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

Հատոր
Том 112 2012 № 3
Volume

INFORMATICS

УДК 519.688

G. E. Harutyunyan, academician S. K. Shoukourian,
foreign member of NAS RA Y. A. Zorian

Fault and Test Algorithm Periodicity Hypothesis in Memory Devices and Its Application to Memory BIST Processor Architecture

(Submitted 4/IV 2012)

Keywords: *fault periodicity, March test, symmetry, memory BIST.*

1. Introduction. Nowadays memory built-in self-test (BIST) controllers come either with hardwired standard test algorithms or with programmability (see [1-3]). In the case of hardwired BIST, these test algorithms should be general in nature and are not necessarily optimal for a novel or proprietary memory design.The user cannot modify or add its own test algorithms to hardwired BIST. The programmable BIST approaches use an architecture that usually requires an externally accessible register (Test Algorithm Register) of predefined format for storing a micro-program that will perform a given march test algorithm. Actually, the programmable BIST controller provides enough flexibility for test algorithm definition. At the same time, the test algorithm is composed of special test mechanisms (test operations, background patterns and addressing methods) which are usually hardwired in the BIST. In this context, the programmability of BIST means that only the existing test mechanisms can be used when building a test algorithm. This limitation specifically implies the flexibility of testing process and test time. In some cases this limitation can even lead to impossibility of programming the required test algorithm. Afterwards, solutions with programmability of separate test mechanisms were also proposed (e.g., [1,3]). However, all the mentioned above solutions do not have a common backbone which will allow considering all these different issues within a unified infrastructure.

In this paper, a new approach for building the mentioned infrastructure is proposed. It is based on a notion of periodicity and regularity for faults and test algorithms, and their interdependence. This is considered as a basis for building a generic BIST architecture. Some explanations are adduced below to clarify the idea.

We consider the known classification of faults, basing on the following factors:

- Complexity of fault sensitization, i.e., the number of operations required to activate the fault.
- Number of cells that the fault involves.

Following this classification, the faults can be grouped into different classes. The number of classes and faults inside them increase with the technology shrinking (see [4]). We have done a systematic investigation of evolution for these fault classes and their detection algorithms, covering a broad range of manufacturing technologies from 90nm to 28nm. Results of this investigation led to determination of a hypothesis that some regularity and periodicity rules exist for that evolution. Moreover, the known property of symmetry (see [5]), which means that each fault usually has its twin, was developed further and a special symmetry measure was introduced to be used in test mechanisms (see [6]). This brought to a representation of the mentioned property in a form of a rule also.

It is proposed to describe all these rules in a form of a special fault periodic table (FPT). Each column of FPT corresponds to some functionality of a test mechanism, while each row of FPT corresponds to a fault family determined by the complexity of fault sensitization. Fault symmetry is also taken into account in FPT. An intersection of a given row and column determines a test algorithm for detection of the given fault family.

This table not only allows building of a generic BIST architecture that supports programmability of test algorithms without limitations mentioned above. It also allows detection of new faults which arise in the field after manufacturing within the same BIST. If these faults can be predicted beforehand then due to regularity of the FPT it is possible to include them into the range of faults covered by the built BIST architecture and to detect them further via programmability.

Another direction of FPT usage is the following. There can be faults that are not realistic in the current technology but can be predicted as realistic in the future technologies. These faults can be also reflected in the BIST architecture. This becomes actual in cases when it is planned to implement a given design, including the BIST design, on a new technology basis without making changes.

The dependencies and regularities in the FPT do not completely imply from the known results. During the investigation, several unknown before periodical dependencies and regularities, as well as unknown use of symmetry in test mechanisms were found, justified and included in the FPT.

Based on discovered fault and test mechanism periodicity and regularity we have defined a new generic architecture of BIST. It has two levels of BIST programmability: first level is test mechanism programmability and second level is test algorithm programmability. The proposed BIST architecture has the following advantages:

- Automatic generation of efficient test algorithms based on FPT;

- Complete programmability of test algorithms and test mechanisms;
- Flexible customization to the specific application and a possibility of finding the optimal trade-off between the area and BIST functionality.

