Debug And Failure Analysis
SWDFT 12 Tutorial

Rob Aitken
ARM Fellow
San Jose, CA
Contents

1. Basic Concepts
2. Design Challenges
3. Getting Started: Silicon Bring-Up
4. Scan-Chain Diagnosis
5. Silicon Validation and Debug
6. Fault Dictionaries and Post-Test Simulation
7. Diagnosis of Delay Faults
8. Debug of Power Issues
9. Diagnosis for Yield Learning
10. Future Directions and Challenges
Acknowledgements

- Many people helped, including...
  - Miron Abramovici, Srikanth Venkataraman, John Biggs, Sounil Biswas, Dave Bull, Brian Cline, Al Crouch, Sid Das, Kris Flautner, Dave Flynn, Anne Gattiker, Bob Madge, Mike Mateja, Peter Maxwell, Teresa McLaurin, Mike Muller, Tom Waayans, and many others
Basic Concepts

- Terminology
  - Error / bug / failure / fault / defect
  - Debug / diagnosis
  - Verification / validation

- **Failure** = Wrong/unexpected/incorrect result

- Our goals:
  - Explain the failure
    - Which gate is bad in this block?
    - Why didn’t this transaction complete?
    - Why is this signal late?
  - Find its root cause
“There are known knowns. There are things we know that we know. There are known unknowns. That is to say, there are things that we now know we don’t know. But there are also unknown unknowns. There are things we do not know we don’t know.”

-- Donald Rumsfeld

**Known unknowns:**

Does the chip have a defect?
Does this chip have a timing problem?
Is this chip is sensitive to low voltage?

**Unknown unknowns:**

Is there incomplete test coverage?
Are problems being masked?
Are there reliability issues lurking?
What else is going on?

**Diagnosis:**
Making Unknowns Known
IC Fault Diagnosis (in One Slide)

Tests -> Observed Behavior

Defective Circuit

Location or Fault -> Diagnosis

Physical Analysis -> Diagnosis

Diagnosis Algorithm

Source: David Lavo, UCSC
Diagnosis: Finding Defects or Faults

- Defects occur post-silicon
  - Random/Spot
  - Systematic
    - Example: shorts caused by lines too close
    - Caused by physical design/process errors
  - **Physical defects** are modeled by **logical faults**
    (short-to-GND defect modeled by stuck-at-0 fault)

- Fault diagnosis may be followed by defect diagnosis

- **Defect analysis** = Finding process errors causing systematic defects (a.k.a. failure analysis, process debug)
Quick Review of Design

- Can’t do diagnosis without understanding design and test
- Things that are hard to design tend to fail more than things that are easy to design
- Let’s look at some tricky design problems…
First Challenge: Clocking

- The clock and data involved in logic operation create a race issues.
  - **Setup**: data slower than next clock
  - **Hold**: data faster than first clock
  - **Skew**: clock mistiming between flip-flops
  - **Jitter**: changes in clock period, edge placement

- Need to make sure race goes the right way with all sources of variation
Keeping Speed In Perspective

- At 1 MHz, light travels nearly 300m in a clock cycle.
- At 20 MHz, light travels 15m in a clock cycle.
- At 300 MHz, light travels only 1m in a clock cycle.
- In 150ps, light travels about 5cm.
- In 6ps, light travels about 2mm.

It is impossible to travel faster than the speed of light, and certainly not desirable, as one’s hat keeps blowing off -- Woody Allen
Second Challenge: Predicting Variation

Die-to-die vs. within-die variability

- Ignoring D2D is a fallacy
- Worst case occurring at a known process corner is a fallacy

Source: C. Visweswariah, IBM
The standard Static Timing Analysis (STA) concept of slack hides an underlying statistical distribution of signal arrival times.

Source: IBM
Modeling On Chip Variation

- Random Variation
  - Derive OCV derate by Monte Carlo simulation
  - $OCV \text{ derate} \propto \text{path variation} / \text{mean path delay}$

- Systematic Variation
  - Location based – derived using test-chips

![Diagram showing OCV derate and % Delay Change](image)
3rd Challenge: Parametric Yield

- Assumes features allowed by the rules yield equally well.
- The design rules do not allow features that do not yield.

Source: Bob Madge
The True Feature Yield Curve

Source: Bob Madge
Feature Types Drive Product Yield

Design with 6 M of these can yield up to 94%

Design with 6 M of these can only yield up to 40%

Allowed

Not Allowed

Yield

Features

Source: Bob Madge
Sources of Variability

- Lithography/Etch
  - Line edge roughness
  - CD variation
  - Influence of neighbors

- Device
  - Implant variation
  - Variation between N and P
  - Stress/strain effects

- Interconnect
  - Dielectric variation
  - Via/contact quality
  - Metal width/height variation

- Deterministic versus Random
Effects of Variability

- **Leakage**
  - Variation in $L$, $V_t$, $\mu$, $t_{ox}$

- **Performance**
  - Changes in $L$, $W$, $R$, $C$, $V_t$, $\mu$

- **Min VDD**
  - Changes in $V_t$, $L$, $W$
  - SRAM bit cell main limiter

- **Dynamic power**
  - Changes in $C$
  - Side effect of changes in performance, leakage

- **Yield**
  - Indirect result of others
  - Parameter goes beyond spec + tolerance
DFM or DFT?

Case 1:
- Signal wire too close to ground wire
- Increased capacitance
- Increased delay
- Range: 0-500ps
- DFM issue

Case 2:
- Signal wire shorted to ground
- Low resistance:
  - Stuck-at fault
  - DFT issue
- High resistance:
  - Delay fault
  - Range: 0-2000ps
  - DFT issue
4th Challenge: Power Delivery

- For iso-power and VDD reduction for future process nodes current drawn will increase (V goes down, I goes up)
  - \( CV^2 F \) constant \( \Rightarrow \) C or F can increase
  - More logic or run faster
- Delivering higher current will impact VDD integrity
  - IR due to increased steady-state current
  - \( \frac{LdI}{dt} \) due to activity rate changes – function of \( \Delta I \) and frequency
  - Timing impact higher at lower VDD
Dual-Core Voltage Noise Implications

- Voltage-noise is reinforced in dual-core configurations with shared voltage rails when noise-events are aligned in time-domain.

