Test Set Development for Cache Memory in Modern Microprocessors

Zaid Al-Ars, *Member, IEEE*, Said Hamdioui, *Member, IEEE*, Georgi Gaydadjiev, *Member, IEEE*, and Stamatis Vassiliadis, *Fellow, IEEE*

***Abstract*—Up to 53% of the time spent on testing current Intel microprocessors is needed to test on-chip caches, due to the high complexity of memory tests and to the large amount of transistors dedicated to such memories. This paper discusses the methodology used to develop effective and efficient cache tests, and the way it is implemented to optimize the test set used at Intel to test their 512-kB caches manufactured in a 0.13-μm technology. An example is shown where a maximal test set of 15 tests with a corresponding maximum test time of 160.942 ms/chip is optimized to only six tests that require a test time of only 30.498 ms/chip.**

***Index Terms*—Fault coverage, memory testing, microprocessor cache, test set development, test time.**

I. Introduction

CACHE memory plays a significant role in todays microprocessors. An ever increasing portion of the microprocessor is being dedicated to this ultra fast on-chip memory, in order to ensure a continuous supply of information to high performance processors. These memories are organized in multiple layers [e.g., level-1 (L1), level-2 (L2), and level-3 (L3) caches], with different performance requirements, and are designed to serve various system requirements (e.g., instruction caches versus data caches) [23]. The amount of on-chip memory embedded alongside the processor occupies about 50% of the chip area [26], and is expected to reach 90% by 2011 [8]. In terms of transistor count, the numbers are even more staggering. Memory consumes up to 75% of the transistors today in a modern processor design [23].

As the size of caches increase, there is a corresponding increase in the impact of memory testing on the overall test time and fault coverage of the microprocessor. At the same time, as the dimensions of manufactured devices decrease, new more complex memory faults are observed that require specialized memory tests to ensure their detection [24]. Keeping in mind that the allowed *defect per million (DPM)* budget for the entire collection of on-chip memory is extremely low, these trends stress the increasing importance of memory testing in microprocessors today and in the future [33].

This paper describes the methodology used industrially in *high volume manufacturing (HVM)* testing to develop cache tests in modern microprocessors. This paper also identifies some important memory tests and evaluates their effectiveness. In addition, it discusses the application of the test development methodology in a test experiment performed at Intel to test their 512-kB caches manufactured in a 0.13-μm technology [22].

This paper is organized as follows. Section II discusses the methodology of cache test development in a current HVM testing environment. The test development process is divided into four main steps: maximal test set generation, test application, test optimization, and, finally, the optimal test set generation. Section III discusses the first step in this methodology, where the maximum test set is generated. Test application (second step) is presented in Section IV, where the results are analyzed of a test experiment performed on caches embedded on Intel microprocessors. Section V shows how to perform the third step of test optimization based on feedback from the test application step. The optimal test set (fourth step) is analyzed in Section VI. Finally, Section VII ends with the conclusion.

II. Industrial Cache Testing

The flow of microprocessor testing is complex and time consuming. Fig. 1 shows a typical representation of such a test flow [14]. Fig. 1 divides the flow into two main phases: *wafer test* (also called *sort test*) and *package test* (also called *class test*). The wafer test, where chips are tested on wafer before dicing and packaging, consists fully of *structural testing*, where internal functional blocks (or structures) are tested separately rather than testing the chip as a whole system through its input/output (I/O) pins. In this stage, cache is tested using *fault localizing tests* with the objective of repairing failing cells with redundant ones.

The package test, which is performed on individual packaged components, is the second phase of the test flow, and consists of three stages: burn-in, structural testing, and, finally, functional testing. The burn-in stage thermally stresses the components to ensure sensitizing early life reliability problems. The structural testing stage is similar to the structural testing stage performed in the wafer test phase of the test flow. Cache is tested again in this stage using *fault detecting tests* with the objective of eliminating faulty chips from the flow. The final stage of the test flow is the functional testing stage, where the whole chip is tested at-speed through its I/O pins to ensure the functionality of the processor, and to sort different chips according to their speed (so-called *speed binning*).

The specific test flow of on-chip cache testing shares the same general characteristics with the flow of stand alone memories [6]. Fig. 2 shows the relative breakdown of test time between different processor components for the Pentium 4 processor [26].

Manuscript received August 8, 2006; revised June 24, 2007.

The authors are with the Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Lab of Computer Engineering, 2628 CD Delft, The Netherlands (e-mail: z.al-ars@tudelft.nl).

S. Vassiliadis, deceased, was with the Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Lab of Computer Engineering, 2628 CD Delft, The Netherlands.

Digital Object Identifier 10.1109/TVLSI.2008.2000257

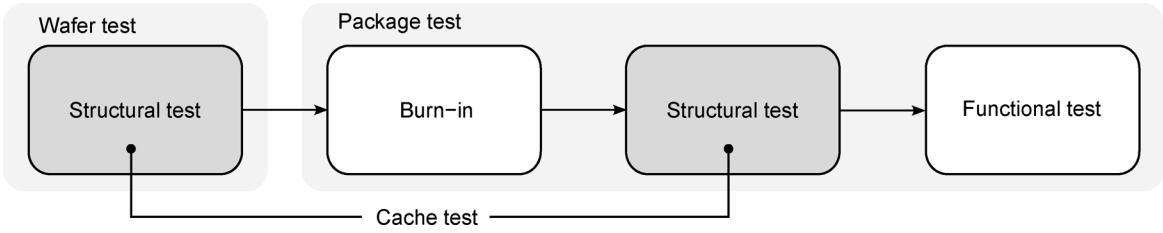


Fig. 1. Typical representation of a microprocessor test flow.

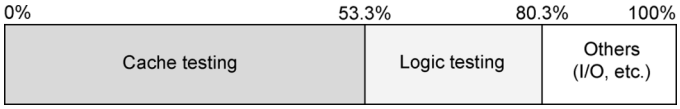


Fig. 2. Relative breakdown of test time between different processor components for the Pentium 4 processor.

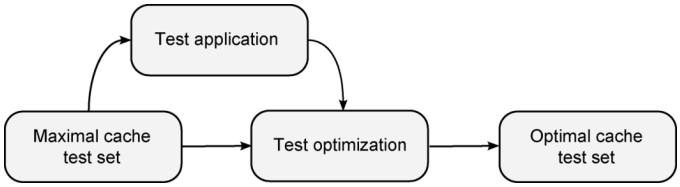


Fig. 3. Cache test development based on the kitchen sink principle.

