UDC 519.7

Tupkalo V. N.

SIGNATURE CHECKING OF ROM MODULES

A method is proposed for ensuring the fault tolerance of read-only memory modules (ROMs) based on the principle of predicting a control signature in each access cycle to memory. The necessary conditions were found for solving the recovery problem for lost data during the scan of the read operation.

The importance of any built-in verification method increases, if along with the solution of the verification problem the method has some additional capabilities. The article shows that the proposed signature method for checking the status of the programmed ROM allows, in principle, to solve the problem of regeneration (recovery) for lost data at the time of checking the read operation.

The solution to this problem is based on the aforementioned principle of signature prediction, expressed in the form of a signature verification rule.

Key words: *memory modules, signature, work check, forecasting, digital control systems.*

The need in fault-tolerant digital control systems (DCS) of different purpose calls for searching new approaches to solving the problem of checkability of DCS functional units and, in particular, memory modules.

Depending on the type of access operations the DCS memory units are classified as random-access memory (RAM) modules and read-only memory (ROM) modules. For RAMs there exists an efficient hardware-microprogram method for checking the state during the operation pauses whose essence reduces to preliminary reading and storing of a cell's contents with its subsequent inversion and with repetition of the write-read cycle to compare the results [1]. This method is inapplicable for ROM because in this case the possibility of the write operation is excluded.

From the standpoint of functional representation of models for checking objects, a DCS ROM is a state checking (diagnostics) object with transfer function Ψ performing a mapping Ψ: $K(A) \rightarrow K(B)$, where $K(A)$ is a tuple of cell addresses and $K(B)$ is a tuple of the cells' contents. The mapping should be regarded here as being many-valued (surjective) because a DCS ROM may contain different cells with the same contents. Therefore the combination of the input sequence in the form $K(A)$ and the output sequence in the form $K(B)$ cannot be interpreted as ROM test in the general case.

The set A of the ROM addresses is always finite, and under sampling it can be represented by $N!$ (N is the number of memory cells) tuples of the same power, which are ordered. Therefore the unitariness (determinism) principle for checking will hold if, in accordance with the well-known concept in [2], the test is formed as a pair of tuples $\{K(A), K(A\#B)\}$, where # is a superposition (composition) operation for the sets of addresses A and contents B such that the mapping

$$\Psi_{\#}: K(A) \rightarrow K(A\#B) = K(A)\#K(B). \qquad (1)$$

Problem statement. In view of condition (1), the problem of synthesizing a test for checking the states of a programmed ROM module can be stated in the following way: given the contents B of the ROM cells, it is required to synthesize an input test action $K^*(A)$ using the given set A of ROM addresses such that the function $\Psi_{\#}$ for the established (chosen) operation # is algorithmically computable (a computing algorithm for the function $\Psi_{\#}$ must necessarily exist [3]).

It should be noted that in the context of the general problem of checking the states of the ROM during the pauses in its operation the requirement of algorithmical computability of the function $\Psi_{\#}$ must be regarded as being only a necessary but not sufficient condition for test synthesis because the mapping (1) preassigns no principle for forming the template. Therefore, proceeding from the requirement that the unitariness principle should hold for the functional checking, we state a sufficient condition for synthesizing the ROM test in the following way: if, in accordance with (1), the function $\Psi_{\#}$ is algorithmically computable, then checking the ROM state can be reduced to checking a characteristic feature of the current (i+l)-th n-digit binary vector $(A \# B)_{i+1}$ of the tuple $K(A \# B)$ by using the characteristic features of the directly preceding i-th vector $(A \# B)_i$ (i = 1, 2, .., are the memory access cycles) if the functional relationship

$$(A\#B)_{i+1} = \Psi_{\#}(A\#B)_i. \tag{2}$$

corresponds to the function computing algorithm.

We will prove that, in principle, for any contents of ROM cells the condition (2) is satisfied guaranteeing the test checkability of the ROM according to the principle of prediction of checked binary vectors. To this end we use the fact that the number of memory cells in the existing and newly elaborated ROMs is a multiple of a power of two. Moreover, based on the notion of "algorithmic solvability" [3] known from the algorithm theory we introduce the following definition.

