Efficient Tests and DFT for RAM Address Decoder Delay Faults

Said Hamdioui Zaid Al-Ars
Delft University of Technology/ Faculty of EE, Mathematics and CS
Computer Engineering Laboratory
Mekelweg 4, 2628 CD Delft, The Netherlands
E-mail: {S.Hamdioui,Z.Alars}@tudelft.nl

Abstract

This paper presents an analysis, at the electrical level, of address decoder faults caused by resistive opens within (a) dynamic address decoders and (b) static address decoders, which have special circuits that deactivate them at fixed moment. Efficient algorithms are proposed to cover the targeted faults. DFT circuit, to facilitate the BIST implementation of the proposed tests, is also provided. Furthermore, the limitations of the current/existing approaches in detecting delay faults are addressed.

Key words: *Dynamic address decoders, delay faults, memory testing, open defects.*

1. Introduction

Faults within *Random Access Memories (RAMs)* can be divided into faults in the memory cell array, in the address decoders, and in the read/write logic. Much has been published on functional fault models and tests for faults in the memory cell array [1, 2, 3, 4, 5], because, for larger memories, the memory cell array occupies most of the area of a memory die. Faults in the address decoders and address decoder paths, so-called *address decoder faults*, have only gotten limited attention. Several authors have shown the importance of this class of faults [6, 7, 8, 9, 10, 11]. Most authors have solved the problem of detecting *Delay Faults* in the address decoders, by using a test called *Moving Inversion 'MOVI'* [8, 9]. [7] even uses the time consuming GalPat test [4]. [6] has solved the problem by adding a decoder specific set of patterns to an existing march test. [10] reported that traditional march tests may cover the address decoder delay faults when varying the duty cycle of the internal clock of the address decoder. [11] modified the known March C- test to target these faults, while [13] developed a systematic approach to address such faults and proposed a set of tests for *static* address decoders. However, all developed solutions targeting address decoder delay faults are complex, time consuming.

This paper presents an analysis, at the electrical level, of address decoder faults caused by resistive opens within *dynamic* address decoders and *static* address decoders which have special circuits that deactivate them at *fixed* moment; such designs are dominating today's memory designs. Detection conditions and new efficient and less time-consuming tests are derived to detect the resulting faults. Due to the nature of the considered address decoder designs, the test procedure for delay faults can be dramatically simplified as will be shown in the paper.

The paper is organized as follows. Section 2 addresses the causes of delay faults in address decoders. Section 3 describes the detection conditions for such delay faults. Section 4 derives the tests. Section 5 gives a BIST circuit for the newly introduced tests, while Section 6 addresses the limitations of existing approaches in detecting delay faults and gives some directions. Section 7 ends with the conclusions.

2. Delays faults in address decoders

Opens are the major cause of delays in the address decoder paths, and can therefore cause *Address decoder Delay Faults*. Figure 1 shows a sequence of memory accesses, accessing memory locations with a *good Word Line* 'WLg' and a potentially *faulty* WL 'WLf'. In case of an ADF, the activation and/or the deactivation of WLf will de delayed, causing an *Activation Delay 'ActD'* fault and/or a *Deactivation Delay 'DeactD'* fault.

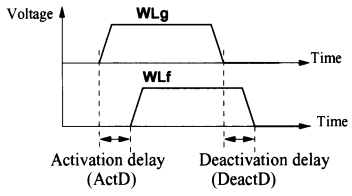

Figure 1. Activation and deactivation delays

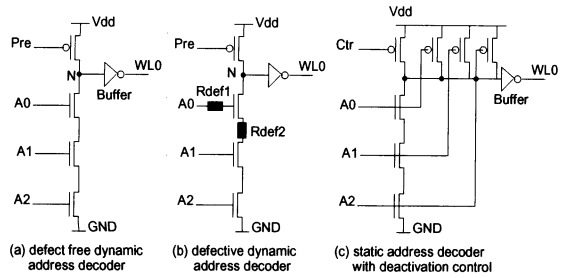

Figure 2. Typical dynamic address decoder

Figure 2(a) depicts a part of a typical CMOS dynamic address decoder. Before an address A_0, A_1, A_2 is presented on the input of the decoder to select WL0, the decoder node 'N' will be precharged using the signal 'Pre'. To select WL0, the address $A_0, A_1, A_2 = 111$ is required. In that case, the three pull-down transistors will be on and the node N will be discharged to GND; hence turning WL0 on.

