Fault Simulation of Interconnect Opens in Digital CMOS Circuits

Haluk Konuk
haluk_konuk@hp.com
California Design Center-ICBD, Hewlett-Packard Company

Abstract

We describe a highly accurate but efficient fault simulator for interconnect opens, based on characterizing the standard cell library with SPICE; using transistor charge equations for the site of the open; using logic simulation for the rest of the circuit; taking four different factors, that can affect the voltage of an open, into account; and considering the oscillation and sequential behavior potential of opens. A novel test technique based on controlling the die surface voltage is also described. We present simulation results of ISCAS85 layouts using stuck-at and IDDQ test sets.

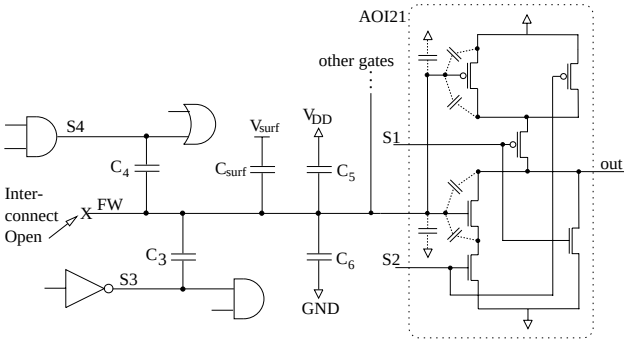

Figure 1: An example for an interconnect open.

1 Introduction

Breaks are one of the common types of defects that occur during an IC manufacturing process [1]. Breaks fall into different categories depending on their location in a digital CMOS circuit. A break can occur inside a CMOS cell affecting transistor drain and source connections [2, 3, 4, 5]. A break can disconnect a single transistor gate from its driver [6, 7]. Yet another break can disconnect a set of logic-gate inputs from their drivers; thus causing the voltage of these inputs to float. In order for this to happen a break needs to occur in the interconnect wiring. In today's CMOS ICs with five or more metal layers, interconnect wiring seems to be the most likely place for a break to occur. This is well supported by the critical area analysis by Xue, *et al.* [8]. Also, vias are especially susceptible to breaks [9], and the number of vias is exceeding the number of transistors in some microprocessor designs [10].

We call the fault created by a break in the interconnect wiring an **interconnect open**. In this paper, we describe a fault simulation algorithm for interconnect opens, that take into account all known factors affecting the voltage of a floating wire created by an interconnect open. We also make use of a novel test technique based on controlling the surface voltage of a die. To the best of our knowledge, this is the first fault simulator reported for interconnect opens; therefore, comparison to prior work is not applicable. In the following section, we discuss the factors affecting the voltage of a floating wire. Then, we describe our fault simulation algorithm, followed by the description of our novel test technique for interconnect opens. Finally, we present the results of our experiments with ISCAS85 circuits.

2 Floating Wire Voltage

Knowing the factors that determine the voltage of a floating wire is essential in building a fault simulator for interconnect opens. The following subsections describe four such factors. Due to lack of space, we do not cover here how the effect of charge collector diodes can be handled, which is described in Chapter 4 in [11].

2.1 Wiring Capacitances

Figure 1 shows an example interconnect open. C_5 is the total wiring capacitance from floating wire FW to the n-wells and to the V_{DD} supply wires. C_6 is the total wiring capacitance from FW to the substrate and to the GND supply wires. The sizes of C_3, C_4, C_5, and C_6 together with the voltages on signal lines $S3$ and $S4$ contribute to the value of FW voltage. What V_{surf} and C_{surf} mean is discussed later.

One needs to know the exact location of an open in

0-89791-993-9/97 $10.00

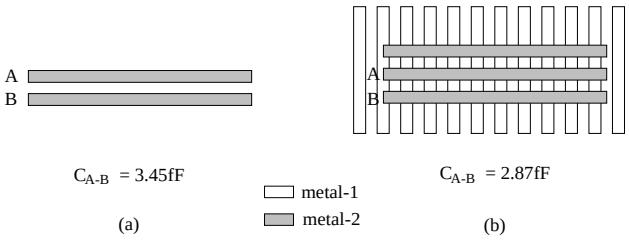

Figure 2: Wire-to-wire capacitance variation.

the interconnect in order to obtain the list and sizes of wiring capacitances to the corresponding floating wire. If an open is going to occur on a piece of straight metal track, we are not aware of any way of predicting where the open defect will land on this metal track. Besides, contacts are much more susceptible to opens than metal tracks are, according to the defect distribution statistics by Feltham and Maly [9]. Also, the increasing number of metal layers in IC processes tends to increase the number of vias per metal layer. In this work, each via, that produces a floating wire when broken, is considered to be a potential interconnect open defect site. The fault list for our simulator is produced by considering each such via.

