ISSN 0002-306X. Изв. НАН РА и ГИУА. Сер. ТН. 2010. Т. LXIII, № 3.

UDC 681.3.07 COMPUTER SCIENCE AND INFORMATICS

**G.E. HARUTYUNYAN, D.V. MELKUMYAN**

# FAULT LOCATION AND DIAGNOSIS ALGORITHM FOR STATIC AND DYNAMIC FAULTS IN SRAMS

A multiphase March algorithm of complexity 74N+C, where N is the number of memory words, C is a constant, for location and diagnosis of static and dynamic faults in Static Random Access Memories (SRAM) proposed. Previously fault location and diagnosis algorithms were proposed for static faults or dynamic faults only. This algorithm locates the failed bits and diagnoses the fault types within the space of all simple (unlinked) static faults, as well as of a special set of two-operation dynamic faults.

***Keywords:*** static fault, dynamic fault, March test, detection, location, diagnosis.

**1. INTRODUCTION.** The problems of fault detection, location and diagnosis in SRAMs (see [1]) are of prime importance in connection with the increasing density of embedded memories and their dominating portion in system-on-chips (SoCs). In addition to the previously known static functional fault models (FFMs) [1], new dynamic FFMs based on Inductive Fault Analysis (IFA) (see [2]) were introduced.

In [3], a method of partial diagnosis with further fault location was proposed for the traditional FFMs. In [4], an efficient March-based fault location and full diagnosis algorithm was proposed only for dynamic FFMs in bit-oriented SRAMs.

In the paper, we present a March algorithm of complexity 74N+C for location and diagnosis of both static and dynamic faults, where N is the number of memory words, C is the constant. The general algorithm consists of three phases. In phase 1, sequentially 3 March algorithms are applied for fault detection and partial diagnosis: March VLP1, VLP2 and VLP3 of overall complexity 74N. Phases 2 and 3 are optional and are intended for fault diagnosis and location. In phase 2, a diagnosis / location algorithm of constant complexity 7…325 is applied. In phase 3, algorithms of complexity 17…181 should be applied. It should be noted that for some faults not all the three phases are needed.

**2. MAIN NOTATIONS.** The definition of the fault primitive (FP) concept used to define the static and dynamic faults can be found in [2], [5]. A March algorithm $M$ is a test algorithm with a finite number of March elements $M=\{M_1, M_2, \ldots, M_k\}$ where each March element $M_i$ consists of an addressing direction and a finite number of Read/Write operations (see [1]).

For a given memory failure, <S/F/R> denotes a FP where S denotes a sensitizing operation sequence, i.e. an operation sequence that results in a difference between the observed and the expected memory behavior. The faulty behavior F is the observed memory behavior that deviates from the expected one, and $R \in \{0, 1, -\}$ is the result of the Read operation of S applied to the faulty cell in case S ends with operation Read. A "-" in R means the output data is not applicable. Table 1 describes all single-cell static FFMs (see [5]). Note

that the symbol "~" used in Tables 1-5 denotes logical negation, and x, y, z, t ∈ {0, 1}. Table 2 describes all two-cell static FFMs.

In [2], a total of 30 two-operation single-cell dynamic FPs were described and compiled into a set of five FFMs (see Table 3). A two-operation 2-cell dynamic FP can be represented as $<S_a; S_v/F/R>$ where $S_a$ (respectively, $S_v$) describes the sequence of operations applied to the aggressor (victim) cell "a" ("v") or its state. A dynamic FP is sensitized by applying two operations sequentially to the aggressor or victim cell. Depending on the number of operations applied to the a-cell and to the v-cell, and on the order in which they are applied, four types of S can be distinguished:

1. $S_{aa}$; the two sequential operations are applied to the a-cell, the v-cell is in a certain state.
2. $S_{vv}$; the two sequential operations are applied to the v-cell, the a-cell is in a certain state.
3. $S_{av}$; the first operation is applied to the a-cell, followed immediately with a second one to the v-cell.
4. $S_{va}$; the first operation is applied to the v-cell, followed immediately with a second one to the a-cell.