Simulation of VDD noise wave starting at core 1 (a) and the VDD noise wave reinforced at core 2. (b)

Source: IBM
VDD Droops – Frequency Spectrum

First Order Droop
High Frequency (100M-1GHz+)
Package L and Die C
Both global and local Components

Second Order Droop
Mid-frequency (1M-10M)
PCB/Socket L and Package Decap
Only Global

Third Order Droop
Low-frequency (<100KHz)
Regulator L and Board Decap
Only Global

On-die VDD Time-Domain Response to a Ldi/dt emergency showing various frequency components
(Source: Muhtaroglu et al.)
Environmental Variation

- Varies spatially and with time (switching activity)

From: C. Visweswariah, IBM
Back to Basics: Defects $\Rightarrow$ Faults

Physical domain $\Rightarrow$ Logical domain

- **Fabrication errors**: caused by human errors
- **Fabrication defects**: caused by problems in manufacturing process
- **Physical failures**: caused by wear-out and/or environmental factors

- **stuck-at faults**
  - single
  - multiple
- **bridging faults**
  - non-feedback
  - feedback
- **delay faults**
  - gate
  - path
- **functional faults**
- ...
What are Systematic Defects?

Example #1: Spacing Dependent Tunnel Bridge

Example #2: Systematic Open Defect

Example #3: Non-redundant P-diffusion Contact
- Minimum diffusion enclosure contacts are susceptible to high resistance failure

Particular Design Features / Process Step Susceptible
What are Random Defects?

No Obvious Design Features / Process Step Susceptibility
Defect Models or Fault Models?

- “Models are to be used, but never to be believed”
  - Henri Thiel

- Why not use defect models?
  - Because there is no such thing!

- Defects are continuous
  - can happen anywhere on die, on one or multiple layers
  - can consume arbitrary area
  - have arbitrary electrical properties (R, C, etc.)

- Two possible approaches
  - attempt to model defects as accurately as possible
  - make assumptions about defect behavior in order to reduce the infinite set of possible defect behaviors into a finite set of faults

- Fault diagnosis uses some of each...
Faults $\Rightarrow$ Errors

- A fault exists in the circuit (model)
  - permanent
  - intermittent
  - transient (hit-and-run)

- An error (or fault effect) is created by applying stimulus to activate the fault
  $\Rightarrow$ at the fault site: fault-free value $\neq$ faulty value

- A fault effect may propagate through the circuit.

- Detection: a fault effect propagates to a PO $\Rightarrow$ observed error

![Diagram showing fault effect propagation](image)
Diagnosis: Errors ⇒ Faults ⇒ Defects

Mapping of test results into:
- defective “components”
  - faulty IC (on a board)
  - faulty cell in a RAM
  - faulty RAM in an array
  - faulty board (in a system)
- defective interconnections
  - shorts
  - opens
- delay faults
- IC manufacturing defects

Goals:
- repair (replacement, bypass,...)
- failure analysis (process improvement)
Cost of Locating Faults

Your mileage may vary, but you get the idea…
Debug = Finding/Locating/Identifying Errors

- Errors (bugs) occur pre-silicon during design
  - Logic errors
  - Timing errors
    - Example: not enough margin to account for process variations
  - Physical design/Process errors
    - Example: design rule not tight enough

- Many errors are found during pre-silicon design verification – Design error debug or Logic error debug
  - Simulation
  - Timing verification
  - Design-rule checks
  - Formal verification

- Some errors are found post-silicon – Silicon debug
Debug: Observed Errors $\Rightarrow$ Root-Cause Errors

Mapping of test results into:
- incorrect functional blocks
- incorrect interfaces
- logic implementation errors
- timing errors

Goals: error fix
- Pre-silicon: redesign, resynthesis
- Post-silicon:
  - respin
  - FIB
  - software workaround
  - soft fix
Why Silicon Validation and Debug?

- The Expectation: Correctness by Design
  - SOC designs are modeled and completely verified before tapeout
  - Good luck with that…

- Facing Reality: 100% Correctness Pre-Silicon Is Not Achievable
  - Specifications are inherently incomplete and flawed
  - Models are always a simplification
  - Simulation can not cover full functional space
  - New IP introduces non-verified behavior
  - Formal verification can only cover “ideal” functionality
  - Actual physical device behavior impacts functionality
  - Plus, all those “design challenges”
Getting Started: Silicon Bring-Up

- After weeks or months of waiting, your chip comes back from the fab
- You put it on the tester to see what it does and…

- Nothing.

- Now what?

- Time to peel the onion
  (thanks to Adam Cron for the metaphor)
- Assume packaged parts
  - Wafer similar
Prepare First

- Static Timing Analysis in all modes
- Power/Rail analysis
- Check and recheck the bonding diagrams, pin maps, package, substrate, and load board designs
- Validate test program
  - ATE simulation
  - Make sure it fails with an empty socket
- Prepare any hardware debug equipment
  - Scopes
  - Power supplies
  - Multimeters
  - Analysis tools (more later)
Step 1: ATE setup

- Check pad connectivity and diode drops
- Try multiple chips
  - Not all at once, in case you’re breaking them!
- Is the chip getting power?
  - Does it get warm?
  - Is the tester power draw different with the chip in the socket?
  - Does the power change with voltage?
    - Linear, high power consumption == resistive problem
- Does it produce anything at the chip outputs with power applied?
  - Check at different levels
- What happens when clock is applied?
  - Look for waves with a scope
  - Keep slowing it down
  - Try higher voltages (but not too high)

“Oops” Examples:
- Chip bonded incorrectly
- Missing metal layers
- Load board wiring problems
- Decap
Step 2: JTAG

- Can you talk to the TAP controller?
  - TRST
  - BYPASS, CHIPID instructions
  - Slow down TCK if needed

- Can you talk to the boundary-scan chain?
  - Can you capture from pads?
  - Can you drive the pads?
  - The right ones? Verify pin mapping

- How does it behave across voltage and temperature?

Challenge: Reset problems
Design errors or manufacturing problems can result in power-on states that prevent operation
Telltale symptom: behaves differently every time power applied, sometimes voltage, temperature dependent
Remember, Shmoos Are Your Friends

Shmoo plotting, the black art of IC diagnosis

(See [Baker and Van Beers D&T-95] for more on shmoos)

- Repeat a test over a range of parameters; e.g. voltage and temperature, and record pass/fail information
- The intent is to have variability!
- For diagnosis/debug, look for interesting fails
- For chip bring up, look for solid passes
Step 3: Scan and ATPG

Either via JTAG or other scan pins, try the ATPG vectors

- Start with a basic shift pattern (0101 or 01000111)
  - Does data come back out again?
  - Are some chains broken?
    - Common to a block?
    - Common to a clock?
  - Any bits/transitions missing? (more later)

- May need to generate additional patterns based on working/broken chains

- Try low speed vectors first
  - Keep compression off initially, if possible
  - If successful, graduate to transition, path delay, etc.
Step 4: On-Chip Bus

- Can you talk to SOC elements (CPU, peripherals, memory, etc. via the on-chip bus ports?
- Debug/trace infrastructure and tools can help here
  - Chip based
  - ATE based
  - BIST
  - Connection to simulators
- Verify that units are alive
- See if they do what you want them to
Things To Watch For

- Power problems
  - Go away with reduced activity
- Clocking issues
  - Frequency/domain dependent
  - Shift problems
- Memory problems
  - Voltage dependence
- Defect problems
  - Different behavior on multiple chips
- Systematic defect problems
  - Logically unconnected failures
Memory Failures

- Voltage?
- Defect?
- BIST?
- Power?

