An Investigation on Capacitive Coupling in RAM Address Decoders

Said Hamdioui Zaid Al-Ars Georgi N. Gaydadjiev Ad J. van de Goor
Delft University of Technology
Faculty of Electrical Engineering, Computer Science and Mathematics
Computer Engineering Laboratory
Mekelweg 4, 2628 CD Delft, The Netherlands
S.Hamdioui@ewi.tudelft.nl

Abstract

In this paper, a complete analysis of address decoder delay faults due to capacitive coupling between address lines is presented. Detection conditions are used to explore the space of possible tests in order to detect these faults, resulting in new tests. The best test is proposed to be combined with other tests (while using the freedom of march tests) to target other faults.

Key words: *memory testing, delay faults, capacitive coupling, address decoder faults.*

1 Introduction

Memory technology has been used for a long time to push the state of the art in the semiconductor industry. Because of the regular structure of memories, cell area optimization has a very high payoff such that transistor dimensions, widths, and distances between wires, are minimized. The consequence is a high sensitivity to defects such as opens, shorts, and bridges within the memory cell array and in other parts of the memory. In addition, memories have another special property, different from logic: they have signals with a very high fan out (e.g., word lines). The lines carrying those signals run across the memory area and therefore have, in addition to a high load, also a high capacitance. Process variations make them very sensitive to delay type behavior, because of their capacitive coupling with other signal, power and ground lines.

Figure 1 [1] shows an exponential worsening of crosstalk noise, wiring delay and Vss bounce, with each new generation of DRAM chips. In [2] Nigh reports that new technology generations will exhibit an increasing sensitivity to, and occurrence of, subtle defect types; many of which will cause additional circuit delays, while the increasing clock speeds will make designs more sensitive to these circuit delays. Needham [3] reports that opens were the most likely cause of field returns of Intel microprocesors in 2000, while Grundmann [4] reports that of the Intel microprocessor field returns in 2002, about 67% was due to capacitive coupling and 20% to power droops. Klaus [5] reports that tests for opens in DRAM address decoders reduce the DPM level by as much as 670 DPM. Li [6], Moore [7], Sachdev [8] and Hamdioui [9] have analyzed the detectability of opens in logic circuits and RAM address decoders. In conclusion, faults caused by delays are becoming a more dominant failure mode!

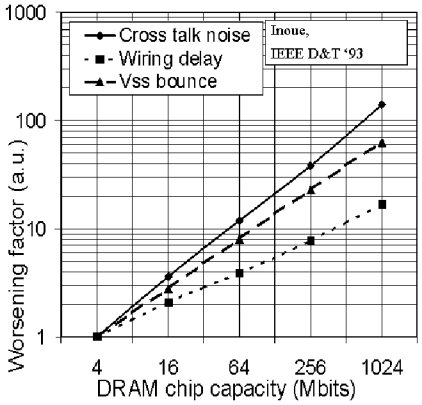


Figure 1. Worsening factors for DRAM technology generations

Delay faults in address decoder paths can be caused by different defect mechanisms. The major causes are: (a) Opens, (b) Resistive bridges and (c) Capacitive coupling. Delays faults due to opens and resistive bridges are addressed in many papers like in [5, 7, 8, 9]. This paper presents an analysis of capacitive coupling in address decoders in *Random Access Memories (RAMs)*; fault models and their appropriate test patterns are presented.

The paper is organized as follows. Section 2 discusses the capacitive coupling and give the fault models. Section 3 presents the detection conditions of such faults. Section 4 derives the tests. Section 5 compare and evaluate the existing tests and their capability in detecting capacitive coupling. Section 6 ends with the conclusions.