Accuracy in wiring capacitance extraction is another important issue. Several layout tools that are capable of two-dimensional capacitance extraction exist in both industry and academia (e.g. *Magic*). The problem with two-dimensional capacitance extraction, which is based on the area and the perimeter of the overlap between two conducting surfaces, is its poor accuracy due to the fact that the same area and perimeter can result in different capacitances depending on the surrounding layout topology.

For example; consider the capacitance between two parallel metal-2 wires A and B in Figure 2. Both wires are 0.9μ by 60.0μ separated by 0.9μ. Using a three-dimensional capacitance extraction program, called SPACE3D [12], and wire height and silicon-dioxide thickness parameters for an HP 0.6μ process, we obtained 3.45fF when there are no other wires in the surroundings, as shown in part (a) of Figure 2. However, when there are other metal-1 and metal-2 wires in the surroundings as shown in part (b), the capacitance goes down to 2.87fF. *Magic* extracts 2.88fF for both cases, while the actual capacitance in part (a) is about 20% larger than the one in part (b).

Three-dimensional capacitance extraction is sufficiently accurate, but not feasible for the entire chip in terms of CPU time. One way to deal with this problem is to use two-dimensional capacitance extraction, and to assume that the actual capacitance is within +/- x% of the computed capacitance. This way, our fault simulator uses a range for each wiring capacitance to compute a voltage range for a given open.

2.2 Transistor Capacitances

These capacitances are in the cells driven by FW in Figure 1. They are gate-drain, gate-source, and gate-bulk capacitances, which are connected to transistor terminals with dotted lines to emphasize that they are not additionally inserted, but they are part of any CMOS transistor. Values of transistor capacitances significantly vary depending on the transistor terminal voltages. For this reason, we use transistor gate charge equations expressed as functions of transistor terminal voltages [13, 2] and transistor geometry, rather than using fixed worst case capacitance values. Transistor geometries can be tightly controlled in today's CMOS processes, and transistor capacitances are much less affected by surrounding structures than wiring capacitances are. Therefore, we do not use a value range for a transistor capacitance as we do for a wiring capacitance.

2.3 Trapped Charge

Experiments by Johnson [14] and Konuk and Ferguson [15] showed that the trapped charge deposited during fabrication can build up a voltage from -4.0V to 2.3V on floating gates with poly extensions, and -1.0V to 1.0V on floating gates connected to metal wires. We are not aware of any technique to predict the amount of trapped charge on a particular interconnect open. Therefore, our fault simulator makes no assumptions for the amount of the trapped charge.

2.4 The Die Surface

Konuk and Ferguson [15] reported an experimental observation that the die surface acted as an RC interconnect, capacitively coupling the floating wire to almost all other signals in a chip. The die surface resistance for this phenomenon to occur is in the tera-Ohms range. When coupled with wiring capacitances in the femto-Farads range, a time constant of one second or less is produced. If the die surface has a sufficiently large resistivity, then the capacitance to the die surface can be ignored. Otherwise, C_{surf} in Figure 1 needs to be taken into account.

3 Processing the Cell Library

We assume that each cell in the library is either a basic cell or is composed of basic cells, where we define a **basic cell** as a network of p-channel transistors

and a complementary network of n-channel transistors, where each cell input drives the gates of one p- and one n-channel transistor, such as the AOI21 cell in Figure 1. All the MCNC cells used in the ISCAS85 circuit layouts satisfy this condition.

We first determine the $\boldsymbol{L0\_th}$ and $\boldsymbol{L1\_th}$ values, which denote the maximum voltage that is still logic-0 and the minimum voltage that is still logic-1 for the cell library, respectively [2], which we computed as 1.05V and 1.90V for the MCNC library using HP 0.6μ HSPICE level-13 parameters from MOSIS.

