ՀԱՅԱՍՏԱՆԻ ԳԻՏՈՒԹՅՈՒՆՆԵՐԻ ԱԶԳԱՅԻՆ ԱԿԱԴԵՄԻԱ
НАЦИОНАЛЬНАЯ АКАДЕМИЯ НАУК АРМЕНИИ
NATIONAL ACADEMY OF SCIENCES OF ARMENIA
ДОКЛАДЫ ԶԵԿՈՒՅՑՆԵՐ REPORTS

110 2010 № 2

INFORMATICS

УДК 519.688

H. S. Avetisyan[1], G. E. Harutyunyan[2], V. A. Vardanian[2]

Minimal March Test Algorithms for Detection of All Realistic Two-Operation, Two-Cell Dynamic Faults from Subclasses S_{av} and S_{va}

(Submitted by academician S.K.Shoukourian 5/III 2010)

Keywords: *dynamic fault, march test, aggressor cell, victim cell*

1. Introduction. New memory technologies and processes introduce new defects that significantly impact on the defect-per-million (DPM) level and yield. Currently for memory testing March tests are mainly used, because they have linear complexity [1]. This paper introduces minimal March test algorithms for detection of "realistic" faults from the well known subclasses S_{av} and S_{va} of two-operation dynamic faults. Earlier, only subclasses S_{aa} and S_{vv} were considered by a few authors (see [2]). In this paper it is shown that the proposed March test algorithms detect all realistic faults (to be defined below) of subclasses S_{av} and S_{va}, and have minimum length with respect to the number of memory words.

2. Definitions and Notations. In [2] the subclass S_{av} of dynamic faults is described. This subclass assumes that operation on the victim cell is performed after applying the first sensitizing operation on the aggressor cell. The article contains also the description of subclass S_{va} of dynamic faults, where operation on the aggressor cell is performed after applying the first sensitizing operation on the victim cell.

As noted in [3-5], we cannot use March tests for detection of these classes of functional fault models (FFMs) without the knowledge of the scramble information (see [6]), because we need to do an operation on the victim cell just after the operation applied on the aggressor cell, and vice versa. But there can be cases when the victim and aggressor cells have not adjacent logical addresses. Due to the technology specifics, usually the coupling faults occur between two physically adjacent cells. So, below we consider the following aggressor-victim physical cell

positions for faults of subclasses S_{av} (Pi) and S_va (Qi):

P1.	Aggressor cell - (i, j), victim cell - (i+1, j);	P2.	Aggressor cell - (i, j), victim cell - (i-1, j);
P3.	Aggressor cell - (i, j), victim cell - (i, j+1);	P4.	Aggressor cell - (i, j), victim cell - (i, j-1).

Q1.	Victim cell - (i, j), aggressor cell - (i+1, j);	Q2.	Victim cell - (i, j), aggressor cell - (i-1, j);
Q3.	Victim cell - (i, j), aggressor cell - (i, j+1);	Q4.	Victim cell - (i, j), aggressor cell - (i, j-1).

i, $0 \leq i \leq n-1$, is the physical row number of the memory, j, $0 \leq j \leq m-1$, is the physical column number of the memory that can be considered as an $m \times n$ array with n (respectively, m) rows (columns).

Based on the scramble information, the March test should be run by physical addresses to be able to test physically adjacent pairs of aggressor and victim cells that are assumed to be the realistic positions of dynamic two-cell, two-operation faults. Thus, we consider the following 4 types of physical addressing: A1. Top to down – "increasing fast row"; A2. Down to top - "decreasing fast row"; A3. Left to right - "increasing fast column"; A4. Right to left - "decreasing fast column".

The proposed test algorithms should be run for these 4 cases to detect all 4 cases of aggressor-victim positions. Note that Ai addressing should be used for detection of cases Pi and Qi. It is easy to check that using Ai addressing the March test cannot detect any fault from cases Pj or Qj, when $i \neq j$. So, if a minimal March test M is proposed for the fixed direction then the overall March test algorithm (that applies March test M for 4 directions) will be again minimal.

