+1)-th iteration of computations according to formulas (1) at U(t)=0 we obtain S(n+1)=S(1), then this generator polynomial of the (n, k)-code is primitive [2, 15].

A cyclic code with polynomial (19) can be considered as the non-binary Hamming (15,13)-code. From it, according to (18), it is possible to obtain the non-binary Abramson (15,12)-code with generator polynomial

$$g_a(x) = (\alpha^1 + \alpha^0 x + \alpha^0 x^2)(\alpha^0 + \alpha^0 x) = \alpha^1 + \alpha^2 x + \alpha^0 x^3,$$

$$GF(2^2). \tag{20}$$

The following characteristic matrices of symbolic LFSM of type 1 correspond to polynomial (20):

$$A_a = \begin{bmatrix} 0 & 0 & \alpha^1 \\ \alpha^0 & 0 & \alpha^2 \\ 0 & \alpha^0 & 0 \end{bmatrix},\ B_a = \begin{bmatrix} \alpha^0 \\ 0 \\ 0 \end{bmatrix}.$$

In order to evaluate the correcting and detecting capabilities of the examined codes, it is sufficient to analyze the graphical models of LFSM of these codes [16].

The graphical model of the non-binary Hamming code with generator polynomial (19) consists of one full ZC of length 15 and trivial ZC of length 1. This code makes it possible to correct the single symbolic errors (two-bit bursts errors of length 2) and to detect a larger number of errors of larger multiplicity.

The graphical model of the non-binary Abramson code with generator polynomial (20) consists of four full ZC of length 15 and four trivial ZC of length 1. This code makes it possible to correct the single symbolic errors (bursts errors of length 2 bits), the bursts symbolic errors of length two (bursts errors of length 4 bits) and to detect all symbolic errors of the double and odd multiplicity [16].

Let us remind that the serial entering of symbols from the channel has been assumed until now, and parallelism is provided for by processing the symbols, not separate bits. But at rapid arrival of symbols, it is possible to organize double parallelism, that is, to process the tuples of symbols.

The principle of symbolic-tuple-parallel method of the CRC computation is no different from the tuple-parallel method of the CRC computation, with the exception of the fact that instead of the binary cyclic codes, their non binary analogs are used.

9. Hardware implementation of accelerating the CRC computation

Hardware implementation of LFSM over Galois field GF(2) includes only two types of the basic elements:

– delay elements (flip-flops);

– the adders of the field elements (gates of logical operation XOR).

There is a mutually one-one mapping between the characteristic matrices of LFSM (4)–(6) and its hardware implementation [12]. Matrix A determines the interconnections of flip-flops. If the entry a_{ij} of the matrix A is equal to unity, then there must exist a connection between the output of the j-th flip-flop and the input of the i-th flip-flop, but if $a_{ij}=0$, then the connection between the specified flip-flops is absent. Matrix B determines the structure of LFSM inputs: if the entry b_{ij} of the matrix B is equal to unity, then there must exist a connection between the j-th input of LFSM and the input of the i-th flip-flop, but if $b_{ij}=0$, then this connection is absent.

As was analytically shown in the previous section, the fast-speed encoders and decoders use not a usual characteristic matrix A, but its r-th degree A^r (it is possible to take a lower degree, but then the performance will decrease proportionally).

The principle of the hardware implementation of matrix A^r is no different from the described principle of the hardware implementation of matrix A. Additional difficulties may arise only in the implementation of increased quantity of gates XOR. In fact, this is an usual task of optimal synthesis of the switching circuit (SC) from the gates XOR. It is possible to select either a variant with a minimal delay in the transmission of signals in SC or the variant with the minimum number of gates XOR.

A basic difference between r-input LFSM (LFSM based on matrix A^r) and usual single-inputLFSM is in the simultaneous. receipt of the r-bit tuple u(j) on its inputs. Table 1 shows the LFSM structures, which are used as the encoders and decoders according to Fig. 1–4.

We shall analyze below particularity of the practical implementation of encoders and decoders based on fast speed LFSM of types 1, 2 and 3.

Fig. 5 shows a generalized circuit of r-input LFSM of type 1. Its functioning is described according to equation (14).

During one cycle of the LFSM operation, the next state S(i+r) of LFSM is formed on the basis of the preceding state S(i) and parallel input tuple u(i). According to theorem 1, LFSM of type 1 may require $\frac{k''}{r}$ cycles of automaton time for CRC computation. For this LFSM, automaton cycle is realized by one hardware clock pulse.