In the decoder of Figure 2(a), defects can cause resistive opens at one of the following locations:

- **Within address path/line**. These are defects that can occur in address lines (e.g., A_0, A_1, A_2). Note that these address lines can be connected to more than one transistor in a real decoders since many word lines have to be decoded. The defect $Rdef1$ in Figure 2(b) shows a defect in address line A_0.
- **Within Vdd-GND path**. These are defects that cause e.g., the discharging path (node 'N' to GND) to be defective. They occur within the connections of the different pull-down transistors. The defect $Rdef2$ in Figure 2(b) shows a defect of such type.

Let us consider the impact of defects within address lines on the address decoder functionality. For sufficiently *high* values of $Rdef1$, the address line will behave as an open connection. Depending on the initial voltage of the floating gate, the transistor will be either always on or always off. This faulty behavior is easy to detect and belong to the conventional address decoder faults [4].

When the value of $Rdef1$ has an *intermediate* value, it will cause a delay in *rising edge* (i.e, ActD) of WL0. This will occur iff the address transition $A_2A_1A_0 = xy0 \rightarrow A_2A_1A_0 = 111$ (where $x, y \in \{0,1\}$) takes place. Because of the defect, when the address $A_2A_1A_0$=111 (after $A_2A_1A_0 = 0xy$) is put on the address lines, the pull-down transistor connected to the address line A_0 will require more time than normal to be turned on. Hence, the discharge of the node 'N' will be delayed. As a consequence, the WL0 will suffer from an ActD. Since the control signal "Pre" will be turned on time (and by assuming that the timing signal does not suffer from defects), WL0 will be put to 0 without any delay. Therefore $Rdef1$ does not cause any DeactD. The assumption is based on the fact precharging is usually done using multiple transistors/contacts to ensure proper deactivation of the word line.

Let us now consider the impact of defects with the discharge path; see $Rdef2$ in Figure 2(b). For sufficiently *high* values of $Rdef2$, the defect will causes the well-known CMOS Stuck-Open fault [12]; this case is a subset of the case when $Rdef2$ is intermediate.

When the value of $Rdef2$ has an *intermediate* value, it will cause a delay in the *rising edge* (i.e, ActD) of WL0. This will occur iff the address transition $A_2A_1A_0 = xyz \rightarrow A_2A_1A_0 = 111$ (where $x, y, z \in \{0,1\}$) takes place. Because of the defect, when the address $A_2A_1A_0$=111 is put on the address lines and the control signal "Pre" is put high, the node N is supposed to be discharged within the timing specification limits. However, because of the defect in the path, the discharge will be delayed and will need more time than the specification. As a consequence, the activation of WL0 will be delayed meaning that WL0 will suffer from ActD. Note that $Rdef2$ will cause no DeactD in WL0 for similar reasons as for $Rdef1$. Note also that the requirements for generating the needed address transition for ActD caused by defects within the discharge path is less severe than those required in case of defects within address lines.

In conclusion, resistive defects (with intermediate values) within both address lines and the Vdd-GND path will cause ActD in dynamic address decoders. However, they will cause no DeactD due to the nature of the decoder design.

It is worth to note that even *static* address decoders may only suffer from ActD faults and *not* from DeactD faults, exactly as it is the case for *dynamic* address decoders. This is the case when static address decoders have special circuits that deactivate them at a *fixed moment*. Figure 2(c) gives a typical static address decoders with fixed moment of deactivation; when the control signal Ctr is low, the word line will be deactivated. For such designs, all the theory and results presented in this paper are also applicable.

Simulations have been performed to validate the delay fault models. A similar row decoder has been considered for the simulation; it is mainly based on Infineon 0.18μm *e*DRAM technology. The simulation has been done while considering a defect in the very last stage of the row decoder im-