3. March test algorithm for subclass S_{av}. Table 1 presents March test MMSAV that detects all realistic faults from subclass S_{av}. The complexity of the algorithm is 109N for a fixed direction and the overall complexity is 436N. The algorithm created was based on idea that the first operation in March element is going to sensitize the fault, the second to detect, and the last to sensitize an aggressor cell. For example for the fault (1W1; 0R0/1/0) initialization of the victim cell is done by M5-1 operation (the first operation in March element M5). This operation sets the value of the victim cell to 0. Then the algorithm runs March element M6, where the first operation M6-1 is used to sensitize the victim cell, M6-2 to detect the fault. The third operation of the March element M6-3 sets the value of the aggressor cell to 1 to provide the needed value 1 for sensitization which is done by operation M6-4.

Theorem. *March test MMSAV is a minimal March test for detection of all realistic faults from subclass S_{av}.*

Proof. Let us evaluate the complexity of the minimal March test algorithms for subclass S_{av}. We will not consider here FFMs dCFrd and dCFir since it is easy to check that if the March test detects dCFdrd then it detects also dCFrd and

dCFir. That is why we will consider here only FFMs dCFdrd, dCFtr and dCFwd that contain in total 36 fault primitives [2-4].

Step 1: The minimal March test algorithm should perform an initialization operation to the memory cells, so one Write operation (W_i) for initialization is mandatory.

Step 2: Taking into account the faults of Subclass S_{av} [3], it is easy to check that 24 sensitizing Write operations should be performed to the aggressor cell (W_a) and correspondingly 24 sensitizing Write operations to the victim cell (W_v).

Table 1. Minimal march test MMSAV (109N)

March elements	Element ¤
⇕(W1)	M0
⇕(R1, R1, W0, R0) ⇕(W1, R1, W0, R0) ⇕(W0, R0) ⇕(R0, R0, W1, R1) ⇕(W0, R0)	M1-M5
⇕(R0, R0, W1, W1) ⇕(R1, R1, W0, W0) ⇕(W0) ⇕(R0) ⇕(R0, R0, W1) ⇕ (W1) ⇕ (R1)	M6-M12
⇕ (R1, R1, W0, W1) ⇕(R1, R1, W0) ⇕ (R0, R0, W1, W0) ⇕(W1, R1) ⇕(W0, R0, W1, R1)	M13-M17
⇕(W0, R0, W0) ⇕(W1, R1, W0, W0) ⇕(W1, R1, W1) ⇕ (W0, R0, W1, W1) ⇕(W0, R0, W1)	M18-M22
⇕(W0) ⇕(R0, R0, W0) ⇕(W1, R1, W0) ⇕(W1) ⇕(R1, R1, W1) ⇕(W1, R1, W0, R0)	M23-M28
⇕(W0, R0, W1, R1) ⇕(W1, R1) ⇕(W1, R1, W0, W0) ⇕(W0, R0, W1, W1) ⇕(W1, R1, W0)	M29-M33
⇕(W0, R0, W1, W0) ⇕ (W0, R0, W1) ⇕(W1, R1, W0, W1)	M34-M36

It is easy to check, that those 24 W_a operations occur at the last positions of March elements, and 24 W_v operations occur at the first positions of March elements. The only case when W_a matches with some W_v is when the first and the last operations of a March element are used to sensitize and the aggressor and the victim cells, i.e. when the March element contains only one Write operation that sensitized both the aggressor and the victim cells (the initial states of the aggressor and the victim cells must be the same). Here we have four cases: ⟨1W1; 1W1/0/-⟩, ⟨0W0; 0W0/1/-⟩, ⟨0W1; 0W1/0/-⟩, ⟨1W0; 1W0/1/-⟩. So for sensitizing the victim and the aggressor cells we need at least 24 + 24 - 4 = 44 Write operations (W_{av}).