Bit Maps: The key to all things memory

- Vertical Pair: Bit Line Contact
- Partial Column: Resistive Bit Line Short
- Multi-Row: Address Decoder
- Swath: CMP Scratch
- Entire Bit: Sense amp, I/O
- Catastrophic: Timing circuit
Summary

- Don’t panic if it doesn’t work
- Proceed methodically
  - Start simple
  - Verify that basics work
  - Add complexity gradually
  - Be sure each step of the way
- When problems arise, don’t jump to conclusions
  - Identify possibilities
  - Gather evidence to support/disprove
  - Evaluate evidence
  - Repeat/continue as needed
Hardware Probing Techniques

- *Guided-probe testing* for board diagnosis (very old)
  - But basically the same as micro/nanoprobing (new)

- *E-beam testing* for IC diagnosis and locating logic errors (less old)

- *Optical probing* techniques (recent)
Backtracing

Trace back errors until find good values

Typical faults/errors located:
- faulty/incorrect components
- timing problems
- logic errors
- initialization problems
- connection errors

Problems:
- simulation complexity (hierarchy, lack of a testbench, etc.)
- data volume (dump file)
- feedback loops
- wired logic
- total diagnosis time
Guided-Probe Testing

- monitors internal board signals via a probe
- ATE provides guidance for moving the probe
- backtrace errors from POs to their source(s)
- independent of fault modeling
- test sequence reapplied for each probing point
- most time-consuming step: moving the probe
## Speed-Up Techniques [Kochan et al. ITC-81]

- Use a dump file to provide a starting point for probing

- Probe inputs of suspected devices before outputs

- Probe only the inputs that can affect the output with errors

- Probe control before data inputs; if no errors, probe selected/enabled data inputs
E-Beam Probing

- monitors internal signals via an electron beam probe
- probe station provides guidance for moving the beam
- trace transitions, not logic values
- independent of fault modeling
- test sequence reapplied continuously to enable sampling
- most time-consuming step: taking samples
Theory of E-Beam Probing

- Electron beam directed at die surface

- Secondary electrons reflected back, allowing voltage contrast measurement
  - positive charge reflects fewer electrons (black)
  - negative charge reflects more electrons (white)

- Can measure change in voltage at GHz frequencies, useful for chip debug as well as diagnosis
Fault Location with E-Beam Probe

Typical faults located:
- breaks, shorts
- faulty transistors
- delay problems

Problems:
- signals not accessible (not at top levels)
- CMP (chemical mechanical polishing): flat surface keeps views to top level metal although tricks exist to get around this
- need somewhere to start looking
- long test lengths
- need for simulation values
- finicky equipment
- total diagnosis time
Example E-Beam Trace
Example Voltage Contrast Image
E-Beam Speedup Techniques

Use shortest test sequence possible

Find an internal node to start at

Use binary search to partition problems

Design-for-debug:

Include probe points as part of design for difficult areas

- signals in top level metal, probe pads, etc.
- use around critical paths, cache, etc.
## IR Techniques and Tools

<table>
<thead>
<tr>
<th>Dynamic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laser Probe</td>
<td>LVP</td>
</tr>
<tr>
<td>Emission Collection</td>
<td>TRE</td>
</tr>
<tr>
<td></td>
<td>LADA</td>
</tr>
<tr>
<td></td>
<td>IREM</td>
</tr>
</tbody>
</table>
IR Techniques and Tools - LVP

- Laser voltage probing
  - Infrared (IR) laser based
  - Signal waveform acquisition for high frequency timing measurement
  - Measure directly from active P-N junction through backside silicon substrate
  - Voltage waveforms of internal nodes - PN junction at drain
  - Picosecond scale resolution
  - Essential for ICs mounted in a flip-chip or MCM package
  - Used in silicon debug and failure analysis
In CMOS circuits, transistors emit a burst of light when they switch
- Electrons in channel are heated (~3000K) and emit a pulse of black body IR radiation
- On average, one photon emerges from Si for every ~10000 switching events
- Special photon counting detectors are the key to building up a histogram of ps-scale arrival times
- High-speed low-noise InGaAs Avalanche Photodiode with below-breakdown bias and gated operation
IR Techniques - IREM

- **IREM**: InfraRed Emission Microscope
  - LN2-cooled HgCdTe array collects Near-IR radiation in 800-2500 nm wavelength range
  - Combined with advanced camera optics and precise stage allows super-accurate location of emissions
  - Tester docked to IREM and failing pattern is applied
  - Substrate is thinned and AR-coating applied
  - Die is observed through the backside
  - Strong abnormal emission can indicate a variety of circuit or process defects
  - Contention, leakage, saturated devices

N diffusion emits - 1
P diffusion emits - 0
Summary

Hardware backtracing techniques useful for debugging unknown failure mechanisms

- timing problems
- incorrect connections
- systematic defects

Usually at least one time-consuming step

Can speed up process by using tricks from guided-probe method

Techniques tend to be ad hoc, need to be careful to avoid simple mistakes, using commercial tools can help
Scan-Chain Diagnosis

- Scan design techniques to test and debug nearly universally accepted

- Scan-chain-related logic takes 10%-30% total circuit area
  - Sequential element, scan overhead, routing, and clocks
  - Proportional number of defects affect scan chains
  - Effective and efficient techniques to isolate scan chain failures
  - Equally important compared to logic failures

- Functional scan chains needed to test, debug, and diagnose logic

- Some critical situations require scan-chain diagnosis
  - Zero or very low yield due to scan-chain failures
  - Design and / or fabrication errors
  - Scan path, latch, or scan-clock problem
  - For example, timing errors (hold-time violation) due to clock skew and / or process variation
Examples of Scan-Chain Problems

General Combinational Logic

- Blocked Shifting
- Shift Timing
- Sample Correctness
- Sample Timing
- Systemic Problem

Courtesy Al Crouch
Scan-Chain Problems

- Impact of defects like rest of logic
  - Front-end related
    - Oxide shorts, source-drain shorts, gate-source/drain shorts, doping
    - Inside the library cells
  - Back-end defects on interconnect / routing
    - Signal-to-signal shorts, shorts to VDD and VSS, metal opens

- Design or process-product interactions
  - Clock-tree errors
  - Power distribution weaknesses
  - Test logic verification thoroughness!

