Test Challenges in Nanometer Technologies

Sandip Kundu, Sanjay Sengupta, Rajesh Galivanche
Intel Corporation, USA

Abstract

Scaling transistor feature size allows greater density, higher performance and lower cost. The unrelenting pursuit of device scaling has enabled MOS gate dimensions to be reduced from 10mm in the 1970's to a present day size of 0.1mm. Conventional scaling of gate oxide thickness, source/drain extension, junction depths, and gate lengths have brought about several new technology issues invalidating some earlier methods for testing ICs. To enable testing devices into the 21st century, new approaches are required in both test and design for testability. In this paper, we will define the problems that arise with device scaling such as Gate Oxide leakage, sub-threshold leakage, power density, electromigration, and soft error problems in qualitative and quantitative terms. The later half of the paper deals with some of the solutions being pursued at Intel.

Introduction

Driven by quest for greater performance, higher density and lower cost, MOS device technologies have been improving at a dramatic rate for more than 30 years. While reducing transistor feature size an array of options exist in terms of deciding oxide thickness, doping profile, junction depth etc. By and large the entire semiconductor industry follows so called *constant-field scaling* that allows the horizontal and vertical electrical fields to remain the same to allow optimum power-performance [1- 3]. This means in every subsequent generation the power supply voltage and the oxide thickness is scaled. However, a host of secondary issues such as Drain Induced Barrier Lowering (DIBL), Gate-Induced Drain Leakage (GIDL), and punch-through do not scale linearly and therefore additional device engineering becomes necessary.

In making compromises to alleviate some device problems, the transistor behavior changes in each generation. The cumulative effect of such changes has a tremendous impact on how the future devices should be tested. That is the topic of discussion in this paper.

Manufacturing test should screen out non-performing parts. A part may fail due to manufacturing defects, due to process marginalities or due to design marginalities that is not cognizant of normal manufacturing process variation. Traditionally, test has focussed on detecting manufacturing defects but there is an increasing trend towards failures due to "soft" defects as well as various marginalities.

Manufacturing Defects

Manufacturing defects have been at the core of test concerns. A defect is a physical aberration or abnormality. Examples of defects include partial or spongy vias, the presence of extra material between metal lines, etc. As the manufacturing process technology changes, so do the underlying defect mechanisms and their distributions. For example, as more metal interconnect layers are stacked, more masking steps are associated with building interconnects rather than devices and increasing number of defects are associated with interconnects.

Interconnects are typically made of aluminum or copper. Aluminum interconnects are based on a process that deposits a layer of aluminum and then removes the extra metal. This tends to bias more of the defects to be *bridge* type of defects. However, copper technology is based on depositing a *seed liner* and then electroplating the metal. Electroplating is more likely to cause opens or resistive/porous/spongy connects. The Chemical Mechanical Polishing (CMP) which is at the heart of planarization may leave *bridges* by failing to remove extra metal.

Some of the common defects associated with devices are missing contacts, dislocations, missing trench isolation, gate oxide defects etc. Not all manufacturing defects cause functional failures. Some manufacturing defects may cause a parametric change such as high leakage or device performance degradation. Yet others may show up as a reliability failure, such as electromigration or gate dielectric breakdown.

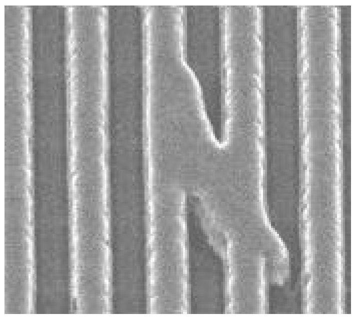

Figure 1 Typical example of a interconnect short

Process Marginalities

When the lithographic patterns are not regular, significant variation exists in device electrical properties such as effective channel length, threshold voltage, substrate biasing sensitivity, temperature dependence etc. The interconnect and inter-layer dielectric thickness also varies significantly as CMP process is not amenable to ultra-precise controls. All this adds up to variation in load and cross coupling capacitance for devices which translates into delay and noise sensitivity variation across die, across wafer and across lots. Due to complex inter-relation among these key electrical parameters the variations are not fully orthogonal, such as variation in effective channel length directly ties into variation in threshold voltage.