Step 3: Let us consider the March elements that should have additional Write operations (W_s) that are needed to bring the aggressor cell to the required state. We should have situations when the last operation of the March element changes the state of the cell, so we must "adjust" the state of the cell to perform an operation for the aggressor. First we should indicate the faults that require additional Write operations. There are 16 such fault primitives: ⟨1W0; 0R0/1/0⟩, ⟨1W1; 0R0/1/0⟩, ⟨0W0; 1R1/0/1⟩, ⟨0W1; 1R1/0/1⟩, ⟨0W0; 0W1/0/-⟩, ⟨1W0; 0W0/1/-⟩, ⟨1W1; 1W0/1/-⟩, ⟨0W1; 1W1/0/-⟩, ⟨0W0; 1W1/0/-⟩, ⟨1W1; 0W0/1/-⟩, ⟨0R0; 0W1/0/-⟩, ⟨0R0; 1W1/0/-⟩, ⟨1R1; 0W0/1/-⟩, ⟨1R1; 1W0/1/-⟩, ⟨1R1; 0R0/1/0⟩, ⟨0R0;

1R1/0/1⟩. For each fault primitive one W_s operation is needed. So, 16 W_s operations are needed to bring the aggressor cell to the required state. Note that the length of a March element that detects such faults must be at least 4. The first operation is needed for victim cell sensitization, the second - for fault detection, the third - for providing the required state on the aggressor cell (W_s), the forth - for aggressor cell sensitization. Note that these W_s operations are not the first and last operations in the March element. Before this operation a Read operation should be present in the same March element for fault detection. So, we can conclude that these W_s operations are different from the mentioned above W_i, W_a and W_v operations.

Step 4: Taking into account the faults of subclass S_{av}, 12 sensitizing Read operations should be performed to the aggressor cell (R_a) and correspondingly 12 Read operations - to the victim cell (R_v). It is easy to check, that those 12 R_a operations occur at the last positions of March elements, and 12 R_v operations occur at the first positions of March elements. Only in case of fault primitives ⟨0R0; 0R0/1/0⟩ and ⟨1R1; 1R1/0/1⟩, the same Read operation can sensitize both the victim and aggressor cells. This means that the mentioned R_a operations are different from the mentioned R_v operations besides two special cases, and we have at least $12 + 12 - 2 = 22$ sensitizing Read operations (R_{av}).

Step 5: The next step is to try to calculate the number of fault detecting Read operations. It is obvious that some sensitizing Read operations can be used for detection purposes. There are only 10 fault primitives that can be detected with sensitizing Read operations (R_{av}). Here they are: ⟨0W0; 0W0/1/-⟩, ⟨1W1; 1W1/0/-⟩, ⟨0W1; 0W1/0/-⟩, ⟨1W0, 1W0/1/-⟩, ⟨0R0; 1W0/1/-⟩, ⟨0R0; 0W0/1/-⟩, ⟨1R1; 0W1/0/-⟩, ⟨1R1; 1W1/0/-⟩, ⟨0R0; 0R0/1/0⟩, ⟨1R1; 1R1/0/1⟩. The remaining 26 fault primitives require that detecting Read operation should be the second March operation but not the last operation in the March element. Thus, the R_d operations are different from R_{av} operations since R_{av} operations are placed either at the last position of the March element or at the first place. For each such fault one detecting Read operation is needed. So, additionally 26 detecting Read operations (R_d) are needed to detect those 26 faults.

Based on the considerations above, we can conclude that any March test that detects all realistic faults from subclass S_{av} should apply at least 109 operations for a fixed direction: 1 W_i, 44 W_{av}, 16 W_s, 22 R_{av} and 26 R_d. March test algorithm MMSAV given in Table 1 also has 109 operations, so it is the minimal. Thus, we can conclude that the theorem is proved. March test MMSAV should be applied for 4 directions mentioned above. So, the overall complexity of the proposed test algorithm is $109N \times 4 = 436N$.