- Manifestation
  - Scan shift timing
  - Scan capture / Sample problems
  - Blocked scan chains
  - Scan-shift hold-time problems **
Overview of Techniques

- **Tester based**
  - [De and Gunda ITC-95], [Song et al. IC-99]

- **Hardware based**
  - Hardware modifications beyond basic scan design
  - Special scan cells and / or additional circuitry
  - [Schafer et al, VTS-92], [Edirisoorya and Edirisoorya VTS-95], [Narayanan and Das ITC-97], [Wu SDFT-98]

- **Special tests**
  - IDDQ [Hirase et al. ATS-99]

- **Software based**
  - Use basic scan design and failing tests
  - Sequential ATPG [Kundu VTS-93]
  - Random pattern and matching algorithm [Stanley YOT-00]
  - Modified ATPG patterns and matching [Guo and Venkataraman ITC-01]
Basic Principles

- Scan cells downstream of faulty cell(s) are corrupted
- Scan chain unload values indicate the nature of defect

Good response

Stuck-at 0 response
All 0's at output

Slow to rise response
01 changed to 11
Basic Principles...

Shift in before the block is good
We just can’t see it

Shift is Blocked past here
All bits past the block get the blocking value

Courtesy Al Crouch
Basic Principles…

- Toggle from good chain to bad chain
  - Suspected Bad Bit
  - Observable

- Toggle from good chain to bad chain
  - Unobservable

- Toggle from Pre-Break to After-Break
  - Observable

- Toggle from bad chain to good chain
  - Invalid
  - Good Chain

- Faulty Chain
- Good Chain

Courtesy Al Crouch
Basic Principles…

- Scan cells downstream of faulty cell can be used for isolation
- Scan cells may be naturally set by initialization
- Diagnosis features can be used to initialize values
- Combinational logic can be used to initialize values
Tester-Based Techniques

Static or DC defects in scan chains

- [De and Gunda ITC-95]
- Load a pattern in the scan chain that excites the defect
- Binary search by probing to locate fault cell (slow!)
- Log N steps, N is the length of the failing scan chain

Issues:

1. Limited observability, expense for probing
2. Potential for inadvertent damage and invalidating the diagnosis effort
Software-Based Techniques

[Kundu VTS-93]

- Test generation for diagnosis
- Determine the type of defect
  - Stuck-at 0 or stuck-at 1
- Generate tests
  - Start from the first scan cell
  - Assume initialization downstream of target cell
  - Generate sequence to write opposite value at target cell
  - If aborted or no test, iterate to next cell
- Apply test
  - Passing test implies faulty cell upstream of target cell
- Pros and cons
  - No hardware modifications and overhead
  - Computational complexity, dynamic test generation
  - Combinational logic should be fault free
Software-Based Techniques...

[Guo and Venkataraman ITC-01]

- Algorithmic technique
  - No hardware modifications
  - Applicable to generic scan designs
  - Stuck-at and timing defects
  - Analyze from both scan-in and scan-out side

- Fault models
  - Stuck-at (0/1), slow-to-rise/fall, fast-to-rise/fall, hold-time

- Faulty chain and type
  - Response to three patterns
- Candidate range determination
- Use ATPG patterns
  - Modify load values of faulty chain
  - Mask fault effect during load
  - Use X value in simulation
    - Patterns unaltered on tester
  - Change to all 0’s, all 1’s, only 0-1 transitions or only 1-0 transitions
    - Patterns altered on tester

For stuck-at-0 fault:
- Expected: X 1 0 X 1 0
- Observed: 0 0 0 1 1 0
- Failed propagation of 0 to 1 transition (upper bound)
- Successful propagation of 0 to 1 transition (lower bound)

For slow-to-rise fault:
- Expected: X 1 0 X 1 0
- Observed: 0 0 0 1 1
- Failed propagation of 0 to 1 transition (upper bound)
- Successful propagation of 0 to 1 transition (lower bound)
Phase 3 - Score and rank
- Each cell in the candidate range
- Errors downstream of cell during load
- Errors upstream of cell during unload

Experimental results
- Three designs
- A: 526K gates, 62 chains, 25K scan cells, 404 - chain length
- B: 2.4 M gates, 81 chains, 105K scan cells, 1327 - chain length
- C: 1.3 M gates, 40 chains, 59K scan cells, 1624 - chain length

Design A - Resolution

Design B

<table>
<thead>
<tr>
<th>Unit</th>
<th>Phase 1</th>
<th>Phase 2</th>
<th>Phase 3</th>
<th>FA Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SA0</td>
<td>10</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>SA1</td>
<td>10</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>SA1</td>
<td>68</td>
<td>2</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Design C

<table>
<thead>
<tr>
<th>Unit</th>
<th>Phase 1</th>
<th>Phase 2</th>
<th>Phase 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SA1</td>
<td>69</td>
<td>3</td>
</tr>
</tbody>
</table>
Dealing with Voltage / Timing Failures

- **Shift Patterns**
  - Shift nominal voltage – fails
  - Shift low/high/range - passes
  - Shift in at passing voltage and then shift out failing voltage
    - Places good state in chain during scan load
    - Scan unload identifies neighborhood of fail

Scan In at Clean Voltage

Scan Out at Bad Voltage

Location of Break
Summary of Scan Chain Diagnosis

- **Tester-based techniques**
  - Simple
  - Manual, iterative, and long throughput time
  - Potential for inadvertent damage if used as primary method
  - Typically needed after techniques for finer resolution and characterization

- **Hardware-based techniques**
  - High resolution and simple
  - Hardware overhead may preclude usage

- **Software-based techniques**
  - No hardware overhead
  - Can provide good resolution
  - Should be the first and primary method
Silicon Validation and Debug

- Functional validation
  - Target: Logic or functional bugs
  - Use architectural and design validation tests
  - Code generation - pseudo-random instructions and data
  - Operate at safe operation points

- Electrical characterization
  - Targets:
    - Circuit sensitivities or marginalites
    - Timing or critical speed-path issues
  - Test to extremes of operation region [Josephson et al. ITC-2001]
  - Vary frequency, voltage, and temperature to extremes
  - Characterize with respect to process variation
  - Skew wafer lots with process parameters varied
Debug Environments

- **Tester environment**
  - Easier and more flexible
    - [Rootselaar and Vermeulen ITC-99], [Josephson et al. ITC-2001]
  - Full control of all pins on component
  - Full access to all debug features
  - Patterns can be looped to perform probing
  - Fully deterministic