2. Definitions and notations. The definition of the fault primitive concept (FP=<S/F/R>), used to define memory faults, can be found in [7-9]. In the notation of FP, S is the sequence of test operations required for fault sensitization (e.g., 0W1, 1W0R0, etc.), F$\in$ {0, 1} is the observed memory behavior that deviates from the expected one. R$\in$ {0, 1, -} is the result of a read operation applied to the victim cell, in case if the last operation of S is a Read operation. “-“ is used when the last operation of S is Write operation.

The difference between static and dynamic faults is determined by #S, the number of operations in S. Depending on #S, the memory faults can be classified into *static* or *dynamic* faults. Static faults are the faults sensitized by performing at most one operation (#S$\leq$ 1). Dynamic faults are the faults that can only be sensitized by performing more than one operation sequentially (#S>1). If #S=2, the fault class will be named *class of two-operation dynamic faults* and, in general, if #S=n, then it will be named *class of n-operation dynamic faults*.

The classification of faults can be done not only with respect to #S, but also with respect to #C, the number of different cells the fault involves. Depending on #C, a fault can be classified into *single-cell faults* involving a single cell (#C=1), *two-cell (coupling) faults* that involve two cells (#C=2) and, in general, *n-cell faults* involving n cells (#C=n).

In the sequel, we will consider only the faults satisfying the following conditions: #C=1 or #C=2, #S$\geq$ 0. Several works (see [10,11]) show that these faults are the most realistic (probable) faults in nowadays memories.

Below we propose a new and simple notation to describe the faults. It is obtained from FP=<S/F/R> in the following way:

- Separate from S the cell initial value and the sensitizing sequence of test operations (*ssto*). For example, if S=1W0R0, then the initial value is 1 and *ssto* is W0R0.
- Do not use F and R since they can be easily obtained from S.

Notation 1. *(x_v, S) denotes a single-cell fault, where $S=OP_1D_1, \dots, OP_nD_n$, $n\geq 0$, is a ssto applied to the victim cell, when the victim cell value is x_v. After fault sensitization the victim cell value becomes $\sim D_n$, if $S \neq \varnothing$, otherwise it becomes $\sim x_v$.*

Notation 2. *(x_a, x_v, S_v) denotes a two-cell (coupling) fault, where $S_v=OP_1D_1, \dots, OP_nD_n$, $n\geq 0$, is a ssto applied to the victim cell, when the victim cell value is x_v and the aggressor cell value is x_a. After fault sensitization the victim cell value becomes $\sim D_n$, if $S \neq \varnothing$, otherwise it becomes $\sim x_v$.*

Notation 3. *(x_a, S_a, x_v) denotes a two-cell (coupling) fault, where $S_a=OP_1D_1, \dots, OP_nD_n$, $n\geq 0$, is a ssto applied to the aggressor cell, when the aggressor cell value is x_a and the victim cell value is x_v. After fault sensitization the victim cell value becomes $\sim x_v$.*

Definition 1. *FG(x, S) is a fault group that contains all faults that are sensitized by ssto S applied to a cell containing value x: FG(x, S)={(x, S), (y, x, S), (x, S, y)}, x, y∈{0, 1}, S=OP_1D_1, …, OP_nD_n, n≥0.*

In case of two-cell faults, the relation (lower/higher address in the memory) between the aggressor cell “a” and the victim cell “v” is essential, since a test algorithm can detect the same coupling fault if a<v and not detect if a>v. When we need to indicate the aggressor and victim cell relations for a fault, we will denote it by a<v or a>v. For example, $(0, W1, 1)_{a<v}$.

A March test algorithm M is a test algorithm with a finite number of March elements M=M_1; M_2; …; M_k (see [5]), where each March element M_i consists of an addressing order A_i and a finite number of Read/Write operations M_i = A_i (O_1 D_1, …, O_mD_m):

- A_i∈ {□, □, ⇕} - addressing order: □ - ascending, □ - descending, ⇕ - arbitrary;
- O_j∈ {R, W} - test operations: R – Read, W – Write;
- D_j - background pattern.

