Address Sequence Generator for Memory BIST

Mikalai Shauchenka

Student of the Lichtenbergschule Darmstadt, Ludwigshöhstraße 105, 64285 Darmstadt, Germany

Abstract

In this work a structure that generates memory address sequences for built-in memory self tests is proposed. The idea is to significantly expand the set of address sequences. The structure consists of three components, namely the base address sequence generator, a device for generating and storing a matrix of binary vectors, and a device to calculate the address values. The idea behind this structure is to significantly expand the set ofdifferent address sequences including the standard well known and extensively used sequences for memory testing.

Keywords — *MBIST, Built in memory self-test, pseudo-random numbers, pseudo-random sequences.*

I. INTRODUCTION

For recent integrated circuits designs, a Built-in Self-Test (BIST) for the memory is becoming more abd more important, especially for embedded memory, which is a most significant part of System on Chip (SoC) designs, occupying more than 70% of the area overhead and is expected to rise up to 95% (!) [1, 2]. Because of this, Memory Built-In Self-Tests (MBIST) have become a much more common field of research and are now seen more and more often in the industry [2, 3, 4]. For high embedded memory quality, regardless of their architecture, size and levels, the fault detection capability of the BIMST plays a significant and crucial role. BIMST can offer some benefits, the most significant is at-speed testing and therefore high fault coverage [5]. Traditionally, MBIST based on March test algorithms consists of a set of March elements with the desired memory address order. The memory accesses speed and area overhead of the MBIST mainly depends on the address sequence (AS) generator, which is the most critical part of a goodMBIST. AS generator designs are very different and the area requirement for their implementation is varying between 26 and 33% of MBIST [3]. Mainly binary counters and Linear Feedback Shift Register (LFSR) are used to generate ASs, which can be succeedingly applied to the memory core to form a test. Linear feedback shift registers are an efficient way of describing ASs and generating them in hardware implementations. An LFSR reduces the amount of required logic (area overhead), minimizes routing complexity and increase testing speed. Binary LFSR have been studied for more than half a century [6]. The maximum-length sequence generated by the LFSR, called M-sequences (maximum length sequence) or pseudo-random number (PN) sequences, are the best-known and most thoroughly studied special case of binary sequences [6, 7]. There are numerous LFSR applications in different areas, for example digital circuit testing, spread spectrum communications, cryptographic stream cipher, PN number generation, and many others [7, 8, 9]. In modern MBIST s, LFSRs are playing an important role as address generators [3, 12, 13]. The address sequences properties and implementation aspects of several mostly used ASs have been considered in [3]. As the result a novel, very systematic, high speed, low-power and low-overhead implementation, based on an Up-counter and a set of multiplexors have been presented and analyzed. The proposed solution [3] is concentrated on the restricted set of address sequences. The paper [12] provides a direct comparison between a fast binary counter, built using a hierarchical Manchester carry chain, and a counter built using a LFSR. The investigation is focused on speed, power consumption and area overhead. It was demonstrated the main benefits of using of LFSRs as an alternative to conventional binary counters. Multi-stage LFSR counter with reduced decoding logic for large-scale array applications was considered in [13]. As have been shown in the paper LFSR counter to be well suited to applications required large arrays of counters. In the papers [14, 15], papers there are approaches of developing the architectures of address generators with low-transition. It has been proven and validated that efficient implementation of the AS generator has cut-down the switching activity of MBIST sufficiently [15]. The proposed approaches based on specifically modified LFSRs structures and that is why allowed to generate the restricted sets of the address sequences belong to the M- sequences family. The reducing of power consumption during the memory core testing of System on a Chip is one of the most important issues. To reduce the power consumption of MBIST the design proposed in [16] concentrated on just only three types of the ASs, namely LFSR based, Linear and Gray Code ASs. The comparison with the standard solutions in terms of the area overhead and consumed power have been presented and analyzed. For only one LFSR based AS the same power reducing issue was investigated in [17]. In order to detect complex and speed-related memory faults the address sequences implemented in MBIST should cover a wide range of different varieties of such sequences should be extended and

flexible [3]. At the same time the area overhead as well as address generation speed also are very important characteristics for the address sequence generator.