4. March test algorithm for subclass S_{va}. Table 2 presents March test

MMSVA that detects all faults from subclass S_{va}. The complexity of the algorithm is $107N$ for a fixed direction and the overall complexity is $428N$. The algorithm was created based on the idea that the first operation in a March element is going to sensitize the aggressor cell, the second to detect, and the last to sensitize the fault. For example for the fault ⟨0W0/1/-; 1W1⟩ initialization of the aggressor cell is done by operation M49-3. This operation sets the value of the aggressor cell to 0. Then the algorithm runs March element M51, where the first operation M51-1 is used to sensitize the aggressor cell, the second M51-2 for bringing the cell value to 0 for the victim state, M51-3 to sensitize the fault. The detection is done by operation M52-1.

Theorem. *March test MMSVA is a minimal March test for detection of all realistic faults of subclass S_{va}.*

Proof. Let us evaluate the complexity of minimal March test algorithms for subclass S_{va}. We will not consider here FFMs dCFrd and dCFir since it is easy to check that they are logically impossible. That is why, we will consider here only FFMs dCFdrd, dCFtr and dCFwd that contain in total 36 fault primitives.

Step 1: The minimal March test algorithm should perform an initialization operation to the memory cells, so one Write operation (W_i) for initialization is mandatory.

Step 2: For faults listed in [5] for subclass S_{va}, it is easy to check that 24 sensitizing Write operations should be performed to the aggressor cell (W_a) and correspondingly 24 sensitizing Write operations to the victim cell (W_v). It is easy to check, that those 24 W_a operations occur at the first positions of March elements, and 24 W_v operations occur at the last positions of March elements. The only case when W_a match with some W_v is when the first and last operations of a March element are used to sensitize and aggressor and victim cells, i.e. when March element contains only one Write operation that sensitized both the aggressor and victim cells (initial states of the aggressor and victim cells must be the same). Here we have four cases: ⟨1W1/0/-; 1W1⟩, ⟨0W0/1/-; 0W0⟩, ⟨0W1/0/-; 0W1⟩, ⟨1W0/1/-; 1W0⟩. So for sensitizing the victim and aggressors cells we need at least 24 + 24 - 4 = 44 Write operations (W_{va}).

Step 3: Now let us consider the March elements that should have additional Write operations (W_s) that are needed to bring the victim cell to the required state. To calculate these additional Write operations, first we should indicate such faults primitives. Here they are: ⟨0R0/1/0; 0W1⟩, ⟨0R0/1/0; 1W1⟩, ⟨1R1/0/1; 0W0⟩, ⟨1R1/0/1; 1W0⟩, ⟨0W1/0/-; 1W1⟩, ⟨0W0/1/-; 0W1⟩, ⟨1W0/1/-; 0W0⟩, ⟨1W1/0/- ; 1W0⟩, ⟨1W1/0/-; 0W0⟩, ⟨0W0/1/-; 1W1⟩, ⟨0W1/0/-; 1R1⟩, ⟨1W1/0/-; 0R0⟩, ⟨0W0/1/-; 1R1⟩, ⟨1W0/1/-; 0R0⟩, ⟨0R0/1/0; 1R1⟩, ⟨1R1/0/1; 0R0⟩. For each fault primitive, one W_s operation is needed. So, 16 W_s operations are needed to bring

the victim cell to the required state. Note that the length of a March element that detects such faults must be at least 3. The first operation is needed for the aggressor cell sensitization, the second for providing the required state on the aggressor cell (W_s), the third for the victim cell sensitization. Note that these W_s operations are not the first and last operations in the March element. Taking into account this we can conclude that these W_s operations are different from the mentioned above W_i, W_a and W_v operations.