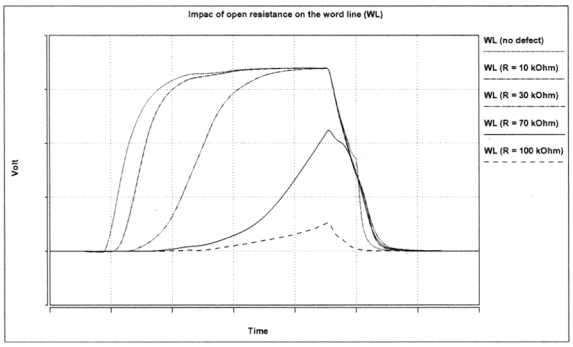

Figure 3. Impact of Rdef on WL timing

pacting the timing of the word line 'WL'. Figure 3 shows five WL waveforms with gradually increasing open defect values: $Rdef1 = 0\Omega$, $10K\Omega$, $30K\Omega$, $70K\Omega$, and $Rdef1 = 100K\Omega$. The defect clearly causes the ActD but not DeactD because the considered implementation of the row decoder has a special circuit/ control signal that deactivates the WL at a *fixed moment*; e.g., 'Pre' signal in Figure 2. Note that for large values of the defect, the word line will be always off; this fault belongs to the static/conventional address decoder faults [4].

3. Detection conditions

In case of static/traditional address decoder faults [4], it is assumed that the faults are detectable using read and write memory operations, applied using a particular *address order (AO)*. However, the sensitization of delay faults in the address decoders (e.g., ActD) is more complex and has two requirements:

- *Sensitizing address transitions*: these can be generated by an address pair or an address triplet. A *Sensitizing Address Pair 'SAP'* consists of a sequence of *two* addresses $\{Ag, Af\}$ or $\{Af, Ag\}$ of Figure 1, which have to be applied in sequence because delays are sensitized by *address transitions*. (Note: A_g is the address of WL_g and A_f is the address of WL_f). When the two SAPs $\{Ag, Af\}$ and $\{Af, Ag\}$ are applied in sequence, the *Sensitizing Address Triplet 'SAT'* $\{Ag, Af, Ag\}$ can be applied instead. This is more efficient because only three addresses have to be applied for a SAT, rather than four addresses when the two SAPs are applied.
- *Sensitizing operation sequences*: To each address of a SAP or a SAT at least one operation has to be applied, resulting in a *Sensitizing Operation Sequence 'SOS'* consisting respectively of 2 operations for a SAP, and 3 operations for a SAT.

Table 1. Address transitions for ActD faults

WL	L.	address transition
WL0	A_2	$\mathbf{0}xy \rightarrow 111$
	A_1	$x\mathbf{0}y \rightarrow 111$
	A_0	$xy\mathbf{0} \rightarrow 111$
WL1	A_2	$\mathbf{0}xy \rightarrow 110$
	A_1	$x\mathbf{0}y \rightarrow 110$
	$\overline{A_0}$	$xy\mathbf{1} \rightarrow 110$
...	...	...
WL7	$\overline{A_2}$	$\mathbf{1}xy \rightarrow 000$
	$\overline{A_1}$	$x\mathbf{1}y \rightarrow 000$
	$\overline{A_0}$	$xy\mathbf{1} \rightarrow 000$

3.1 Sensitizing address transitions

In the previous section, it has been shown that resistive defects (with intermediate values) within both address lines and Vdd-GND path will cause delay faults in dynamic address decoders; these faults are of type ActD (see Figure 1).

Defects within address path/line. For the defect $Rdef1$ within address line A_0, it has been shown that in order to sensitize ActD fault, the address transitions $xy0 \rightarrow 111$ of $A_2A_1A_0$ is required; this can be presented by the *Sensitizing Address Pair 'SAP'*= $\{xy0, 111\}$ with $x, y \in \{0, 1\}$. If the defect is in the address line A_1 or A_2, the sensitization of the ActD fault will require the address transitions $x0y \rightarrow 111$, respectively, $0xy \rightarrow 111$ of $A_2A_1A_0$ resulting into the two SAPs $\{x0y, 111\}$ and $\{0xy, 111\}$.

Thus the only requirement the SAPs have to satisfy for the detection of ActD is that an $x \rightarrow \overline{x}$ transition has to be made for the line containing *Rdef*; other lines also may, or may not, make a transition. Because *Rdef* can be present in any input of any address line, the set of SAPs has to contain $x \rightarrow \overline{x}$ transitions for each address line. Note that for an address decoder with e.g., three inputs $A_2A_1A_0$, there will be $2^3 = 8$ address lines, each corresponding to a binary combination of $A_2A_1A_0$. Table 1 summarizes the required address transitions for the three-bit address decoder; the part of the three-bit decoder decoding WL0 is shown in Figure 2. The table lists the required address transitions for sensitizing the ActD faults for resistive defects with address lines. E.g., an ActD in WL1 due to a resistive open in the line (denoted as 'L.') $\overline{A_0}$ requires the address transition $xy\mathbf{1} \rightarrow 11\mathbf{0}$