II. GENERAL MATHMATICAL MODEL

Consider the m-dimensional binary vectors in binary space of 2^m binary vectorsto be the address sequence A(n) = $a_m(n)$ $a_{m-1}(n)$ $a_{m-2}(n)$ … $a_2(n)$ $a_1(n)$, where $a_i(n) \in \{0,1\}$, $I \in \{1, 2, \ldots, m\}$,and $n \in \{0, 1, 2, \ldots, 2^m-1\}$. Then the problem of generating the desire address sequence can be seen as the generation of m-dimensional binary $B(n) = b_m(n)b_{m-1}(n)\ b_{m-2}(n)$ … $b_2(n)\ b_1(n)$;$b_i(n) \in \{0,1\}$, $i \in \{1, 2, \ldots, m\}$in this relation forman entire set of 2^m binary vectors. Then the vector space A(n) formed according to (1) is of dimension m and consists of 2^m vectors, this is why vectors A(n) can be used as an address sequence [18].

To simplify further investigations, the binary value of B(n) will take the value of an integer n, representing a binary notation of decimal numbers. For example in a case of m = 4 and n = 5, B(5) = $b_4(5)$ $b_3(5)$ $b_2(5)$ $b_1(5)$ = 0101. The key elements of this approach (1) to generate address sequences is basis $\{v_1, v_2, \ldots, v_m\}$, which formed the generating binary m by m matrix V. The only restricting factor for such amatrix V is it's maximal rank. A matrix V has a maximal rank if it consists out of a set of linearly independent vectors v_i [18]. For the same value of m = 4, four binary vectors v_1 = 1001, v_2 = 0100 v_3 = 0001 v_4 = 0010 are independent. That is why can be considered as the basis of binary space. For this basis, the linear combination to binary coefficients $A(5) = a_4(5)\ a_3(5)\ a_2(5)\ a_1(5) = v_1 \times b_1(5) \oplus v_2 \times b_2(5) \oplus v_3 \times b_3(5) \oplus v_4 \times b_4(5) = v_1 \oplus v_3 = 1001 \oplus 0001 = 1000$.

The linear vector combination (1) based on generating a *m* by *m* matrix *V* of linear independent vectors $v_i = v_{i1}v_{i2} \ldots v_{im}$, $v_{ij} \in \{0,1\}$ $j = \overline{1,m}$ the can be considered as the matrix V^T:

$$A(n) = \begin{vmatrix} a_1(n) \\ a_2(n) \\ \cdots \\ a_m(n) \end{vmatrix} = \begin{vmatrix} v_{11} & v_{21} & \cdots & v_{m1} \\ v_{12} & v_{22} & \cdots & v_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1m} & v_{2m} & \cdots & v_{mm} \end{vmatrix} \rightleftarrows \times \begin{vmatrix} b_1(n) \\ b_2(n) \\ \cdots \\ b_m(n) \end{vmatrix}.$$

It should be noted that matrix V^T is the transposed matrix *V* and all bit $a_i(n) \in \{0,1\}$, $I \in \{1, 2, \ldots, m\}$ of the address sequence *A*(*n*) area linear combination of the corresponding vectors v_i bit

$$a_i(n) = v_{1i} \times b_1(n) \oplus v_{2i} \times b_2(n) \oplus v_{3i} \times b_3(n) \oplus \ldots \oplus v_{mi} \times b_m(n). \quad (3)$$

Relations (2) and (3) can be used as a mathematical model for an address sequence generator, which consists of three components, namely the base address sequence generator *B*(*n*), a device for generating and storing a matrix of binary vectors V^T, and a device for actually calculating the address values *A*(*n*). Let´s review the individual componentsofthegenerator.

III. ADRESS GENERATOR IMPLEMENTATION

The goal of the proposed address sequence generator is to significantlyexpand the set of different address sequences for MBIST including the well known and extensively used sequences for MBIST.