Cache testing consumes 53.3% of the total test time, and thereby represents the largest portion of the test flow. Second on the list is logic testing, which consumes 27.0% of the total test time. The remaining 19.7% of test time is distributed among I/O testing, parametric testing as well as others.

This paper is mainly concerned with the process of test development and application of fault detecting tests for caches, applied during the structural testing stage of the package test, as shown in Fig. 1. The test development process described here is also very similar to the one used for fault localizing cache tests implemented in the wafer test phase. The cache test development process is based on the kitchen sink principle, as depicted in Fig. 3. The test development process starts out with a maximal set of cache tests that ensures detecting all possible faults in the memory. In general, there are more than 50 test algorithms implemented in this stage [33]. This set is not optimal and takes an excessively long test time to complete. All of these tests are applied to a large sample of microprocessor caches, as part of the manufacturing test set used to test microprocessor chips on the production line in the fab. Based on the feedback from this analysis, it is possible to optimize the maximal test set to construct an optimal test set that is particularly suited to the cache under test.

III. Maximal Cache Test Set

This section describes the construction and content of the maximal cache test set used to kick off the process of constructing an optimal industrial test set as shown in Fig. 3. The test set is constructed from a number of different sources: 1) some well-known traditional tests; 2) tests developed specifically for the cache under test; and 3) theoretically derived tests using the *fault primitive (FP)* analysis [5]. Typically, there are more than 50 tests in the maximal test set [33], but to keep the discussion simple this paper will discuss the test development process for a subset of the maximal test set, referred to here as the *base tests (BTs)*. An overview of the used BTs is given in Section III-A, while the needed stresses during test application are presented in Section III-B.

A. Overview of Used BTs

Table I lists the used BTs along with their test length (TL), where n denotes the number of bits in the cache, C the number of columns, and R the number of rows. The used march notation is explained as follows [30]. A complete march test is delimited by the "$\{\ldots\}$" bracket pair, while a march element is delimited by the "$(\ldots)$" bracket pair. March elements are separated by semicolons, and the operations within a march element are separated by commas. Note that all operations of a march element are performed at a certain address, before proceeding to the next address. The latter can be done in either an increasing ($\Uparrow$) or a decreasing ($\Downarrow$) address order. When the address order is not relevant, the symbol $\Updownarrow$ is be used.

As mentioned previously, the set of used BTs consists of three FP-based BTs (theoretically derived) and only 12 well-known traditional BTs. The BTs with the most promising fault coverage and unique fault detection are discussed here [1], [2], [7], [17], [29], [31], [32].

1) FP-Based BTs: The FP-based BTs consist of three march tests listed in the first block of Table I.

- March SS [20] to target all *simple static* memory faults. *Static faults* are faults sensitized by performing *at most* one operation (e.g., the state of the cell is always stuck at *one*, a read operation to a certain cell causes that cell to flip). *Simple faults* are faults which cannot influence the behavior of each other. That means that the behavior of a simple fault cannot change the behavior of another one, and therefore *masking* cannot occur.
- March RAW [19] to target some *dynamic* faults. *Dynamic faults* are faults that can only be sensitized by performing more than one operation *sequentially* (e.g., two *successive* read operations cause the cell to flip, however, if only one read operation is performed, the cell will not flip [3], [11]). March RAW is designed to target dynamic faults caused by read-after-write operations, which have been observed in real designs [19].
- March SL [21] to target all *simple linked* faults. *Linked faults* are faults that do influence the behavior of each other [4], [28], [31]. That means that the behavior of a certain fault can change the behavior of another, such that *masking*

TABLE I
DESCRIPTION OF THE USED BTS

Test	TL	Description
March SS [20]	$22n$	$\{\Updownarrow(w0); \Uparrow(r0,r0,w0,r0,w1); \Uparrow(r1,r1,w1,r1,w0); \Downarrow(r0,r0,w0,r0,w1); \Downarrow(r1,r1,w1,r1,w0); \Updownarrow(r0)\}$
March RAW [19]	$26n$	$\{\Updownarrow(w0); \Uparrow(r0,w0,r0,r0,w1,r1); \Uparrow(r1,w1,r1,r1,w0,r0); \Downarrow(r0,w0,r0,r0,w1,r1); \Downarrow(r1,w1,r1,r1,w0,r0); \Updownarrow(r0)\}$
March SL [21]	$41n$	$\{\Updownarrow(w0); \Uparrow(r0,r0,w1,w1,r1,r1,w0,w0,r0,w1); \Uparrow(r1,r1,w0,w0,r0,r0,w1,w1,r1,w0); \Downarrow(r0,r0,w1,w1,r1,r1,w0,w0,r0,w1); \Downarrow(r1,r1,w0,w0,r0,r0,w1,w1,r1,w0)\}$
SCAN [1]	$4n$	$\{\Uparrow(w0); \Uparrow(r0); \Uparrow(w1); \Uparrow(r1)\}$
MATS+ [27]	$5n$	$\{\Updownarrow(w0); \Uparrow(r0,w1); \Downarrow(r1,w0)\}$
MATS++ [12]	$6n$	$\{\Updownarrow(w0); \Uparrow(r0,w1); \Downarrow(r1,w0,r0)\}$
March C- [25]	$10n$	$\{\Updownarrow(w0); \Uparrow(r0,w1); \Uparrow(r1,w0); \Downarrow(r0,w1); \Downarrow(r1,w0); \Updownarrow(r0)\}$
PMOVI [15]	$13n$	$\{\Downarrow(w0); \Uparrow(r0,w1,r1); \Uparrow(r1,w0,r0); \Downarrow(r0,w1,r1); \Downarrow(r1,w0,r0)\}$
March SR [18]	$14n$	$\{\Downarrow(w0); \Uparrow(r0,w1,r1,w0); \Uparrow(r0,r0); \Uparrow(w1); \Downarrow(r1,w0,r0,w1); \Downarrow(r1,r1)\}$
March G [30]	$23n$	$\{\Updownarrow(w0); \Uparrow(r0,w1,r1,w0,r0,w1); \Uparrow(r1,w0,w1); \Downarrow(r1,w0,w1,w0); \Downarrow(r0,w1,w0); \Uparrow(r0,w1,r1); \Uparrow(r1,w0,r0)\}$
Hammer [32]	$49n$	$\{\Uparrow(w0); \Uparrow(r0,10*w1,r1); \Uparrow(r1,10*w0,r0); \Downarrow(r0,10*w1,r1); \Downarrow(r1,10*w0,r0)\}$
GalColumn[12]	$6n+4nR$	$\{\Uparrow(w0); \Uparrow_b(w1_b, \Uparrow_{col}(r0,r1_b), w0_b); \Uparrow(w1); \Uparrow_b(w0_b, \Uparrow_{col}(r1,r0_b), w1_b)\}$
GalRow [12]	$6n+4nC$	$\{\Uparrow(w0); \Uparrow_b(w1_b, \Uparrow_{row}(r0,r1_b), w0_b); \Uparrow(w1); \Uparrow_b(w0_b, \Uparrow_{row}(r1,r0_b), w1_b)\}$
WalkColumn [31]	$8n+2nR$	$\{\Uparrow(w0); \Uparrow_b(w1_b, \Uparrow_{col}(r0), r1_b, w0_b); \Uparrow(w1); \Uparrow_b(w0_b, \Uparrow_{col}(r1), r1_b, w0_b)\}$
WalkRow [31]	$8n+2nC$	$\{\Uparrow(w0); \Uparrow_b(w1_b, \Uparrow_{row}(r0), r1_b, w0_b); \Uparrow(w1); \Uparrow_b(w0_b, \Uparrow_{row}(r1), r1_b, w0_b)\}$