Based on the table, we can conclude that the required number of SAPs (i.e., address transitions) to sensitize all the ActD faults is $3 * 2^3 = 24$; i.e., the total number of addresses (2^3) times the number of inputs of the decoder. In general the required number of SAPs will be $N * 2^N$ where N is the number of inputs of the decoder. Since each SAP consists of two addressed, the total required addresses to

Table 2. Minimal set of SAT to sensitize ActD

Word lines	defect location	Required SAT
WL0 and WL7	any address line	$\{000, 111, 000\}$
WL1 and WL6	any address line	$\{001, 110, 001\}$
WL2 and WL5	any address line	$\{010, 101, 010\}$
WL3 and WL4	any address line	$\{011, 100, 011\}$

sensitize all ActD faults will be $2 * N * 2^N$. However, this number of addresses can be reduced by 'merging' SAPs required for the sensitization of ActD fault for each WL. For example, the three required address transitions/SAPs for WL0 (see Table 1) can be merged into one address transition $000 \rightarrow 111$. When using the 'merged' SAP based addressing, the total number of address (required for the sensitization of ActD faults) will be reduced from $2*N*2^N$ to $2 * 2^N$.

One can further reduce the number of required addresses by using *Sensitizing Address Triplets 'SAT'* instead of SAPs (Sensitizing Address Pair). When two SAPs, say $\{Ag, Af\}$ and $\{Af, Ag\}$, are applied in a sequence, they can be combined into a single SAT $\{Ag, Af, Ag\}$. For example, the SAP=$\{000, 111\}$ is required to sensitize ActD fault in WL0 and SAP=$\{111, 000\}$ is required to sensitize the ActD fault in WL7 (see Table 1). These two SAPs can be combined into SAT=$\{000, 111, 000\}$ sensitizing ActD faults in both WL0 and WL7. Table 2 give the required SATs to sensitize all ActD faults for a 3-bit address decoder. Using SATs, the total addresses required to sensitize all ActD faults in 3-bit address decoder will be $4 * 3 = 12$ (i.e., 4 is the total addresses divide by 2, and 3 is the number of addresses per SAT). In general, for a N-bit address decoder, the total required addresses to sensitize all targeted faults will be $3 * \frac{2^N}{2} = 3 * 2^{N-1}$; a reduction of 25% as compared with the total number of addresses required when using 'merged SAP' version. Section 5 will give a BIST implementation that can be used to generate the SAT based addressing.

Defects within Vdd-GND path. For the defect $Rdef2$ within the discharge path, it has been shown that in order to sensitize the ActD fault, the address transitions $xyz \rightarrow 111$ of $A_2A_1A_0$ is required; this can be presented by the *Sensitizing Address Pair 'SAP'*= $\{xyz, 111\}$ with $x, y, z \in \{0, 1\}$. Note that if the defects occurs between any two pull-down transistors, the required SAP remains the same. It is clear that the requirements for generating the needed address transition for ActD caused by defects within the discharge path are less severe and are a *subset* of those required in case of defects within address lines. Therefore, any addressing method that will guarantee the generation of address transition to sensitize ActD faults due to defects within address lines will also cover ActD faults due to defects within the discharge path; e.g. the SAT-based addressing method (see Table 2) will guarantee the sensitization of the ActD faults due to both defect classes.

Table 3. Read-Write-Sequences (SOT)

Name	RWS
RaRaR	$(rx_g, r\overline{x}_f, rx_g)$
RaRaW	$(wx_g, r\overline{x}_f, rx_g)$
RaWaR	$(rx_g, w\overline{x}_f, rx_g)$
WaRaR	$(rx_g, r\overline{x}_f, wx_g)$
RaWaW	$(wx_g, w\overline{x}_f, rx_g)$
WaRaW	$(wx_g, r\overline{x}_f, wx_g)$
WaWaR	$(rx_g, w\overline{x}_f, wx_g)$
WaWaW	$(wx_g, w\overline{x}_f, wx_g)$

3.2 Sensitizing Operation sequences

The required addresses to sensitize all targeted ActD faults in dynamic address decoders can be realized using SAT-based address method covered in the previous section. To specify a test, operations (i.e., read, write) have to be associated to generated addresses. As SAT consists of three addresses, say '$\{Ag, Af, Ag\}$', three operations are needed for each SAT; hence *Sensitizing Operation Triplet 'SOT'* is required. Note that the first address and the third address are the same; see also Table 2.