The first block of the address sequence generator is used for the base address sequence *B*(*n*) generation. Binary counters and LFSR are mainly used to generate memory addresses that can be seuccessively applied to the memory core under test. That is why a binary counter and a LFSR can be chosen as base sequence *B*(*n*) generators. Binary counters generally use flip-flops, half adders, and a high-speed carry chain. The high-speed counter in the most applications uses a hierarchical Manchester carry chain for carry propagation [12]. The delay associated with a binary counter depends on the number of bits in the adder/carry chain circuit. In contrast, LFSR counters use only flip-flops and XOR gates [12]. The delay of LFSR with internal XOR gates is independent of the number of bits in the counter. The only problem with LFSR is the absence ofthe zero code in their output and thus it cannot generate all zero addresses. To overcome this restriction the de Bruijn sequences generator for addresses generation can be used. For the primitive polynomial $\Pi(x) = 1 \oplus x^3 \oplus x^4$ the structural scheme of the de Bruijn generator is shown in Fig. 1.

Like the standard LFSR with internal XOR gates, the above presented generator consists of successively connected *D*-type flip-flops and two-input XORs according to the used primitive polynomial. Additional (*m*−1) inputs of the NOR gate allow to generate the whole zero code [7].

All memory tests use two kinds of chosen address sequences *A*(*n*), namely, up-sequence ⇑*A*(*n*) and down-sequence ⇓*A*(*n*) which is the sequence with reverse addresses order. Then the generator of the base sequence should operate in two modes: *up* and *down*. This slightly increase the complexity of the first block. For example in a case of the de Bruijn generator the shift register have to perform shifts in both sides and additional XOR gates are needed.

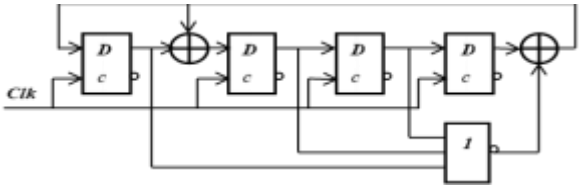


Fig. 1. De Bruijn sequence generator

The second, and as already noted the main block of the address generator is a memory device consisting of m cells, each consisting of m bits. This allows to store mm-bit binary vectors $v_i = v_{i1}v_{i2} \dots v_{im}$, $v_{ij} \in \{0,1\}$. The contents of this device, which are the vectors vi of the generating matrix V determine the form of the address sequence $A(n)$ (2). Matrix V^T is transposed matrix V, and vice-versa, V is transposed matrix V^T and both matricesmust consist of linear independent vectors. That is why the main problem is to generate and store this kind of vectors.

To drastically reduce the complexity of the memory device for generation and storing the values of mm-bit linearly independent binary vectors v_i, we proposed to use a LFSR based structure described by the primitive polynomial $\phi(x)$ with $\deg\phi(x) = m$. The main idea behind this proposal is the fact that any consecutive mm-bit LFSR states represent the set of linear independent vectors v_i [7]. The initial state of the LFSR determines all m binary vectors that can be generated based on $2m-1$ D-type flip-flops or m flip-flops and additional XOR gates. Fig. 2 shows the example of the memory device for mm-bit binary linearly independent vectors vi generation and storing for the case of primitive polynomial $\phi(x) = 1 \oplus x1 \oplus x4$.

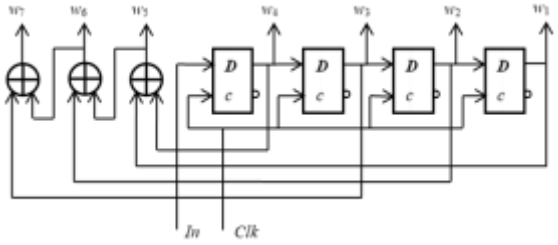


Fig. 2. Example of Memory device implementation

For the general case, the memory device (see Fig. 2) consists of m-bit shift register and additional XOR gates. The shift register is used tostore the first non-zero binary vector $v_1 = v_{11}v_{12} \dots v_{1m}$, $v_{1j} \in \{0,1\}$ $j = \overline{1,m}$. This vector sequentially is loaded into the register bit by bit, applying the current bit to the input In and clock signals to the input Clk. For example in a case is shown in Fig. 2 $v_1 = v_{11}v_{12}v_{13}v_{14} = w_4w_3w_2w_1$. The second vector v_2 is composed of the first $m-1$ bits of v_1 and one additional bit that is obtained according to the chosen primitive polynomial $\phi(x)$. For polynomial $\phi(x) = 1 \oplus x^1 \oplus x^4$ this bit w_5 is $w_5 = w4 \oplus w1$, and then $v_2 = v_{21}v_{22}v_{23}v_{24} = w_5w_4w_3w_2$. The rest of the binary vectors $v_i = v_{i1}v_{i2} \dots v_{im}$ for $i = 3, 4, \dots, m-1$ are obtained the same way as vector v_2, where in our case $v_3 = w_6w_5w_4w_3$ and $v_4 = w_7w_6w_5w_4$. Then the equation for address sequences generation (2) has the form (4).