We will consider only classes of FPs caused by $S_{aa}$ and $S_{vv}$ since, as noted in [6], it is impossible to test the faults from $S_{av}$ and $S_{va}$ by means of March algorithms, in general. Table 4 describes all FPs caused by $S_{aa}$. Table 5 describes all FPs caused by $S_{vv}$.

For a given March algorithm, the corresponding dictionary of fault syndromes is constructed in the following way (see [3]): each row (respectively, column) of the dictionary corresponds to a certain fault class (respectively, a Read operation from the March test). If the March test contains r Read operations, then $<s_i(R_0), s_i(R_1), \ldots, s_i(R_{r-1})>$ is the signature of the $i^{th}$ fault named March syndrome, where $s_i(R_k)=0$ (respectively, 1), if the $k^{th}$ Read operation of the March test returns to a fault-free (respectively, faulty) value.

TABLE 1. SINGLE-CELL STATIC FFMS

| FFM | Fault primitives |
|---|---|
| SF | <x/~x/-> |
| TF | <xW(~x)/x/-> |
| WDF | <xWx/~x/-> |
| RDF | <Rx/~x/~x> |
| DRDF | <Rx/~x/x> |
| IRF | <Rx/x/~x> |

TABLE 2. TWO-CELL STAIC FFMS

| FFM | Fault primitives |
|---|---|
| CFst | <x; y/~y/-> |
| CFds | <Rx; y/~y/->, <xWy; z/~z/-> |
| CFtr | <x; yW(~y)/y/-> |
| CFwd | <x; yWy/~y/-> |
| CFrd | <x; Ry/~y/~y> |
| CFdrd | <x; Ry/~y/y> |
| CFir | <x; Ry/y/~y> |

TABLE 3. SINGLE-CELL DYNAMIC FPS

| FFM | Fault primitives |
|---|---|
| dRDF | <xWyRy/~y/~y>, <xRxRx/~x/~x> |
| dIRF | <xWyRy/y/~y>, <xRxRx/x/~x> |
| dDRDF | <xWyRy/~y/y>, <xRxRx/~x/x> |
| dTF | <xWyW(~y)/y/->, <xRxW(~x)/x/-> |
| dWDF | <xWyWy/~y/->, <xRxWx/~x/-> |

TABLE 4. FPS FROM $S_{AA}$

| FFM | Fault primitives |
|---|---|
| dCF$ds_{ww}$ | <xWyWt; z/~z/-> |
| dCF$ds_{wr}$ | <xWyRy; z/~z/-> |
| dCF$ds_{rw}$ | <xRxWy; z/~z/-> |
| dCF$ds_{rr}$ | <xRxRx; z/~z/-> |

TABLE 5. FPS FROM $S_{VV}$

| FFM | Fault primitives |
|---|---|
| dCFrd | <x; yWzRz/~z/~z>, <x; zRzRz/~z/~z> |
| dCFdrd | <x; yWzRz/~z/z>, <x; zRzRz/~z/z> |
| dCFir | <x; yWzRz/z/~z>, <x; zRzRz/z/~z> |
| dCFtr | <x; yWzW(~z)/z/->, <x; zRzW(~z)/z/-> |
| dCFwd | <x; yWzWz/~z/->, <x; zRzWz/~z/-> |

**3. THE PROPOSED ALGORITHM.** The algorithm consists of 3 phases. Phase 1 is for fault detection and partial diagnosis. After this phase all considered faults are detected. Using March syndromes obtained from phase 1 faults are grouped. Two faults are in the same group if their corresponding syndromes are the same. Phases 2 and 3 are for fault diagnosis and location. Note that the algorithm can apply only the first phase or only the first and the second phases for specific fault groups. But there are fault groups that require all three phases. If a syndrome is obtained that is out of our list, the algorithm stops its work with a message "Unknown fault". It is natural to obtain such "unknown" syndromes, since we do not consider all the space of realistic faults.