Definition 2. F_1=(x_1, {$OP_{11}D_{11}$, …, $OP_{1k1}D_{1k1}$}), F_2=(x_2, {$OP_{21}D_{21}$, …, $OP_{2k2}D_{2k2}$}).

F_3=(y_1, x_1, {$OP_{11}D_{11}$, …, $OP_{1k1}D_{1k1}$}), F_4=(y_2, x_2, {$OP_{21}D_{21}$, …, $OP_{2k2}D_{2k2}$}).

F_5=(y_1, {$OP_{11}D_{11}$, …, $OP_{1k1}D_{1k1}$}, x_1), F_6=(y_2, {$OP_{21}D_{21}$, …, $OP_{2k2}D_{2k2}$}, x_2).

A pair of faults F_1 and F_2 (F_3 and F_4, F_5 and F_6) is called a *pair of symmetric faults* (or *symmetric faults*) and denoted by $F_1 \leftrightarrow F_2$ (respectively, $F_3 \leftrightarrow F_4$, $F_5 \leftrightarrow F_6$), if it satisfies the following conditions:

- k_1=k_2, x_1=~ x_2, y_1=~ y_2 (“~ ” means opposite value);
- OP_{11}=OP_{21}, …, OP_{1k1}=OP_{2k2} and D_{11}=~ D_{21}, …, D_{1k1}=~ D_{2k2}.

For example, (0, ∅)$\leftrightarrow$(1, ∅), (0, 1, W0R0)$\leftrightarrow$(1, 0, W1R1).

We can denote also FG(x, S)$\leftrightarrow$FG(~ x, ~ S) and call them as *a pair of symmetric fault groups* (or *symmetric fault groups*). This means that:
∀F∈ FG(x, S), ∃ G∈ FG(~ x, ~ S), that F$\leftrightarrow$G and ∀F∈ FG(~ x, ~ S), ∃ G∈ FG(x, S), that F$\leftrightarrow$G.

Definition 3. *A pair of March elements M_1=A_1($O_{11}D_{11}$, …, $O_{1n1}D_{1n1}$) and M_2=A_2($O_{21}D_{21}$, …, $O_{2n2}D_{2n2}$) is called a pair of symmetric March elements and denoted by $M_1 \leftrightarrow M_2$, if it satisfies the following conditions:*

- *n_1=n_2 and OP_{11}=OP_{21}, …, OP_{1n1}=OP_{2n2};*
- *(D_{11}=D_{21}, …, D_{1k1}=D_{2k2}) OR (D_{11}=~D_{21}, …, D_{1k1}=~D_{2k2}).*

Definition 4. *March test M=M_1; M_2; …; M_k is called symmetric if it satisfies one of the following conditions:*

Condition A. March test M consists only of pairs of symmetric March elements.

Condition B. M_1=⇕(WD_1) and M'=M_2; …; M_k satisfies Condition A.

Condition C. M_k=⇕(RD_k) and M'=M_1; …; M_{k-1} satisfies Condition A.

Condition D. M_1*=⇕(WD*$_1$*),* M_k*=⇕(RD*$_k$*) and M´=*M_2*;…;* M_{k-1} *satisfies Condition A.*

Four symmetric March tests are presented below. Each of them satisfies one of the conditions described in Definition 4.

□(W0, R0); □(W1, R1) - satisfies Condition A.

⇕(W0); □(R0, W1); □(R1, W0) (MATS+ [5]) - satisfies Condition B.

□(W0, R0, W1); □(W1, R1, W0); ⇕(R0) - satisfies Condition C.

⇕(W0); ⇑(R0, W1); ⇑(R1, W0); ⇓(R0, W1); ⇓(R1, W0); ⇕(R0) (March C- [5]) - satisfies Condition D.