The requirement for the three operations to sensitize an ActD is:

- '$Ox_g, O\overline{x}_f, Ox_g$', where $x \in \{0, 1\}$ and $O \in \{r = read, w = write\}$. The operation on the Af has to be performed with the *complementary* of the data value applied to Ag in order for the fault to be sensitized. Because of ActD in e.g., Af, the operation may fail. The ActD fault will shorten the active period of word line (i.e., high; see Figure 3), which is required to perform an operation correctly. Since the timing of the WL is impacted, the operation may fail.

  Note: x should take on the value $x = 0$ *as well as* the value $x = 1$, because of the likely asymmetric sensitivities to the 0 and 1 state; this is an engineering requirement.

Depending on the selected operations in $O \in \{r, w\}$, eight *Read Write Sequences 'RWSs'* for SOTs are possible (see Table 3): RaRaR (i.e., Read-afer-Read-after-Read), RaRaW (i.e., Read-after-Read-after-Write), RaWaR, RaWaW, WaRaR, WaRaW, WaWaR and WaWaW. The RWSs for SOTs use addresses triplets 'Ag, Af, Ag' in order to allow for SATs; the first and the third addresses are identical. E.g., for WaRaR, first a 'rx_g' is applied to 'Ag', next a '$r\overline{x}_f$' is applied to 'Af', and last a 'wx_g' applied to 'Ag'.

Table 4. Tests based on SAT addressing and RWS

#	RWS	Time	Test description
1	WaWaW-based	$10n$	$\{\Uparrow^{SAT}(w0_g, w1_f, w0_g, \mathbf{r1_f}, \mathbf{r0_g}); \Downarrow^{SAT}(w1_g, w0_f, w1_g, \mathbf{r0_f}, \mathbf{r1_g})\}$
2	WaWaR-based	$12n$	$\{\Uparrow^{SAT}(w0_g, r0_g, w1_f, w0_g, \mathbf{r1_f}, \mathbf{r0_g}); \Downarrow^{SAT}(w1_g, r1_g, w0_f, w1_g, \mathbf{r0_f}, \mathbf{r1_g})\}$
3	WaRaW-based	$10n$	$\{\Uparrow^{SAT}(w1_f; w0_g, r1_f, w0_g, \mathbf{r0_g}); \Uparrow^{SAT}(w0_f; w1_g, r0_f, w1_g, \mathbf{r1_g})\}$
4	WaRaR-based	$12n$	$\{\Uparrow^{SAT}(w0_g, w1_f; r0_g, r1_f, w0_g, \mathbf{r0_g}); \Uparrow^{SAT}(w1_g, w0_f; r1_g, r0_f, w1_g, \mathbf{r1_g})\}$
5	RaWaW-based	$8n$	$\{\Uparrow^{SAT}(w0_g, w1_f, r0_g, \mathbf{r1_f}); \Downarrow^{SAT}(w1_g, w0_f, r1_g, \mathbf{r0_f})\}$
6	RaWaR-based	$10n$	$\{\Uparrow^{SAT}(w0_g; r0_g, w1_f, r0_g, \mathbf{r1_f}); \Downarrow^{SAT}(w1_g; r1_g, w0_f, r1_g, \mathbf{r0_f})\}$
7	RaRaW-based	$8n$	$\{\Uparrow^{SAT}(w1_f; w0_g, r1_f, r0_g); \Downarrow^{SAT}(w0_f; w1_g, r0_f, r1_g)\}$
8	RaRaR-based	$10n$	$\{\Uparrow^{SAT}(w0_g, w1_f; r0_g, r1_f, r0_g); \Downarrow^{SAT}(w1_g, w0_f; r1_g, r0_f, r1_g)\}$

4. Tests for address decoders

Based on the addressing method generated in Section 3.1 and the required RWSs generated in Section 3.2, tests for detecting delay faults in dynamic address decoders can be constructed. The results are given in Table 4. The first column lists the RWS on which the test is based, the second column the test length, while the third column gives the description of the test; in the test 'SAT' denotes the SAT-based addressing method described in Section 3.1.