Table 6 presents the three March test algorithms that are used in phase 1. These algorithms must be applied sequentially (first March VLP1, then March VLP2, then March VLP 3) and join the obtained March syndromes. The overall complexity of phase 1 is 74N.

Diagnostic algorithms are listed in Table 7, location algorithms in Table 8. Algorithm LD1 from Table 8 additionally has also diagnosis capability. In Tables 7 and 8, x, y, z $\in$ {0, 1}, "$W_V$" / "$R_V$" ("$W_A$" / "$R_A$") means Write / Read operation applied to the victim (aggressor) cell. In these algorithms the aggressor cell (A) is considered that it is physically placed on the left (L), up (U), right (R) or down (D) side of the victim cell. "For each A cell" means apply all $W_A$ and $R_A$ operations sequentially first on cell L, then on cells U, R and D. "$W_L$" means Write operation to the L cell only.

TABLE 6. PHASE 1 ALGORITHMS

| Name | Description |
|---|---|
| March VLP1 | ⇑(W0); ⇑(R0, W1, W1, R1, W1, W1); ⇑(R1, W0, W0, R0, W0, W0);<br>⇓(R0, W1, W1, R1, W1, W1); ⇓(R1, W0, W0, R0, W0, W0); ⇑(R0). |
| March VLP2 | ⇑(W0); ⇑(R0, W1, R1, W1, R1, R1); ⇑(R1, W0, R0, W0, R0, R0);<br>⇓(R0, W1, R1, W1, R1, R1); ⇓(R1, W0, R0, W0, R0, R0); ⇑(R0). |
| March VLP3 | ⇑(W0); ⇑(R0, W0, W1, W0, W1); ⇑(R1, W1, W0, W1, W0);<br>⇓(R0, W0, W1, W0, W1); ⇓(R1, W1, W0, W1, W0); ⇑(R0). |