Fig. 6 shows a generalized circuit of r-input LFSM of type 2. Its functioning is described according to equation (17). For the hardware formation of the next state S(i+r), two clock pulses from the current state S(i) should be transmitted to LFSM: by the first clock pulse, with the aid of shift register and gates XOR1, intermediate word S(i)+u(i) is formed, while by the second – with the aid of shift register and gates XOR2, the new state S(i+r) is calculated.

Table 1

LSC structures for 4 methods of the CRC computation

Computation methods	Encoder	Decoder
Serial	binary LFSM, 1=output	binary LFSM, 1-input
Tuple-parallel	binary LFSM, 1-output	binary LFSM, h-input
Symbolic-parallel	symbolic LFSM, h-output	symbolic LFSM, h-input
Symbolic-tuple-parallel	symbolic LFSM, h-output	symbolic LFSM, z×h-input

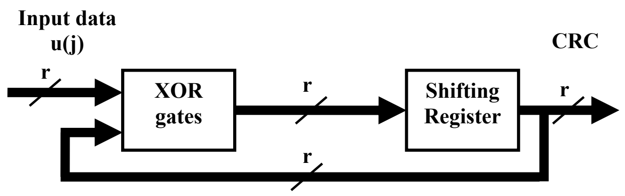

Fig. 5. Generalized circuit of r-input LFSM of type 1

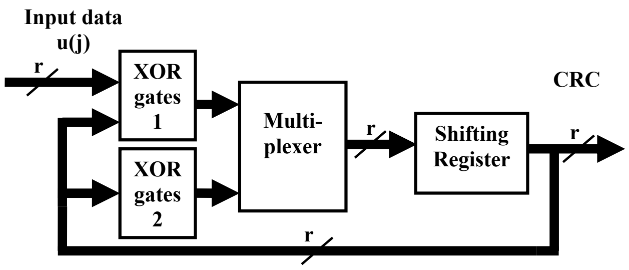

Fig. 6. Generalized circuit of r-input LFSM of type 2

The generalized circuit of r-input LFSM of type 3 is shown in Fig. 7. Its functioning is described according to Theorem 3. For the hardware formation of the next state S(i+r), it is necessary to feed three clock pulses from the current state S(i) to LFSM.

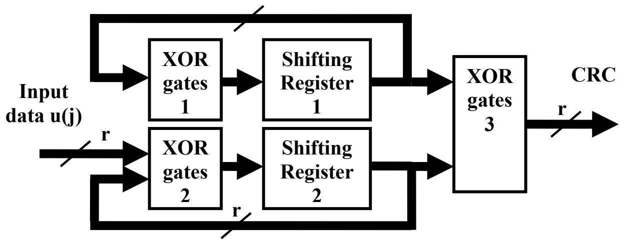

Fig. 7. Generalized circuit of r-input LFSM of type 3

By the first clock pulse, shift register 1 and gate XOR1, the computation of expression $S'=A^r \times S(i)$, GF(2) is performed. By the second clock pulse, with the aid of shift register 2 and gate XOR2, the expression $S''=L_r \times u(i)$, GF(2) is calculated. By the third clock pulse, with the aid of gates XOR3, state S(i+r) is formed.

Thus, mathematical acceleration of the procedures for encoding and decoding is differently realized in practice. More preferable are the fast speed LFSM of type 1.

The principles of hardware implementation of LFSM over field GF(2^h) are analogous to the above-presented principles in the binary Galois field, but there are two basic differences. At first, the elements of multiplication by the constant are added, secondly, all three basic elements are more difficult to realize in the non-binary Galois field.

10. Software implementation of accelerating the CRC computation

A hardware method of the CRC computation is the fastest; however, it cannot be always realized in practice. That is why a software method for the CRC computation is used more frequently.

The software method is slower and it is of course relevant to accelerate the process of computations. Solving this problem is possible on the basis of the automaton representation of the CRC-codes. The principle of acceleration is the same as with the hardware implementation – the computation of LFSM states with a certain interval, avoiding intermediate states. For the generator polynomial (3), it is possible to compute states only with interval r:

$$S(i), S(i+r), S(i+2r)...$$

The theoretical substantiation of this method of computation has been already presented in theorems 1–3. But what is the peculiarity of software implementation?

An analysis of formula (13) reveals that for the transfer from state S(i) to next S(i+r), the computation of term

$$A^r \times S(i),\ \ GF(2) \qquad (21)$$

is the most difficult part of this formula.