We define a **composite input** or a **c-input** for a cell as either a single cell input or multiple input ports of the same cell tied together. $\boldsymbol{V_{L0,g,ci}}$ denotes the logic-0 threshold voltage for c-input ci of cell g, that is, the voltage on the c-input ci of g such that the output of g is at L1_th and is sensitized to ci. $\boldsymbol{Q_{L0,g,ci}}$ denotes the total electrical charge on the transistor gates that are driven by ci, when the voltage on ci is $V_{L0,g,ci}$ with g's output sensitized to ci. $\boldsymbol{V_{L1,g,ci}}$ and $\boldsymbol{Q_{L1,g,ci}}$ are similarly defined.

For IDDQ testing [16], a threshold current $\boldsymbol{I_{ddq,th}}$ needs to be determined such that a quiescent power supply current larger than $I_{ddq,th}$ indicates a defective chip. $\boldsymbol{V_{iddq0,g,ci}}$ denotes the logic-0 voltage on c-input ci such that $I_{ddq,th}$ flows through cell g. $\boldsymbol{Q_{iddq0,g,ci}}$ denotes the total electrical charge on the transistor gates that are driven by ci, when the voltage on ci is $V_{iddq0,g,ci}$, and $I_{ddq,th}$ flows through g. $\boldsymbol{V_{iddq1,g,ci}}$ and $\boldsymbol{Q_{iddq1,g,ci}}$ are similarly defined. For every cell g and c-input ci in the library, $V_{L0,g,ci}$, $V_{L1,g,ci}$, $V_{iddq0,g,ci}$, $V_{iddq1,g,ci}$, $Q_{L0,g,ci}$, $Q_{L1,g,ci}$, $Q_{iddq0,g,ci}$, and $Q_{iddq1,g,ci}$ are computed and recorded using HSPICE. In addition, the slope and the y-intercept values for the straight line defined by points ($V_{iddq0,g,ci}$, $Q_{iddq0,g,ci}$) and ($V_{L0,g,ci}$, $Q_{L0,g,ci}$) are recorded to be used for charge interpolation in our fault simulation algorithm. Similarly, the slope and the y-intercept for the straight line defined by points ($V_{L1,g,ci}$, $Q_{L1,g,ci}$) and ($V_{iddq1,g,ci}$, $Q_{iddq1,g,ci}$) are recorded. For example; consider the HSPICE charge-voltage plot in Figure 3 for the a input of the 2-input NAND gate in the MCNC library using HP 0.6μ process parameters. The V_{iddq0}, V_{L0}, V_{L1}, and V_{iddq1} points are marked using $I_{ddq,th} = 50\mu A$.

4 Fault Simulation Algorithm

The top level structure of our algorithm is as follows:

FOREACH 32-vector DO
  FOREACH interconnect open DO
    Perform parallel pattern single fault propagation [17] after flipping the fault-free logic value on the floating wire.
    FOR vector 1 THROUGH 32 DO
      IF vector can detect the open THEN
        Compute the range of trapped charge for this vector to detect the open, and add this range to the detection ranges computed from previous vectors.
ENDFOR; ENDFOR; ENDFOR

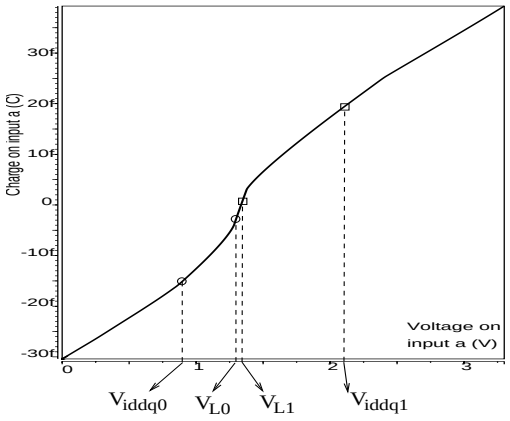

Figure 3: Charge on the a input of a NAND gate.

After all the vectors and opens are processed, an interconnect open is marked detected if the detection range for the trapped charge spans from $-\infty$ to $+\infty$. Alternatively, our program lets the user specify a voltage range, such that the voltage created by the trapped charge on any interconnect open is guaranteed to fall within this range. If specified, then this range is used to decide whether an open is detected or not. Detection of a defect can be accomplished by either voltage sensing (stuck-at testing) or current sensing (IDDQ testing) or both.