Design Marginalties

Design is based on a suite of tools used as design aid. Each tool in turn relies on an electrical model and not all tools are amenable to statistical analysis. As a result, model approximations, numerical errors, data abstractions eat away at the core accuracy. For example, performance analysis which is often based on static timing analysis fail to capture statistical variations and even though a design is validated under nominal process corner, it may fail to operate when underlying delay assumptions are changed. On the other hand if overly pessimistic assumptions are made about process variation, it may impose a performance penalty to overcome such variation during design process. Design marginalities may cause failure of a part due to race introduced by delay variations on silicon.

Test Goals

Test has several goals. It should be a screen for defects, a screen for potential reliability problems as well as a vehicle for performance sort.

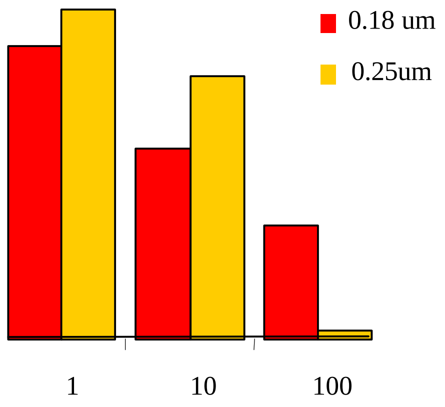

Figure 2 Bridge Resistance Data from Failure Analysis

Even though the defect mechanism may remain same, the impact is ever changing. For example it was observed from failure analysis data that with scaling the population of bridge defects with higher resistance increases. Thus many failures can not be observed as hard fails and can only be observed as delay defects under very specific patterns and a controlled test environment.

IddQ test had been very effective in terms of detecting bridge defects. Unfortunately, when the bridge resistance is higher the defect current trends lower. With rising background leakage in newer technologies such small rise in defect current is not detectable.

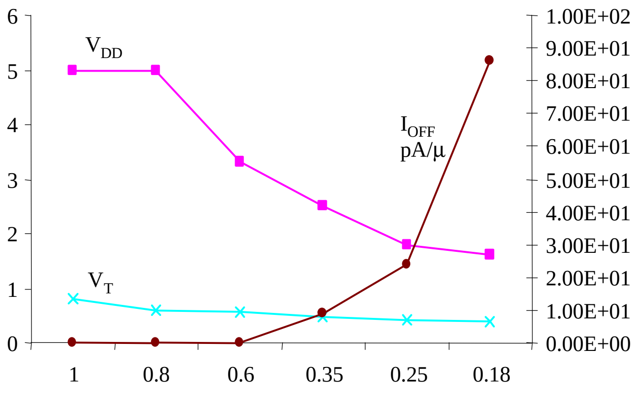

Figure 3 V_{dd},V_t and I_{off} vs. technology generation

Rising leakage current arises from both subthreshold as well as gate leakages.

Figure 3 shows power supply and threshold voltage trends for Intel's microprocessor process technologies [1,2]. As seen, the threshold voltage is going down in each technology generation driving up subthreshold leakage current in each successive generation.

As oxide thickness continues to scale (Figure 5) gate leakage becomes a significant contributor (Figure 6) as well.

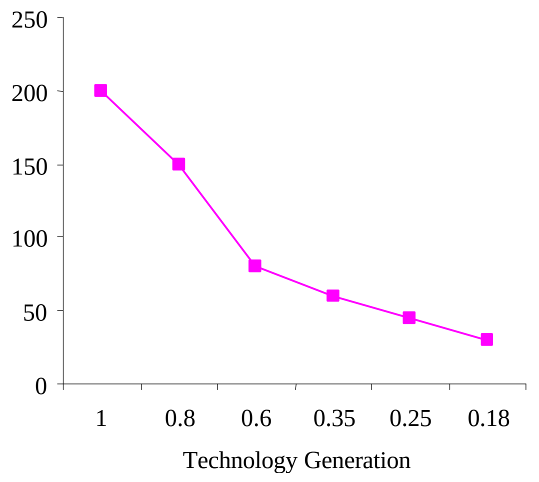

Figure 4 Gate oxide thickness in A° vs. technology generation

Simulation studies show that loss of IddQ test for bridge detection can be made up somewhat by implementing necessary DFT to enable detection of speed detects [4].