In the hardware CRC implementation, matrix A^r is realized by a change in quantity and connections of gates XOR, which causes almost no influence on the speed of computations. In order not to waste time on each iteration for the complicated computation of matrix A^r, it is possible to calculate in advance and to store in the memory the values of term (21) for all possible values of state S(i). This is basically a popular method for the software CRC computation based on the tables lookup [7, 8]. With the aid of this table, it is possible to form the corresponding state S(i+r) for each address (state S(i)).

It is necessary to note the difference in the work with tables lookup for LFSM of different types. For LFSM of type 1, it is sufficient to have one table, formed with the aid of matrix A^r. As follows from Theorem 2, for LFSM of type 2, also one table lookup is used based on the matrix A^r, but it is approached twice in each iteration. As follows from Theorem 3, for LFSM of type 3, it is necessary to have two tables lookup, one based on matrix A^r, and the second one – formed with the aid of matrix L_r.

Let us examine the algorithms for the CRC computation as the checksum by the tuple-parallel method with the aid of the tables lookup for different types of LFSM (both on the side of transmitter and on the side of receiver).

Algorithm 1. CRC computation for LFSM of type 1

Initial data:

– characteristic matrices A, B and initial state S(0) of LFSM;

– table lookup;

– data stream M of length k'' split into r-bit tuples u(i).

1. Assign the number of iteration i=0.

2. By the tabular address, equal to $S(i \times r)$, to find tabular value σ(i).

3. To compute the following state of LFSM:

$$S((i+1)r) = \sigma(i) + u'(i),\ \ GF(2),$$

where u'(i) is the tuple u(i) with the mutual transposition between the low-order and high-order digits:

$$u_j(i) = u_{r-j+1}(i),\ \ u_j(i) \in u'(i), u_{r-j+1}(i) \in u(i),$$

4. Assign $i = i+1$. If $i \le \frac{k''}{r}$ then go to 2.

5. $S((i+1)r)$ – CRC as the checksum.

6. End.

Algorithm 2. CRC computation for LFSM of type 2.

Initial data:

– characteristic matrices A, B and the initial state S(0) of LFSM;

– table lookup;

– data stream M of length k'', split into r-bit tuples.

1. Assign the number of iteration i=0.

2. By the tabular address, equal to $S(i \times r)$, to find tabular value $\sigma_1(i)$.

3. By the tabular address, equal to u(i), to find tabular value $\sigma_2(i)$.

4. To compute the following state of LFSM:

$$S((i+1)r) = \sigma_1(i) + \sigma_2(i),\ \ GF(2),$$

5. Assign $i = i+1$. If $i \le \frac{k''}{r}$, then go to 2.

6. $S((i+1)r)$ – CRC as the checksum.

7. End.

Algorithm 3. CRC computation for LFSM of type 3.

Initial data:

– characteristic matrices A, B and the initial state S(0) of LFSM;

– two tables lookup;

– data stream M of length k'', split into r-bit tuples.

1. Assign the number of iteration i=0.

2. In the first table, by the address, equal to $S(i \times r)$, to find tabular value $\sigma_1(i)$.

3. In the second table, by the address, equal to u(i), to find tabular value $\sigma_2(i)$.

4. To compute the following state of LFSM:

$$S((i+1)r) = \sigma_1(i) + \sigma_2(i),\ \ GF(2),$$

5. Assign $i = i+1$. If $i \le \frac{k''}{r}$, then go to 2.

6. $S((i+1)r)$ – CRC as the checksum.

7. End.

Let us note that the principles of the software implementation of accelerating the CRC computation over field

GF(2^p) are analogous to the above-presented principles in the binary Galois field. The difference is in the more complex representation of the elements of field GF(2^p); the memory size for their storage increases in p times, accordingly.

11. Discussion of results

More than 60 years have passed since the time the historical papers were published by K. Shannon that marked the beginning of development of the theory of error correction coding. The predictions of certain specialists about the completion of creating the foundation of this theory, at least for classic codes, proved to be premature. If we speak only about the cyclic codes, then only one stage of their development was completed, which was based on classic radio engineering and electrical communication.

Practice posed new problems, which traditional theory cannot solve. For example, the selection of complex generator polynomials of the CRC codes has remained almost the art until now. For the majority of the codes, it has been impossible to obtain accurate estimations of their correcting capability.