***Definition* 1.** An ROM test $\{K^*(A); K^*(A\#B)\}$ is said to be (algorithmically) solvable if for its tuple $K^*(A\#B)$ there exists a formation algorithm coinciding with the computing algorithm for the function (2). In this case the tuple $K^*(A)$ is called a solvable test action.

***Theorem* 1.** An ROM test is algorithmically solvable if the number of test words in the tuple $K(A)$ is equal to $|K(A)| = 2^n-1$, where $n = \log_2 N$ and N is the number of memory cells.

Proof. First, it is necessary to prove that in the context of condition (2) there always exists a composition operation # such that the mapping (1) is one-to-one. Indeed, since the tuple $K(A)$ is strictly ordered, the tuple $K(A \# B) = K(A) \# K(B)$ is also strictly ordered for any cell contents, if # is the logical operation on binary numbers (the concatenation operation ∇). For example, if A_i = 0110111 and B_i = 10101, then $A_i \# B_i = A_i \nabla B_i$ = 011011110101. The requirement that the mapping (l) be injective implies that $|K(A)| = |K(A) \# K(B)| = 2^n$. Therefore, according to Definition 1, the tuple $K(A)$ is a solvable test action if for the tuple $K(A) \# K(B)$ there is an algorithm that gives the answer to the question whether A_j belongs to the tuple $K(A)$ or not for any of the numbers $A_j \in K(A)$. As is known from coding theory, such an algorithm exists for a sequence of numbers formed with the help of a shifting register with feedback (in the form of polynomial number generators) [4]. In this case the maximum number of different n-digit numbers is generated when the feedback is described by an irreducible primitive polynomial $G(x) = g_n x^n + g_{n-1}x^{n-1} + \ldots + g_1 x + 1$ of degree n, i.e., $|K(A)| = 2^n-1$ (a zero number in the tuple $K(A \# B)$ is excluded).

Based on Theorem 1, we can conclude that the method for solving the problem of checking the state of a programmed ROM must involve the following two consecutive synthesis stages: when data are written in the ROM, the relation (1) is fulfilled with condition $B_j \# A_j = 00...0 \notin K(A \# B)$ being satisfied; and, with account of the selected polynomial $G(x)$ for generating a test of degree $n = \log_2 N$, the set $A \# B$ is ordered so that the tuple $K(A \# B)$ is predictably checkable when checking the state of the programmed ROM, i.e., the algorithmic computability of the function (2) is ensured. In this case the resulting tuple $K(A)$ is the tuple $K^*(A)$. When these synthesis stages are realized, the programmed ROM is regarded as being predictably checkable.

The proof of Theorem 1 implies that since the algorithmically solvable function for generating the test with $|K(A)| = 2^n-1$ is a known function generating binary numbers with the aid of a shift register toward higher digits with modulo-two adder feedback described by an irreducible primitive polynomial $G(x)$ [4], the general relationship (2) for an n-digit binary notation satisfies the recurrence relation

$$(A\#B)_{i+1} = 2(A\#B)_i \oplus r_{[n]}\left[\sum_{\alpha=1}^{n} g_\alpha (a_\alpha \# b_\alpha)_i\right], \tag{3}$$

where $(i + 1)$ and i are, respectively, the subsequent and preceding shift cycles, $r_{[n]}$ is an n-digit constant with unity only in the least significant digit, g_α are the coefficients in nonzero terms of the polynomial $G(x)$ for generating binary numbers, and $(a_\alpha \# b_\alpha)_i$ is the value of the a-th digit of the number $(A \# B)_i$.

In view of the fact that the Boolean equivalent of the arithmetic function for adding two identical numbers R has the form $2R = R \oplus R \oplus H(R+R) = H(R+R)$ [5] and $H(R+R)$ is the Boolean operation (function) producing numbers whose code characterizes the positions of carry unities in the addition of identical numbers, we finally obtain the following expression for (3)

$$(A\#B)_{i+1} = \widehat{H}[(A\#B)_i+(A\#B)_i] \oplus r_{[n]}[\sum_{\alpha=1}^{n} g_\alpha(a_\alpha \# b_\alpha)_i], \quad (4)$$

where $\widehat{H}[(A\#B)_i+(A\#B)_i]$ is obtained by truncating the code of the number $H[(A\#B)_i+(A\#B)_i]$ on the left (discarding the highest-order digit).