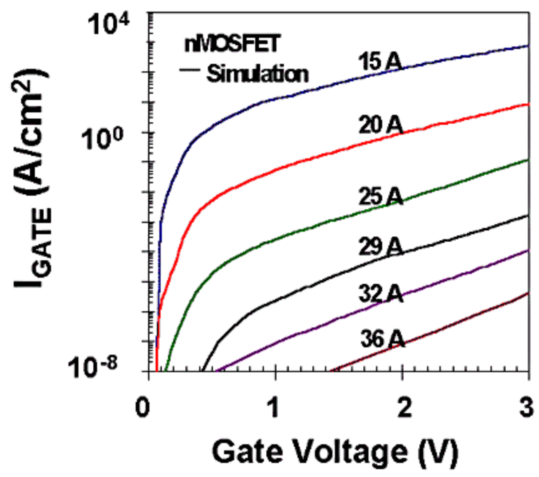

Figure 5 Gate leakage vs. oxide thickness and voltage

Aside from loss of IddQ test, scaling of Vdd introduces new factors in failures. First, sharply higher signal slew rates introduce more noise and that coupled with lower supply voltage sharply erodes noise margin. In this environment, process variation such as inter-layer dielectric thickness variation that can introduce greater noise due to cross talk effects may cause failure. Thus the trend towards failure from process marginalities is on the rise.

Another effect taking wings that can cause failure during or after test is Soft Error. Soft Error is a non-permanent or transient error that is introduced when a node discharges due to ionizing cosmic or radioactive radiation.

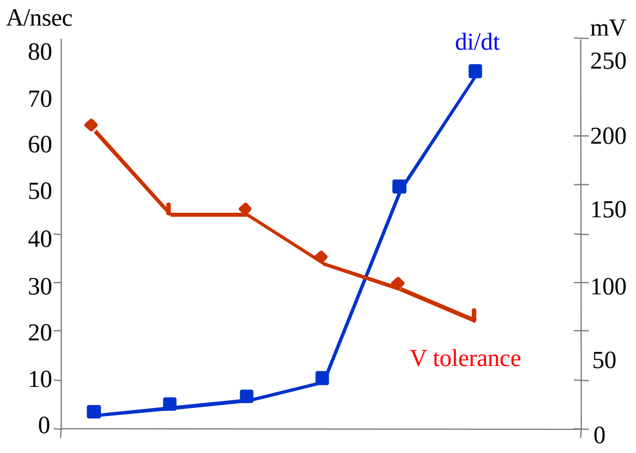

Figure 6 Supply voltage and noise voltage trends

Ionizing radiation cause circuit nodes to discharge with greater likelihood when the amount of stored charge on a node is small. The following figure shows how the node capacitances and supply voltages are scaling with technology. Simultaneous reduction in node capacitance and supply voltage leads to quadratic reduction of node charge. This increases susceptibility to discharge from cosmic or radioactive radiation.

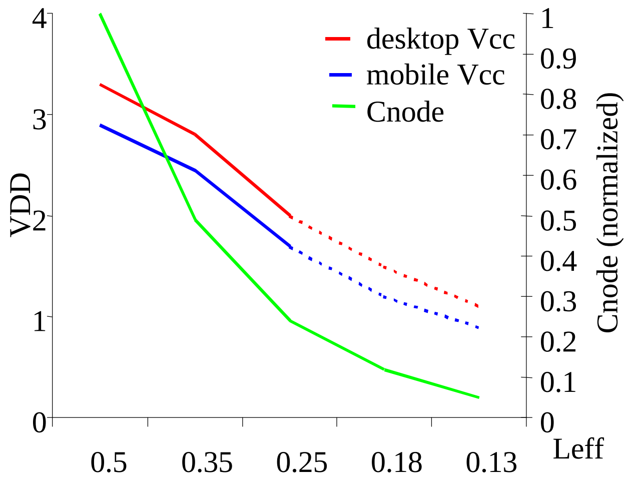

Figure 7 Vdd and Cnode scaling with technology

Soft Error is not a manufacturing test problem. Nevertheless, it introduces additional requirements for manufacturing test. This is because, the most common techniques to solve the Soft Error problem revolves around introducing redundancy (error correction, duplication, triple module redundancy, voting etc). During manufacturing, if a hard fault affects a redundant element, the impact may get corrected out, thereby degrading the Soft Error tolerance. Thus additional DFT becomes necessary to test redundant or spare logic in a special mode where the impact of hard faults can be seen.

Another impact of scaling is that while the dimensions of transistors shrink and the supply voltage goes down, the current per unit width remains more or less the same. Thus, if interconnect dimensions are scaled, current density goes up; if they are not scaled, density targets are missed. Increased current density causes greater electromigration problem. Electromigration is the phenomenon of physical dislocation of metal due to unidirectional current flow. Electromigration causes failure at a later time. Accelerating electromigration during burn-in by higher temperature and toggle may adversely impact the performance of transistor due to acceleration in Hot Electron effect that degrades transistor performance. Thus the reliability considerations must be carefully weighed against performance loss.