- **System environment**
  - Cheaper and easier to generate and apply correct vectors
  - Uncovers issues when running real programs (OS and applications)
  - Non-determinism due to interrupts and memory refreshes
  - Harder to control and lack of access to all debug features
  - If possible: quickly locate sections of failing test and replicate problem on a tester
    - [Holbrook et al. ITC-94], [Hao et al. ITC-95], [Carbine and Feltham ITC-97],
    - [Rootselaar and Vermeulen ITC-99], [Josephson et al. ITC-2001]
In-System Problems

- “What’s going on inside the chip?”
- On-chip clock frequency often too high for external access
- Non-deterministic operation → pseudo-intermittent behavior
- Time-specific expected values not known
- Defects that escaped manufacturing test may be detected in system → “No Trouble Found” on tester
- Expensive and unpredictable
  - (at 90nm): 3 to 6 months
  - Economic impact of delays: $M to $10’s M
- Embedded software debug: separate job
- No link pre-silicon ↔ post-silicon
Pre-Silicon vs. Post-Silicon

- Pre-silicon verification environment
  - Tool-rich
  - Automation
  - Advanced concepts
    - Transactions
    - Assertions

- Silicon validation and debug environment
  - Few tools, little automation, bit-level analysis
  - Most commercial debug is for software debug, not hardware

- Separate environments
Typical Debug Flow

1. Find pattern that excites the bug
   - Found during system validation / debug or customer sighting!
   - Pattern already on tester
   - Pop out of the HVM environment due to process shifts
   - Sequential ATPG using existing scan to generate patterns for embedded arrays and queues [Kwon et al. ITC-98]

2. Find the root-cause of the bug
   - Use design for debug features to extract data
   - Use simulation tools and deductive reasoning to arrive at hypotheses
   - Use probing, FIB edits, and modified tests to verify the hypotheses

3. Determine a fix to correct the bug
Find the Root Cause

- Design for debug features

- Controlled operation of chip
  - Stop chip close to point of first internal error and place in test mode
  - Trigger mechanisms programmed to events
  - Clock control mechanisms
  - Clock manipulation

- Internal access (signals and memories) of the chip
Controlled Operation of Chip

- Trigger mechanisms programmed to events
  [Carbine and Feltham ITC-97], [Rootselaar and Vermeulen ITC-99], [Josephson et al. ITC-2001]
  - Control register [Carbine and Feltham ITC-97]
  - Probe mode in system - state monitored
  - Matchers and brakes [Rootselaar and Vermeulen ITC-99]

- Clock control mechanisms
  - Step one clock at a time
  - Step through specified number of clocks

- Clock manipulation
  - Skip cycle or phase
  - Move clock edge - stretch or shrink clock cycle
  - Skew clock region relative to other clock regions
Cycle Stretching

Track down timing failures with an extra-wide cycle, created with a missing clock pulse

- Start with missing clock pulse just before point of observed error
- Move back in time until test passes
- Time of stretched cycle shows time of delay problem
- Can use scan or e-beam probe to proceed further

- Potential difficulty:
  - on-chip clock generation (PLL etc.)
- Design-for-debug: bypass mode, or hardware cycle stretcher (pulse gater)
Clock Control for Silicon Debug and Diagnosis

[Nadeau-Dostie 2000], [Wood ITC-99], [Rootselaar and Vermeulen ITC-99]…

- Clock prescaler: reduce the system clock frequency before it is applied to the BIST circuit ⇒ helps determine if failure is speed-related

- Clock freezing: stop system clock after capture cycle to scan out captured data
  - breakpoints:
    - determined by counting clock cycles
    - activated by hardware matchers programmed to monitor different conditions
  - concatenate all internal scan chains into a single one for “scan dump”
  - allow circular scan register to restore state (if desired to continue debugging)
  - allow clock freezing based on test results (stop after first fail)
Clock Control cont…

- Clock stretching
- Bypass internal (PLL) clock
- ...but allow its use for at-speed test
- Facilities for single-stepping
- Separate clock domains must have independent control
- All clock control implementation: programmable via boundary-scan TAP using private debug instructions
- Debugging support software required - allow user to work without having to know the scan chain ordering, control sequences, etc.
Using Scan for Debug

- Scan provides access to the internal state
- Can be used with cycle stretching, dictionary techniques etc.
- Potential difficulties:
  - stopping clocks at correct point
  - state corruption when switching to from normal to scan mode
  - counting clocks correctly
  - comparing with simulation values (what values are “don’t care”?)
  - time-consuming process
Find the Root Cause...

- Design for debug features...

- Internal access (signals and memories) of the chip
  - Scan snapshots and dumps
    - Destructive or non-destructive to the functional state
    - Sample on the fly - capture state and shift out while chip in operation
    - Restart after reloading scan state and initializing array state
  - Array freeze and dump
  - Observation-only registers

- Typical debug methodology
  - Use trigger to get close to cycle of interest
  - Sample on the fly to narrow range of clocks
  - Clock manipulation and/or scan dumps to isolate cycle of interest
  - Take complete internal observation at offending cycle
  - Use simulation and deductive reasoning to isolate offending circuitry
The Algorithmic IC Diagnosis Process

- Need to identify fault, but more importantly to locate defect
- Physical failure analysis identifies defects and their causes and allows manufacturing process improvement, increased yield, increased profit
Why Do IC Diagnosis?

Goal: identify site of actual failure

- Success: actual site among small number of reported sites
  - ideally 1
  - realistically no more than 10
- Better to report no sites than a list of misleading ones
  - don’t want to waste time/effort on an incomplete suspect list

Different faults may point to same defects

- Defects manifest themselves as different types of faults
  - short $\Rightarrow$ stuck-at, bridge, delay, or IDDQ fault, depending on resistance

Techniques must work for wide range of defect types

- Once a defect type has been found, fab can work to eliminate it
- Specific diagnosis procedures can fail if they are too narrowly targeted
Requirements for IC Diagnosis

In addition to diagnosis software, also need:

- Almost full scan design (99%+)
- High quality scan test vectors (>95% coverage)
- Few unknown values in vectors
- Logical to physical mapping (placement information)
- ATE to apply tests
- Working scan chains
  - Scan chain diagnosis covered previously
- Mapping from ATE fails to diagnosis software
- Desirable to resolve to single logical site
  - Fault equivalence

Diagnosis

Short to GND

From upstream logic

To downstream logic
Extracting Failure Data from Scan Tests

Tester cycle not enough. Need to know which scan elements failed for which tests

- Examples: cycle 1=bit 1 vector 1; cycle 1001=pins, vector 1; cycle 1047=bit 46 vector 2
- Easiest when ATE takes care of this!
Models for Fault diagnosis

- Fault model used for detection
  - model needs to produce a vector that fails when defect fails

- Fault model used for diagnosis
  - model needs to predict, as closely as possible, exactly the behavior produced by the modeled defect
  - much harder problem
  - can validate models by deliberately crafting defective circuits using focused-ion-beam (FIB) and observing their behavior on the tester (Aitken, ITC95)
  - Can also treat model as a noisy predictor and use better quality algorithms to match observations to predictions (Lavo et al, ITC98-02)
Why are stuck-at faults so useful?