can occur. Masking makes the testing of linked faults very complex as compared with testing of simple faults.

2) Traditional BTs: A set of 12 well-known BTs has been selected, with the most promising fault coverage and unique faults detected. These BTs are listed in the second block of Table I. For Hammer, the notation $10*w1$ means that the write 1 operation is performed 10 times successively to the same cell. Two versions of Galpat and of Walking 1/0 tests are used, each with a complexity of $O(n \cdot \sqrt{n})$. As an example, the read operation in GalColumn is restricted to only the cells in the same column as the *base cell (b)*, instead of galloping throughout the whole memory. In these tests, the notation $\Uparrow_b$ means to go through all the bit of the memory in an incrementing fashion, while considering the current cell as the base cell b. For GalRow, the notation $\Uparrow_{row}\ (r0, r1_b)$ means to apply a $r0$ (read 0) operation in an incrementing order to the cells of the row of the *base cell*, and apply $r1$ (read 1) operation to the base cell *after each* $r0$ operation. A similar explanation applies to $\Uparrow_{col}\ (r0, r1_b)$ in GalColumn. Similarly, for WalkRow and WalkColumn, the notation $\Uparrow_{row}\ (r0)$ ($\Uparrow_{col}\ (r0)$) means apply a $r0$ operation using an incrementing address order to the row (column) of the base cell, and skip the base cell.

B. Used Stresses

Each BT has to be applied using several different *stress combinations (SCs)*. An SC specifies the way the test is performed and, therefore, it influences the sequence and/or the type of the memory operations. The used SCs are the *addressing directions* and the *data-backgrounds*.

The used addressing directions consist of fx and fy:

"Fast x" (fx): "Fast x" addressing is simply incrementing or decrementing the address in such a way that each step goes to the next row.

"Fast y" (fy): "Fast y" addressing is simply incrementing or decrementing the address in such a way that each step goes to the next column.

TABLE II
LIST OF THE USED BTS AND THEIR STRESS COMBINATIONS

#	Base Test (BT)	TT/SC (ms)	# SC	Stress combination							
				fx				fy			
				s	c	cs	rs	s	c	cs	rs
1	GalColumn	21.217	1	+	-	-	-	-	-	-	-
2	GalRow	1.556	1	-	-	-	-	+	-	-	-
3	Hammer	2.000	1	+	-	-	-	-	-	-	-
4	March C-	0.410	6	+	-	+	+	+	-	+	+
5	March G	0.942	2	+	-	-	-	+	-	-	-
6	MATS+	0.205	2	+	-	-	-	+	-	-	-
7	MATS++	0.246	2	+	-	-	-	+	-	-	-
8	PMOVI	0.532	8	+	+	+	+	+	+	+	+
9	March RAW	1.065	8	+	+	+	+	+	+	+	+
10	Scan	0.164	4	-	+	+	+	-	+	-	-
11	March SL	1.679	8	+	+	+	+	+	+	+	+
12	March SR	0.573	8	+	+	+	+	+	+	+	+
13	March SS	0.901	8	+	+	+	+	+	+	+	+
14	WalkColumn	10.813	1	+	-	-	-	-	-	-	-
15	WalkRow	0.983	1	-	-	-	-	+	-	-	-

A *data-background (DB)* is defined as the pattern of ones and zeros as seen in an array of memory cells. The used DBs are as follows.

1) *Solid (s)*: All 0s, all 1s.
2) *Checkerboard (c)*: $0101\ldots/1010\ldots/0101\ldots/1010\ldots$.
3) *Column stripe (cs)*: $0101\ldots/0101\ldots/0101\ldots/0101\ldots$.
4) *Row stripe (rs)*: $0000\ldots/1111\ldots/0000\ldots/1111\ldots$.

Table II lists the 61 tests applied at both high voltage and low voltage. A *test* consists of a BT (i.e., test algorithm) applied using a particular SC. The total number of tests is, therefore, the number of BTs (15), multiplied by the corresponding number of SCs (#SC) and with two voltages (high and low), a total of $= \Sigma(BT_i \cdot (\#SC_i) \cdot 2) = 122$. The column "TT/SC" in Table II gives the test time, in milliseconds (ms), of each BT using a single SC for the tested chip. To calculate the test time per BT, the "TT/SC" has to be multiplied by "#SC" and with two (high and low voltage). The total test time of all tests is

TABLE III
UNION AND THE INTERSECTION FAULT COVERAGE OF BTS AT HVCC (TOTAL FC = 202)

#	BT Name	FC	UFs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	GalColumn	164	0	**164**	188	179	183	186	180	178	183	186	177	186	181	184	164	181
2	GalRow	176	9	152	**176**	192	192	*195*	189	188	190	193	190	*195*	192	193	188	178
3	Hammer	171	1	156	155	**171**	186	188	186	184	185	186	181	188	184	186	178	183
4	March C-	183	0	164	167	168	**183**	186	184	183	185	187	186	186	187	185	183	183
5	March G	185	0	163	166	168	182	**185**	186	186	186	188	188	188	189	187	186	186
6	MATS+	175	0	159	162	160	174	174	**175**	179	183	186	181	185	185	184	179	181
7	MATS++	177	0	163	165	164	177	176	173	**177**	183	186	181	186	185	184	177	181
8	PMOVI	181	0	162	167	167	179	180	173	175	**181**	184	185	187	187	185	183	181
9	March RAW	184	0	162	167	169	180	181	173	175	181	**184**	188	188	188	186	186	184
10	Scan	168	1	155	154	158	165	165	162	164	164	164	**168**	188	178	186	173	181
11	March SL	185	1	163	166	168	182	182	175	176	179	181	165	**185**	188	186	186	186
12	March SR	170	0	153	154	157	166	166	160	162	164	166	160	167	**170**	187	178	184
13	March SS	184	0	164	167	169	182	182	175	177	180	182	166	183	167	**184**	184	184
14	WalkColumn	160	0	160	148	153	160	159	156	160	158	158	155	159	152	160	**160**	181
15	WalkRow	168	0	151	166	156	168	167	162	164	168	168	155	167	154	168	147	**168**