2. Capacitive coupling between address lines

With modern high-speed address decoders an additional source of inter-gate signal delays may be present in the form of capacitive coupling between signal lines [10, 11]. This is caused by the close proximity of the signal lines, and is aggravated by variations in the manufacturing process [12]. Consider the capacitively coupled lines Lv (the victim line) and La (the aggressor line) of Figure 2. The rise and fall times of Lv depend on the activity on line La as follows:

- *Nominal*: If line La does not make a transition; the timing of Lv is not affected (not shown in the figure).
- *Delayed*: Upon a simultaneous $x \rightarrow \overline{x}$ transition on Lv and an $\overline{x} \rightarrow x$ transition on La. The dashed lines in Figure 2 show the delayed rise and fall times of Lv. They cause an *Activation Delay 'Act***D***'* fault and a *Deactivation Delay 'Deact***D***'*; i.e., both are present.
- *Accelerated*: Upon simultaneous $x \rightarrow \overline{x}$ transitions on La and Lv; $x \in \{0, 1\}$. The dashed lines show the accelerated rise and fall times of Lv. They cause an *Activation Acceleration 'Act***A***'* fault and a *Deactivation Acceleration 'Deact***A***'*; i.e., both are present.

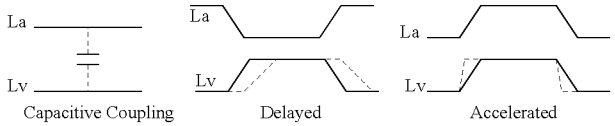


Figure 2. Address lines capacitive coupling

Sathe [11] states that the rise and fall times of Lv can change by a factor of 2 to 3, due to activities on La. Note that capacitive coupling can cause four different faults; each two occur at the same time; i.e., ActD with DeactD and ActA with DeactA. To understand the impact of such faults, consider for example the presence of ActD and DeactD in a certain address line Lv caused by the address line La. Assume that these lines are driving a word line WL. The presence of capacitive coupling will cause a delay fault in WL. Figure 3 presents the concept of a delay fault in word lines; it shows a sequence of memory accesses, sequentially accessing memory locations with a *good Word Line* 'WLg' with an address A_g, a potentially *faulty Word Line* 'WLf' with an address A_f. Due to the capacitive coupling, the activation and/or the deactivation of WLf will be delayed, causing an *Activation Delay 'ActD'* fault and/or a *Deactivation Delay 'DeactD'* fault in the word line.

Most companies involved in testing memories consider the likelihood of functional faults due to capacitive coupling to be an order of magnitude lower than that of inter-gate opens; hence, they do not apply special production tests for this. However, according to Kundu [12], manufacturing process variations in future technology generations will increase the probability of faults due to capacitive coupling.

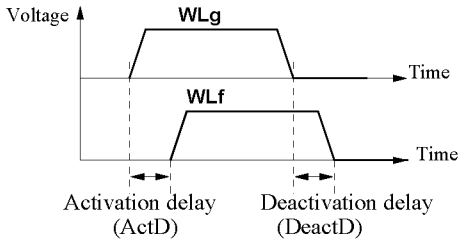


Figure 3. Activation and deactivation delays

3. Detection conditions for capacitive coupling

In case of functional address decoder faults, it is assumed that the fault is detectable using read and write operations, applied using a no specific address order [13]. However, the sensitization of address decoder delay and acceleration faults is more complex and has two requirements:

1. Sensitizing address transitions, and
2. Sensitizing operation sequences.

Sensitizing address transition(s) can be caused 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 3, which have to be applied in sequence because address decoder delay/acceleration faults are sensitized by *address transitions*. (Note: Ag presents the address of WLg and Af that of WLf). 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, rather than four addresses when the two SAPs are applied. SAPs/SATs are generated using an *Addressing Method 'AM'* as will be shown in Section 3.1.

Sensitizing Operation Sequence: 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, since a least one operation has to be applied to each address of a SAP and SAT. Sensitizing Operation Sequence will be discussed in Section 3.2.

3.1 Sensitizing address transitions

The addresses of the SAPs/SATs, required for sensitizing address decoder delay/acceleration faults, are generated using an *Addressing Method 'AM'*. An AM describes the method used for generating the sequence of addresses. A well-known AM is the *Binary AM 'Bin'*; for $N = 3$, it consists of the address sequences $\Uparrow^{Bin} = \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $\Downarrow^{Bin} = \{7, 6, 5, 4, 3, 2, 1, 0\}$.