- Stuck-at model introduced by Eldred (1959, Bell Sys Tech J)
  - This is a somewhat controversial statement…
- The death of stuck-at faults has been predicted almost ever since
  - Less controversial
- Several reasons why stuck-at faults refuse to die:
  - **simplicity**: easy to apply to a circuit
  - **tractability**: can be applied to millions of gates at once
  - **logical behavior**: fault behavior can be determined logically, so simulation is straightforward and deterministic
  - **measurability**: detection/non-detection are easy
  - **adaptability**: can apply stuck-at faults on gates, systems, transistors, etc.
- **Fearless prediction**: stuck-at faults will continue to be used as long as ATPG exists.
Bridging faults

- Still most common defect type
- Most useful models
  - Dominance
  - Voting

<table>
<thead>
<tr>
<th>1N</th>
<th>1P</th>
</tr>
</thead>
</table>
| 1  | 0  | N wins
| 0  | 1  | P wins

Dominance bridging fault

Vote 1: 1N vs 1P: N wins

Vote 2: 1N vs 2P: P wins
Net Faults

Work at Intel also uses composite models [Venkataraman and Drummonds ITC-00]

- Net fault = composite stuck-at signatures of net stem and all branches

Three sorting criteria

- vectorwise intersection - sort by number of hits by vector
- raw intersection - number of hits including I/Os
- mispredictions - reverse sort by misses
Net Faults

Better accuracy, resolution and precision over stuck-at diagnosis

Very successful on complex nets
- via defect involved 7 stuck-at faults, 1 net fault
- contact defect involved 10 stuck-at faults, 1 net fault
- break defect involved 28 stuck-at faults, 1 net fault

Equivalence
Class of Stuck-at Faults
Design Feature Extraction

Feature Extraction

Prioritized Fault candidates

1. N1 to N5 – 10um
2. N1 to N3 – 5um
3. N1 to N2 – 2.5um

Bridging Lines by Run Length

1. N2 – 8 singles
2. N1 – 4 singles
3. N4 – 4 singles
4. N5 – 4 singles
5. N3 – 2 singles

Open Lines by Run Length

1. N1 – 5um M1, 5um M2
2. N2 – 4um M1, 6um M2
3. N5 – 5um M1, 4um M2
4. N3 – 6um M1, 0um M2
5. N4 – 2um M1, 2um M2

Open Vias by Count

1. N1 – 5um M1, 5um M2
2. N2 – 4um M1, 6um M2
3. N5 – 5um M1, 4um M2
4. N3 – 6um M1, 0um M2
5. N4 – 2um M1, 2um M2

- Extract defect prone features and create targeted tests or characterize existing test sets for diagnosis
I Have Fault Models and Tester Data. Now What?

- Need to reconcile models and observations.
- Two approaches:
  - fault dictionaries – presimulate all faults
  - post-test fault simulation – figure it out later
- Commercial tools use post-test, no longer realistic to build a dictionary, but it’s useful to explain the process…
- Matching techniques, diagnostic algorithms similar for both
Fault Dictionaries

List predicted behavior of all faults in a model

A complete fault dictionary lists:

- every failing output \((m)\)
- for every vector \((n)\)
- for every fault \((f)\)
- Total size: \(O(mnf)\)

<table>
<thead>
<tr>
<th>Vector</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Fault 1</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

fault1: v1 - o2, o3, o4; v2 - o3, o4; v5 - o2, o4; v6 - o1, o2, o3, o4;
fault2: v2 - o3; v3 - o3; v6 - o3;

Several compression/compaction schemes

- example: error sets (repeated failure patterns such as fault 2’s)
Post-Test Fault Simulation

Dynamic cause-effect technique

[Waicukauski and Lindbloom D&T-89]

1. round-up suspects (actual response → plausible faults)
2. drop suspects with good alibis (fault simulate and drop faults whose response ≠ actual response)

- significant reduction in the # of simulated faults (per defective device, compared with static techniques)
- may need to repeat simulations for new models, devices

- remaining suspects (if any) completely match symptoms
- not restricted to one type of fault
- may change fault model if no more suspects
The Preliminary Investigation

1. Trace back from failing POs

2. Combine traced cones
   - single stuck faults: $\cap C_i$
   - multiple stuck faults and bridging faults: $\cup C_i$

3. Eliminate faults based on parity
   - $v = \text{stuck value}$
   - $o = \text{observed error value}$
   - $p = \text{parity of sensitized path}$
Finding and Clearing Suspects

Assume single stuck faults:

- cone intersection: \{F, B, C1, C\}
- the paths fault \(\rightarrow D\) and fault \(\rightarrow E\) must have opposite parities
  \(\Rightarrow\) only C may be the fault site
- \(C\) s-a-1 cannot cause \(D=0\)

- \(C\) s-a-0 is the only suspect left

- remove others from suspect list (don’t have to simulate them for post-test techniques)
Dealing with Gangs

- Suspects are always divided into gangs, each dominated by a leader.

- **Gang** = suspects living together (faults residing in the same single-output subcircuit).

- **Leader** = suspect at the top (activated fault on the subcircuit output = stem fault).

- Any crime committed by a gang member also involves the leader (Any internal fault is always detected at the same POs as the active stem fault.)

- Certain alibis for the leader are also good for all gang members
  - leader detected in other places (different POs) at the time the crime was committed (failing test), or,
  - leader not detected at the time the crime was committed (failing test)
  \[\Rightarrow\] Clear all gang members (drop all the faults in the subcircuit)
No Suspects Left! Now What?