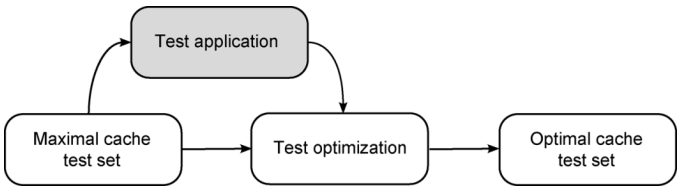


Fig. 4. Test application step of the cache test development process.

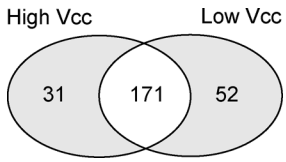


Fig. 5. Venn-diagram of the FC.

160.942 ms/chip, where the four nonlinear BTs consume about 43% of the total test time. In Table II, the solid, the checkerboard, column stripe, and row stripe data-background are denoted as "s," "c," "cs," and "rs," respectively. The different addressing orders are denoted as "fx" and "fy." A "+" in Table II indicates that the corresponding SC is applied, and a "−" denoted that it is not (e.g., WalkRow is used with fy (fast y) and *s* (solid) data-background). Due to test time constraints, only a subset of SCs have been selected for traditional BTs, while all SCs have been implemented for the FP-based BTs, as shown in Table II. The impact of SCs on the coverage of traditional BTs is not very interesting, since this has already been studied in detail and published many times in the literature [1], [2], [7], [17], [29], [31], [32].

IV. TEST APPLICATION RESULTS

The second step in the test optimization process is "test application," as indicated by the shaded block in Fig. 4. This section presents the results of running the tests in Table I on a huge number of Intel 512-kB caches. The exact number of caches tested is not given due to confidentiality reasons.

All SCs have been implemented at two different voltage levels: *high voltage (HVcc)* and *low voltage (LVcc)*. These voltages are generated externally by the tester and applied at the inputs of the microprocessor. Testing of the 512-kB caches resulted in the following.

- *HVcc testing:* 1545 chips failed, of which 1343 chips failed all 61 tests, and 202 chips failed only *some* tests.
- *LVcc testing:* 1543 chip failed, of which 1320 chips failed all tests, and 223 chips failed only *some* tests.

From now on, this paper will only concentrate on the chips that did not fail all tests, since they are the most interesting ones for further study and analysis.

Fig. 5 shows a *Venn-diagram* of the influence of the voltage levels on detectable faults, as derived from the database of the test results. The total number of devices found to be faulty is $31 + 171 + 52 = 254$. The *fault coverage (FC)* at HVcc testing is 202 out of 254, while the FC at LVcc testing is 223 out of 254. Note that 171 faults are detected at both LVcc and HVcc. In addition, 52 faults are detected at LVcc *only* while 31 faults are detected at HVcc *only*. This clearly explains the necessity of testing at both voltages in order to achieve a good FC. Low voltage testing is important for detecting faults caused by resistive bridges [13], [16], [29], while high voltage testing is important for detecting resistive open defects [9]–[11].

The FC of a BT is defined as the *union* of the fault coverages of its corresponding SCs. A die belongs to the union (i.e., considered detected by a BT) if at least one SC of that BT detects the die to be faulty. For example, MATS+ is implemented using fx-s (i.e., "fast x" and solid data-background) and fy-s. The fault is considered detected if at least one of the two MATS+ tests detects the fault (see Table II).

Table III shows the *unions* and the *intersections* of the 15 BTs for HVcc, while Table IV shows the results for LVcc. A die belongs to the union of two BTs if at least one of the two BTs finds the die to be faulty, and belongs to the intersection of two BTs if *both* BTs find the die to be faulty. The first column in each table gives the BT number, while the second column gives the name of the BT. The column "FC" lists the fault coverage of the corresponding BT, and the column "UFs" gives number of *unique faults (UFs)* each BT detects. Unique faults are faults that are only detected once by a single test. As an example of

TABLE IV
UNION AND INTERSECTION FAULT COVERAGE OF BTS AT LVCC (TOTAL FC = 223)

#	BT Name	FC	UFs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	GalColumn	184	0	**184**	212	210	215	204	199	197	205	212	195	211	205	209	186	202
2	GalRow	194	0	166	**194**	212	218	213	211	211	212	214	208	215	210	214	211	196
3	Hammer	199	0	173	181	**199**	217	216	211	210	214	215	208	216	212	217	209	209
4	March C-	215	0	184	191	197	**215**	218	215	215	219	*220*	215	219	217	218	215	215
5	March G	196	0	176	177	179	193	**196**	200	200	206	212	202	213	209	210	203	205
6	MATS+	191	0	176	174	179	191	187	**191**	193	204	211	196	209	204	208	196	202
7	MATS++	191	0	178	174	180	191	187	189	**191**	204	211	194	210	203	208	194	201
8	PMOVI	201	0	180	183	186	197	191	188	188	**201**	209	204	213	209	210	204	205
9	March RAW	208	0	180	188	192	203	192	188	188	200	**208**	211	213	212	213	211	208
10	Scan	189	0	178	175	180	189	183	184	186	186	186	**189**	210	201	208	194	199
11	March SL	209	0	182	188	192	205	192	191	190	197	204	188	**209**	210	211	210	211
12	March SR	193	0	172	177	180	191	180	180	181	185	189	181	192	**193**	209	202	204
13	March SS	208	0	183	188	190	205	194	191	191	199	203	189	206	192	**208**	208	210
14	WalkColumn	179	0	177	162	169	179	172	174	176	176	176	174	178	170	179	**179**	201
15	WalkRow	182	0	164	180	172	182	173	171	172	178	182	172	180	171	180	160	**182**

TABLE V
BTS DETECTING UNIQUE FAULTS

BT	HVcc		LVcc	
	FC	# UFs	FC	# UFs
GalRow	176	9	-	-
Hammer	171	1	-	-
Scan	168	1	-	-
March SL	185	1	-	-
Total		12		0

TABLE VI
REDUCED SET OF BTS WITH 100% FC

#	For HVcc	For LVcc	Type BT
1	GalRow	GalRow	$O(n.\sqrt{n})$
2	Hammer	Hammer	$O(n)$
3	March C-	March C-	$O(n)$
4	March G	March G	$O(n)$
5	-	PMOVI	$O(n)$
6	March RAW	March RAW	$O(n)$
7	Scan	-	$O(n)$
8	March SL	March SL	$O(n)$
9	March SR	March SR	$O(n)$
10	March SS	March SS	$O(n)$

unique faults at HVcc, GalRow detects nine UFs that are not detected with any other test.