Eight tests can be distinguished. They have been derived from the test structures of RWSs in Table 3, by repeating those for the data values $x = 0$ and $x = 1$, adapting the initializing operations to the appropriate addressing when needed; the initialization operations the first in the march elements and separated by ';' from the three operations of RWS. If the test is based on a RWS of the form XaYaR or of the form XaRaY (with X,Y $\in$ {R,W})) then the initialization of Ag, respectively, Af is required; this is because a cell can only be read after it is written. Note that the RaWaW-based test does not require any initialization. Moreover, operations to detect the faults are also added when needed (they are given in bold font). As an example, consider the WaWaR-based test. The test has to access the addresses generated in Section 3.1. Let's assume a 3 bit decoder. The test will first access cell with address 000 and initialize it to 0. Then, the same address will be read followed by writing the complementary data value to next address which is '111' (see Table 2), and then writing the initial address 000 to 0 again. In the presence of a delay fault, the write operation(s) may not succeed. The read operations to the two addresses will then detect the faults.

It is worth to note that all the three operations of RWS have to be applied *back-to-back*, as they are responsible for the sensitization of the delay faults together with SAT-based addressing method. Therefore, no delays and/or operations can be inserted between the RWS operations. This is the reason why the detection operations are added after the RWS operation. Note that the initialization and detection operations do not require back-to-back operations.

The main question is now which of the proposed eight tests is the best to use. It is known that the faults for memories consists also of faults within the memory array and faults within the peripheral circuits (e.g., sense amplifiers, precharge circuits, write drivers, etc). Hence, the choice of the appropriate test has to take into considerations the following: (a) either it is easy to merge with existing tests targeting other faults, or (b) easy to extend to cover additional faults.

Let's assume that we want to select delay faults test that can be extended in order to cover additional faults. For example *dynamic* peripheral circuits faults [14] which consist of: (a) *Slow Sense Amplifier Fault (SSAF)*, *Slow Precharge fault (SPRF)*, *Slow Write Driver Fault (SWDF)* and *Bit Line Imbalance Fault (BLIF)*. According to [14], any addressing method can be used to detect such faults, however special RWS are required. The detection of SSAF and SPRF requires WaW sequence (i.e, Write-after-write), while the detection of SWDF and BLIF requires RaW sequence; both of these RWSs have to be used with complementary data values. These two RWSs can be combined into single RWS with three operations; i.e., RaWaW. Hence one can use RaWaW-based test of Table 4 to cover both delay faults in the address decoder and dynamic peripheral circuit faults.

5. DFT and BIST feature

Figure 4 gives an BIST implementation example to generate the required SATs for an address decoder with $N = 10$. The design consists of binary counter, a modulo-3 counter and some logic gates. The binary counters is controlled by the modulo-3 counter; it is incremented after three clock cycles of the modulo-3 counter. For example, if the initial state of the binary counter is $b_0 b_1 ... b_9$ ='0000000000', then the generated address will be 000000000 in the first clock period, 111111111 in the second clock period and 000000000 in the third clock period. After the first clock period, c0 of the modulo-3 counter will be 1, therefore inverting all the output values of the binary counter using XOR gates. Once the binary

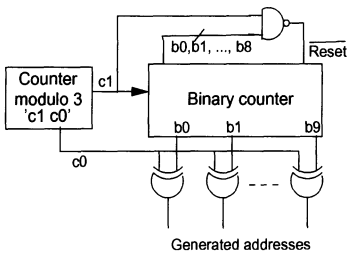

Figure 4. BIST implementation

counter reaches $b_0b_1...b_9$ = 0111111111' (i.e., $\frac{2^N}{2}$), the counter will be reseted when $c_1 = 1$. All the required addresses are then generated.

Note that if there are N address lines, only 2^{N-1} address triplets have to be generated. E.g., for $N = 3$, only $2^2 = 4$ SATs are needed (see Table 2)

6. Limitation of existing approaches

The traditional way of memory testing usually uses *pseudo-random addressing* or *binary addressing* as it is easy to generate. Although such addressings have been very useful to target traditional faults (e.g., stuck-at-fault, transition faults, coupling faults, etc), it is clear that they are inadequate to target delay (or time-related) faults. Delay/time-related faults are becoming very important with technology scaling and have to be considered for all serious test purposes (e.g., critical applications). Therefore address generators in today's memory BIST (e.g., Linear-Feedback Shift Register LFSR-based for pseudo-random addressing, counter-based for binary addressing) have to be made more flexible to allow for the generation of address transitions required for the detection of delay faults.