“He’s always got an alibi, and one or two to spare”

*Macavity the Mystery Cat, TS Eliot*

If everybody has an alibi, need some more clues

- Partial matching: guilt by association
- Composite fault models: connections between suspects
- Improved ranking: technology to the rescue
Identifying Suspects via Matching

**Dictionary**

<table>
<thead>
<tr>
<th>Vector</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Fault 1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Fault 2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- entries same as before

**Observed errors:** v1 - o2, o4; v2 - o4; v5 - o2, o4; v6 - o1, o2, o4; (same as fault 1 except o3)

<table>
<thead>
<tr>
<th>Vector</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ob s1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Observed errors:** v6 - o1, o2, o3, o4; (matches fault 1 for v6 only)

<table>
<thead>
<tr>
<th>Vector</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ob s2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Formalization of Partial Matching

Consider correct prediction of observed failures ("hits"), Prediction of failures which were not observed ("misses"), and Observed failures which were not predicted ("nonpredictions")

<table>
<thead>
<tr>
<th>Vector</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Fault 1</td>
<td>H</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>Fault 2</td>
<td>N</td>
<td>N</td>
<td>M</td>
<td>N</td>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>obs s1</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Observed errors: v1 - o2, o4; v2 - o4; v5 - o2, o4; v6 - o1, o2, o4;
- fault 1: 8 hits, 3 misses, 0 nonpredictions
- fault 2: 0 hits, 3 misses, 8 nonpredictions
Partial Matching (cont.)

<table>
<thead>
<tr>
<th></th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Fault 1</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>Fault 2</td>
<td>M</td>
<td>M</td>
<td></td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>ob s1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Observed errors:** v6 - o1, o2, o3, o4;

- fault 1: 4 hits, 7 misses, 0 nonpredictions
- fault 2: 1 hit, 2 misses, 3 nonpredictions

- Suggest giving much higher weight to hits than misses, which in turn should be weighted higher than nonpredictions

- Experimental results good for 10:1:0 weighting for hits, misses, and nonpredictions [Aitken and Maxwell HPJ-95]
Composite Fault Example: Bridging Faults

Simple observation about bridging and stuck-at behavior:
- Bridging fault could not be detected unless some stuck-at fault was detected on at least one of the two bridged nodes A and B [Will73]

Build composite signatures out of stuck-at simulations
- Combine sets of failures for each of 4 faults

<table>
<thead>
<tr>
<th>A s-a-0</th>
<th>A s-a-1</th>
<th>B s-a-0</th>
<th>B s-a-1</th>
</tr>
</thead>
</table>

composite bridging fault signature for A and B (some entries for A and B may overlap)

Match restrictions [Chess et al. ICCAD-95], [Lavo et al. ITC-96]
- Ignore any vectors which do not place opposite values on the two bridged lines
- Not affected by Byzantine Generals

Match requirements
- Require that vectors that detect complementary stuck-at faults be present in signature (e.g. A/0, B/1 or A/1, B/0)
- Could be affected by Byzantine Generals
- Apply a lexicographic ordering to try and recover

Fault location usually in top 10, often in top 3.
IC Diagnosis Summary

- IC fault diagnosis is first step in failure analysis process
- Goal: identify a small number of potential defect sites
- Diagnosis supported by commercial ATPG tools
- Composite fault models, partial matching, and advanced scoring can help find unmodeled defects
- Careful preparation is the key to success
Delay Faults

- Classic delay failures
  - Resistive shorts
  - Resistive opens
  - Crosstalk, IR drop, noise
  - Weakened drive (broken transistor legs)
  - Excess transistor variability

- New mechanisms
  - Variation in interconnect R,C
    - Thin/thick/tall/short wires
    - Systematic portion can be addressed by improved extraction tools
  - “Capacitive faults”
Diagnosis of Delay Faults

Delay faults:
- transition (line) delays
- gate delays
- path delays
- path segment delays

Delay fault testing [*Krstic and Cheng 1998*]:
- 2 vectors ⇒ generate transition(s)
- sample output when it should be stable

Generating transitions in specified FFs: not easy (even in full-scan circuits)

Testing non-scan sequential circuits:
- at-speed test
- variable-speed test *(slow-fast-slow)* ⇒ delay faults activated only once
Generalized Critical Path Tracing

[Girard et al. D&T-92]

- **sensitive input** (●): if input is delayed ⇒ output is delayed
- stable gate inputs (without transitions) cannot be sensitive
  
  ![Diagrams showing sensitive inputs and stable gate inputs](diagrams)

- **critical line**: line whose delay fault *may* be observed at a PO
- POs are always critical
- if gate output is critical ⇒ all its sensitive inputs are critical
- stems are not analyzed; any branch critical ⇒ stem is critical
  (Reason: it *may* be critical for some delay values)
An Example

- Backtrace in turn from every late PO
- Intersect to find common suspects (single fault assumption)
- Union to find all possible suspects (multiple fault assumption) [Ghosh-Dastidar and Touba SDFT-98]
Example...

⇒ Delay fault located on the path segment B→H

Note: this delay fault is compatible with E being on time

Lower bound for the size of the delay fault (look at late paths) [Ghosh-Dastidar and Touba VTS-00]:

\[ D_{BH} > \text{Clock\_period} - \min\{D_{HG}, D_{HF}\} \]
Deriving Information from Passing Tests

Consider only robustly tested paths!

Assume that segment $B \rightarrow H$ is suspect. Why isn’t this fault also detected at $G$?

Upper bound for the size of the delay fault:

$$D_{BH} < \text{Clock\_period} - D_{HG}$$

- Lower bound $< D_{BH} <$ Upper bound
- Lower bound $\geq$ Upper bound $\Rightarrow$ Alibi for suspect
- Extension for sequential circuits [Girard et al. VTS-95]
- Note that this type of thing is complicated and can be very tedious!
Debug of Power Issues

- Low Power Design Overview
- Static and Dynamic Power
- Functional Power Debug
- Fault Diagnosis and IDDQ
Reduced Operating Voltage Is Key

\[ E = \int_{0}^{t} (CV_{DD}^{2} f_c + V_{DD} I_{\text{leak}}) dt \]

Minimize \( I_{\text{leak}} \) by:
- Reducing operating voltage
- Fewer leaking transistors

Minimize \( I_{\text{switch}} \) by:
- Reducing operating voltage
- Less switching cap
- Less switching activity
Example Challenge: Multiple Voltage Domains

- Power \( \sim CV^2 \)
- High voltage only where needed, low elsewhere
- Design requirements: level shifters
- Fault behavior:
  - Bridging faults between logic 1 and logic 1!
  - Reduced VDD has worst case delay at low temperature!
Bus transaction based save and restore to memory
- Bus master implements CRC-32 on the fly
- Useful diagnostic check for “soft” errors while power gated
  - Used to explore effects of in-rush current induced voltage drop
- Could also be used to restore a check point
Debugging Power

- Check for functional failures as well
  - Shmoos are your friends
  - Look for clues in pass/fail behavior
- Dynamic current
  - Gating not working
  - Improper models
  - Design error
  - Defect present
- Static current
  - Sneak current paths, more later
IR Drop and Power Delivery

- In-rush currents with power switches
- Poorly connected IP blocks
- Standard cell rows with weak power connections
  - Failures in seemingly unrelated cells
  - Look for common placement
- Several tools available
Scan and Low Power Design

- DFT rules for power (aka first places to look for debug…)
- Rule #1 for low power:
  - Don’t clock the scan too fast (noise, IR drop)
- Rule #2:
  - Don’t cross power domains with scan chains!
- Rule #3:
  - Put retention flops on their own chain
    - Might consider putting it in “always on” domain
Leakage and Temperature

- Leakage worst at high temperature, fast process corner
  - I_off approximately 10X for every 40-50C
  - Severe leakage-related behavior may occur only at high temperature
  - For debug, try different temperatures!