The union and the intersection of each pair of BTs is shown in the rest of the tables. The numbers on the diagonal give the FC of the BTs, which are also listed in the column "FC" (for example, at HVcc, March SS has $\mathrm{FC} = 184$). The part above the main diagonal shows the union for each BT pair, while the part under the diagonal lists the intersection of each BT pair (for example, at HVcc the union of March C- and PMOVI is 185 and their intersection is 179). Based on the two tables and the Venn-diagram, one can conclude the following.

A. HVcc Testing

1) The total number of faulty chips detected is 202.
2) The best BTs, in terms of FC, are: March SL and March G with $\mathrm{FC} = 185$, March SS and March RAW with $\mathrm{FC} = 184$, and March C- with $\mathrm{FC} = 183$.
3) There are 12 unique faults, detected with four tests. These are listed in Table V, together with their FC and the number of unique faults (# UFs) each BT detects.
4) The best union pair in terms of the FC is 195 achieved with GalRow and March G, and with GalRow and March SL (see Table III).

B. LVcc Testing

1) The total number of faulty chips detected is 223.
2) The best BTs, in terms of FC, are: March C- with $\mathrm{FC} = 215$, March SL with $\mathrm{FC} = 209$, and March SS and March RAW with $\mathrm{FC} = 208$.
3) There are no unique faults detected at LVcc testing.
4) The best union pair in terms of the FC is 220 achieved with March C- and March RAW, see Table IV.

It is important to note here that the three FP-based BTs (i.e., March SS, March SL, and March RAW) score very high for both HVcc and LVcc testing.

Using Tables III and IV, it is possible to determine BTs detecting supersets of faults in comparison with other BTs in *this experiment*. For example, GalColumn detects a superset of WalkColumn at HVcc testing (see Table III). This is because the intersection of the two tests is 160 (which is the FC of WalkColumn), and their union is 164 (which is the FC of GalColumn). Keep in mind that in this experiment the number of stresses used with each BT is not the same for all BTs, see Table II. Determining the BTs detecting supersets allows for deriving a reduced set of BTs that has the same FC as the initial test set (see Table I). The reduced set is given in Table VI; it consists of nine BTs for HVcc as well as for LVcc, where eight BTs are common BTs.

V. TEST OPTIMIZATION

The third step in the cache test development process is "test optimization," as shown in the shaded block of Fig. 6, where the maximal test set is reduced based on the FC feedback from the test application step. In the following, the different BTs are analyzed and compared with each other first and then the impact of stress combinations (SCs) is analyzed.

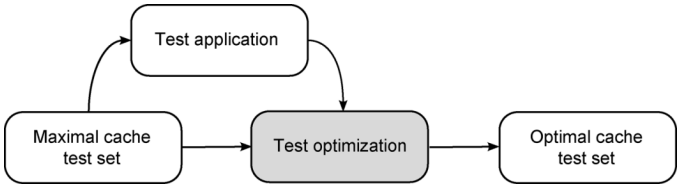


Fig. 6. Test optimization step of the cache test development process.

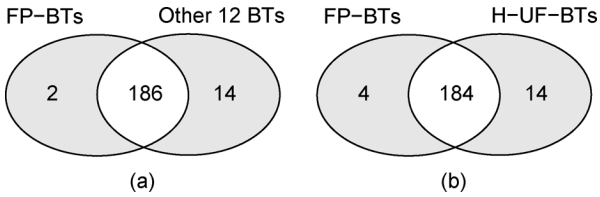


Fig. 7. FC of FP-based BTs at HVcc showing (a) total FC and (b) FC of tests detecting UFs.

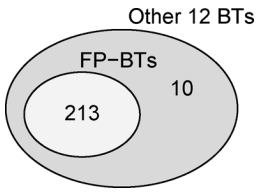


Fig. 8. FC of FP-BTs at LVcc.

A. Analysis of BTs

Here, the FC is evaluated of the three *FP-based BTs* (i.e., March SS, March SL, and March RAW, denoted as *FP-BTs*) and compare it with the FC of the other 12 BTs. One useful way to do that is to calculate the *union* of the FC of the FP-BTs and compare it with the union of the FC of the other 12 BTs.

1) Analysis of HVcc Testing: Fig. 7(a) shows the Venn-diagram of the FC union of the three FP-BTs as compared with the 12 traditional BTs (see Table II). The total FC is 202. Fig. 7(a) shows that 188 out of 202 faults can be detected with the FP-BTs only, while the other 12 BTs detect 200 out of the 202 faults. There are 14 faults that are not covered with the FP-BTs, 11 of them are unique faults (see Table V). Note that the total number of UFs is 12, and that March SL (an FP-BT) detects one of them.

Consider now the set of the three BTs shown in Table V, which detect UFs at HVcc (March SL is excluded), and let "H-UF-BTs" denote this set of BTs (i.e., $\text{H-UF-BTs} = \{\text{GalRow}, \text{Hammer}, \text{Scan}\}$). The analysis of the FC of H-UF-BTs reveals that the union of their FC is 198 out of 202 faults, as is shown in Fig. 7(b). In addition, the union of H-UF-BTs with the FP-BTs achieves 100% FC (i.e., 202 from 202). Note that 188 out of 202 faults are covered by the FP-BTs, and that the latter detect 4 faults that are missed by H-UF-BTs. Thus, the FC achieved with the initial test set of 15 BTs can also be achieved with a short test set consisting of six BTs: three FP-BTs and three H-UF-BTs.

Any fault detected with FP-BTs can (probably) be explained since these BTs target well-known predefined faults. However, most detected UFs (by empirical tests) cannot be explained with the well-known fault models. This means that additional faults exist which still should be modeled. The detected UFs call for a detailed analysis in order to understand the defect mechanisms behind them. A deep understanding of the defect mechanisms and their faulty behavior will allow for modeling the faults and for introducing shorter/optimal BTs that cover such faults.