4.1 Stuck-at Detection

For a given vector and a given logic value on floating wire FW, let's first describe how we split a c-input of a cell driven by FW into sensitized and unsensitized parts. A **sensitized c-input** is formed by those input ports of a c-input, that drive those transistors through which an IDDQ current flows if the c-input voltage turns both p- and n-channel transistors on, while all other inputs of the cell are kept at V_{DD} or GND. The remaining part of the c-input is called an **unsensitized c-input**.

If an applied vector detects a stuck-at-0 or a stuck-at-1 fault on FW, then the first step to determine the trapped charge range for this vector to detect the open is to find the maximum FW voltage that is still logic-0 in case of stuck-at-0, or the minimum FW

voltage that is still logic-1 in case of stuck-at-1. We assume the stuck-at-0 case for the rest of this subsection without loss of generality. We determine the maximum logic-0 voltage on FW for the applied vector vec, $\boldsymbol{V_{L0,FW,vec}}$, as the minimum logic-0 threshold voltage (minimum $V_{L0,g,ci}$) over all the sensitized c-inputs driven by FW.

Then, we compute the trapped charge on FW, that corresponds to $V_{L0,FW,vec}$. Section 4.3 describes the computation of the trapped charge in detail. Let's call this computed charge $\boldsymbol{Q_0}$. Intuitively, if the actual trapped charge is smaller than Q_0, then the FW voltage will be smaller than $V_{L0,FW,vec}$, that is, the interconnect open will be detected as a stuck-at-0 fault by vec. However, this is not always true due to the potential sequential behavior of interconnect opens, due to feedback capacitive coupling, as described by Konuk and Ferguson[18]. For this reason, we re-compute the trapped charge by using $V_{L0,FW,vec}$ with still vec applied to the circuit, but this time with logic-1 on FW propagated to all circuit nodes that are sensitized to FW. We call this computed charge $\boldsymbol{Q_1}$.

Then, the maximum actual trapped charge with which vec is guaranteed to detect the open as a stuck-at-0 fault is $Q_{trapped,max,sa0} = min(Q_0, Q_1)$. In other words, the detection range for vector vec is $(-\infty, Q_{trapped,max,sa0})$. In addition to sequential behavior, an interconnect open can also oscillate under certain conditions [18]. The way we compute $Q_{trapped,max,sa0}$ guarantees that neither sequential behavior nor oscillation can invalidate our results.

4.2 Current sensing (IDDQ detection)

For IDDQ detection, we first check whether the given vector vec and a logic-0 on FW create any sensitized c-input driven by FW. If so, we determine the minimum voltage on FW, $\boldsymbol{V_{iddq0,FW,vec}}$, such that an IDDQ that is larger than or equal to $I_{ddq,th}$ still flows through a cell driven by FW. Note that V_{iddq0} is less than V_{L0}, and V_{iddq1} is greater than V_{L1} for a basic cell, as illustrated in Figure 3. So, we determine $V_{iddq0,FW,vec}$ as the minimum logic-0 IDDQ threshold voltage (minimum $V_{iddq0,g,ci}$) over all the sensitized c-inputs driven by FW.

Then, we compute the trapped charge on FW, that corresponds to a voltage of $V_{iddq0,FW,vec}$, as explained in Section 4.3. Note that the FW voltage needs to be larger than $V_{iddq0,FW,vec}$ for IDDQ detection. Let's call this computed charge $\boldsymbol{Q_{trapped,min,iddq0}}$.

Similarly, we compute $\boldsymbol{Q_{trapped,max,iddq1}}$, that corresponds to the maximum actual trapped charge with which vec can detect the open as an IDDQ fault. Note that if a sensitized c-input exists when FW is logic-0, a sensitized c-input must also exist when FW is logic-1, because there will be at least one cell driven by FW, whose other inputs do not have any combinational path to them starting from FW. Then, the IDDQ detection range for vec is $(Q_{trapped,min,iddq0}, Q_{trapped,max,iddq1})$. Oscillation [18] is possible within this detection range; however, we assume that if oscillation occurs, it will draw sufficient IDDQ current so that the open will still be detected.