Step 4: From [5] we can see, that for faults of subclass S_{va}, there are 12 sensitizing Read operations that should be performed to the aggressor cell (R_a) and correspondingly 12 Read operations to the victim cell (R_v). It is easy to check, that those 12 R_a operations occur at the first positions of March elements, and 12 R_v operations occur at the last positions of March elements. Only in case of fault primitives ⟨0R0/1/0; 0R0⟩ and ⟨1R1/0/1; 1R1⟩, the same Read operation can sensitize both the victim and aggressor cells. This means that the mentioned R_a operations are different from the mentioned R_v operations besides two special cases, and we have at least $12 + 12 - 2 = 22$ sensitizing Read operations (R_{va}).

Step 5: The next step is to try to calculate fault detecting Read operations. It is obvious that some sensitizing Read operations can be used for detection purposes.

Table 2. Minimal march test MMSVA (107N)

March elements	Element #
⇕(W1)	M0
⇕ (W0, R0) ⇕ (R0) ⇕(R0, W1, R1) ⇕(R1, W0, R0) ⇕(R0, W1) ⇕(R1) ⇕(R1, W0, W1) ⇕(R1, W0)	M1-M8
⇕ (R0, W1, W0) ⇕(R0, W0) ⇕ (R0, W1, W1) ⇕ (R1, W0, W0) ⇕(R0) ⇕(W1, W0, R0) ⇕(R0)	M9-M15
⇕ (W0, W1, R1) ⇕(R1, W1) ⇕(R1) ⇕(W1, W0, R0) ⇕(R0) ⇕(W1, R1) ⇕(R1) ⇕(W1, R1) ⇕(R1)	M16-M24
⇕(W0, R0) ⇕(R0) ⇕ (W0, W1) ⇕(R1) ⇕(W0, W1, R1) ⇕(R1) ⇕(W1, W0, W1) ⇕(R1) ⇕(W0, W1)	M25-M33
⇕ (R1) ⇕(W1, W0) ⇕(R0) ⇕(W1) ⇕(R1) ⇕(W0) ⇕(R0) ⇕(W0, W1, W0) ⇕(R0) ⇕(W1, W0) ⇕(R0)	M34-M44
⇕(W0) ⇕(R0) ⇕(W1, W0, W0) ⇕(R0) ⇕(W0, W1, W1) ⇕(R1) ⇕(W1, W0, W0) ⇕(R0) ⇕(W1, R1)	M45-M53
⇕(R1) ⇕ (W1) ⇕(R1) ⇕(W0, W1, W1) ⇕(R1) ⇕(W0, W0) ⇕(R0)	M54-M60

There are only 12 fault primitives that can be used for detection purposes, because of Read operations in aggressor (R_a) detected with sensitizing Read operations (R_{av}): ⟨0W1/0/-; 1R1⟩, ⟨1W1/0/-; 0R0⟩, ⟨0W0/1/-; 1R1⟩, ⟨1W0/1/-; 0R0⟩, ⟨0R0/1/0; 1R1⟩, ⟨1R1/0/1; 0R0⟩, ⟨1W0/1/-; 1R1⟩, ⟨0W0/1/-; 0R0⟩, ⟨1W1/0/-; 1R1⟩,

⟨0W1/0/-; 0R0⟩, ⟨1R1/0/1; 1R1⟩, ⟨0R0/1/0; 0R0⟩. So we need at least 24 additional Read operations for detection (R_d).

Based on the considerations above, we can conclude that any March test that detects all faults from subclass S_{av} should apply at least 107 operations for fixed direction: 1 W_i, 44 W_{av}, 16 W_s, 22 R_{av} and 24 R_d. The March test algorithm MMSVA given in Table 2 also has 107 operations so it is the minimal. Thus, we can conclude that the theorem is proved. Note that the March test MMSVA should be applied for 4 directions mentioned above. So, the overall complexity of the minimal March test MMSVA is 428N.