Depending on the activity on the aggressor line La (see Figure 2), an activation or a deactivation delay can be caused on line Lv. Based on this, the detection conditions

for transition accelerations and transition delays, due to capacitive coupling between address lines, can be established:

1. Lv is only coupled to *one* neighboring line La.
   Detection condition: For all *pairs* of lines $\{Lv, La\}$, and for $x = 0$ and $x = 1$:
   - Lv makes an $x \rightarrow \overline{x}$, and La an $\overline{x} \rightarrow x$ transition (delayed), and
   - Lv makes an $x \rightarrow \overline{x}$, and La an $x \rightarrow \overline{x}$ transition (accelerated)

   This requires all address pairs with $H = 2$, where H stands for the *Hamming distance* between the two addresses say Aa (driving the line La) and Av (driving the line Lv) of the SAP $\{Aa, Av\}$. H is defined as the number of bit positions in which the addresses Aa and Av of an address-pair differ. Only AMs which generate SAPs or SATs with $H = 2$ can cover the ActD, DeactD, ActA and DeactA (see Section 2).
2. Lv is coupled to *two* neighboring lines $La1$ and $La2$ (Not shown in Figure 2).
   Detection condition: For all *triplets* of lines $\{Lv, La1, La2\}$, and for $x = 0$ and $x = 1$:
   - Lv makes an $x \rightarrow \overline{x}$, and $\{La1, La2\}$ an $\overline{x} \rightarrow x$ transition (delayed)
   - and Lv makes an $x \rightarrow \overline{x}$, and $\{La1, La2\}$ an $x \rightarrow \overline{x}$ transition (accelerated)

   This requires all address pairs with $H = 3$.

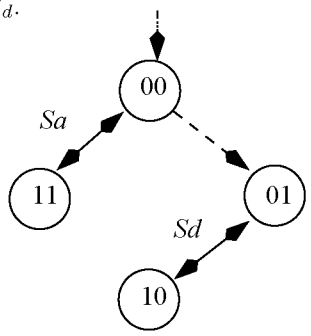
Below, descriptions of addressing methods, capable of sensitizing faults due to capacitive coupling, are given.

H2 addressing method

Figure 4a shows the state diagram for a 2-bit address sequence (i.e., $N = 2$) using the H2 AM; each address is represented by a node, and each address line by a bit; i.e., $L1, L0 \in \{0, 1\}$. The SATs for $\{La, Lv, La\}$ with $H = 2$ consist of the nodes connected by solid bi-directional arrows, labeled S_a and S_d. The SAT $S_a = \{00, 11, 00\}$ sensitizes *edge* **a***ccelerations*, because both lines, La and Lv, make $0 \rightarrow 1 \rightarrow 0$ transitions; the SAT $S_d = \{01, 10, 01\}$ sensitizes *edge* **d***elays*. Figure 4b shows the address sequence; '#' is the step number in the sequence, columns '$L1L0$' list the address sequence, column 'G' denotes the size of the **G**roup, and column 'S' lists whether an edge acceleration fault 'S_a' or an edge delay fault 'S_d' has been sensitized. A *Group* is defined as a contiguous string of either 0s or 1s; and the *size* of the Group is the number of elements of the group. Note that in general an address can have multiple groups; e.g., the address '11000' has two groups '11' and '0000' with size 2 and 4 respectively.

In Figure 4b, for which $N = 2$, G takes on the value $G = 1$ (this is the only value for $G < N$) or $G = 2$ (this is the value for $G = N$). For $G = 2$, *all* address lines are in the same Group, and make the same transition such that edge **a**cceleration faults will be sensitized; see 'S_a' in column 'S'.

For $G < N$, in Figure 4b this will be $G = 1$, the set of N address lines is divided into $N/2G$ Groups of 0s and 1s. The edge **d**elay faults (denoted by 'S_d' in column 'S') will be sensitized, because a group of G lines makes $x \rightarrow \overline{x}$ transitions, while the G lines of the neighboring group (which have the opposite data value) make opposite, i.e., $\overline{x} \rightarrow x$, transitions. The number of addresses to be generated for 2 bit address sequence is: $N_{H2}(N = 2) = 6$; 3 for S_a and 3 for S_d.



a. State diagram

#	L1	L0	G	S
0	0	0	2	
1	1	1	2	Sa
2	0	0	2	Sa
3	0	1	1	
4	1	0	1	Sd
5	0	1	1	Sd

b. Address sequence

Figure 4. Minimal set of SAPs for $H = 2$

Table 1 shows the way the address sequence for $N = 2$ can be extended for $N = 8$ [14, 15]. The column '#' indicates the *step* in the sequence of addresses, the columns '$L1$ till $L7$' (the 'L' is placed *above* the digits) show the value of the address; the column G gives the size of the group, while the last column lists the type of faults sensitized by each transition.