TABLE 7. DIAGNOSIS ALGORITHMS

| Name | Description | Length |
|---|---|---|
| D1(x) | $W_V(\sim x)$, $W_V(x)$, $R_V(x)$, $W_V(\sim x)$, $R_V(\sim x)$, $W_V(x)$, $R_V(x)$ | 7 |
| D2(x) | $W_V(\sim x)$, $W_V(x)$, $R_V(x)$, $R_V(x)$, $W_V(x)$, $W_V(x)$, $R_V(x)$, $R_V(x)$, $W_V(x)$, $R_V(x)$, $W_L(x)$, $R_V(x)$ | 12 |
| D3 | $W_V(0)$, $W_V(1)$, $W_V(0)$, $R_V(0)$, $W_V(1)$, $W_V(1)$, $W_V(0)$, $R_V(0)$ | 8 |
| D4 | {$W_V(1)$}; {For each A cell: $W_A(0)$}; {$R_V(1)$, $W_V(0)$, $W_V(1)$, $W_V(1)$, $R_V(1)$, $R_V(1)$, $W_V(0)$, $W_V(0)$, $R_V(0)$, $R_V(0)$, $W_V(1)$, $R_V(1)$, $R_V(1)$, $W_V(1)$, $R_V(1)$, $W_V(0)$, $R_V(0)$, $R_V(0)$, $W_V(0)$, $R_V(0)$}; {For each A cell: $W_A(1)$}; {$R_V(0)$};<br>{For each A cell: $W_A(0)$}; {$R_V(0)$, $W_V(1)$, $W_V(0)$}; {For each A cell: $W_A(1)$}; {$R_V(0)$, $W_V(1)$, $W_V(0)$, $W_V(0)$, $R_V(0)$, $R_V(0)$, $W_V(1)$, $W_V(1)$, $R_V(1)$, $R_V(1)$, $W_V(0)$, $R_V(0)$, $R_V(0)$, $W_V(0)$, $R_V(0)$, $W_V(1)$, $R_V(1)$, $R_V(1)$, $W_V(1)$, $R_V(1)$}; {For each A cell: $W_A(0)$}; {$R_V(1)$}; {For each A cell: $W_A(1)$}; {$R_V(1)$} | 71 |
| D5 | {$W_V(0)$}; {For each A cell: $W_A(0)$}; {$W_V(0)$, $W_V(1)$, $R_V(1)$, $W_V(0)$, $W_V(1)$, $R_V(1)$, $W_V(1)$, $W_V(1)$, $R_V(1)$, $R_V(1)$, $R_V(1)$, $W_V(1)$, $W_V(0)$, $R_V(0)$, $W_V(1)$, $W_V(0)$, $R_V(0)$, $W_V(0)$, $W_V(0)$, $R_V(0)$, $R_V(0)$, $R_V(0)$}; {For each A cell: $W_A(1)$}; {$W_V(0)$, $W_V(1)$, $R_V(1)$, $W_V(0)$, $W_V(1)$, $R_V(1)$, $W_V(1)$, $W_V(1)$, $R_V(1)$, $R_V(1)$, $R_V(1)$, $W_V(1)$, $W_V(0)$, $R_V(0)$, $W_V(1)$, $W_V(0)$, $R_V(0)$, $W_V(0)$, $W_V(0)$, $R_V(0)$, $R_V(0)$, $R_V(0)$}; {For each A cell: $W_A(0)$ , $W_V(0)$, $W_A(0)$, $W_A(0)$, $R_V(0)$, $W_A(1)$, $W_A(0)$, $R_A(0)$, $R_V(0)$, $W_A(0)$, $W_A(1)$, $R_V(0)$, $W_A(0)$, $W_A(0)$, $R_A(0)$, $R_A(0)$, $R_V(0)$, $W_A(1)$ , $W_V(0)$, $W_A(1)$, $W_A(1)$, $R_V(0)$, $W_A(0)$, $W_A(1)$, $R_A(1)$, $R_V(0)$, $W_A(1)$, $W_A(0)$, $R_V(0)$, $W_A(1)$, $W_A(1)$, $R_A(1)$, $R_A(1)$, $R_V(0)$}; {For each A cell: $W_A(0)$, $W_V(1)$, $W_A(0)$, $W_A(0)$, $R_V(1)$, $W_A(1)$, $W_A(0)$, $R_A(0)$, $R_V(1)$, $W_A(0)$, $W_A(1)$, $R_V(1)$, $W_A(0)$, $W_A(0)$, $R_A(0)$, $R_A(0)$, $R_V(1)$, $W_A(1)$, $W_V(1)$, $W_A(1)$, $W_A(1)$, $R_V(1)$, $W_A(0)$, $W_A(1)$, $R_A(1)$, $R_V(1)$, $W_A(1)$, $W_A(0)$, $R_V(1)$, $W_A(1)$, $W_A(1)$, $R_A(1)$, $R_A(1)$, $R_V(1)$} | 325 |