The generation of appropriate address transitions (used with read-write sequences) is *necessary*, but *not sufficient* to guarantee the detection delay faults in address decoders. The address transitions will sensitize the delay fault, and the read-write sequence has to be able to detect the fault. However, the detection will be guaranteed only of the delay is large enough to impact the active period of word line/ bit line resulting in an incorrect operation(s). For example, in Figure 3 one can see that the impact of the defect when $Rdef$=70KΩ is large and therefore the detection is guaranteed since the back-to-back operations of RWS will not succeed to correctly access the cell for which the word line has a delay. However, for a defect value of 10KΩ, the impact is small, therefore the operations may pass correctly and the fault will be not detected. In conclusion, if the impact of the defect is not large enough, the defect may escape the test resulting larger *Part-Per-Million PPM* level. These escapes may cause reliability problems in the field due e.g. to the fast degradation of the parameters of the design.

It is clear from the above that the traditional functional test approach has limited fault/defect coverage especially for advanced fault models (e.g., delay faults, time-related faults). Therefore new *structural* approaches have to be developed in order to guarantee the required product quality and reduce the escapes. Approaches like new DFT techniques and accelerated stresses combinations (to make the defect/fault more visible) could help in this.

7. Conclusions

In this paper delays in RAM address decoders are addressed. An analysis at the electrical level is presented. Using a systematic approach, the detection conditions for targeted delay faults are developed and compiled in a set of eight efficient test algorithms. The tests require special addressing sequences/transitions to sensitize the faults. Required DFT to generate such addressing is also provided. Finally, the papers discusses the limitation of the existing and the proposed approach in testing delay address decoder faults and give some recommendations.

References

[1] J.H. de Jonge and A.J. Smeulders, 'Moving Inversion Test Pattern is Thorough, Yet Speedy', Computer Design, May 1976, pp. 169-173

[2] R.D. Adams and E.S. Cooley, 'Analysis of deceptive destructive read memory fault model and recommended testing'. In Records of the North Atlantic Test Worshop, May 1996; pp. 27-32

[3] B. Nadeau-Dostie et al., 'Serial Interfacing for Embedded-Memory Testing', IEEE Design & Test of Comp., Vol. 7, No. 2, 1990, pp. 52-63

[4] A.J. van de Goor, 'Testing Semiconductor Memories: Theory and Practice', *ComTex Publishing, Gouda, The Netherlands*, 1998.

[5] S. Hamdioui, 'Testing Static Random Access Memories: Defects, Fault Models, and Test Patterns', *Kluwer Academic Publishers, Boston Hardbound*, ISBN 1-4020-7752-1, March 2004.

[6] M. Sachdev, "Open Defects in CMOS RAM Address Decoders", *IEEE Design & Test of Computers*, pp. 26-33, April - June 1997.

[7] S. Nakahara et al., 'Built-in self-test for GHz embedded SRAMs using flexible pattern generator and new repair algorithm', In Proc. IEEE Int. Test Conf., 1999, pp. 301-310

[8] M. Klaus and Ad J. van de Goor, 'Tests for Resistive and Capacitive Defects in Address Decoders', In Proc. of the Asian Test Symposium, 2001, pp. 31-36

[9] M.T. Fragano, J.H. Oppold, M.R. Ouellette and J.P. Rowland, 'Self-Test Pattern to Detect Stuck-Open Faults', US Patent No.: US 6,442,085 B1, Date of Patent: Aug. 27, 2002

[10] M. Azimane and A.K. Majhi, "New Test Methodology for Resistive Open Defect Detection in Memory Address Decoders", *In Proc. of IEEE VLSI Test Symposium*, pp. 123-128, 2004.

[11] L. Dilillo, et. al, "March iC-: An Improved Version of March C- for ADOFs Detection", *In Proc. of IEEE VLSI Test Symposium*, pp. 129-134, 2004.

[12] Niraj Jha and Sandeep Gupta, 'Testing of Digital Systems', Cambridge University Press, Cambridge, UK, 2003; ISBN 0-521-77356-3

[13] S. Hamdioui, Z. Al-Ars, A.J. van de Goor, "Opens and Delay Faults in CMOS RAM Address Decoder", *IEEE Transactions on Computers*, pp. 1630-1639, December 2006.

[14] A.J. van de Goor, S. Hamdioui, and R. Wadsworth, "Detecting Faults in the Peripheral Circuits and Evaluation on SRAM Tests", Proc. IEEE Int'l Test Conf., pp. 114-123, 2004.