At the layout level, all address lines are typically located at the same metal layer, such that capacitive coupling can only occur between physically neighboring lines. Next, it is assumed that the layout is not known, such that capacitive coupling can occur between any two lines. In case the layout is known, the following presentation can be simplified, because only the cases $G = N$ and $G = 1$ apply.

The sizes of the groups (see column 'G') are a power of 2. For $G < N$ edge delay faults will be sensitized between a line from a group of G 0s and a line from a group of G 1s. The second column of Table 1 shows the required sequence of addresses, and the fourth column lists the sensitized faults. Entries 0 through 2 (with $G = 8$) sensitize all S_a faults between any two lines (Li, Lj), with $j \neq i$ $(0 \leq i, j \leq N - 1)$. The sensitization of the S_d faults is done in several clusters of 3 entries: for entries 3 through 5, with $G = 1$, S_d faults are sensitized between any two lines $(Li, Li + 1 + 2k)$ with $i + 1 + 2k \leq N - 1$. La has to be taken from the 1-line groups with the 0(1) value, and Lv from the 1-line groups with the 1(0) value. Entries 6 through 8, with $G = 2$, sensitize all faults between any two

lines $(Li, Li+2+4k)$ with $i+2+4k \leq N-1$ from 2-line groups, and in entries 9 through 11 from 4-line groups.

By inspection, one can see that this method can be extended for arbitrary values of N [14, 15]. The number of addresses required is: $N_{H2}(N) = 3$(for $G = N$) $+3$(the number of of addresses used for each additional Group) $*\lceil log_2(N) \rceil$ (the number of groups) $= 3 + 3 * \lceil log_2 N \rceil$.

Table 1. H2 address sequence for $N = 8$

#	L 0	L 1	L 2	L 3	L 4	L 5	L 6	L 7	G	S_a=acceleration, S_d=delay Fault (La,Lv) sensitized
0	0	0	0	0	0	0	0	0	8	
1	1	1	1	1	1	1	1	1	8	S_a: $(Li, Lj); j \neq i$
2	0	0	0	0	0	0	0	0	8	S_a: $(Li, Lj); j \neq i$
3	0	1	0	1	0	1	0	1	1	
4	1	0	1	0	1	0	1	0	1	S_d: $(Li, Li+1+2k^{(a)})$
5	0	1	0	1	0	1	0	1	1	S_d: $(Li, Li+1+2k^{(a)})$
6	0	0	1	1	0	0	1	1	2	
7	1	1	0	0	1	1	0	0	2	S_d: $(Li, Li+2+4k^{(b)})$
8	0	0	1	1	0	0	1	1	2	S_d: $(Li, Li+2+4k^{(b)})$
9	0	0	0	0	1	1	1	1	4	
10	1	1	1	1	0	0	0	0	4	S_d: $(Li, Li+4+8k^{(c)})$
11	0	0	0	0	1	1	1	1	4	S_d: $(Li, Li+4+8k^{(c)})$
a: $i+1+2k \leq N-1$ b: $i+2+4k \leq N-1$ c: $i+4+8k \leq N-1$										

H3 addressing method

Similar to the case for $H = 2$, the set of SATs $\{La, Lv, La\}$ for $H = 3$ can be divided into two subsets: S_a and S_d. For $N = 3$, they can be derived using the state diagram of Figure 5. $S_a = \{000, 111, 000\}$ and is responsible for edge acceleration; $S_d = \{\{001, 110, 001\}, \{010, 101, 010\}, \{100, 011, 100\}\}$ and is responsible for edge delay. The number of addresses to be generated is: $N_{H3}(3) = 12$.