$$A(n) = \begin{vmatrix} a_1(n) \\ a_2(n) \\ a_3(n) \\ a_4(n) \end{vmatrix} = \begin{vmatrix} w_4 & w_3 & w_2 & w_1 \\ w_5 & w_4 & w_3 & w_2 \\ w_6 & w_5 & w_4 & w_3 \\ w_7 & w_6 & w_5 & w_4 \end{vmatrix} \times \begin{vmatrix} b_1(n) \\ b_2(n) \\ b_3(n) \\ b_4(n) \end{vmatrix}. \quad (4)$$

It should be noted that proposed memory device is very easy to implement on hardware, see example Fig. 2. The hardware overhead depends only on the chosen primitive polynomial $\phi(x)$ and inmost cases consists of mD-type flip-flops and $m-1$ two inputs XOR gates. The only variations in hardware overhead can be additional XOR gates. For the general case, the new memory device allows to get the m by m matrix V of linear independent vectors v_i for structures like the memory device, implemented as a LFSR in Fig. 2. The primitive polynomial guarantees the linear independents for m consecutive states of the LFSR, and that is why it allows to avoid linear dependences for vectors v_i in the matrix V. In the case of de-Bruijn generator, the only restriction is the all zero state that reduces the number of possible generating matrix $V$to $2m - m$. A little bit complicated preparation procedure is for $2m-1$ shift register serving as memory device. The problem is that the first m bits of the register initial state $wm \dots w3\ w2\ w1$ can be chosen arbitrarily except for the zero code but the next bits can lead to linear dependencies between the vectors vi. That is why ananalyses of linear dependency should be done. Nevertheless, for both structures of the memory device, LFSR based (see Fig. 2) and $2m-1$ shift register the following, so called *Toeplitz* matrix or diagonal-constant generating matrix (5) [18] will be obtained.

$$V = \begin{vmatrix} w_m & \cdots & w_3 & w_2 & w_1 \\ w_{m+1} & \cdots & w_4 & w_3 & w_2 \\ w_{m+2} & \cdots & w_5 & w_4 & w_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{2m-1} & \cdots & w_{m+2} & w_{m+1} & w_m \end{vmatrix}. \quad (5)$$

The third and the last block of the proposed structure of the memory addresses generator is required for implementing the equation (3). This block consists of m m-input XOR gates and m^2 two-input AND gates. At the XOR gates outputs the corresponding bits $ai(n)$ from sequences$A(n)$ are obtained and two-inputs AND gates are used for m-bit binary vector multiplication (3).

Examples of such a type of AS, generated according to (4) are shown in Table 1 for the case of

memory device shown in Fig. 2 and the base address sequence $B(n)$ generation by de Bruijn generator (see Fig. 1) and Up-counter. The first address $A(1) = a_4(1)a_3(1)a_2(1)a_1(1) = 0111$ have been obtained based on the de Bruijn sequence code $B(1) = b_4(1)\ b_3(1)\ b_2(1)\ b_1(1) = 1100$ and matrix V_1 according to (4), where $a1(1) = w_4 \times b_1(1) \oplus w_3 \times b_2(1) \oplus w_2 \times b_3(1) \oplus w_1 \times b_4(1) = 1 \times 0 \oplus 0 \times 0 \oplus 1 \times 1 \oplus 0 \times 1 = 1$; $a2(1) = w_5 \times b_1(1) \oplus w_4 \times b_2(1) \oplus w_3 \times b_3(1) \oplus w_2 \times b_4(1) = 1 \times 0 \oplus 1 \times 0 \oplus 0 \times 1 \oplus 1 \times 1 = 1$; $a_3(1) = w_6 \times b_1(1) \oplus w_5 \times b_2(1) \oplus w_4 \times b_3(1) \oplus w_3 \times b_4(1) = 1 \times 0 \oplus 1 \times 0 \oplus 1 \times 1 \oplus 0 \times 1 = 1$; $a_4(1) = w_7 \times b_1(1) \oplus w_6 \times b_2(1) \oplus w_5 \times b_3(1) \oplus w_4 \times b_4(1) = 1 \times 0 \oplus 1 \times 0 \oplus 1 \times 1 \oplus 1 \times 1 = 0$. As can be seen from the Table 1, the output address sequences depends on the base address sequence, as well as the particular generating matrix V. For both matrixes V_1 and V_2 and both base addressing the different address sequences $A(n)$ have been obtained.