Definition 5. *A March test is called partial-symmetric if it is not a symmetric March test but contains at least one pair of symmetric March elements. For example, well-known March B test algorithm [5] is a partial-symmetric March test: March B: ⇕(W0); ⇑(R0, W1, R1, W0, R0, W1); ⇑(R1, W0, W1); ⇓(R1, W0, W1, W0); ⇓(R0, W1, W0).*

3. Fault and test algorithm periodicity. To study the possible regularities in memory faults and test algorithms, we have investigated their evolution starting from 90nm to 45nm technology nodes. The investigations show that every new technology brings new and more complex faults. In its turn test algorithms also become more complex. The main regularities that we have noticed during our investigations are introduced below.

Regularity 1. Newly discovered faults have similar behavior as the known faults. For example, the same notation is usually used for description of known and new faults.

Regularity 2. For detection of a new fault, a new test algorithm is usually constructed/extended from an existing test algorithm. For example, March test ⇑(W0, W0, R0); ⇑(W1, R1, R1) is constructed from March test ⇑(W0, R0); ⇑(W1, R1).

Regularity 3. Each fault has its twin fault (e.g., Stuck-At-0 and Stuck-At-1). In other words, for each fault F there is a fault G, that F↔G (*symmetric faults*).

Regularity 4. *Symmetric faults* are usually detected by symmetric March tests. For example, the pair (0, R0) and (1, R1) is detected by symmetric March test ⇑(W0, R0, R0); ⇑(W1, R1, R1).

During investigation of fault and test algorithm regularity we have noticed that there is a periodicity in evolution of faults and test algorithms.

Fault periodicity hypothesis. Memory faults are evolved in periodic way. This means that the new faults are the periodic extensions of the existing faults.

Fault families. Based on the length of the *ssto* the faults can be divided into *fault families*. All faults that are sensitized by *ssto* of length k are from F_k-family. For example, (0, ∅)∈ FF_0, (0, 1, W1R1W0)∈ FF_3, (1, W0R0R0R0, 1) ∈ F F_4.

FF_0={FG(0, ∅), FG(1, ∅)}, FF_1={FG(0, W0), FG(1, W1), FG(0, W1), FG(1, W0), FG(0, R0), FG(1, R1)}.

Test algorithm periodicity hypothesis. Test algorithms are evolved in periodic way. This means that the new test algorithms are the periodic extensions of the existing test algorithms.

Let us compare March C- and March MSS1. March C- is a minimal March test for detection of all traditional faults, while March MSS1 is a minimal March test for detection of all static faults. Note that static faults are superset of traditional faults.

March C-: $\Updownarrow$(W0); $\Uparrow$(R0, W1); $\Uparrow$(R1, W0); $\Downarrow$(R0, W1); $\Downarrow$(R1, W0); $\Updownarrow$(R0)

March MSS1 [12]: $\Updownarrow$(W0); $\Uparrow$(R0, R0, W1, W1); $\Uparrow$(R1, R1, W0, W0); $\Downarrow$(R0, R0, W1, W1); $\Downarrow$(R1, R1, W0, W0); $\Updownarrow$(R0)

Both March tests has the same structure with a difference that Write and Read operations of 2nd, 3rd, 4th and 5th March elements in March MSS1 are doubled. This means that March MSS1 can be easily extended from March C-.

In memories the faults usually occur as *pairs of symmetric faults* (see Regularity 3). Since the *symmetric faults* are usually detected by symmetric March tests (see Regularity 4) we have developed a *March test template* to construct symmetric March tests. The template allows obtaining March tests without using time consuming March test generation tools.