Summary of Test Problems

Manufacturing process is imperfect and introduces defects. Manufacturing process also leads to variance of electrical parameters across die, wafer and lots. The design process is simplified by layering assumptions about transistor parameter values, gate delays, static analysis that are not true representation of actual silicon behavior. Thus a chip may fail due to defect or excessive process variation that was not accounted for or due to model approximations that were not fully justified for all process corners. Additionally, newer technologies are more susceptible to reliability problems and Soft Error problems. A chip is not tested for its entire specifications range (minimum voltage to maximum voltage, minimum operating temperature to maximum operating temperature, minimum driver load to maximum driver load etc.) and therefore predictions must be made for behavior over the entire range by testing a few points in this space. Such predictions are increasing under stress in newer technologies.

Test Challenges

The test problems are tackled by DFT, by changes in test methods and conditions and by applying smarter stimulus. DFT is governed by the availability of automation and the economics of intrusion in area-performance-power space. Test methods are based on economics of available equipment and empirical knowledge on best test conditions (temperature, voltage, frequency, signal slew etc) while smarter stimulus is again related to DFT and automation. Thus DFT and automation are highly inter-related. That is why we turn our focus on to automation next.

Test Automation Challenges

One approach to dealing with these test challenges is to build upon the framework that exists for the single stuck-at fault model [4]. While many of the concepts and tools carry over, targeting defects poses a new set of problems:

First, there is a need for a fault model for each defect type that can be implemented in a conventional boolean fault simulator or ATPG tool. Such a fault model typically consists of a set of activation conditions, and may include some restrictions on the way the fault effect, once activated, is propagated to an observation node. We will use the term *constraints* to refer to these conditions. It is of great practical importance that the fault models we use are simple enough to enable an efficient implementation. The approach we have taken is to start with a *simple first-order model*, implement tools to automate the test generation process, and establish the effectiveness of the model at catching defects. If necessary, the model is refined based on feedback from this process.

A second common theme in testing for defects is the automation required to identify the target faults. Since it is impractical to test for every possible defect of a certain type (e.g. all 2-line bridges), it becomes necessary to rank the defect sites based on probability. This function, which we call *fault extraction*, often requires analysis of lower levels of design abstraction, such as the physical or circuit level, while incorporating data from failure analysis.

Even when a ranked list of faults is available, the list of all interesting faults is very large – there are not only multiple defect types to consider, but also the same site may participate in multiple defects of a given type. For example, a long interconnect could bridge with several other lines. The sheer volume of faults we have to deal with necessitates *high performance* simulation and test generation tools. In our experience, this requirement rules out switch-level or full-timing simulation.

In the following sections, we will review the tool infrastructure being developed to deal with one example of defect-oriented models, namely the bridge fault.

Fault Simulation Engine

The Blast fault simulator was developed at Intel to serve as a mainstream stuck-at fault simulator as well as an extensible infrastructure to target new fault models based on defects. Blast is an event-based, zero-delay, gate-level fault simulator. It has support for flexible primitives to model multi-ported RAMs and ROMs, with variable address and data ranges per port.

A key design decision was to use a single fault simulation algorithm, as opposed to a parallel-fault algorithm that leverages word parallelism. Some of the factors that contributed to this decision were:

- At the end of each simulation vector, parallel-fault simulators require faulty values in state machines to be stored away, and retrieved when the next vector is simulated. This incurs a bit packing and unpacking overhead.
- Bit-parallel operations cannot be used when evaluating complex gates and behavioral models (such as memory arrays), again requiring bit packing/unpacking in a parallel fault simulator.
- In general, a parallel-fault approach is effective only if the faults in a concurrently simulated group are closely related, and cause events on the some of the same gates. This analysis does not yield consistent results.
- Defect-based faults can sometimes change the structure of a circuit, introducing new primitives in the fault circuit. Using machine parallelism gets complex when each faulty machine has a different structure.

Event processing is done based on rank order levelization of combinational gates. All combinational loops are broken by non-clocked storage elements. The fault simulator can handle some limited asynchronous behavior. Asynchronous loops may exist by design, or may be introduced by a fault. In a given simulation vector (one set of input changes), the simulation wheel continues to turn until the events settle. Oscillation is detected when user specified number of wheel turns is exceeded.