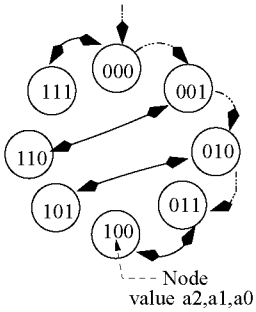


Figure 5. State diagram for N=3

If the layout of the address lines is known, then this method becomes the *'H3k'* AM (the **k** is from *k*nown); it can be extended for any value of N. Table 2 consists of only two SATs; they are row 0 through 5 of Table 1. The SATs have to consider only the address lines $Li-1$, Li and $Li+1$; $1 \leq i \leq N-2$, for address lines L_0 through L_{N-1}. The number of addresses to be generated is: $N_{H3k}(N) = 6$.

In case the layout of the address lines are not known, the capacitive coupling can be assumed to occur between any two address lines. In that case, the following addressing (HN1 addressing) can be applied.

Table 2. H3k address sequence for $N = 8$ with adjacent address lines

#	L 0	L 1	L 2	L 3	L 4	L 5	L 6	L 7	G	S_a=acceleration, S_d=delay Fault (La,Lv) sensitized
0	0	0	0	0	0	0	0	0	8	
1	1	1	1	1	1	1	1	1	8	S_a: $(Li-1, Li, Li+1)$
2	0	0	0	0	0	0	0	0	8	S_a: $(Li-1, Li, Li+1)$
3	0	1	0	1	0	1	0	1	1	
4	1	0	1	0	1	0	1	0	1	S_d: $(Li-1, Li, Li+1)$
5	0	1	0	1	0	1	0	1	1	S_d: $(Li-1, Li, Li+1)$
Note: $1 \leq i \leq N-2$										

HN1 addressing method

The $H = N-1$ addressing method 'HN1' is based on the observation that the SATs for $N = 2$ can be extended for any arbitrary value of N, by modifying S_a and S_d as follows:

- S_a will become the SAT with N equal values: $S_a = \{00...000, 11...111, 00...000\}$; it detects all edge acceleration faults.
- S_d will consist of a set of N SATs, based on a moving '1', as follows: {00...001, 11...110, 00..001}, {00...010, 11...101, 00..010}, ..., and last {10...000, 01...111, 10..000}. This set of SATs detects all edge delay faults.

The result is that the S_a and S_d faults are detected on all lines, irrespective of the position of the victim line and the aggressor lines. The resulting number of addresses to be generated is: $N_{HN1}(N) = 3 + 3N$.

3.2 Sensitizing Operation Sequence

To each address of a SAP or a SAT at least one operation has to be applied, resulting in a *Sensitizing Operation Pair (SOP)* for a SAP, and *Sensitizing Operation Triplet (SOT)* for a SAT. Inspecting the H1, H3 and HN1 AMs reveals that SATs are generated instead of SAPs. Therefore SOTs are required.

Sensitizing Operation Triplets 'SOTs'

The required SOTs for the SATs $\{Ag, Af, Ag\}$ and $\{Af, Ag, Af\}$ consist of two transitions requiring three operations (see Figure 3 and Figure 5):

- Ox_g; $x \in \{0, 1\}$, $O \in \{r, w\}$ [Note: r denotes a read operation and w a write operation] The operation is applied to Ag with data value x.

Table 3. Read-Write-Sequences SOT

RWS	March element 'ME'
RaRaR	$\Updownarrow (wx); \Updownarrow_f (w\overline{x}_f, \Updownarrow_g^{AM} (rx_g, r\overline{x}_f, rx_g), wx_f)$
RaRaW	$\Updownarrow_f (w\overline{x}_f, \Updownarrow_g^{AM} (wx_g, r\overline{x}_f, rx_g), wx_f)$
RaWaR	$\Updownarrow (wx); \Updownarrow_f (\Updownarrow_g^{AM} (rx_g, w\overline{x}_f, rx_g), wx_f)$
WaRaR	$\Updownarrow (wx); \Updownarrow_f (w\overline{x}_f, \Updownarrow_g^{AM} (rx_g, r\overline{x}_f, wx_g), wx_f)$
RaWaW	$\Updownarrow_f (\Updownarrow_g^{AM} (wx_g, w\overline{x}_f, rx_g), wx_f)$
WaRaW	$\Updownarrow_f (w\overline{x}_f, \Updownarrow_g^{AM} (wx_g, r\overline{x}_f, wx_g), wx_f)$
WaWaR	$\Updownarrow (wx); \Updownarrow_f (\Updownarrow_g^{AM} (rx_g, w\overline{x}_f, wx_g), wx_f)$

- $O\overline{x}_f$; $x \in \{0,1\}$, $O \in \{r,w\}$ This operation applied to the address Af with the *complement* of the data value applied to Ag guarantees the first transition required for the sensitization of the ActD and ActA faults.
- Ox_g; $x \in \{0,1\}$, $O \in \{r,w\}$ This operation applied to the address Ag with the *complement* of the data value applied to Af guarantees the second transition required for the sensitization of the DeactD and DeactA faults.