If voltage and current are both measured for a given vector, then the open might be detected as a stuck-at and/or an IDDQ fault. Due to lack of space, we defer the discussion of this case to a future paper.

4.3 Trapped Charge Computation

This subsection describes how we compute $Q_{trapped}$ for a given voltage V_{FW} and a given test vector vec. It is important to emphasize that what we compute here is not the actual trapped charge on a floating wire, since we are not aware of any method to compute that. The trapped charge computed here is a consequence of the V_{FW} and the vec applied, and it is used as a boundary value for the actual trapped charge in order for the open to be detected.

The total charge on FW has two components: (i) the charge on the transistor gates driven by FW, denoted by $\boldsymbol{Q_{gate}}$, and (ii) the charge on FW due to the wiring capacitances between FW and other nodes (including the substrate and the die surface), denoted by $\boldsymbol{Q_{wire}}$. From the law of charge conservation, the trapped charge on FW is $Q_{trapped} = Q_{gate} + Q_{wire}$.

The equation for Q_{gate} is as follows, where $\boldsymbol{SCI}$ and $\boldsymbol{UCI}$ denote the sets of sensitized and unsensitized c-inputs driven by FW, respectively.

$$Q_{gate} = \sum_{ci \in SCI} (a_{ci} + b_{ci} \cdot V_{FW}) + \sum_{ci \in UCI} Q_{eqn}(ci, V_{FW}, vec) \qquad (1)$$

where a_{ci} and b_{ci} are the y-intercept and the slope values recorded for c-input ci during our library processing step in Section 3. Recall that two sets of y-intercept and slope values are computed per ci; one for the line passing through the IDDQ threshold and the logic threshold points on the Q-V curve as shown in Figure 3 when the c-input voltage is logic-0, and the other for the line when the c-input voltage is logic-1. This interpolation-based computation in Equation 1 is an efficient way of approximating the charge on a sensitized c-input with reasonable accuracy.

For an unsensitized c-input ci, the charge on the gate of each transistor driven by ci is computed by

using Equations 3 and 5 in [2]. These gate charges are added up to find $Q_{eqn}(ci, V_{FW}, vec)$ in Equation 1. We could not use interpolation to compute $Q_{eqn}(ci, V_{FW}, vec)$; because, unlike a sensitized c-input, V_{FW} alone is not sufficient to determine the drain/source voltages of the transistors driven by an unsensitized c-input. Transistor drain/source voltages are determined, when needed, using the algorithm described in [2].

With regard to computing Q_{wire}, recall from Section 2.1 that our algorithm uses a range for each wiring capacitance due to the accuracy limitations of extraction tools. Let $\boldsymbol{C_{w0}}(\boldsymbol{C_{w1}})$ denote a wiring capacitance from FW to a neighboring node that is at logic-0 (logic-1) value. Then, the following equation shows how to pick the worst case values for C_{w0} and C_{w1} in order to guarantee detection. The detection of the open might occur below V_{FW}, for instance, when V_{FW} is the maximum logic-0 voltage for the floating wire, and the vector applied is a test for FW stuck-at-0. In this case, $C_{w1,max}$ will be used for C_{w1} as the worst case.

$$C_{w0}(C_{w1}) = \begin{cases} C_{w0,min}(C_{w1,max}) \text{ if } FW \text{ voltage} \\ \quad \text{needs to be} < V_{FW} \text{ for detection} \\ C_{w0,max}(C_{w1,min}) \text{ if } FW \text{ voltage} \\ \quad \text{needs to be} > V_{FW} \text{ for detection} \end{cases}$$

If the die surface does not have a big enough resistivity, then the min and max values for the capacitance between FW and the surface needs to be considered, also, together with a voltage range $V_{surf,min}$ and $V_{surf,max}$ the die surface can acquire.

5 A Novel Test Technique

The passivation layer deposited on top of the last metal layer in a fabrication process usually has a similar thickness with the inter-metal dielectric. Therefore, if a conducting plate is placed on the die surface during wafer probing as illustrated by Figure 4, there will be capacitances to this plate from almost all the wires in the chip. If there is an interconnect open in the chip, then there will also be a capacitance from the floating wire to this plate, as denoted by C_{surf} in Figure 1. Playing with the voltage of this plate, V_{surf}, provides a handle to control the voltage of the floating wire; thus changing the amount of IDDQ current flowing in the cells driven by the floating wire. Note that if a chip is defect-free, a change in the die surface voltage will not cause any change in its IDDQ current, which must be below $I_{ddq,th}$, anyway.