The simulator is divergence-based in that it re-uses good machine computations to only simulate fault effect propagation in faulty machines. Because there can be multiple wheel turns in a given vector, the good machine can undergo irreversible state changes that would prevent event reuse for computing faulty machine divergence. A phased approach is used to prevent this problem. Consider the case where the wheel requires two turns for events to settle. The good machine state is captured at the end of the first wheel turn (response to PI changes), and the faulty machine divergences are computed. If there are residual events in the good machine, the wheel is turned again in the good machine, and events due to state divergences are propagated in the faulty machine. The algorithm is illustrated in Figure 8 below. The terms *GM(i)* and *FM(i)* refer to the state of the circuit in the *i*th time frame.

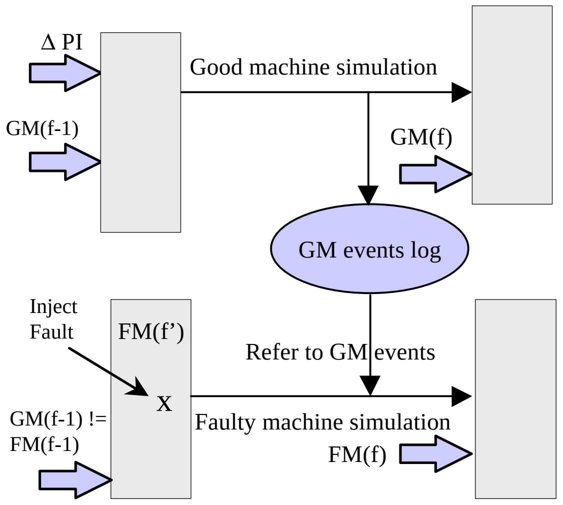

Figure 8 Divergence-based fault simulation

The fault simulator is currently being used by several next-generation microprocessor design teams to grade both functional and ATPG vectors. To give an estimate of the performance, we quote run times on a 32-bit microprocessor design (2.2M primitives) below. Functional tests were run on fault samples, and CPU seconds are recorded on an HP J5000. Total memory requirement was less than 300 MB in each case.

Test	Target Faults	Vectors	CPU Time
Test1	1076	97185	17716
Test2	883	27327	10081
Test3	709	30477	6489
Test4	361	33933	6277
Test5	354	43509	12702
Test6	294	33933	12950

Figure 9 Stuck-at fault simulation performance

Because the algorithm is divergence-based, the incremental computation per fault is very small compared to good machine simulation. As the fault size decreases, logic simulation dominates CPU time.

Bridge Defects

The bridge defects we will consider are two-line bridges between inter-cell interconnects. Intra-cell bridges are not considered due to their low likelihood. All fanouts of a bridged net are assumed to behave similarly in the presence of a fault. Bridges can be low-resistance or resistive; we will see how the bridge resistance plays a role in the way the fault is modeled.

Since it is practical only to target a small subset of bridge faults, we need an extraction tool to rank faults by likelihood. We use the weighted critical area (WCA) [5] to measure this likelihood. Nets in a physical design are assumed to be composed of rectangles (non-Manhattan nets are approximated piecewise with rectangles). The critical area of two rectangles is the area in which a defect of a certain size has to lie in order for the nets to bridge. The weighted critical area is the sum of critical areas for different defect sizes, weighted by the probability of the defect size.

For nets composed of multiple rectangles, the critical area for a given defect size is the union of the critical areas of each rectangle pair, with one member of the pair drawn from each net. This definition applies to nets on the same layer. The concept of critical area can be extended to inter-layer nets, where defects are 3-dimensional in nature. The WCA of a net-pair is the sum of its inter-layer and intra-layer WCA constituents.

A fault extraction tool called EIFFEL has been developed at Intel to rank net pairs by WCA. While the tool can compute the probability of multi-node bridges, we will focus on the 2-line case here. EIFFEL uses novel algorithms to address both the capacity and performance aspects of computing the WCA for a large microprocessor design.

Interval trees are used for rectangle intersection to efficiently compute the union of critical areas of rectangle pairs. The critical area of the largest defect size is computed first. For each smaller defect size, the critical area is computed by resizing the critical area rectangles, and then using a new merging algorithm to come up with the reduce critical area. These algorithms give about two orders of magnitude speed-up over previously available tools [6].