To solve these, and other, problems will become possible if the new theory of cyclic codes will be based not usual signal-code constructions, described in the textbooks on coding, but on the theory of finite automaton in the Galois fields, that is, on the theory of LFSM. The fundamentals of the new theory of cyclic codes are presented in the Author's monograph [13]. Present paper demonstrates special features of its application to the parallel CRC codes. Primary attention here is paid to the binary CRC codes and the ways are outlined for the consequent development of the nonbinary CRC codes, which will be very useful in the high speed multichannel communication systems.

12. Conclusions

We examined a method for the CRC codes representation on the basis of mathematical apparatus of LFSM and conducted comparative theoretical analysis of the LFSM properties of three types. It is demonstrated that the peculiarities of LFSM architecture, its performance efficiency and error detection capability are determined by the structure of its characteristic matrices.

Three methods of parallel computation of the CRC codes are proposed (tuple- parallel, symbolic-parallel and symbolic-tuple-parallel) with the aid of bit and symbolic LFSM. It is theoretically proven that different types of LFSM have different productivity.

Three variants of hardware implementation and three algorithms of software implementation are presented, as well as mathematical substantiation of the popular method of computations by the tables lookup, and special features of application of different types of LFSM are demonstrated.

For the parallel LSC we proposed the non-binary Hamming and Abramson codes, which provides for the high detection and correction capability for the CRC on their basis.

Thus, for the parallel CRC codes within the framework of one mathematical apparatus (theory of LFSM), the two problems are solved simultaneously: we conducted theoretical substantiation of the computational speed and the degree of detection capability.

References

1. Peterson, W. Cyclic Codes for Error Detection [Text] / W. Peterson, D. Brown // Proceedings of the IRE. – 1961. – Vol. 49, Issue 1. – P. 228–235. doi: 10.1109/jrproc.1961.287814
2. Semerenko, V. P. Theory and practice of crc codes: new results based on automaton models [Text] / V. P. Semerenko // Eastern-European Journal of Enterprise Technologies. – 2015. – Vol. 4, Issue 9 (76). – P. 38–48. doi: 10.15587/1729-4061.2015.47860
3. Walma, M. Pipelined cyclic redundancy check (CRC) calculation [Text] / M. Walma // 2007 16th International Conference on Computer Communications and Networks. – 2007. doi: 10.1109/icccn.2007.4317846
4. Krishna Reddy, K. V. An Optimization Technique for CRC Generation [Text] / K. V. Krishna Reddy // International Journal of Computer Trends and Technology (IJCTT). – 2013. – Vol. 4, Issue 9. – P. 3260–3265. – Available http://www.ijcttjournal.org
5. Hemant, S. FPGA implementation of 4-bit parallel Cyclic Redundancy Code [Text] / S. Hemant, H. Sharma, S. Tomar, J. Kanungo // International Journal of Research in Engineering and Technology. – 2015. – Vol. 04, Issue 11. – P. 111–113. doi: 10.15623/ijret.2015.0411021
6. Gawande, S. Design and Implementation of Parallel CRC for High Speed Application [Text] / S. Gawande, S. A. Ladhake // International Journal of Science and Research (IJSR). – 2015. – Vol. 04, Issue 2. – P. 90–92. – Available at: http://www.ijsr.net/archive/v4i2/SUB15590.pdf
7. Koopman, P. Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks [Text] / P. Koopman, T. Chakravarty // International Conference on Dependable Systems and Networks, 2004. – 2004. doi: 10.1109/dsn.2004.1311885
8. Nguyen, G. D. Fast CRCs [Text] / G. D. Nguyen // IEEE Transactions on Computers. – 2009. – Vol. 58, Issue 10. – P. 1321–1331. doi: 10.1109/tc.2009.83
9. Sheidaeian, H. Parallel Computation of CRC Using Special Generator Polynomials [Text] / H. Sheidaeian, B. Zolfaghari // International journal of Computer Networks & Communications. – 2012. – Vol. 4, Issue 1. – P. 39–47. doi: 10.5121/ijcnc.2012.4104
10. Albertengo, G. Parallel CRC generation [Text] / G. Albertengo, R. Sisto // IEEE Micro. – 1990. – Vol. 10, Issue 5. – P. 63–71. doi: 10.1109/40.60527
11. Campobello, G. Parallel CRC realization [Text] / G. Campobello, G. Patane, M. Russo // IEEE Transactions on Computers. – 2003. – Vol. 52, Issue 10. – P. 1312–1319. doi: 10.1109/tc.2003.1234528
12. Gill, A. Linear sequential machines [Text] / A. Gill. – Moscow: Science, 1974. – 288 p.
13. Semerenko, V. P. Teorija cyklichnyh kodiv na osnovi avtomatnyh modelej [Text]: monografija / V. P. Semerenko. – Vinnycja: VNTU, 2015. – 444 p.