An extension of this technique is to use a set of conducting plates forming tiles on the die surface, whose voltages can be individually controlled. This way, a floating wire can be localized to a chip area depending on the size of each tile used on the die surface. This technique has been filed for a patent.

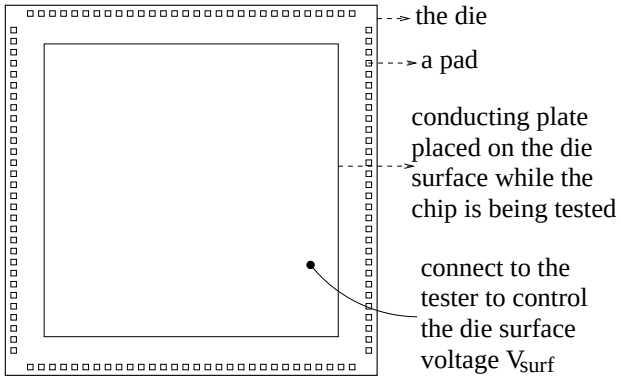

Figure 4: Controlling the die surface voltage.

6 Experiments and Results

We used the two metal layer channel routed layouts of the ISCAS85 circuits for our experiments. Due to lack of space, we are reporting our results only for the largest five ISCAS85 circuits in this paper. Since we consider each via (the contact between metal 1 and 2) as an interconnect open site, we process each layout with a program that is an extended version of *Carafe* [19] to analyze each via to find out whether a floating wire is created if the via is broken. The program removes each via, that can create a floating wire, from the layout, and labels the wire pieces in a systematic fashion. Then, we run *Magic* using the HP 0.6μ parameters in the technology file 8.2.8 from MOSIS to extract the capacitances from the layout.

There is a metal-2 wire across the height of each MCNC cell layout for every input port of the cell. A via connects this metal-2 wire to a small piece of metal-1, which is connected to poly that drives transistor gates. When this via is broken, the floating poly together with a small piece of metal-1 form a very short floating wire (s-FW), which has very small capacitances to neighboring nodes and to the die surface. The numbers of breaks corresponding to these vias are listed in the first column of Table 1. The remaining via-breaks are considered to create "long" floating wires (l-FW), and are listed in the second column. The IDDQ and stuck-at vectors are generated by *Nemesis* [20]. The pseudo-stuck-at fault model is used for IDDQ ATPG.

For a given chip, the wiring capacitances are fixed, even though their values may not be accurately

Circuit	# of l-FW via-breaks	# of s-FW via-breaks	# of IDDQ vecs	IDDQ cov.	# of stuck-at vecs	stuck-at cov.
c2670	3141	1286	42	97.78 %	117	96.33 %
c3540	4420	1854	67	97.71 %	175	96.58 %
c5315	7652	2938	60	99.62 %	137	98.79 %
c6288	8260	4232	34	99.40 %	33	99.42 %
c7552	9054	3464	86	99.62 %	221	98.89 %

Table 1: Via-break and test vector statistics

known due to limitations of extraction tools. So, we decided to use the extracted capacitance values as the real values, and assume that it is theoretically possible to modify the given circuit layout to exactly match these capacitance values, and the actual design is this modified layout. We also assumed that the die surface effect of Section 2.4 does not apply.

Table 2 shows our simulation results. Columns 2 to 5 show the % of s-FW and l-FW opens guaranteed to be detected no matter what the actual trapped charge is[1], when stuck-at (SA) vectors are used alone or in combination with IDDQ vectors. SA vectors are good at detecting interconnect opens when the magnitude of the actual trapped charge is large, forcing the floating wire to behave as stuck at 0 or 1. In contrast, IDDQ vectors tend to detect opens when the trapped charge magnitude is small, resulting in floating wire voltages being pulled to the vicinity of $V_{DD}/2$ by wiring and transistor capacitances. This is why SA and IDDQ vectors together produce much better guaranteed coverages in columns 4 and 5.