Therefore under the transition from the numbers in (4) to their signatures sg the signature checking rule for the state of a predictably checkable ROM is written as

$$\mathrm{sg}(A\#B)_{i+1} = \mathrm{sg}\widehat{H}[(A\#B)_i+(A\#B)_i] \oplus r_{[n]}[\sum_{\alpha=1}^{n} g_\alpha(a_\alpha \# b_\alpha)_i]. \quad (5)$$

An example demonstrating the essence of the stages in the suggested technique for synthesizing an algorithmically solvable test (see Definition 1) for checking the states of a programmed ROM with organization 32 X 4 is presented in Tables 1 and 2. It is assumed that the tuple elements in Table 2 follow from top to bottom. The role of the operation # is played by the modulo two addition of two digits with the same polynomial weight.

Table 1

B_j	A_j	$B_j \# A_j$	B_j	A_j	$B_j \# A_j$
0000	00001	00001	0111	11111	11000
0001	00010	00011	1000	00111	01111
0001	00011	00010	1000	10011	11011
0001	00100	00101	1000	10100	11100
0001	01101	01100	1001	10110	11111
0010	00110	00100	1010	11001	10011
0010	01111	01101	1010	11110	10100
0010	10000	10010	1011	01100	00111
0011	01010	01001	1100	10001	11101
0011	01001	01010	1100	11011	10111
0100	10010	10110	1101	11101	10000
0100	10101	10001	1101	10111	11010
0101	01011	01110	1110	00101	01011
0101	11100	11001	1110	01000	00110
0110	01110	01000	1111	11010	10101
0110	11000	11110			

A fundamental property of an algorithmically solvable test $\{K^*(A); K^*(A\# B)\}$ is that it can cycle an unbounded number of times and that the tuple $K^*(A)$ must not necessarily return to the initial address after an interruption of the diagnosing process (owing to the end of a pause in the ROM operation).

***Definition* 2.** A test of a predictably checkable ROM is said to be enumerable if

$$|K(A)| = 2^n - 1.$$

The notion of an enumerable test (enumerable set [3]) relates to a typical situation when the amount and the values of the numbers meant for storage make it possible to synthesize the test without

cycling. Such a test does not permit the diagnosis of the first cell addressed by the tuple $K(A)$. We will illustrate this situation for the example of Table 2. Let the tuple $K(B)$ of the ROM contents be confined to the first twelve elements in the test synthesis. The cell with address 00001 is undiagnosable. In this case the cycling becomes possible if in the realization of condition (2) provision is made for a checkable transition from the address 01101 to 00001. Table 2 shows directly that such a transition exists if, with account for the checking rule (5) for predicting signatures, and depending on the selected polynomial $P(x)$ for generating signatures, the test tuples $K(A)$ and $K(B)$ are supplemented with subsequent three (boldfaced) elements ($P(x) = x^4+x^3+1$), or with four (marked by asterisks) elements ($P(x) = x^3+x+1$). As a result, the set becomes cyclically enumerable. Moreover, as is seen from Table 2, there can be several cyclically enumerable tests and they are characterized by different lengths (redundancy). The question of redundancy is decided at the stage when the operation # and the polynomials $G(x)$ and $P(x)$ are chosen. Neither is it excluded that the generation polynomial $G(x)$ can be irreducible.

Table 1

$K(B)$ $b_4b_3b_2b_1$	$K(A)$ $a_5a_4a_3a_2a_1$	$K(B)\#K(A)$ $G(x) = x^5+x^2+1$	$K(\mathrm{sg}(B\#A))$ $P(x) = x^4+x^3+1$	$K(\mathrm{sg}(B\#A))$ $P(x) = x^3+x+1$
0000	00001	00001	0001	001
0001	00011	00010	0010	011
0001	00100	00101	0101	110
0011	01001	01010	1011	101
1111	11010	10101	0110	011
1110	00101	01011	1010	100
1100	11011	10111	0100	000
0101	01011	01110	1111	010
1100	10001	11101	1111	101
1000	10011	11011	1001	001
0100	10010	10110	0101	001
0001	01101	01100	1101	001
0111	11111*	11000	1010	Oil
0100	10101*	10001	0010	100
0001	00010*	00011	0011	010
1011	01100	00111	0111	101
1000	00111	01111	1110	011
1001	10110	11111	1101	110
0110	11000	11110	1100	111
1000	10100	111000	1110	100
0101	11100	11001	1011	010
1010	11001	10011	0000	111
1110	01000	00110	0110	100
0010	01111	01101	1100	000
1101	10111	11010	1000	000
1010	11110	10100	0111	010
0011	01010	01001	1000	111
0010	10000	10010	0001	110
0010	00110	00100	0100	111
0110	01110	01000	1001	110
1101	11101*	10000	0011	101