In short, the required SOT for the SAT '$\{Ag, Af, Ag\}$' consists of three operations: $Ox_g, O\overline{x}_f, Ox_g$; $x \in \{0,1\}$, $O \in \{r,w\}$. Because of the capacitive coupling faults, $O\overline{x}_f$ and/or Ox_g 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. If the most sensitive value is known (for example via simulation), then x only has to take on that value.

Depending on the selected operations in $O \in \{r,w\}$, eight *Read Write Sequences (RWSs)* for SOTs are possible (see Table 3): RaRaR, RaRaW, RaWaR, WaRaR, RaWaW, WaRaW, WaWaR and WaWaW. (Note: R denotes Read, W denotes Write and 'a' denotes 'After'; for example: WaRaR means Read-after-Read-after-Write). The RWS WaWaW will not be considered from here on, because at least one 'r' operation has to be present in order to detect the faults; this violates the WaWaW requirement. The RWSs for SOTs use address triplets 'Ag, Af, Ag' in order to allow for SATs; the first and the third address are identical. In Table 3, the nested ME '$\Updownarrow_g^{AM} (Ox_g, O\overline{x}_f, Ox_g)$' is performed for each address 'Af'. The nested ME for the WaRaR RWS has the following form: '$\Updownarrow_g^{AM} (rx_g, r\overline{x}_f, wx_g)$'; first a '$rx_g$' is applied to '$Ag$', next a '$r\overline{x}_f$' is applied, and last a 'wx_g'.

All RWSs *ending* with an 'R' (i.e., that means that the nested ME *starts* with a 'rx_g' operation and has the form 'XaYa**R**' with X,Y $\in$ {R,W}) require initialization of 'Ag'; this is accomplished by the ME '$\Updownarrow(wx)$'. Initialization of 'Af' is required for the RWSs of the form 'Xa**R**aY'. This requires *two* extra operations: one to write the value '$\overline{x}$' and one to write back the original value 'x'. This is performed only *once* for each 'Af' address by the ME '$\Updownarrow_f (w\overline{x}_f, \Updownarrow_g^{AM} (Ox_g, r\overline{x}_f, Ox_g), wx_f)$'. The RWSs of the form 'Xa**W**aY' require the extra 'wx_f' operation to write back the original value 'x' in 'Af'.

4. Tests for capacitive coupling

Based on the AMs of Section 3.1 and the SOTs of Section 3.2, tests for detecting address decoder faults due to capacitive coupling in address lines can be constructed. The results are given in Table 4. The left column in the table shows the test #, the second the RWSs; the third the test time in terms of required operations for the three AMs; the last column presents the description of the test. In the table, AM denotes either H2, H3 or NH1 addressing. Note that for each RWS, the description of the tests are the same for the three AMs since the SOTs are the same.

Seven tests based on each AM can be distinguished. They have been derived from the test structures of Table 3, by repeating those for the data values x=0 and x=1, and by adapting the initializing operations to the appropriate AMs. The latter is important, because the H2, H3k and HN1 AMs have the property that it does not access all 2^N words. Hence, the initializing operations do not have to be applied to all words.

As an example, consider The RaRaR (test#1) for the 'HN1' addressing (see Section 3.1). Table 5 will be used to explain the test. Each row in the table describes a complete 3-address SAT; with addresses 'Ag', 'Af' and 'Ag'. The last column lists the operations to be applied to the SAT, consisting of *two* initializing operations, separated by ';' from the *three* operations for the RWS. The number of addresses generated by HN1 (see Section 3.1) is $N_{HN1}(N) = 3 + 3N$. That means that the total number of SATs is $\frac{N_{HN1}}{3} = 1 + N$ (Note: each SAT has three addresses). The total time, in terms of required operations, is then: $= 10 + 10N = (1 + N)$ (i.e., the number of SATs) $\times$ 2 (for $x = 0$ and $x = 1$) $\times$ 5 (per SAT: 2 initializing operations + 3 operations due to the RaRaR RWS).