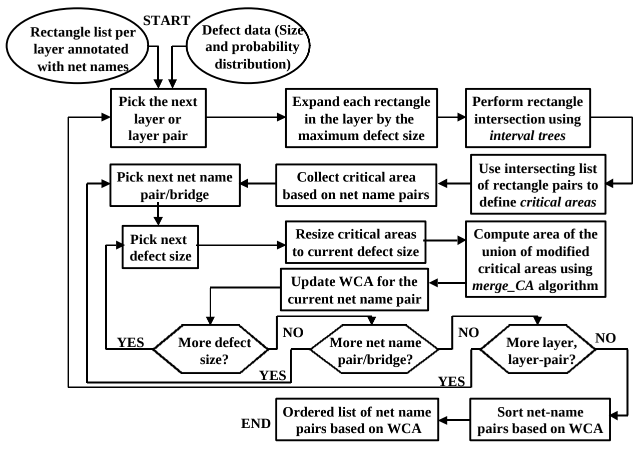
START
Rectangle list per layer annotated with net names
Defect data (Size and probability distribution)
Pick the next layer or layer pair
Expand each rectangle in the layer by the maximum defect size
Perform rectangle intersection using *interval trees*
Use intersecting list of rectangle pairs to define *critical areas*
Collect critical area based on net name pairs
Pick next net name pair/bridge
Pick next defect size
Resize critical areas to current defect size
Compute area of the union of modified critical areas using *merge_CA* algorithm
Update WCA for the current net name pair
More defect size?
More net name pair/bridge?
More layer, layer-pair?
YES
NO
YES
NO
YES
NO
Sort net-name pairs based on WCA
Ordered list of net name pairs based on WCA
END

Figure 10 High-level WCA computation flow

At a higher level, EIFFEL uses a divide-and-conquer approach to address the capacity issue. The layout is partitioned into segments, and each segment is expanded by a small amount to account for boundary effects. The segments are then flattened (hierarchy is removed), and WCA computation and fault list extraction are performed on each segment. The final step is to merge partial fault lists extracted from each segment to get the chip-level results.

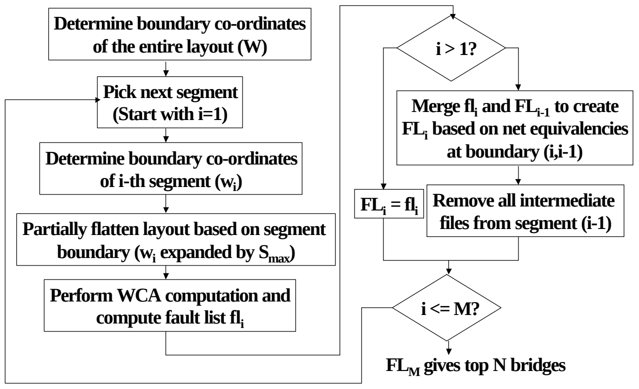
Determine boundary co-ordinates of the entire layout (W)
Pick next segment (Start with i=1)
Determine boundary co-ordinates of i-th segment (w_i)
Partially flatten layout based on segment boundary (w_i expanded by S_{max})
Perform WCA computation and compute fault list fl_i
i > 1?
Merge fl_i and FL_{i-1} to create FL_i based on net equivalencies at boundary (i,i-1)
$FL_i = fl_i$
Remove all intermediate files from segment (i-1)
i <= M?
FL_M gives top N bridges

Figure 11 Divide-and-conquer flow.

The above methodology was implemented and run on large layout databases. We used defect data containing 9 different defect sizes for each layer and layer pair. The layout data L1-L7 used for our experiments are partitions a microprocessor layout. Experiments were run on a Pentium II 466MHz workstation with 1GB memory (running Linux OS) to determine the top 30000 bridges. The table below summarizes the results.

Circuit	Transistor Count (millions)	Segment Count M	Execution time (CPU seconds)	Memory Usage (MB)
L1	0.39	300	10791	202.90
L2	0.54	300	32682	205.06
L3	0.87	300	52699	97.30
L4	1.18	300	61722	123.11
L5	1.70	300	139549	168.95
L6	2.28	300	229000	861.93
L7	2.32	450	214081	846.26

Figure 12 Experimental results from EIFFEL on large databases

Various fault models have been proposed for bridging defects with varying degrees of complexity. Examples include AND- and OR-bridges, where the bridge nodes behave like wired-AND or wired-OR logic, and the voting model, which takes some pattern dependency into account.