2) Analysis of LVcc Testing: Fig. 8 shows the Venn-diagram of the FC of the three FP-BTs, as compared with the rest of 12 BTs at LVcc testing. All faults detected by the FP-BTs are also detected by the union of the other 12 BTs; these consist of 213 faults out of 223 (i.e., 95.51%).

As it has been shown in Section IV, there are no BTs detecting UFs at LVcc (see Table V). The question is now what are the faults missed by the FP-BTs, and which BTs (from the initial BT set) have to be added to the FP-BTs in order to achieve the complete FC (i.e., 223/223). A detailed analysis showed that a least Hammer should be added. The next question is then which kind of faults Hammer detects, and how they can be modeled. These questions remain still to be worked out.

Based on the previous analysis, one can derive an optimal set of BTs detecting all faults at HVcc, as well as at LVcc (see Table VII). Testing at HVcc requires 6 BTs and at LVcc requires 4 BTs; 4 BTs are common. Inspecting the table reveals that some of the BTs are *empirical* tests (e.g., GalRow, Hammer), not designed to target well-defined faults models. Such tests detect faults that cannot be explained with well-know fault models, and still remain to be understood and to be modeled. This will allow for developing low-cost fault model-based tests.

TABLE VII
OPTIMAL BTS SET ACHIEVING 100% FC

BT	Test length	HVcc		LVcc	
		BT	FC	BT	FC
GalRow	$6n + 4nR$	+	176	-	-
Hammer	$49n$	+	171	+	199
March RAW	$26n$	+	184	+	208
Scan	$4n$	+	168	-	-
March SL	$41n$	+	185	+	209
March SS	$22n$	+	184	+	208

TABLE VIII
MINIMAL SET OF SCS FOR FP-BTS

SC	March SS		March RAW		March SL	
	HVcc	LVcc	HVcc	LVcc	HVcc	LVcc
(fx,c)	171	**186**	175	**183**	175	**188**
(fx,cs)	170	**187**	(**178**)	196	**176**	**191**
(fx,rs)	171	(**193**)	**181**	**200**	173	(**191**)
(fx,s)	170	183	175	184	**174**	183
(fy,c)	179	188	(**175**)	**185**	179	(**187**)
(fy,cs)	**180**	(**195**)	(**178**)	**197**	182	(**196**)
(fy,rs)	**182**	**195**	(**181**)	196	(**180**)	193
(fy,s)	177	**199**	(**177**)	175	(**179**)	**199**
Total	2	5	2	4	3	4

B. Impact of SCs on BTs

In order to identify the best SCs needed to maximize the FC of the three FP-BTs, the impact of the SCs on these three tests is discussed here. The results of a detailed analysis of the SCs are summarized in Table VIII, where the FC of each SC is listed along with the three FP-BTs. Table VIII also lists the minimal number of SCs to be used with each of the three FP-BTs in order to achieve 100% FC. The minimal SCs that have to be

TABLE IX
LIST OF MINIMAL STRESS COMBINATIONS

#	Base Test (BT)	TT/SC (ms)	# SC		Stress combination							
					fx				fy			
			HVcc	LVcc	s	c	cs	rs	s	c	cs	rs
1	GalRow	1.556	1	0	-	-	-	-	H	-	-	-
2	Hammer	2.000	1	1	HL	-	-	-	-	-	-	-
3	March RAW	1.065	2	4	-	L	-	HL	-	L	HL	-
4	Scan	0.164	3	0	-	-	H	H	-	H	-	-
5	March SL	1.679	3	4	H	L	HL	-	-	-	L	HL
6	March SS	0.901	2	5	-	L	L	-	L	-	HL	HL

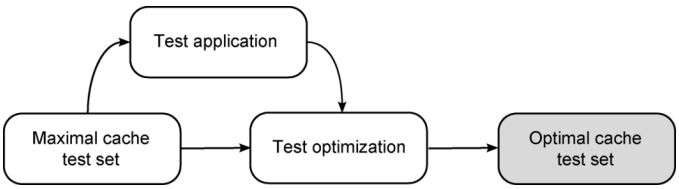


Fig. 9. Formulation of the optimal cache test set.

used with each FP-BT are shown in bold. For example, at HVcc testing March SS requires the use of only two SCs: $(\mathrm{fy}, \mathrm{cs})$ and $(\mathrm{fy}, \mathrm{rs})$. If the number is given between "( )," then it means that only one of such SCs is required. For example, March SL at LVcc requires the use of $(\mathrm{fx}, \mathrm{c})$, $(\mathrm{fx}, \mathrm{cs})$, $(\mathrm{fy}, \mathrm{s})$, and one of the following SCs: $(\mathrm{fx}, \mathrm{rs})$, $(\mathrm{fy}, \mathrm{c})$, or $(\mathrm{fy}, \mathrm{cs})$. Based on Table VIII we can conclude the following.

- Instead of using an initial set of 48 SCs for FP-BTs (i.e., # of SCs including HVcc and LVcc $\times$ the three $\mathrm{FP\text{-}BTs} = 16 \times 3$), one can only use 20 SCs while achieving the same FC: 7 SCs at HVcc and 13 SCs at LVcc.
- For achieving a 100% FC, the number of SCs required at HVcc is much smaller than that required at LVcc. For example, March SS requires only two SCs at HVcc, while it requires 5 SCs at LVcc (see last row of Table VIII).
- Generally speaking, using FP-BTs with fy addressing scores better than with fx addressing.
- A special analysis showed the following.
  - For March SS: $(\mathrm{fy}, \mathrm{rs})$ covers $(\mathrm{fx}, \mathrm{c})$, $(\mathrm{fx}, \mathrm{cs})$, $(\mathrm{fx}, \mathrm{rs})$, and $(\mathrm{fx}, \mathrm{s})$ at HVcc testing.
  - For March RAW: $(\mathrm{fy}, \mathrm{rs})$ covers $(\mathrm{fy}, \mathrm{s})$ irrespective of the voltage at which testing is done. In addition, at HVcc testing $(\mathrm{fy}, \mathrm{rs})$ also covers $(\mathrm{fx}, \mathrm{cs})$ and $(\mathrm{fy}, \mathrm{cs})$.
  - For March SL: $(\mathrm{fx}, \mathrm{cs})$ covers $(\mathrm{fx}, \mathrm{rs})$, and $(\mathrm{fy}, \mathrm{cs})$ covers $(\mathrm{fy}, \mathrm{c})$ at HVcc testing.