Note that the time for tests using H3k addressing is independent of N. Remember that H3k is applicable only when the layout of the address lines is known, and therefore one needs to check only for the physical neighbors lines.

Table 5. RaRaR-HN1 operation sequence

#	Sa or Sd	SAT			Operations
		Ag	Af	Ag	
1	Sa	000	111	000	$w0_g, w1_f; r0_g, r1_f, r0g$
2	Sd	001	110	001	$w0_g, w1_f; r0_g, r1_f, r0g$
3	Sd	010	101	010	$w0_g, w1_f; r0_g, r1_f, r0g$
4	Sd	100	011	100	$w0_g, w1_f; r0_g, r1_f, r0g$

5 Discussion

The question is now which test of Table 4 is the best to be used? First, the addressing NH1 is the best to be used for two reasons: (a) usually the layout of the memory chips is not know and (b) it covers the capacitive coupling even when more than one aggressor is considered. Therefore the use of HN1 addressing (together with appropriate RWS)

Table 4. Tests for address decoder delay faults due to capacitive coupling

#	RWS	Time			Test description
		H2	H3k	HN1	
1	RaRaR	$10+10\beta$	60	10+10N	$\{\Updownarrow^{AM}(w0_g, w1_f; r0_g, r1_f, r0_g); \Updownarrow^{AM}(w1_g, w0_f; r1_g, r0_f, r1_g)\}$
2	RaRaW	$8+8\beta$	48	8+8N	$\{\Updownarrow^{AM}(w1_f; w0_g, r1_f, r0_g); \Updownarrow^{AM}(w0_f; w1_g, r0_f, r1_g)\}$
3	RaWaR	$8+8\beta$	48	8+8N	$\{\Updownarrow^{AM}(w0_g; r0_g, w1_f, r0_g); \Updownarrow^{AM}(w1_g; r1_g, w0_f, w1_g)\}$
4	WaRaR	$8+8\beta$	60	10+10N	$\{\Updownarrow^{AM}(w0_g, w1_f; r0_g, r1_f, w0_g); \Updownarrow^{AM}(w1_g, w0_f; r1_g, r0_f, wr1_g)\}$
5	RaWaW	$6+6\beta$	36	6+6N	$\{\Updownarrow^{AM}(w0_g, w1_f, r0_g); \Updownarrow^{AM}(w1_g, w0_f, r1_g)\}$
6	WaRaW	$8+8\beta$	48	8+8N	$\{\Updownarrow^{AM}(w1_f; w0_g, r1_f, w0_g); \Updownarrow^{AM}(w0_f; w1_g, r0_f, w1_g)\}$
7	WaWAR	$8+8\beta$	48	8+8N	$\{\Updownarrow^{AM}(w0_g; r0_g, w1_f, w0_g); \Updownarrow^{AM}(w1_g; r1_g, w0_f, w1_g)\}$
Note: The ';' separates the initializing operations from the operations required by the RWS N = number of address lines; $N_{H2}(N) = 3+3\beta$; $\beta = \lceil log_2(N) \rceil$; $N_{HN1}(N) = 3+3N$					

will guarantee the detection of capacitive coupling faults irrespective of the layout and of the number of aggressor lines involved. Second, from the fault coverage point of view, the test based on RaRaR is the best since it has the maximal number of reads as compared with the other tests. More reads means more detectability and therefore higher fault coverage. So the best test to be used to cover capacitive coupling faults in address decoders will be:

$\{\Updownarrow^{HN1}(w0_g, w1_f; r0_g, r1_f, r0_g); \Updownarrow^{HN1}(w1_g, w0_f; r1_g, r0_f, r1_g)\}$

The most used tests in the industry consist of march tests like: Scan, MATS+, March C-, PMOVI, March G, etc. Usually these algorithms are used with the binary addressing. That means that even if RWSs sequence within these algorithms are the same as those required for the detection of capacitive coupling, the latter faults will be not covered since they require special addressing method.