The importance of any built-in checking method increases if, along with the solution of the checking problem, the method possesses some additional potentialities. In particular, we will show that the suggested signature method for checking the state of a programmed ROM makes it possible,

in principle, to solve the regeneration (recovery) problem for lost data at the time when the read operation is checked. The solution of this problem is based on the above-mentioned signature prediction principle expressed in the form of the signature checking rule (5) and the proof of the following assertions.

***Theorem* 2.** If the numbers A_i, and their signatures $\mathrm{sg}A_i$ are elements of the field $GF(2^m)$, then the result of the summation $A_i \oplus \mathrm{sgsg}A_i$, is the zero element of this field in the case when the generating polynomial for signatures has the form $P(x) = x^m + x^{m-1} + 1$.

Proof. Because, [6]

$$\mathrm{sg}A_i\,(x) = M^{-1}[A_i\,(x) \bmod P(x)],$$

where M^{-1} is the inversion of the matrix M of nonzero coefficients of the irreducible primitive polynomial for generating signatures, we have

$$\mathrm{sgsg}A_i\,(x) = M^{-1}\{M^{-1}[A_i\,(x) \bmod P(x)]\} \bmod P(x) = M^{-1}M^{-1}[A_i\,(x) \bmod P(x)].$$

Therefore the relation

$$M^{-1}M^{-1}[A_i\,(x) \bmod P(x)] = A_i\,(x).$$

can hold if for $\deg P(x) > \deg A_i\,(x)$ the product $M^{-1}M^{-1}$ is a unit matrix. The condition that the numbers and their signatures belong to the field $GF(2^m)$ implies that the order of this matrix is equal to $m-1$.

***Theorem* 3.** An algorithmically solvable (cyclically solvable) test for an ROM with organization $N \times m$ is recovering if the operation # is the modulo two addition and the polynomial for generating signatures has the form $P(x) = x^m + x^{m-1} + 1$.

Proof. Based on the rule (5) for ROMs, Definition 1 implies that in each i-th access cycle the standard signature $\mathrm{sg}(A_{i+1} \# B_{i+1})$ is predicted, and in the $(i+1)$-th cycle the current value of the signature $\mathrm{sg}(A^t_{i+1} \# B_{i+1})$ is determined from the test address $A^t_{i+1} \in K(A)$. Therefore, as is known, the superposition

$$\mathrm{sg}(A_{i+1} \# B_{i+1})_s = \mathrm{sg}A^s_{i+1\ +1} \# \mathrm{sg}B^s_{i+1}$$

is possible if the operation is the modulo two addition [3]. Hence, we have

$$\mathrm{sg}(A_{i+1} \oplus B_{i+1})_s \oplus \mathrm{sg}A^t_{i+1} = \mathrm{sg}B^s_{i+1}, \qquad (6)$$

and, in view of the assertion of Theorem 2, the code of the number B^s_{i+1} can be recovered ($\mathrm{sgsg}B^s_{i+1} = B^s_{i+1}$) only in the case when $\deg P(x) > \deg B^s_{i+1}$ and $P(x) = x^m + x^{m-1} + 1$.

Recovery. The proof of Theorem 3 implies that at the time of the checking (comparison) we have $\mathrm{sg}(A_{i+1} \oplus B_{i+1})_s = 0100$ for the standard signature, i.e., it is the value of the right-hand side of the inequality. Because the value of the current chosen address $A^t_{i+1} = 11011$ is known, the signature $\mathrm{sg}\,A^t_{i+1} = \mathrm{sg}(1101) = 1001$ is also known. Therefore, based on relation (6), we have $\mathrm{sg}B^s_{i+1} = 0100 \oplus 1001 = 1101$, and Theorem 2 implies $\mathrm{sgsg}B^s_{i+1} = \mathrm{sg}[\mathrm{sg}(1\ 101)] = 1100$. i.e., the true contents of the cell with address $A_{i+1} = 11011$ is recovered.