VI. OPTIMAL TEST SET AND SCS

The last step in the cache test development process is the formulation of the optimal test set to be used for high volume manufacturing (HVM) testing of the memories, as represented by the shaded block in Fig. 9.

It has been shown in Section V-A that in order to achieve the same FC as that of the initial 15 BTs (with a total of 122 SCs) only a minimal set of six BTs is required (see Table VI). In order to get an idea about the impact of selecting appropriate SCs on the overall test time while keeping the same FC, the minimal number of SCs that have to be used with the minimal test set (i.e., six BTs) will be presented.

Table IX gives the SCs needed to be used with each of the six BTs. The column "TT/SC" lists the test time of each BT per SC. The column "#SC" gives the number of SCs each BT requires at HVcc and LVcc. For example, March SS has to be used with 2 SCs at HVcc and 5 SCs at LVcc. An "HL" in Table IX denotes that the SC is used both at HVcc and LVcc, an "L" only at LVcc, an "H" only at HVcc, and a "–" not used. For example, Hammer is used only with $(\mathrm{fx}, \mathrm{s})$ at HVcc and LVcc. The minimal number of SCs, required to achieve the FC achieved with the initial 122 SCs, is only 26: 12 SCs at HVcc and 14 SCs at LVcc. Note that Scan was initially used with 4 SCs at HVcc and at LVcc. However, the impact of the stress on the FC at HVcc showed that only three SCs are required in order to achieve the same FC. At LVcc, Scan is not required (see also Table VII). The required test time for the initial test set was 160.942 ms/chip, however, with the optimal test set, the required test time is just 30.498 ms/chip (i.e., a reduction factor of 5.3).

The previous clearly indicates the importance of test optimization and the overall test time reduction. Optimizing the test set means, in addition to selecting appropriate BTs, also selecting the minimal number of SCs that has to be associated with each BT in order to achieve the maximal FC.

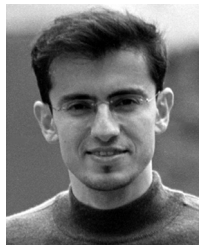
VII. CONCLUSION

This paper presented the process of test set development for on-chip caches of Intel microprocessors, based on the kitchen sink principle. There are four main steps to develop an optimized test set: maximal test set generation, test application, test optimization, and optimal test set generation. This paper also discussed an example of implementing this process in a high volume manufacturing environment. The example shows the way to optimize the test time of an initial test set of 15 base tests, each with up to 16 stress combinations, resulting in a total of 122 tests. Test set optimization resulted in a minimal set of only six base tests instead of 15. In addition the test time has been reduced from a maximum of 160.942 ms/chip, to an optimal test time of just 30.498 ms/chip (a reduction factor of about 5.3).

REFERENCES

[1] M. S. Abadir and J. K. Reghbati, "Functional testing of semiconductor random access memories," *ACM Comput. Surveys*, vol. 15, no. 3, pp. 175–198, 1983.
[2] R. D. Adams, *High Performance Memory Testing*. Norwell, MA: Kluwer, 2003.
[3] Z. Al-Ars and A. J. van de Goor, "Static and dynamic behavior of memory cell array opens and shorts in embedded DRAMs," in *Proc. Des., Autom. Test Euro.*, 2001, pp. 496–503.
[4] Z. Al-Ars, S. Hamdioui, and A. J. van de Goor, "A fault primitive based analysis of linked faults in RAMs," in *Proc. IEEE Int. Workshop Memory Technol., Des., Test.*, 2003, pp. 33–28.