Table 1. Address sequences

n	$B(n)$ De Bruijn	$B(n)$ Up-counter	$V_1 = \begin{vmatrix}1&0&1&0\\1&1&0&1\\1&1&1&0\\1&1&1&1\end{vmatrix}$		$V_2 = \begin{vmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{vmatrix}$	
			$A(n)$ De Bruijn	$A(n)$ Up-counter	$A(n)$ De Bruijn	$A(n)$ Up-counter
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
1	1 1 0 0	0 0 0 1	0 1 1 1	1 1 1 1	1 0 0 1	0 0 1 0
2	0 1 1 0	0 0 1 0	0 0 1 1	1 1 1 0	1 1 0 0	0 1 0 0
3	0 0 1 1	0 0 1 1	0 0 0 1	0 0 0 1	0 1 1 0	0 1 1 0
4	1 1 0 1	0 1 0 0	1 0 0 0	1 1 0 1	1 0 1 1	1 0 0 0
5	1 0 1 0	0 1 0 1	0 1 0 0	0 0 1 0	0 1 0 1	1 0 1 0
6	0 1 0 1	0 1 1 0	0 0 1 0	0 0 1 1	1 0 1 0	1 1 0 0
7	1 1 1 0	0 1 1 1	1 0 0 1	1 1 0 1	1 1 0 1	1 1 1 0
8	0 1 1 1	1 0 0 0	1 1 0 0	1 0 1 1	1 1 1 0	0 0 0 1
9	1 1 1 1	1 0 0 1	0 1 1 0	0 1 0 1	0 0 0 0	0 0 1 1
10	1 0 1 1	1 0 1 0	1 0 1 1	0 1 0 0	0 1 1 1	0 1 0 1
11	1 0 0 1	1 0 1 1	0 1 0 1	1 0 1 1	0 0 1 1	0 1 1 1
12	1 0 0 0	1 1 0 0	1 0 1 0	0 1 1 1	0 0 0 1	1 0 0 1
13	0 1 0 0	1 1 0 1	1 1 0 1	1 0 0 0	1 0 0 0	1 0 1 1
14	0 0 1 0	1 1 1 0	1 1 1 0	1 0 1 1	0 1 0 0	1 1 0 1
15	0 0 0 1	1 1 1 1	1 1 1 1	0 1 1 0	1 0 0 0	1 1 1 1

The chosen construction of the shift register (LFSR) used in memory device shown in Fig. 2 allows to generate only $2m - 4$ different matrices$V$ composed of m m-bit linear independent vectors. For every matrix Va new address sequence $A(n)$ for constant base $B(n)$ sequence will be obtained. In the case of primitive polynomial $\prod(x) = 1 \oplus x^1 \oplus x^4$ with degree $m = 4$ and counter base $B(n)$ sequence generator there are 12 different output sequences $A(n)$.

The sufficient increasing of possible generating matrix V can be generated by the memory device design as the ordinal shift register with $2m - 1$ bits, what allow to get more different sequences $A(n)$. This register, as mentioned earlier, serves as a memory device for storing matrix V. The most common and most widely used in MBISTaddress sequences $A(n)$ [3] generatedwiththe proposed generatorare shown in Table 2.