If the initial voltage on a floating wire due to the trapped charge can be bounded, then the detection ranges our algorithm computes, as explained in Section 4, can be compared to the size of the total possible range to give a probabilistic range coverage number. This is what we did for the rest of Table 2. Assuming that the actual trapped charge voltages are bounded by the [-1.0V, 1.0V] range, and assuming a homogeneous distribution of actual trapped charge voltages in this range, we obtained the range coverage numbers in columns 6 and 7. Using the [-0.5V, 0.5V] range decreased the coverage numbers, as we expected, because we used only the SA vectors, and SA vectors get worse in detection when the trapped charge magnitude gets smaller. Still, these range coverage numbers are much better than the guaranteed coverage numbers in columns 2 and 3.

In order to be certain of the coverage of a set of test vectors for a particular given design, even when the trapped charge voltage is bounded, one needs to take into account the inaccuracy of the capacitance extraction tools. Assuming that the actual value for a wiring capacitance is within the range from $0.7 * C_{extracted}$ to $1.3 * C_{extracted}$, and assuming that trapped charge voltage is bounded by [-0.5V, 0.5V], we obtained the range coverage numbers in Table 3. We also assumed that the die surface voltage can vary, and the "no surf cntrl" columns are for the case where we used either V_{DD} or GND for the die surface voltage depending on which one being the worst case. V_{DD} is 3.3V for all our experiments.

Note that the coverage numbers in column 3 for l-FW opens is particularly low, because long floating wires are affected by the die surface more than the short floating wires are. Applying the SA vectors twice, once with $V_{surf} = V_{DD}$ and once with V_{surf} = GND using the setup in Figure 4, helps long floating wires behave more like stuck-at faults, thus significantly increasing the range coverage numbers as listed in column 5. However, s-FW opens still have fairly low range coverages as listed in column 4, because the die surface voltage does not have as much effect on the short floating wire defects.

Using IDDQ vectors in addition to SA vectors boosts the coverage numbers as listed in columns 6 and 7, compared to the columns 2 and 3. However, the l-FW opens still do not reach the coverage numbers obtained by the two-pass SA testing of column 5. Finally, controlling the surface voltage at $V_{surf} = V_{DD}/2$, and using both SA and IDDQ vectors produce very high range coverage values for both s-FW and l-FW opens as listed in the last two columns of Table 3.

We used an HP-735 99MHz workstation with 180MB memory for our fault simulations. The maximum wall clock time was 4 minutes 24 seconds for circuit c7552, which shows us promise for the feasibility of a fault simulator for real chips.

In conclusion, we presented an accurate but efficient fault simulator for interconnect opens, which take many factors that can affect the voltage of an open into account. We also presented a novel test technique based on controlling the die surface voltage. Our results from ISCAS85 circuit layouts show that

[1] Note that excessive trapped charge may create a dangerous voltage level such that a gate-oxide punch through may occur, creating a gate-oxide short coupled with an interconnect open. We do not consider this case and leave it to future research.

Circuit	% opens guaranteed to be detected				Trapped charge range cov. (%)			
	SA only		SA and IDDQ		SA only, -1.0V to 1.0V trapped Q volt.		SA only, -0.5V to 0.5V trapped Q volt.	
	s-FW	*l-FW*	*s-FW*	*l-FW*	*s-FW*	*l-FW*	*s-FW*	*l-FW*
c2670	0.00	5.19	93.86	97.33	86.52	89.89	77.46	83.61
c3540	0.00	16.99	93.91	97.22	86.85	91.04	79.26	85.27
c5315	0.00	11.43	98.03	97.28	88.78	92.06	80.06	86.76
c6288	0.00	18.67	99.13	99.84	88.35	91.51	77.71	84.44
c7552	0.00	6.57	97.89	99.14	88.89	90.80	80.32	85.57

Table 2: Via-break coverages assuming no surface effect and using extracted capacitance values.