[5] Z. Al-Ars and A. J. van de Goor, "Static and dynamic behavior of memory cell array spot defects in embedded DRAMs," *IEEE Trans. Comput.*, vol. 52, no. 3, pp. 293–309, Mar. 2003.
[6] Z. Al-Ars, S. Hamdioui, G. Mueller, and A. J. van de Goor, "Framework for fault analysis and test generation in DRAMs," in *Proc. Des., Autom. Test Euro.*, 2005, pp. 1020–1021.
[7] Z. Al-Ars, "DRAM fault analysis and test generation," Ph.D. dissertation, Comput. Eng. Lab., Delft Univ. Technol., Delft, The Netherlands, 2005.
[8] A. Allan, D. Edenfeld, W. H. Joyner, Jr., A. B. Kahng, M. Rodgers, and Y. Zorian, "2001 technology roadmap for semiconductors," *Computer*, vol. 35, no. 1, pp. 42–53, 2002.
[9] A. K. Majhi, M. Azimane, G. Gronthoud, M. Lousberg, S. Eichenberger, and F. Bowen, "Memory testing under different stress conditions: An industrial evaluation," in *Proc. Des., Autom. Test Eur.*, 2005, pp. 438–443.
[10] M. Azimane and A. K. Majhi, "New test methodology for resistive open defect detection in memory address decoders," in *Proc. IEEE VLSI Test Symp.*, 2004, pp. 123–128.
[11] S. Borri, M. Hage-Hassan, P. Girard, S. Pravossoudovitch, and A. Virazel, "Defect oriented dynamic fault models for embedded SRAMs," in *Proc. Euro. Test Symp.*, 2003, pp. 23–28.
[12] M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems*. Woodland Hills, CA: Computer Science, 1976.
[13] J.T.-Y. Chang and E. J. McCluskey, "Detecting delay flaws by very-low-voltage testing," in *Proc. Int. Test Conf.*, 1996, pp. 367–376.
[14] Y.-S. Chang, S. Chakravarty, H. Hoang, N. Thorpe, and K. Wee, "Transition tests for high performance microprocessors," in *Proc. IEEE VLSI Test Symp.*, 2005, pp. 29–34.
[15] J. H. de Jonge and A. J. Smeulders, "Moving inversions test pattern is thorough, yet speedy," *Comput. Des.*, pp. 169–173, 1976.
[16] B. Kruseman, A. Majhi, C. Hora, S. Eichenberger, and J. Mierlevede, "Systematic defects in deep-submicron technologies," in *Proc. Int. Test Conf.*, 2002, pp. 290–299.
[17] H. Goto, S. Nakamura, and K. Iwasaki, "Experimental fault analysis of 1 Mb SRAM chips," in *Proc. IEEE VLSI Test Symp.*, 1997, pp. 31–36.
[18] S. Hamdioui and A. J. van de Goor, "Experimental analysis of spot defects in SRAMs: Realistic fault models and tests," in *Proc. Asian Test Symp.*, 2000, pp. 131–138.
[19] S. Hamdioui, Z. Al-Ars, and A. J. van de Goor, "Testing static and dynamic faults in random access memories," in *Proc. IEEE VLSI Test Symp.*, 2002, pp. 395–400.
[20] S. Hamdioui, A. J. van de Goor, and M. Rodgers, "March SS: A test for all static simple RAM faults," in *Proc. IEEE Int. Workshop Memory Technol., Des., Test.*, 2002, pp. 95–100.
[21] S. Hamdioui, Z. Al-Ars, A. J. van de Goor, and M. Rodgers, "Linked faults in random-access-memories: Concept, fault models, test algorithms and industrial results," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 5, pp. 737–757, May 2004.
[22] S. Hamdioui, A. J. van de Goor, J. D. Reyes, and M. Rodgers, "Memory test experiment: Industrial results and data," *IEE Proc. Comput. Digit. Techn.*, vol. 153, no. 1, pp. 1–8, 2006.
[23] D. D. Josephson, S. Poehlman, V. Govan, and C. Mumford, "Test methodology for the McKinley processor," in *Proc. IEEE Int. Test Conf.*, 2001, pp. 578–585.
[24] S. Kundu *et al.*, "Test challenges in nanometer technologies," *J. Electron. Test.: Theory Appl.*, vol. 17, no. 3/4, pp. 209–218, 2001.
[25] M. Marinescu, "Simple and efficient algorithms for functional RAM testing," in *Proc. Int. Test Conf.*, 1982, pp. 236–239.
[26] M. Mayberry, J. Johnson, N. Shahriari, and M. Tripp, "Realizing the benefits of structural test for Intel microprocessors," in *Proc. IEEE Int. Test Conf.*, 2002, pp. 456–463.
[27] R. Nair, "Comments on "An optimal algorithm for testing stuck-at faults in random access memories"," *IEEE Trans. Comput.*, vol. 28, no. 3, pp. 258–261, Mar. 1979.
[28] C. A. Papachristou and N. B. Saghal, "An improved method for detecting functional faults in random access memories," *IEEE Trans. Comput.*, vol. 34, no. 2, pp. 110–116, Feb. 1985.
[29] I. Schanstra and A. J. van de Goor, "Industrial evaluation of stress combinations for march tests applied to SRAMs," in *Proc. Int. Test Conf.*, 1999, pp. 983–992.
[30] D. S. Suk and S. M. Reddy, "A march test for functional faults in semiconductor random access memories," *IEEE Trans. Comput.*, vol. 30, no. 12, pp. 982–985, Dec. 1981.
[31] A. J. van de Goor, *Testing Semiconductor Memories, Theory and Practice*, 2nd ed. Gouda, The Netherlands: ComTex Publishing, 1998.
[32] A. J. van de Goor and J. de Neef, "Industrial evaluation of DRAMs tests," in *Proc. Des., Autom. Test Euro.*, 1999, pp. 623–630.
[33] D. M. Wu, M. Lin, M. Reddy, T. Jaber, A. Sabbavarapu, and L. Thatcher, "An optimized DFT and test pattern generation strategy for an Intel high performance microprocessor," in *Proc. IEEE Int. Test Conf.*, 2004, pp. 38–47.



Zaid Al-Ars (M'04) received the M.Sc. and Ph.D. degrees in electrical engineering, both with honors, from the Delft University of Technology, Delft, The Netherlands.

He is currently working at the Delft University of Technology. He has more than six years of experience with industry and academia as a consultant and a researcher on test issues, in general, and memory testing, in particular. He spent a number of years with Infineon Technologies, Munich, Germany, where he was responsible for constructing new test methodologies to reduce the overall costs of their test flow. He is the author of numerous papers in the field of electrical defect simulation, fault modeling, and test generation in memory devices, most of which are based on cooperation with international industrial partners in the semiconductor industry, such as Intel, ST Microelectronics, and others. He is a member of the reviewing committees of various international conferences and journals. His research interests include systematic fault analysis and test generation and optimization for semiconductor memory products, design for testability, built-in self-testing and repair, logic testing, diagnosis, and IC reliability, and inductive fault analysis (IFA) and yield improvement of defective ICs.

Said Hamdioui (M'01) received the M.S.E.E. and Ph.D. degrees (both with honors) from the Delft University of Technology, Delft, the Netherlands.

He is currently with the Delft University of Technology. He has more than eight years of experience in industry and academia as a consultant and/or as researcher and developer on test issues in general and memory testing in particular. He spent a couple of years with Intel Corporation in Santa Clara and Folsom, California, where he was responsible for developing new low-cost and efficient test solutions for advanced Intel single-port and multiport embedded cache designs in the new generations of microprocessors. In addition, he spent more than one and half years with Philips Semiconductors in France and the Netherlands, where he was responsible for driving advanced product debug and yield ramp activities. He is the author of a book and about 40 conference and journal papers in the area of testing. His research interests include VLSI test and reliability, deep-submicron CMOS IC design and test, systematic fault modeling, test generation and optimization for semiconductor memories, design for testability, BIST, yield enhancement, product engineering, etc.

Dr. Hamdioui was the recipient of the European Design Automation Association (EDAA) Award for 2001.

Georgi Gaydadjiev (M'03) was born in Plovdiv, Bulgaria, in 1964.

He is currently an Assistant Professor with the Computer Engineering Laboratory, Delft University of Technology (TU Delft), Delft, The Netherlands. His research and development experience includes 15 years in hardware and software design at System Engineering Ltd., Pravetz, Bulgaria, and Pijnenburg Microelectronics and Software B.V., Vught, the Netherlands. His research interests include embedded systems design, advanced computer architectures, hardware/software co-design, VLSI design, cryptographic systems, and computer systems testing.

Stamatis Vassiliadis (F'97) was born in Manolates, Samos, Greece, in 1951.

He was a Chair Professor with the Electrical Engineering, Mathematics, and Computer Science (EEMCS) Department, Delft University of Technology (TU Delft), Delft, the Netherlands. He previously served in the Electrical Engineering faculties of Cornell University, Ithaca, NY, and the State University of New York (SUNY), Binghamton, NY. For a decade, he worked with IBM, where he was involved in a number of advanced research and development projects. His 72 U.S. patents rank him as the top all-time IBM inventor.

Dr. Vassiliadis was a recipient of numerous awards for his work, including 24 publication awards, 15 invention awards, and an outstanding innovation award for engineering/scientific hardware design. He received an honorable mention Best Paper Award at the ACM/IEEE MICRO25 in 1992 and Best Paper Awards in the IEEE CAS (1998), IEEE ICCD (2001), and PDCS (2002).