Let us estimate hardware expenditures required to perform the recovery of the contents of ROM cell as compared to the implementation of the checking alone (realization of the rule (5)) and let us find out, for instance, how these expenditures relate for the above- mentioned ROM with organization 32×4 in the case when Hamming's code (information redundancy for recovery) is used.

According to the assertion of Theorem 3, the checking rule (5) must have the form

$$\mathrm{sg}(A^t_{i+1} \oplus B_{i+1}) \underbrace{=}_{\varepsilon=\{0,1\}} \mathrm{sg}\widehat{H}[(A\oplus B)_i + (A\oplus B)_i] \oplus r_{[m]}[\textstyle\sum_{\alpha=1}^{n} g_\alpha (a_\alpha \oplus b_\alpha)_i], \qquad (7)$$

where $\varepsilon = \{0,1\}$ is the checking result: 1 – error is not detected; and 0 – error is detected; n is the number of digits in the representation of the number $A \oplus B$.

Since the right-hand side of (7) is the standard signature for the access cycle i + l predicted in the i-th access cycle:

$$\mathrm{sg}(A_{i+1} \oplus B_{i+1})_s = \mathrm{sg}A_{i+1} \oplus \mathrm{sg}B^s{}_{i+1})_s,$$

it follows from (6) and (7) that

$$\mathrm{sg}B^s{}_{i+1} = \mathrm{sg}(A_{i+1} \oplus B_{i+1})_s \oplus \mathrm{sg}A^t{}_{i+1} =$$

$$= \mathrm{sg}\widehat{H}[(A\oplus B)_i + (A\oplus B)_i] \oplus r_{[m]}[\textstyle\sum_{\alpha=1}^{n} g_\alpha (a_\alpha \oplus b_\alpha)_i] \oplus \mathrm{sg}A^t_{i+1}, \qquad (8)$$

Formula (7) implies that for the zero contents B_{i+1} we have

$$\mathrm{sg}A^t_{i+1} = \mathrm{sg}\widehat{H}(A_i + A_i) \oplus r_{[m]}[\textstyle\sum_{\alpha=1}^{n} g_\alpha a_\alpha]. \qquad (9)$$

Therefore, based on the assertion of Theorem 2, for $\varepsilon = 0$ the substitution of (9) into (8) results in

$$B^s{}_{i+1} = \mathrm{sg}\{\mathrm{sg}\widehat{H}[(A\oplus B)_i + (A\oplus B)_i] \oplus$$

$$\oplus r_{[m]}[\textstyle\sum_{\alpha=1}^{n} g_\alpha (a_\alpha \oplus b_\alpha)] \oplus \mathrm{sg}\widehat{H}(A_i + A_i) \oplus r_{[m]}[\textstyle\sum_{\alpha=1}^{n} g_\alpha a_\alpha] =$$

$$= \mathrm{sg}\{\mathrm{sg}(A_{i+1} \oplus B_{i+1})_s \oplus \mathrm{sg}\widehat{H}[(A\oplus \varepsilon B)_i + (A\oplus \varepsilon B)_i] \oplus r_{[m]}[\textstyle\sum_{\alpha=1}^{n} g_\alpha (a_\alpha \oplus \varepsilon b_\alpha)]\}. \qquad (10)$$

The comparison of (7) and (10) shows that to recover the information distorted in the cell with address A_{i+1} the checking module synthesized in accordance with (7) must be supplemented with two-input logic elements AND whose number is determined by the word length m of the memory cell. Besides, according to (10), to the output of the comparison unit for the standard (predicted) and current signatures an additional combination unit (a signature operator sg before the curly bracket) for producing signatures with the same number m of inputs and outputs must be attached. As is known [5], for the polynomial $P(x) = x^4 + x^3 + 1$ and $m = 4$ this unit must implement the system of Boolean functions $Y = y_4\, y_3\, y_2\, y_1$ where $y_4 = s_4 = b_4$; $y_3 = s_3 = b_3$; $y_2 = s_2 = b_2$; $y_1 = s_4 \oplus s_1 = b_1$; $S = s_4 s_3 s_2 s_1$ is the code of the result of the modulo two addition (the operation in curly brackets in (10)); and b_j are the digits of the recovered number $B^s{}_{i+1}$. The weights of digits in the code of the number S increase from right to left.