March test template. Let us assume that S is the sequence of test operations, i.e., $S=OP_1D_1, \ldots, OP_kD_k$, $k\geq 0$ and $x\in\{0, 1\}$. *March test template* MTT(x, S) has the following structure: $\Uparrow$(W($\sim$D$_k$)); $\Uparrow$([R($\sim$D$_k$)], [W(x)], S); $\Uparrow$([R(D$_k$)], [W($\sim$x)], $\sim$S); $\Downarrow$([R($\sim$D$_k$)], [W(x)], S); $\Downarrow$([R(D$_k$)], [W($\sim$x)], $\sim$S); $\Downarrow$(R($\sim$D$_k$)), where:

- $\sim S=OP_1(\sim D_1), \ldots, OP_k(\sim D_k)$, if $k\geq 1$. $\sim S=\varnothing$, if $S=\varnothing$.
- [W(x)] and [W($\sim$x)] are absent, if $S\neq\varnothing$ and $x=\sim D_k$, otherwise they are present;
- [R(D$_k$)] and [R($\sim$D$_k$)] are absent, if $S\neq\varnothing$, $x=\sim D_k$, OP_1=R, otherwise they are present;
- If $S=\varnothing$, then consider $D_k=x$.

All March tests obtained by MTT are symmetric. This is true since all the March tests obtained by MTT satisfy the Condition D of Definition 4.

Theorem 1. *The March test obtained by MTT(x, S) detects all faults from FG(x, S) and FG(~x, ~S).*

Proof. Let us show that March test obtained by MTT detects the considered faults. There can be 4 different cases depending on values x and S:

Case A: $S=\varnothing$, $x=\forall$. MTA=$\Uparrow$(W($\sim$x)); $\Uparrow$(R($\sim$x), W(x)); $\Uparrow$(R(x), W($\sim$x)); $\Downarrow$(R($\sim$x), W(x)); $\Downarrow$(R(x), W($\sim$x)); $\Downarrow$(R($\sim$x)).

Case B: $S=OP_1D_1, \ldots, OP_kD_k$, $k\geq 1$, $x=D_k$. MTB=$\Uparrow$(W($\sim$x)); $\Uparrow$(R($\sim$x), W(x), S); $\Uparrow$(R(x), W($\sim$x), $\sim$S); $\Downarrow$(R($\sim$x), W(x), S); $\Downarrow$(R(x), W($\sim$x), $\sim$S); $\Downarrow$(R($\sim$x)).

Case C: $S=OP_1D_1, \ldots, OP_kD_k$, $k\geq 1$, $x=\sim D_k$ and OP_1=W. MTC=$\Uparrow$(W(x)); $\Uparrow$(R(x), S); $\Uparrow$(R($\sim$x), $\sim$S); $\Downarrow$(R(x), S); $\Downarrow$(R($\sim$x), $\sim$S); $\Downarrow$(R(x)).

Case D: $S=OP_1D_1, \ldots, OP_kD_k$, $k\geq 1$, $x=\sim D_k$ and $OP_1=R$. MTD=$\Uparrow$(W(x)); $\Uparrow$(S); $\Uparrow$(~ S); $\Downarrow$(S); $\Downarrow$(~ S); $\Downarrow$(R(x)).

Table 1 shows the faults and the corresponding operations of March tests for sensitizing and detecting them for Case A. In the table, $MT_{i,j}$ denotes jth component of ith March element in March test MT, $i\geq 1$, $j\geq 1$. For example, $MTA_{3,1}$ means the first component (i.e., operation R(x)) of the third March element in March test MTA. It is easy to construct such tables for other cases (Cases B, C and D) as well.

March tests for detection of multiple fault groups. In order to construct a March test for detection of n pairs of *symmetric fault groups* FG(x_1, S_1)↔FG(~ x_1, ~ S_1), FG(x_2, S_2)↔FG(~ x_2, ~ S_2), …, FG(x_n, S_n)↔FG(~ x_n, ~ S_n), the following steps should be done:

Step 1. Construct a combined *ssto* CS which application to a cell containning value x will sensitize all the considered faults. For example, CS=S_1, W(x_2), S_2, …, W(x_n), S_n applied to a cell containing value x=x_1.

Step 2. Obtain March test M by MTT(x, CS) that will detect all the considered faults.