Interconnect bridging defects exhibit a range of behavior based on different values of bridge resistance. This effect is illustrated for the circuit in Figure 13. There is a bridge defect between node *j* and *k* in this example. Node *k* is held at logic 0 as *j* changes from 0 to 1. The signal transition is propagated and observed at output *v*.

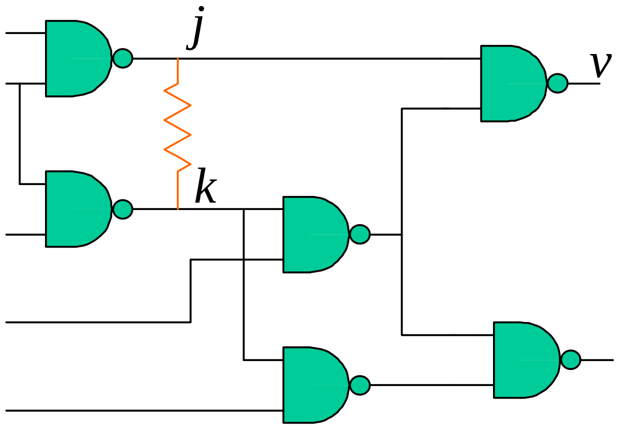

Figure 13 Example circuit with bridge defect

Figure 14 shows the output response of the circuit for different values of the bridge resistance. Receiver threshold voltages are marked using horizontal dashed lines, and the vertical dashed line shows the required arrival time at node v for the transition to be captured in a downstream latch.

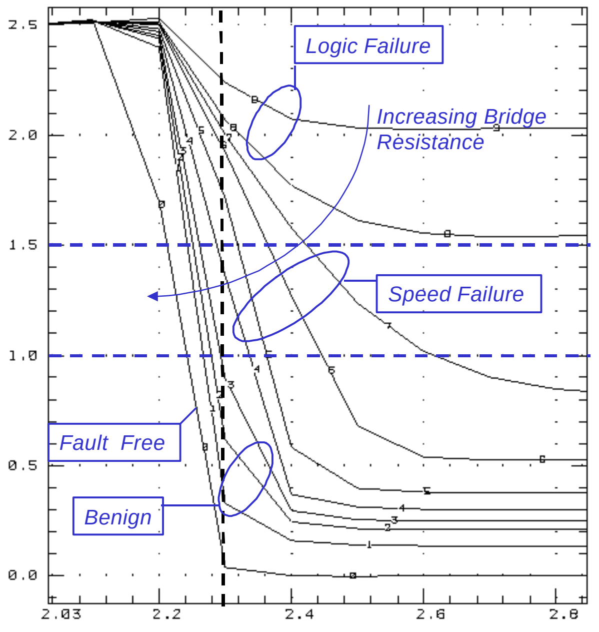

Figure 14 Output responses for various bridge resistances

The plot shows three distinct circuit behaviors for varying bridge resistance. For low resistance values, the output never reaches the correct logic value, and the defect shows up as a static logic failure. For intermediate resistances, the output goes to logic 0 too late, resulting in a speed failure. Very high bridge resistances are benign from the viewpoint of correct logical operation of the circuit.

For low resistances, the defect can be modeled as node *j* stuck at logic 0 with the condition that *k* is at logic 0. Speed failures can be modeled as a slow-to-rise delay fault at node *j*, with the condition that *k* is held at logic 0.

Blast supports various models for bridge faults that behave like stuck at faults, including the wired-AND and wired-OR models. A 4-way bridge model has also been implemented, and is useful when the relative strengths of the drivers and pattern dependencies are not known. This model consists of a set of four constrained stuck at faults: $\{(j{=}0 \Rightarrow k{=}0),\ (j{=}1 \Rightarrow k{=}1),\ (k{=}0 \Rightarrow j{=}0),\ (k{=}1 \Rightarrow j{=}1)\}$. Implementation of constrained delay faults is under way. As a first approximation, the delay faults will be modeled as transition or gross delay faults, and propagation path slacks will be considered if necessary.

The bridge fault extraction, modeling and simulation tool suite is currently being used on a next generation microprocessor design. An ordered fault list extracted by EIFFEL was truncated using a WCA threshold for individual bridges. The reduced fault list was then mapped onto driver gates in the gate-level netlist. Legacy stuck-at tests were run on a set of functional blocks with the highest bridge WCA. The figure below shows the stuck-at fault coverage and the percentage WCA coverage for those blocks.