Circuit	SA only				SA and IDDQ			
	no surf cntrl		two pass: V_{surf} = V_{DD} and GND		no surf cntrl		V_{surf} = V_{DD} / 2	
	s-FW	*l-FW*	*s-FW*	*l-FW*	*s-FW*	*l-FW*	*s-FW*	*l-FW*
c2670	64.21	17.74	72.85	94.31	99.63	66.28	99.65	99.76
c3540	67.06	19.47	74.74	94.83	99.63	70.48	99.66	99.73
c5315	67.08	18.99	75.44	96.01	99.97	64.89	99.97	99.93
c6288	64.02	22.24	72.98	94.63	99.62	75.85	99.62	99.85
c7552	66.97	17.35	75.35	95.32	99.99	61.57	100.00	99.95

Table 3: Trapped charge range cov.'s with cap. variation and trapped charge volt. bounded to [-0.5V, 0.5V]

combination of high coverage stuck-at and IDDQ test sets can result in high coverage of interconnect opens, especially when the die surface voltage is controlled, while stuck-at tests alone may not always guarantee such high coverage.

ACKNOWLEDGMENT: Prof. F. Joel Ferguson of UC Santa Cruz is gratefully acknowledged for his guidance and valuable technical discussions.

References

[1] C.F. Hawkins, J.M. Soden, A.W. Righter, and F.J. Ferguson. Defect classes - an overdue paradigm for CMOS IC testing. In *Proceedings of ITC*, Oct. 1994.

[2] H. Konuk, F.J. Ferguson, and T. Larrabee. Charge-based fault simulation for CMOS network breaks. *IEEE Transactions on Computer-Aided Design*, Dec. 1996.

[3] C. Di and J.A.G. Jess. On accurate modeling and efficient simulation of CMOS opens. In *Proceedings of ITC*, 1993.

[4] M. Favalli, M. Dalpasso, P. Olivo, and B. Ricco. Modeling of broken connections faults in CMOS ICs. In *Proceedings of European Design and Test Conference*, 1994.

[5] W.M. Maly, P.K. Nag, and P. Nigh. Testing oriented analysis of CMOS ICs with opens. In *Proceedings of ICCAD*, 1988.

[6] V.H. Champac, A. Rubio, and J. Figueras. Electrical model of the floating gate defect in CMOS IC's: Implications on IDDQ testing. *IEEE Transactions on Computer-Aided Design*, March 1994.

[7] M. Renovell and G. Cambon. Electrical analysis and modeling of floating-gate fault. *IEEE Transactions on Computer-Aided Design*, pages 1450–58, Nov. 1992.

[8] H. Xue, C. Di, and J.A.G. Jess. Probability analysis for CMOS floating gate faults. In *Proceedings of European Design and Test Conference*, 1994.

[9] D.B.I. Feltham and W. Maly. Physically realistic fault models for analog CMOS neural networks. *IEEE Journal of Solid-State Circuits*, Sep. 1991.

[10] K.M. Thompson. Intel and the myths of test. In *The Keynote Address of ITC*, 1995.

[11] H. Konuk. Testing for opens in digital CMOS circuits. *Ph.D. Thesis, Computer Eng. Dept., UC Santa Cruz*, Dec. 96.

[12] A.J. van Genderen and N.P. van der Meijs. *Space3d Capacitance Extraction User's Manual*. Delft University of Technology, 1997.

[13] B.J. Sheu, W.-J. Hsu, and P.K. Ko. An MOS transistor charge model for VLSI design. *IEEE Transactions on Computer-Aided Design*, pages 520–527, April 1988.

[14] S. Johnson. Residual charge on the faulty floating gate MOS transistors. In *Proceedings of ITC*, October 1994.

[15] H. Konuk and F.J. Ferguson. An unexpected factor in testing for CMOS opens: The die surface. In *IEEE VLSI Test Symposium*, 1996.

[16] A.D. Singh, H. Rasheed, and W.W. Weber. IDDQ testing of CMOS opens: An experimental study. In *Proceedings of ITC*, 1995.

[17] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and Th. McCarthy. Fault simulation for structured VLSI. In *VLSI Systems Design*, Dec. 1985.

[18] H. Konuk and F.J. Ferguson. Oscillation and sequential behavior caused by interconnect opens in digital CMOS circuits. In *Proceedings of ITC*, 1997.

[19] A. Jee and F.J. Ferguson. Carafe: An inductive fault analysis tool for CMOS VLSI circuits. In *Proceedings of the IEEE VLSI Test Symposium*, 1993.

[20] T. Larrabee. Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, January 1992.