March tests for detection of fault families. Let us construct March test MT_0 for detection of all faults from F_0-family. Since FF_0={FG(0, ∅), FG(1, ∅)}, then MT_0=MTT(0, ∅)=$\Updownarrow$(W0); $\Uparrow$(R0, W1); $\Uparrow$(R1, W0); $\Downarrow$(R0, W1); $\Downarrow$(R1, W0); $\Updownarrow$(R0). This is the well-known March C- test algorithm which is the minimal March test for detection of all traditional faults.

Now let us construct March test MT_1 for detection of all faults from FF_1. FF_1={FG(0, W1), FG(1, W0), FG(1, W1), FG(0, W0), FG(1, R1), FG(0, R0)}. To construct the combined *ssto*, the following cases should be considered: W1 should be applied to a cell containing value 0, W1 should be applied to a cell containing value 1, R1 should be applied to a cell containing value 1.

Table 1.Case A. Fault sensitization and detection

Fault Group	Fault	Sensitization	Detection	Fault Group	Fault	Sensitization	Detection
FG(x, ∅)	(x, ∅)	$MTA_{2,2}$	$MTA_{3,1}$	FG(~ x, ∅)	(~ x, ∅)	$MTA_{1,1}$	$MTA_{2,1}$
	$(x, x, \varnothing)_{a<v}$	$MTA_{4,2}$	$MTA_{5,1}$		$(x, \sim x, \varnothing)_{a<v}$	$MTA_{2,2}$	$MTA_{2,1}$
	$(x, x, \varnothing)_{a>v}$	$MTA_{2,2}$	$MTA_{3,1}$		$(x, \sim x, \varnothing)_{a>v}$	$MTA_{4,2}$	$MTA_{4,1}$
	$(\sim x, x, \varnothing)_{a<v}$	$MTA_{3,2}$	$MTA_{3,1}$		$(\sim x, \sim x, \varnothing)_{a<v}$	$MTA_{3,2}$	$MTA_{4,1}$
	$(\sim x, x, \varnothing)_{a>v}$	$MTA_{5,2}$	$MTA_{5,1}$		$(\sim x, \sim x, \varnothing)_{a>v}$	$MTA_{1,1}$	$MTA_{2,1}$
	$(x, \varnothing, x)_{a<v}$	$MTA_{4,2}$	$MTA_{5,1}$		$(\sim x, \varnothing, x)_{a<v}$	$MTA_{3,2}$	$MTA_{3,1}$
	$(x, \varnothing, x)_{a>v}$	$MTA_{2,2}$	$MTA_{3,1}$		$(\sim x, \varnothing, x)_{a>v}$	$MTA_{5,2}$	$MTA_{5,1}$
	$(x, \varnothing, \sim x)_{a<v}$	$MTA_{2,2}$	$MTA_{2,1}$		$(\sim x, \varnothing, \sim x)_{a<v}$	$MTA_{3,2}$	$MTA_{4,1}$
	$(x, \varnothing, \sim x)_{a>v}$	$MTA_{4,2}$	$MTA_{4,1}$		$(\sim x, \varnothing, \sim x)_{a>v}$	$MTA_{1,1}$	$MTA_{2,1}$

The combined *ssto* is W1W1R1, that should be applied to a cell containing value 0. MT_1=MTT(0, W1W1R1)=$\Updownarrow$(W0); $\Uparrow$(R0, W1, W1, R1); $\Uparrow$(R1, W0, W0, R0); $\Downarrow$(R0, W1, W1, R1); $\Downarrow$(R1, W0, W0, R0); $\Updownarrow$(R0). This is the well-known March MSS [13] test algorithm which is the minimal March test for detection of all static faults.

Symmetry measure. In [6], a new method of symmetry measurement for March test algorithms is introduced. The dependency between symmetry measure and BIST optimization range is discussed. Based on the proposed metric and experiments, it is stated that the higher symmetry of a test algorithm brings to a greater BIST area saving.

In [14], an efficient test algorithm is generated by a new method that is sufficiently general and efficient to generate symmetric March test algorithms for different combinations of static and dynamic faults. The method is based on the observation that almost all known minimal or efficient March test algorithms are symmetric.