To guarantee the detection of different targeted fault types, different tests are need. However, one can merge also different tests in order to achieve the same fault coverage for a low test time (i.e., cost). E.g., the above test targeting capacitive coupling faults can be merged with other tests to target other fault like memory cell array faults [13]. Operations can be added (while keeping the RWS required for the detection of capacitive coupling) in order to cover other faults. The result will be then a modified test covering capacitive coupling as well as memory cell array faults. If, in addition one wants to cover also faults in the peripheral circuits [16] (e.g., Slow Sense Amplifier Fault, Slow Precharge Circuit Fault), then the combined tests should be used with an appropriate data-background (i.e., checkerboard or row stripe) and address direction (i.e., Fast X).

Based on the above short discussion, one can conclude that by understanding the faults models (either related to address decoders, memory array, peripheral circuits), their detection conditions and the freedom the march tests provide, we can combine different march tests to optimize the test time while maintaining a complete fault coverage of the target faults. From a practical point of view, an experiment on large sets of chips, using many of the proposed and old tests has to be performed in order to establish the importance and the effectiveness of each test; this is the future focus of this work.

6 Conclusions

In this paper the Address decoder faults due to capacitive coupling have been analyzed. These faults have been divided onto *Activation Delay (ActD)*, *Deactivation Delay (DeactD)*, *Activation Acceleration (ActA)* and *Deactivation Acceleration (DeactA)*. Thereafter a systematic method has been used to develop several tests targeting such faults. The method is based on generating Addressing Methods, together with the required Read-Write-Sequences. The best test to be used is proposed and the possibilities to merge it with other tests targeting other fault classes is discussed.

References

[1] M. Inoue et al., 'A New Test Acceleration Chip for Low-Cost Memory Tests', IEEE Design & Test of Computers, Vol. 1, pp. 15-19, 1993.

[2] Phil Nigh and Anne Gattiker, 'Test Method Evaluation Experiments & Data', In Proc. of the IEEE Int. Test Conference, pp.454-463, 2000.

[3] W. Needham et al., 'High Volume Microprocessor Test Escapes, an Analysis Our Tests are Missing', In Proc. of the IEEE Int. Test Conference, pp.25-34, 1998.

[4] B. Grundmann, R. Galivanche and S. Kundu, 'Circuit and Design Platform Challenges in technologies beyond 90nm', In Proc. of the Design, Automation and Test in Europe, pp. 44-47, 2003.

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

[6] J.C.M. Li et al., 'Testing for Realsitic Opens and Stuck Opens" In Proc. of the IEEE Int. Test Conference, pp.1049-1058, 2001.

[7] W. Moore, G. Gronthoud, K. Baker, M. Lousberg, 'Delay-Fault Testing and Defects in Deep Sub-Micron ICs-Does Critical Resistance Really Mean Anything?' *In IEEE Proc. of Int. Test Conference*, pp. 95-104 , 2000.

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

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

[10] W.Y. Chen et al., 'Analytic models for crosstalk delay and pulse analysis for non-ideal inputs'. In Proc. of the IEEE Int. Test Conference, pp.809-818, 1997.

[11] A.D. Sathe et al., 'Analog Macromodeling of Capicitive Coupling Faults in Digital Circuit Interconnects', In Proc. of the IEEE Int. Test Conference, pp.375-383, 2002.

[12] S. Kundu et al., 'Test Challenges in Nanometer Technologies', Journal of Electronic Testing: Theory and Applications, Vol. 17, No. 3/4, pp. 209-218, June/August 2001.

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

[14] S. Hamdioui and J.D Reyes, 'New Data-Background Sequences and Their Industrial Evaluation for Word-Oriented Random-Access Memories', IEEE trans. on CAD, Vol. 24, Issue 5, pp. 892-904, June 2005.

[15] A.J. van de Goor and I.B.S. Tlili, 'A Systematic Method for Modifying March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories', IEEE Trans. on Computers, Vol. 52, No. 10, pp.1320-1331, October 2003.

[16] A.J. van de Goor, S. Hamdioui and R. Wadsworth, " Detecting Faults in the Peripheral Circuits and Evaluation on SRAM Tests", *In Proc. of IEEE Int. Test Conference*, pp. 114-123, 2004.