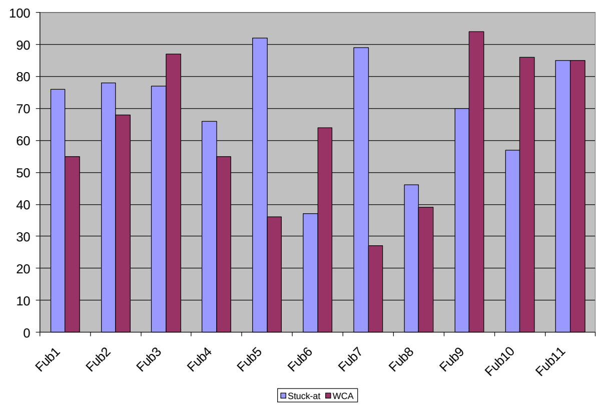

Figure 15 Stuck-at and bridge coverage correlation

The key point to note here is that there is no direct correlation between stuck-at and bridge coverage. This contradicts conventional wisdom that high stuck-at coverage is good for catching defects that do not behave like stuck-at faults.

Efforts are under way to compute the WCA coverage of complete test set on the reduced bridge fault list. Once this is done, additional tests will be created as necessary. An ATPG capability with bridge fault test generation is also being developed for future designs.

What Lies in the Future?

Defect based test while promising has a downside. It involves extraction of most likely defect sites or nodes (bridge, open, cross-talk, min-delay testing, charge retention test for dynamic logic, gate and path delay faults, SER etc.), requires behavioral modeling of these defects as faults on an less detailed model (typically gate level). Modeling at the transistor level is impractical due to large data volume and long processing times. Mapping defect information to a faulty behavior on a higher level model is complex due to the fact that changing ambient conditions such as voltage and temperature changes the faulty behavior. ATPG for defect based faults is non-trivial and setting proper coverage goals is non-trivial. The modeling and the computational effort can be large and the return in terms of quality improvement may not justify the cost. Another difficulty is to balance costs versus priorities. Is it better to get 80% bridge fault coverage and 60% coverage on cross-talk faults or is it better to achieve 90% bridge fault coverage at the expense of a lower cross-talk fault coverage when the computational resources or the tester pattern capacity is limited.

Testing for defect-based models requires a multi-disciplinary approach, tying in various aspects of the design flow, such as physical design and timing, with test generation. Given the rate of change in process technology (and the underlying defect mechanisms), it becomes impractical to go after all physical defect mechanisms during component test.

SIA roadmap is advocating Logic Built-In Self-Test to solve the future quality problems. It offers an alternative approach, where a large number of patterns are applied with the hope that some of the patterns actually help in terms of catching defects. This is an approach that is used in many quarters. However, there are some issues with BIST as well. First and foremost is that how can asynchronous races be exposed using BIST ? How much impact does power related noise have on BIST environment ? How does one perform speed test/binning using BIST ?

An important goal in test is quality versus cost management. If component test cost becomes prohibitive, is it cheaper to postpone some defect coverage till system test ?

The answer is going to be design, process, application, manufacturing volume and quality expectation specific. It is unlikely that a single definitive answer or approach will emerge for designs .

References

1. Scott Thompson, Paul Packan and Mark Bohr, "MOS Scaling: Transistor Challenges in the 21st Century", *Intel Technology Journal*, Q3, 1998 (http://developer.intel.com/technology/itj/q31998/articles/art_3.htm).

2. Ali Keshavarzi, Kaushik Roy and Charles F. Hawkins, "Intrinsic Leakage in Low Power Deep Submicron CMOS ICs", *IEEE International Test Conference*, 1997.

3. Yuan Taur; Yuh-Jier Mii Logan, R. and Hon-Sum Wong, "On effective channel length in 0.1µm MOSFETs," *IEEE Electron Device Letters*, Volume: 164 , April 1995 , Page(s): 136 -138

4. Sanjay Sengupta et al, "Defect-Based Test: A Key Enabler for Successful Migration to Structural Test", *Intel Technology Journal*, Q1, 1999 (http://developer.intel.com/technology/itj/q11999/articles/art_6.htm).

5. J. F. Ferguson and J. P. Shen, "Extraction and Simulation of Realistic Faults Using Inductive Fault Analysis", *IEEE International Test Conference*, 1988, pp. 475-484.

6. S. T. Zachariah, S. Chakravarty and C. D. Roth, "An Efficient Algorithm to Extract Two-Node Bridges", *Design and Automation Conference*, 2000, to appear.