Table 2. Mos commonly used address sequence generation

n	$B(n)$ Up-counter	*Up-counter* $\begin{vmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{vmatrix}$	*Gray Code* $\begin{vmatrix}1&1&0&0\\0&1&1&0\\0&0&1&1\\0&0&0&1\end{vmatrix}$	*Random* $\begin{vmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\0&0&0&1\end{vmatrix}$
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0	0 0 1 1	0 0 1 1
3	0 0 1 1	0 0 1 1	0 0 1 0	0 0 1 0
4	0 1 0 0	0 1 0 0	0 1 1 0	0 1 1 1
5	0 1 0 1	0 1 0 1	0 1 1 1	0 1 1 0
6	0 1 1 0	0 1 1 0	0 1 0 1	0 1 0 0
7	0 1 1 1	0 1 1 1	0 1 0 0	0 1 0 1
8	1 0 0 0	1 0 0 0	1 1 0 0	1 1 1 1
9	1 0 0 1	1 0 0 1	1 1 0 1	1 1 1 0
10	1 0 1 0	1 0 1 0	1 1 1 1	1 1 0 0
11	1 0 1 1	1 0 1 1	1 1 1 0	1 1 0 1
12	1 1 0 0	1 1 0 0	1 0 1 0	1 0 0 0
13	1 1 0 1	1 1 0 1	1 0 1 1	1 0 0 1
14	1 1 1 0	1 1 1 0	1 0 0 1	1 0 1 1
15	1 1 1 1	1 1 1 1	1 0 0 0	1 0 1 0

Up-counter (*Linear*) sequence, also called the *counting* address sequence is the first one in the set of the address sequence family [3]. For the generation of up-counter sequences formed by binary counting circuits (counters), it is necessary to form a generating matrix V with the all zeros, except forthe main diagonal, like it is shown in Table 2.

To minimize stresses during memory testing, sequences with minimal switching activity are used, mainly the set of *Gray Code* sequences [3]. An example of such a type of sequence is shown in Table 2.

Based on the proposed mathematical model a FPGA implementation was made. On Fig.3 simulation for address sequences from table 1. Fig.4 shows the result of it'sRTL synthesis.

Fig. 3.

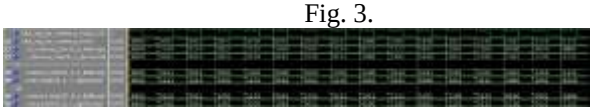
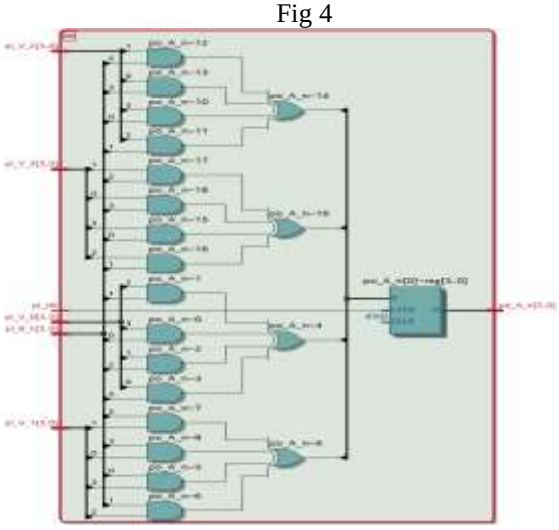


Fig 4



IV. CONCLUSION

Though this work, a new architecture for an address generator which occupies major part of modern BIMSThas been introduced. The main goal behind the proposed address sequence generation method is the significant expansion of the set of different address sequences including the standard well known and extensively used sequences for MBIST. The lower bound of number of address sequences can be estimated by the value $2m - m$, andthe upper bound does not exceed $22m-1$. The peculiar properties of the generation of the*Toeplitz* matrix (5), allows to obtain address sequences with different characteristics and properties.

REFERENCES

[1] International Technology Roadmap for Semiconductors, Test and Test Equipment, 2015.

[2] Bushnell, M. L. Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits / M.L. Bushnell, V.D. Agrawal. – New York : Kluwer Academic Publishers, 2000. – 690 p.

[3] Goor, A.J. Optimizing memory BIST Address Generator implementations / A.J. Goor, H. Kukner, S. Hamdioui // Proc. of 2011 6th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), Athens, Greece. – 2011. – P. 572–576.