4. Fault Periodic Table. Since we have already stated about *fault periodicity* and *fault families*, we have tried to introduce the faults by a Fault Periodic Table (FPT) (see Table 2). In the table #C is the number of different cells the fault involves. Though we have restricted $\#C \leq 2$, the FPT can be extended also for cases #C>2. "SCF" stands for single-cell fault, TCFv–two-cell fault where *ssto* is applied to the victim cell and TCFa–two-cell fault where *ssto* is applied to the aggressor cell.

FPT allows studying and remembering the properties of a large number of faults in a simpler way. In FPT the locations of unknown/not discovered faults are left blank. Usually it becomes possible to predict them beforehand based on the properties of the existing faults.

Table 2. Fault Periodic Table (FPT)

Fault Family	Fault Group	#C=1	#C=2			
		SCF	TCFv	TCFv	TCFa	TCFa
FF_0	FG(0, ∅)	(0, ∅)	(0, 0, ∅)	(1, 0, ∅)	(0, ∅, 0)	(0, ∅, 1)
	FG(1, ∅)	(1, ∅)	(1, 1, ∅)	(0, 1, ∅)	(1, ∅, 1)	(1, ∅, 0)
FF_1	FG(0, W0)	(0, W0)	(0, 0, W0)	(1, 0, W0)	(0, W0, 0)	(0, W0, 1)
	FG(1, W1)	(1, W1)	(1, 1, W1)	(0, 1, W1)	(1, W1, 1)	(1, W1, 0)
	FG(0, W1)	(0, W1)	(0, 0, W1)	(1, 0, W1)	(0, W1, 0)	(0, W1, 1)
	FG(1, W0)	(1, W0)	(1, 1, W0)	(0, 1, W0)	(1, W0, 1)	(1, W0, 0)
	FG(0, R0)	(0, R0)	(0, 0, R0)	(1, 0, R0)	(0, R0, 0)	(0, R0, 1)
	FG(1, R1)	(1, R1)	(1, 1, R1)	(0, 1, R1)	(1, R1, 1)	(1, R1, 0)
FF_2	FG(0, W0W0)	(0, W0W0)	(0, 0, W0W0)	(1, 0, W0W0)	(0, W0W0, 0)	(0, W0W0, 1)
	FG(1, W1W1)	(1, W1W1)	(1, 1, W1W1)	(0, 1, W1W1)	(1, W1W1, 1)	(1, W1W1, 0)
	FG(0, W0W1)	(0, W0W1)	(0, 0, W0W1)	(1, 0, W0W1)	(0, W0W1, 0)	(0, W0W1, 1)
	FG(1, W1W0)	(1,W1W0)	(1, 1, W1W0)	(0, 1, W1W0)	(1, W1W0, 1)	(1, W1W0, 0)

	…	…	…	…	…	…
…	…	…	…	…	…	…

5. Conclusions. Basing on a systematic investigation of faults and test algorithms during their evolution a periodicity and regularity of faults is discovered. A hypothesis on fault and test algorithm periodicity and their interdependence is proposed. According to this, Fault Periodic Table (FPT) and March test template (MTT) are created to reflect formally their interdependence. FPT allows to consider any large number of faults in one table and MTT allows to obtain March tests without March test generation tools. It is justified that the proposed MTT leads to effective test algorithms.

Synopsys

G. E. Harutyunyan, academician S. K. Shoukourian,
foreign member of NAS RA Y. A. Zorian

Fault and Test Algorithm Periodicity Hypothesis in Memory Devices and Its Application to Memory BIST Processor Architecture

This paper introduces a new approach for building a generic built-in self-test (BIST) processor architecture for memory testing that is based on a hypothesis of periodicity and regularity for faults and test algorithms. It is proposed to describe all the periodicity and regularity rules in a form of a special fault periodic table (FPT) and March test template (MTT). FPT allows to consider any large number of faults in one table and MTT allows to obtain March tests without using special tools for their generation.