5. Conclusions. In this paper, we proposed a minimal March test algorithm for detection of all two-operation, two-cell "realistic" dynamic functional fault models from subclass S_{av} and S_{va} when the aggressor and victim cells are physically adjacent. Here we also gave a proof, that the proposed test algorithms are minimal.

[1]Russian-Armenian (Slavonic) University

[2]Virage Logic

H. S. Avetisyan, G. E. Harutyunyan, V. A. Vardanian

Minimal March Test Algorithms for Detection of All Realistic Two-Operation, Two-Cell Dynamic Faults from Subclasses S_{av} and S_{va}

This paper introduces minimal March test algorithms for detection of "realistic" faults from the well known subclasses S_{av} and S_{va} of two-operation dynamic faults. Earlier, only subclasses S_{aa} and S_{vv} were considered. In this paper it is shown that the proposed March test algorithms detect all realistic faults (to be defined below) of subclasses S_{av} and S_{va}, and have minimum length with respect to the number of memory words.

Հ. Ս. Ավետիսյան, Գ. Է. Հարությունյան, Վ. Ա. Վարդանյան

Մինիմալ մարշ թեստային ալգորիթմ երկբջիջ, երկու գործողությամբ դինամիկ անսարքությունների S_{av} եւ S_{va} ենթադասերի բոլոր անսարքությունների հայտնաբերման համար

Ներկայացվում են մինիմալ մարշ թեստային ալգորիթմներ, որոնք կարողանում են հայտնաբերել S_{av} եւ S_{va} դասերի բոլոր դինամիկ անսարքությունները. դասերը երկու գործողությամբ զգայունացվող դինամիկ անսարքությունների ենթադասեր են, այսինքն՝ անսարքություններ, որոնք զգայունացվում են հիշողության բջջի նկատմամբ հաջորդական

երկու գործողություն կատարելիս: Նախկինում դիտարկված են եղել S_{aa} եւ S_{vv} ենթադասերը, որոնց համար ներկայացվել էին մարշ ալգորիթմներ: S_{av} եւ S_{va} դասերը ուսումնասիրված չեն եղել: Հոդվածում ներկայացված են նաեւ նշված երկու ալգորիթմների մինիմալության ապացույցները:

А. С. Аветисян, Г. Э. Арутюнян, В. А. Варданян

Минимальные марш тестовые алгоритмы, выявляющие все "реалистические" неисправности из подклассов двухклеточных, двухоперационных динамических неисправностей S_{av} и S_{va}

Представлены минимальные марш тестовые алгоритмы, которые способны выявлять все реалистические неисправности из класса динамических неисправностей S_{av} и S_{va}: классы являются подклассами неисправностей, которые восприимчивы к двум операциям над оперативной памятью.

Ранее были изучены только подклассы S_{aa} и S_{vv}, для которых были предложены марш алгоритмы: подклассы S_{av} и S_{va} не были изучены. Нами представлены также доказательства минимальности предложенных алгоритмов.

References

1. *Van de Goor A. J.* Testing semiconductor memories: Theory and Practice, John Wiley & Sons. 1991.

2. *Hamdioui S., Al-Ars Z., van de Goor A.J.* - Proc. IEEE VLSI Test Symposium. 2002. P. 395-400.

3. *Avetisyan H., Harutunyan, Vardanian V. A.* - 30th Proceeding of NAS (Математические вопросы кибернетики и вычислительной техники. Т. 30), Yerevan, Armenia. 2008. P. 18-24.

4. *Avetisyan H., Harutunyan G., Vardanian V. A., Zorian Y.* - IEEE East-West Design & Test Symposium. 2009. P. 175-178.

5. *Avetisyan H.* - 3th Annual Conference of RAU. Yerevan, Armenia. 2008. P. 34-41.

6. *Van de Goor A. J., Schanstra I.* - Proc. IEEE Workshop DELTA. 2002. P. 128-136.