TABLE 8. LOCATION ALGORITHMS

| Description | Length |
|---|---|
| {For each A cell: $W_A(x)$}; {$W_V(z)$}; {For each A cell: $W_A(y)$, $R_A(y)$, $R_V(z)$} | 17 |
| {For each A cell: $W_A(1)$}; {$W_V(x)$}; {For each A cell: $W_A(0)$, $W_V(0)$, $R_V(0)$} | 17 |
| {For each A cell: $W_A(0)$}; {$W_V(0)$}; {For each A cell: $W_A(0)$, $W_A(1)$, $R_V(0)$} | 17 |
| {For each A cell: $W_A(\sim x)$}; {For each A cell: $W_A(x)$, $W_V(\sim y)$, $W_V(y)$, $W_V(y)$, $R_V(y)$, $R_V(y)$, $R_V(y)$, $W_V(y)$, $W_V(y)$, $R_V(y)$, $W_V(y)$, $W_V(\sim y)$, $W_V(y)$, $W_V(\sim y)$, $R_V(\sim y)$, $W_V(y)$, $R_V(y)$, $R_V(y)$, $W_V(y)$, $R_V(y)$)} | 84 |
| {$W_V(y)$}; {For each A cell: $W_A(\sim x)$, $W_A(x)$, $W_A(x)$, $W_A(x)$, $R_A(x)$, $R_A(x)$, $W_A(x)$, $W_A(\sim x)$, $W_A(x)$, $W_A(\sim x)$, $W_A(x)$, $R_A(x)$, $R_V(y)$} | 53 |
| {For each A cell: $W_A(0)$}; {$W_V(x)$}; {For each A cell: $W_A(0)$, $W_A(1)$, $R_A(1)$, $R_V(x)$, $W_V(x)$, $W_A(0)$, $W_A(1)$, $R_V(x)$, $W_A(1)$, $W_A(0)$, $R_A(0)$, $R_V(x)$, $W_V(x)$, $W_A(1)$, $W_A(0)$, $R_V(x)$, $W_A(1)$, $W_A(1)$, $R_A(1)$, $R_V(x)$, $W_A(1)$, $W_A(1)$, $R_V(x)$, $W_A(0)$, $W_A(0)$, $R_A(0)$, $R_V(x)$, $W_A(0)$, $W_A(0)$, $R_V(x)$, $R_A(0)$, $R_A(0)$, $W_A(0)$, $R_V(x)$, $R_A(0)$, $W_A(1)$, $R_V(x)$, $R_A(1)$, $R_A(1)$, $W_A(1)$, $R_V(x)$, $R_A(1)$, $W_A(0)$, $R_V(x)$} | 181 |

The aggressor cell is found in the following way: when operation $R_V$ returns a faulty value at the first time, then the current A-cell (L, U, R, D) is considered as an aggressor cell. The maximum length of the diagnosis/location algorithms is 325 which is a constant number and does not depend on the number of memory cells. Note that the syndromes of phase 2 are constructed only by the results of operations $R_V$ since operation $R_A$ is used only for fault sensitization but not for fault detection.

Due to the limitation of the paper, we do not bring here all the cases describing which algorithm should be used in the next phase for each obtained March syndrome. But for each specific case we will bring an example to explain how the algorithm works. Fault groups obtained after phase 1 can be divided into 5 types:

Type 1. The groups contain only one single-cell fault. For these groups further phases are not needed since they are already located and diagnosed. For example, syndrome (10011001110000111110000111110101) corresponds to the FP <0/1/->. Thus if this syndrome is obtained in phase 1, then the FP is diagnosed and located.

Type 2. The groups contain more than one fault but they are only single-cell faults. For these groups only a diagnosis algorithm is needed since the faulty cell is already located. For example, if the syndrome (00000000000000010000000100000000) is obtained in phase 1, then either FP <1W0R0/0/1> or FP <1R1W0/1/-> is detected. Since both of them are single-cell faults, then there is no aggressor cell to locate and only diagnosis algorithm should be applied. For this group D1(0) algorithm should be used in phase 2. If the syndrome of D1(0) is (101), then it means that FP <1W0R0/0/1> is diagnosed, if (001), then FP <1R1W0/1/-> is diagnosed.

Type 3. The groups contain only one fault which is a two-cell fault. For these groups only a location algorithm is needed since the fault is already diagnosed. For example, if (10000001010000000000000111010000) syndrome is obtained in phase 1, it means that the FP <1;0/1/-> is detected. For this FP a location algorithm is needed to locate the aggressor cell. For this purpose L1(0,1,0) algorithm should be used in phase 2.