Thus, the unit for producing $\mathrm{sgsg}B^s{}_{i+1}$ contains three one-digit communication lines and one two-input modulo two adder.

For the above case of four-digit ROM with organization 32×4 there exists a Hamming (8, 4) code [4] capable of correcting all single errors and detecting all double errors (the minimum code distance is equal to 4). In this case the four data digits are supplemented with the same number of checking digits. Consequently, when data redundancy is introduced for checking and recovering only single errors based on Hamming's (8, 4) code, it is required to pass from 32×4 to 32×8 ROM, i.e., a 100% additional ROM capacity is needed to store the Hamming-coded data.

Thus, in contrast to the well-known signature checking methods for ROM devices (e.g., see [7], [8]), the signature checking of ROM based on the predictability principle for the checking signature in the $(i+1)$-th read cycle with account of the value of the cell address determined by the signature of the result read in the foregoing i-th cycle requires no additional memory for storing the standard

signatures. In this case the originally stored data can be recovered without using the information redundancy.

REFERENCES

1. B. M. Kagan, Elektronnye vychislitelnye mashiny i sistemy (Electronic computers and computing systems).– Moscow: Energiya, 1979.
2. GOST 20911-89. Tekhnicheskaya diagnostika. Terminy i opredeleniya (Standards 20911-89. Technical diagnostics. Terminology and definitions).–Moscow: Izd-vo standartov, 1989.
3. O. P. Kuznetsov and G. M. Adelson-Velskiy, Diskretnaya matematika dlya inzhenerov (Discrete mathematics for engineers).–Moscow: Energiya, 1980.
4. P. Blayhoot, Teoriya i praktika kodov konrtoliruyushchikh oshibki (Theory and application of error-controlling codes).– Moscow: Mir, 1986.
5. V. N. Tupkalo, Avtomatika i Telemekhanika.– No. 1: 167-172.–1993.
6. V. N. Tupkalo, Electronic Modeling 10, No. l .–1992.
7. I. V. Ognev and K. F. Sarychev, Nfidezhnost zaporninayushchikh ustroistv (Reliability of storage devices).–Moscow: Radio i Svyaz, 1988.

В. Н. Тупкало

ПРОВЕРКА ПОДПИСИ ПЗУ МОДУЛЕЙ

Предложен способ обеспечения отказоустойчивости модулей памяти только для чтения (ПЗУ) на основе принципа прогнозирования контрольной сигнатуры в каждом цикле доступа к памяти. Найдены необходимые условия для решения проблемы восстановления для потерянных данных во время проверки операции чтения.

Важность любого встроенного метода проверки возрастает, если наряду с решением проблемы проверки метод обладает некоторыми дополнительными возможностями. В статье показано, что предлагаемый метод подписи для проверки состояния запрограммированного ПЗУ позволяет, в принципе, решить проблему регенерации (восстановления) для потерянных данных в момент проверки операции чтения.

Решение этой проблемы основано на вышеупомянутом принципе предсказания подписей, выраженном в форме правила проверки подписи.

Ключевые слова: *модули памяти, подпись, проверка работы, прогнозирование, системы цифрового управления.*

В. М. Тупкало

ПЕРЕВІРКА ПІДПИСИ ПЗУ МОДУЛІВ

Запропоновано спосіб забезпечення відмовостійкості модулів пам'яті тільки для читання (ПЗУ) на основі принципу прогнозування контрольної сигнатури в кожному циклі доступу до пам'яті. Знайдено необхідні умови для вирішення проблеми відновлення для втрачених даних під час перевірки операції читання.

Важливість будь-якого вбудованого методу перевірки зростає, якщо поряд з вирішенням проблеми перевірки метод володіє деякими додатковими можливостями. У статті показано, що пропонований метод підписи для перевірки стану запрограмованого ПЗУ дозволяє, в принципі, вирішити проблему регенерації (відновлення) для втрачених даних в момент перевірки операції читання.

Вирішення цієї проблеми грунтується на вищезгаданому принципі передбачення підписів, вираженим у формі правила перевірки підпису.

Ключові слова: *модулі пам'яті, підпис, перевірка роботи, прогнозування, системи цифрового управління.*

Рецензент: д.т.н., професор Азарсков В.М., НАУ