Գ. Է. Հարությունյան, ակադեմիկոս Ս. Կ. Շուքուրյան,
ՀՀ ԳԱԱ արտասահմանյան անդամ Ե. Ա. Զորյան

Անսարքությունների և թեստ ալգորիթմների պարբերականության հիպոթեզ հիշող սարքերում և դրա կիրառումը հիշող սարքերի ներդրված ինքնաթեստավորող պրոցեսորի ճարտարապետությունում

Ներկայացվում է նոր մոտեցում՝ կառուցելու համընդհանուր ներդրված ինքնաթեստավորող պրոցեսորի ճարտարապետություն հիշող սարքերի թեստավորման համար, որը հիմնված է անսարքությունների և թեստ ալգորիթմների պարբերականության և կանոնավորության հիպոթեզի վրա: Առաջարկվում է պարբերականության և կանոնավորության բոլոր կանոնները նկարագրել անսարքությունների FPT պարբերական աղյուսակի և MTT Մարշ թեստի շաբլոնի տեսքով: FPT-ն թույլ է տալիս նկարագրել կամայական մեծ քանակի անսարքություններ մեկ աղյուսակի մեջ, իսկ MTT-ն թույլ է տալիս ստանալ Մարշ թեստեր՝ առանց օգտագործելու դրանք ծնող հատուկ գործիքներ:

Г. Э. Арутюнян, академик С. К. Шукурян,
иностраный член НАН РА Е. А. Зорян

Гипотеза периодичности ошибок и тестовых алгоритмов для устройств памяти и ее применение в архитектуре процессора встроенного тестирования памяти

Предлагается новый подход к построению архитектуры универсального процессора встроенного тестирования для устройств памяти, основанный на гипотезе периодичности и регулярности ошибок. Предлагается описать все правила периодичности и регулярности в виде специальной периодической таблицы ошибок FPT и шаблона марш-тестов MTT. FPT позволяет описать произвольно большое количество ошибок в единой таблице, а MTT – получать марш-тесты без использования специальных средств их генерации.

References

1. *Boutobza S., Nicolaidis M., Lamara K.M., Costa A.* - International Test Conference. 2005. P. 1155-1164.
2. *Zarrineh K., Upadhyaya S. J.* - Conference on Design, Automation and Test in Europe. 1999. P. 708–713.
3. *Hakhumyan A., Harutyunyan G.* – International Conference on Computer Science and Information Technologies. 2011.P. 287-290.
4. *Hamdioui S., Wadsworth R., Reyes J. D., Van de Goor A. J.* - Journal of Electronic Testing: Theory and Applications (JETTA). 2004. V. 20., N 3. P. 245-255.
5. *Van de Goor A. J.* Testing semiconductor memories: Theory and Practice. 1991.
6. *Harutyunyan G., Hakhumyan A., Shoukourian S., Vardanian V., Zorian Y.* - Journal of Electronic Testing: Theory and Applications (JETTA). 2011.V. 27, N 6. P. 753-766.
7. *Van de Goor A. J., Al-Ars Z.* - VLSI Test Symposium, 2000. P. 281-289.
8. *Hamdioui S., Van de Goor A. J., Rodgers M.* - International Workshop on Memory Technology, Design, and Testing. 2002. P. 95-100.
9. *Hamdioui S., Gaydadjiev G. N., Van de Goor A. J.* - Workshop on Circuits, Systems and Signal Processing. 2003. P 84-89.
10. *Dilillo L., Girard P., Pravossoudovitch S., Virazel A.* - European Test Symposium. 2004. P. 140-145.
11. *Hamdioui S., Al-Ars Z., Van de Goor A. J.* - VLSI Test Symposium. 2002. P. 395-400.
12. *Harutunyan G., Vardanian V. A., Zorian Y.* - VLSI Test Symposium. 2005. P. 53-59.
13. *Harutunyan G., Vardanian V. A., Zorian Y.* –Design and Diagnostics of Electronic Circuits and Systems, 2006. P. 260-265.
14. *Harutyunyan G., Shoukourian S., Vardanian V., Zorian Y.* - Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). 2012.V. 31.N 6. P. 941-949.