Type 4. The groups contain more than one fault and at least one of them is a two-cell fault. For these faults, location and diagnosis algorithms are needed. For example, if syndrome (00100010000001000000001000000010) is obtained in phase 1, it means that

either FP <0R0;1/0/-> or FP <1R1;1/0/-> is detected. Thus we should diagnose in the next step which FP exactly from these 2 in the memory occurred. Since the group contains at least one coupling fault, then the location of the aggressor cell should be detected as well. Using LD1(1) algorithm we will diagnose the fault as well as the aggressor cell. If the syndrome obtained from algorithm LD1(1) is (001000111111), then FP <0R0;1/0/-> is diagnosed. If the algorithm returns syndrome (100011111111), then FP <1R1;1/0/-> is diagnosed.

Type 5. The groups contain more than one fault and at least one of them is a two-cell fault. This type is the same as Type 4, but for these faults two further phases are needed, one for diagnosis, and the second for location. For example, if syndrome (00100100000001000011100000001000) is obtained in phase 1, it means that either FP <0;1/0/-> or FP <1;1/0/-> is detected. In phase 2, algorithm D4 should be applied. If syndrome (11100111000000000000000010) is obtained in phase 2, it means FP <0;1/0/-> is diagnosed. If syndrome (00000000000000001100011101) is obtained in phase 2, then it means that FP <1;1/0/-> is diagnosed. In phase 2 we diagnosed the fault but not the aggressor cell position. That is why we need one additional phase. In phase 3, algorithm L4(0, 0) should be used if FP <0;1/0/-> is diagnosed in phase 2, and the algorithm L4(1, 0) should be used if FP <1;1/0/-> is diagnosed in phase 2.

**4. EXPERIMENTAL RESULTS.** For our experiments we have used Virage Logic Yield Accelerator tool that allows to create test patterns that can be applied to the chip in the automatic test equipment (ATE) environment. It can process tester output files providing detection, location and diagnosis of faults.

We injected two coupling faults on a memory of the size 16 rows x 16 columns. The static fault <1W0;0/1/-> was injected in the position (victim cell = (5,1), the aggressor cell = (6,1)). The second injected fault was dynamic fault <0;1R1R1/1/0>. It was injected on the position (victim cell = (2,5), aggressor cell = (2,4)). The expected information was the following:

- Detect FP <1W0;0/1/-> on (5,1), (6,1) position. It belongs to a group of Type 3.
- Detect FP <0;1R1R1/1/0> on positions (2,5), (2,4). It belongs to a group of Type 5.

Then we ran the algorithm and it detected both faults with the following message:

```
Errors were detected:
Memory, Fault, Victim Row, Victim Col, Aggr. Row, Aggr. Col
dpram_lt[1], <1W0;0/1/->, 5, 1, 6, 1
dpram_lt[1], <0;1R1R1/1/0>, 2, 5, 2, 4
```

FP <1W0;0/1/-> was detected and diagnosed in phase 1. For fault location, the algorithm L1(1,0,0) was used in phase 2. We can see that this fault indeed belongs to a group of Type 3. FP <0;1R1R1/1/0> was detected in phase 1, diagnosed in phase 2 and located in phase 3. We can see that this FP belongs to a group of Type 5. This experiment proves the validity and the correctness of the proposed algorithm.

**5. CONCLUSIONS.** A new algorithm for static and dynamic fault location and diagnosis is presented. The general algorithm consists of three phases. In phase 1 we apply sequentially 3 March algorithms for fault detection and partial diagnosis of overall complexity 74N, N is the number of memory words. Phases 2 and 3 are optional and are intended for fault diagnosis and location. In phases 2 and 3 diagnosis/location algorithms of constant complexity are applied.