[4] Marinissen, E.J. Challenges in Embedded Memory Design and Test / E.J. Marinissen, B. Prince, D. Keitel-Schulz, Y. Zorian // Proc. of Design, Automation and Test in Europe Conference and Exhibition, Munich, Germany. – 2005. – P. 722–727.

[5] Aswin, A.M. Implementation and Validation of Memory Built in Self-Test (MBIST) –Survey / A.M. Aswin, S.S. Ganesh // International Journal of Mechanical Engineering and Technology (IJMET). – 2019. – Vol. 10, № 3. – P. 153–160.

[6] Golomb, S.W. Shift Register Sequences / S.W. Golomb. – San Francisco : Holden-Day, Inc., 1967. – 224 p.

[7] Yarmolik, V.N. Generation and application of pseudorandom sequences for random testing / V.N. Yarmolik, S.N. Demidenko. – New York,NY,USA:John Wiley & Sons, Inc., 1988. – 167 c.

[8] Mohan, M. Review on LFSR for Low Power BIST / M. Mohan, S.S.A. Pillai // Proceedings of 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India. – 2019. – P. 788-454.

[9] Hellebrand, S. A mixed mode BIST scheme based on reseeding of folding counters / S. Hellebrand, H.G. Liang,H-J. Wunderlich // Journal of Electronic Testing. – 2001. – №17. – P. –341–349.

[10] Ren,H. A Multi-polynomial LFSR Based BIST Pattern Generator for Pseudorandom Testing / H. Ren Z.Xiong// Proceedings of 2nd International Conference on Information Science and Control Engineering, Shanghai, China. – 2015. – P. 788-454.

[11] 15. Vennelakanti, S. Design and Analysis of Low Power Memory Built in Self-Test Architecture for SoC based Design / S. Vennelakanti, S. Saravanan // Indian Journal of Science and Technology. – 2015. – Vol 8, №14. – P. 1–5.

[12] Ajane, A. Comparison of binary and LFSR counters and efficient LFSR decoding algorithm / A. Ajane, P.M. Furth, E.E. Johnson // Proceedings IEEE 54th International Midwest Symposium Circuits Systems (MWSCAS), Seoul, Korea. – 2011. – P. 1–4.

[13] Morrison, D. Multistage Linear Feedback Shift Register Counters With Reduced Decoding Logic in 130-nm CMOS for Large-Scale Array Applications / D. Morrison, D. Delic, M.R. Yuce, J.-M. Redouté // IEEE Transaction on Very Large Scale Integration (VLSI) Systems. – 2019. – Vol 27, №1. – P. 103–115.

[14] Kumar, S. Efficient Memory Built in Self-Test Address Generator Implementation / S. Kumar, M. Rajkumar // International Journal of Applied Engineering Research. – 2015. – Vol 10, № 7. – P. 16797–16813.

[15] Saravanan, S. Design and Analysis of Low-Transition Address Generator / S. Saravanan, M. Hailu, G.M. Gouse, M. Lavanya, R. Vijaysai // Proc. of 6th EAI International Conference, ICAST, Bahir Dar, Ethiopia. – 2018. – P.

[16] Singh, B. Address Counter / Generators for Low Power Memory BIST / B. Singh, S. B. Narang, A. Khosla // IJCSI International Journal of Computer Science Issues. – 2011. – Vol. 8, Issue 4, № 1. – P. 561–567.

[17] Awad, A.N. Low Power Address Generator for Memory Built-In Self-Test / A.N. Awad, A.S. Abu-Issa //The Research Bulletin of Jordan ACM. – 2011. – Vol. II (III). – P. 52–56

[18] Boyd, S. Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares / S. Boyd. – Cambridge, United Kingdom :University Printing House, 2018. – 463 p.

[19] Yarmolik, V.N. Generating Modified Sobol Sequences for Multiple Run March Memory Tests / V.N. Yarmolik, S.V. Yarmolik // Automatic Control and Computer Sciences. – 2013. –Vol. 47, № 5. – P. 242–247.

[20] Chen, T.Y. Quasi random testing / T.Y. Chen, R. Merkel // IEEE Trans. Reliability. – 2007. – Vol. 56, №3. – P. 562–568