14. Blahut, R. Theory and Practice of Error Control Codes [Text] / R. Blahut. – Moscow: Myr, 1986. – 576 p.
15. Ahmad, A. Selection of polynomials for cyclic redundancy check for the use of high speed embedded – an algorithmic procedure [Text] / A. Ahmad, L. Hayat // Wseas Transactions on Computers. – 2011. – Vol. 10, Issue 1. – P. 16–20.
16. Semerenko, V. P. Estimation of the correcting capability of cyclic codes based on their automation models [Text] / V. P. Semerenko // Eastern-European Journal of Enterprise Technologies. – 2015. – Vol. 2, Issue 9 (74). – P. 16–24. doi: 10.15587/1729-4061.2015.39947

На прикладах успішних реалізацій технологій у проектах ненаселених привязних та автономних підводних апаратів (АПА) показано ефективність автоматизованих систем управління (АСУ) з гібридною системою підтримки прийняття рішень (СППР). Поставлена та розв'язана задача аналітичного визначення залежності похибки від властивостей АПА та параметрів процесу, як кількісного критерію вибору альтернатив моделі, алгоритму, керуючих правил у ході фукціюнування АСУ підводних технологій

Ключові слова: координаційне управління, оцінка похибки моделі, гібридна СППР, АСУ підводних технологій

На примерах успепешных реализаций в проектах необитаемых привязных и автономных подводных аппаратов (АПА) показана эффективность автоматизированных систем управления (АСУ) с гибридной системой поддержки принятия решений (СППР). Поставлена и решена задача аналитического определения зависимости ошибки от свойств АПА и параметров процесса, как количественного критерия выбора альтернатив модели, алгоритма, управляющих правил в ходе функционирования АСУ подводних технологий

Ключевые слова: координационное управление, оценка ошибки модели, гибридная СППР, АСУ подводных технологий

UDC 519.816:009. 826:629.585
DOI: 10.15587/1729-4061.2016.85585

CRITERIA FOR THE EVALUATION OF MODEL'S ERROR FOR A HYBRID ARCHITECTURE DSS IN THE UNDERWATER TECHNOLOGY ACS

A. Trunov
PhD, Associate Professor, First Vice-Rector
Department of automation and computer-integrated technologies
Petro Mohyla Black Sea National University
68 Marines str., 10, Mykolayiv, Ukraine, 54000
E-mail: trunovalexandr@gmail.com

1. Introduction

The inability of classic methods of the theory of automatic control to effectively resolve the problems of automation of enterprises and their management has been paid more and more attention to in the technical literature [1–3]. An analysis of negative results and the reasons that caused them indicates that the coordination, coordinational control [1], as the main principle of functioning in the overall management system, plays a role of a subsystem in the stabilization process in relation to the predetermined strategy [4]. When defining the role of coordination in the process of control, it should be noted that for any intellectual, industrial, social and everyday human activity, mandatory is a typical procedure for making a decision [1–3, 5, 6]. At present, scientists, based on the study and systematization of technologies [7–28], including underwater technology [4, 7–15, 29, 30], determined and formed a generalized structure of underwater technological complex and generalized models of the technological process (TP) automation control system (ACS). In addition, an analysis of the methods of control over complex automated systems was carried out [4, 7, 13–15, 17–20]. As evidenced by the results of analysis, structural constituents of such systems are mostly designed with automatic or automated control systems [15, 17]. It is established that the application of methods of designing control system processes that are suitable for the functioning in the surface marine technologies, is complicated in the underwater technology [30]. The latter is due to the fact that the magnitude of time of the transition process in the executive mechanisms is comparable to the magnitude of the system transition from one state to another and to the magnitude of time necessary for decision making [7, 30]. In addition, it is predetermined by nonlinearity of the processes of interaction with the environment and complexity of the adequate modeling, which is caused by insufficient exploration of dynamics of underwater apparatus, manipulators and technological equipment [8]. A significant obstacle is also the problem of changing the angular position of the device, as the carrier of technological equipment in space, which occurs when implementing the control algorithms and is caused by a change in the centers of mass and the moments of inertia [7]. Another obstacle is the unknown features of the implementation of technologies and a lack of methods that enable the prediction of possible changes, while under design, in the functional purpose of the entire complex [30]. This particularly, concerns individual