## REFERENCES


1. **Van de Goor A.J.** Testing semiconductor memories: Theory and Practice, John Wiley & Sons, 1991.
2. **Hamdioui S., Gaydadjiev G. N., Van de Goor A.J.** A Fault Primitive Based Analysis of Dynamic Memory Faults // IEEE 14th Annual Workshop on Circuits, Systems and Signal Processing. - Veldhoven, the Netherlands, 2003. - P. 84-89.
3. **Vardanian V.A., Zorian Y.** A March-based Fault Location Algorithm for Static Random Access Memories // In Proc. of IEEE Int. Workshop MTDT, 2002. - P. 62-67.
4. **Harutunyan G., Vardanian V.A., Zorian Y.** An Efficient March-Based Three-Phase Fault Location and Full Diagnosis Algorithm for Realistic Two-Operation Dynamic Faults in Random Access // In Proc. of IEEE VLSI Test Symposium.- 2008. - P. 95 – 100.
5. **Hamdioui S., Van de Goor A.J., Rodgers M.** March SS: a test for all static simple faults // Records of IEEE Int. Workshop MTDT. – 2002. - P. 95-100.
6. **Harutunyan G., Vardanian V.A., Zorian Y.** Minimal March tests for detection of dynamic faults in random access memories // JETTA. - February 2007. - Vol. 23, N. 1 - P. 55-74.


YSU. The material is received 01.08.2010.

Գ.Է. ՀԱՐՈՒԹՅՈՒՆՅԱՆ, Դ.Վ. ՄԵԼՔՈՒՄՅԱՆ

# ՍՏԱՏԻԿ ՆԱՆՈՄԵՏՐԱՆՈՑ ՀԻՇՈՂ ՍԱՐՔԵՐՈՒՄ ՍՏԱՏԻԿ ԵՎ ԴԻՆԱՄԻԿ ԱՆՍԱՐՔՈՒԹՅՈՒՆՆԵՐԻ ՏԵՂԱՅՆԱՑՄԱՆ ԵՎ ԱԽՏՈՐՈՇՄԱՆ ԱԼԳՈՐԻԹՄ

Ներկայացված է բազմափուլանոց 74N+C բարդությամբ մարշ ալգորիթմ ստատիկ նանոմետրանոց հիշող սարքերում ստատիկ և դինամիկ անսարքությունների տեղայնացման և ախտորոշման համար, որտեղ N-ը հիշողության բառերի քանակն է իսկ C-ն՝ հաստատուն թիվ: Նախկինում հայտնի անսարքությունների տեղայնացման և ախտորոշման ալգորիթմները եղել են միայն ստատիկ կամ միայն դինամիկ անսարքությունների համար: Այս ալգորիթմը տեղայնանում է՝ հիշող սարքի անսարք բիթը և ախտորոշում անսարքության տիպը բոլոր պարզ (չկապակցված) ստատիկ անսարքությունների և երկու գործողությամբ դինամիկ անսարքությունների հատուկ ենթաբազմության համար:

***Առանցքային բառեր.*** ստատիկ անսարքություն, դինամիկ անսարքություն, մարշ թեստ, հայտնաբերում, տեղայնացում, ախտորոշում:

Г.Е. АРУТЮНЯН, Д.В. МЕЛКУМЯН

# АЛГОРИТМ ДЛЯ ЛОКАЛИЗАЦИИ И ДИАГНОСТИКИ СТАТИЧЕСКИХ И ДИНАМИЧЕСКИХ ДЕФЕКТОВ В СТАТИЧЕСКИХ ЗАПОМИНАЮЩИХ УСТРОЙСТВАХ С ПРОИЗВОЛЬНОЙ ВЫБОРКОЙ

Представлен многофазовый марш-алгоритм для локализации и диагностики статических и динамических дефектов в статических запоминающих устройствах с произвольной выборкой со сложностью 74N+C, где N - число слов в запоминающем устройстве, а C - константа. Ранее представленные алгоритмы локализации и диагностики были только для статических или динамических дефектов. Представленный алгоритм локализирует дефектный бит и диагностирует тип дефекта для всех примитивных (несвязанных) статических дефектов и для специальной подгруппы двух операционных динамических дефектов.

***Ключевые слова:*** статический дефект, динамический дефект, марш-тест, обнаружение, локализация, диагностика.