Graph showing normalized sub-threshold current vs. temperature.
Open Faults

- Resistive via behavior
  - Requires low/room temperature delay tests
  - In memory, can lead to read disturb fault
    - March step: R0, W1, R1, R1, W0
- Classical opens (breaks, cracks) still present
- CMP scratches (complex mix of opens, shorts)
Nature of the Defect

- Resistive defect (particle short)
  - Defect current should change linearly with voltage
- Transistor defect
  - Defect current may behave exponentially
- Open defect
  - Current flow due to weakly on transistors (non-linear)
- Temperature dependence (resistive open)
**IDDQ and Diagnosis**

- Benefit: confirm low leakage (even off states!)
- Can use IDDQ for diagnosis even if unusable in production
- The most important requirement is **static design**!
  - Easy to ask for, but can be difficult to achieve
  - All nodes need to be driven to either VDD or GND at all times
  - Fully complementary CMOS design ideal
    - Otherwise, needs to be static during test and may lose coverage
- Single (or few) global observation point
  - Can’t just mask problem bits on a tester
- Many areas where problems can (and do) occur
- Need to distinguish IDDQ design problems from background leakage.
- Measurement on multiple pins can help (e.g. Acharyya & Plusquellic, ITC 03)
But My Design Is Static!

- Are you sure?
  - Parasitic structures
  - Paths through “off” logic
    - Headers float down, footers float up
  - May be functionally inactive
  - Gate level model may not represent silicon
  - Circuit re-use, modification
  - Incorrect test modes (domain isolation, shutdown)
Typical IDDQ Problems

- Design is probably not static in test
- Common examples of problem areas:
  - tristate bus contention
  - floating nodes
  - conflicts with tester (both driving)
  - degraded voltages, e.g. nFET pass gates
Problem Circuit Areas

- Analog behavior with digital models:
  - Analog/mixed signal cores (HSIO, RF, PLL)
  - RAM at different voltage than core
  - Pads (ESD circuitry)
  - Dynamic circuitry
  - Domain isolation, power switches, clamps
- Clamp unused circuitry
- Tristate buses
- Be careful connecting blocks
  - Parasitic paths to nearby voltage domains
What To Do If Patterns Fail On Tester

- Skip problem vectors
  - Doesn’t help if they are functionally important
- Analyze circuit to identify problems
- Check tester timing
  - are bidirectional pads switching?
  - is circuit verified static at measurement point (typically end of cycle)
- Change tester drives
  - disable outputs from chip
  - weak pullups on inputs
Leakage, Variability and Test
(It’s been a problem for a long time!)

*Josephson et al, IEEE Design and Test, June 1995
Find natural breakpoints: Delta IDDQ

300μA visible on 30mA “normal background” == 1%
(Courtesy Anne Gattiker, IBM)
Min VDD Screening

Data from Madge et al, ITC 2002 (LSI)

- Min VDD changes from die to die, lot to lot
- Outliers are of questionable quality, reliability
- Causes not always clear

66 Mhz memBIST MinV_{DD} outlier:

W-stringer resistive bridge

Courtesy: Bob Madge
Summary

- Low power design increasingly common
- Many techniques in use for reducing dynamic/leakage power
- DFD can help
  - Needs to be applied early
  - Often requires customized support
- Static design key for low leakage design
- IDDQ measurements very useful for fault diagnosis and debugging functional leakage issues, even in high leakage processes
- Min VDD screening provides similar clues if IDDQ won’t work
Yield Analysis and Improvement Application

- Two applications of defect diagnosis in yield improvement

- Yield learning and rampup to volume production
  - Initial effort focused on test chips, memories and in-line monitors
  - Regular and repetitive structure of memories makes diagnosis easy
  - After introduction of first product logic diagnosis critical
  - Differences in layout topology and layers used

```
<table>
<thead>
<tr>
<th>Time</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early phase: SRAM vehicle; Test chip; monitors</td>
<td></td>
</tr>
<tr>
<td>Intermediate phase: First product(s); Unexpected low yields due to Process-product interaction</td>
<td></td>
</tr>
<tr>
<td>Mature phase: High volume Product(s) – yield vehicles; Excursions</td>
<td></td>
</tr>
</tbody>
</table>
```
Yield Application

- Manufacturing excursions and low yield situations

![Diagram showing yield application with excursion and high fallout highlighted.](attachment:yield aplicación diagram.png)
Statistical Diagnosis to Support Yield

Hora et al. ITC-02

- Execute during production, on-line with the wafer test
- Test time and memory is an issue
- Entire lots need to be analyzed
- Fast execution on first few failing vectors using a dictionary
- Determine trade-off between resolution and time using 1-2 wafers worth of data to set number of vectors to be logged
- Plot fail histograms of diagnosis result
- Peaks correspond to areas of interest

42 failing die from 17 wafers on one lot
Logic Mapping

[Balachandran et al. ITC-99], [Kinra et al. ITC-00]

- In-line inspection during wafer processing records anomalies
- Used to monitor process and control defects
- Overlay diagnosis data (nets) with in-line inspection data
- Hits: Nets which overlap with in-line defect location
- Hits improve accuracy and precision of diagnosis results
- Identify process steps which are suspect

Hits between in-line defect data and nets identified by diagnosis
Volume Production Fail Statistical Diagnosis

- Emerging defect/yield learning method for complex logic designs
  - Fail set logging for initial ramp and for volume production test
  - Run logic fault isolation on many/all fail sets (thousands)
  - Write diagnosis information into database
  - Statistically sort, analyze, and visualize the cumulative diagnosis results
Statistical Net-Based Analysis

Net a

Net b

Net c

Net d

Net e

Probability

Typ → 0.18μm
Found 0.15μm

D. Appello, et al., ST Micro
Statistical Cell-Based Analysis

Relative occurrence normalized to occupied area

Observed and calculated fails per die vs cell names

A. Leininger, et al., Infineon

D. Apello, et al., ST Micro
Statistical Spatial Analysis

Stacked Die Level Fail Net Density Map

Stacked Shot Level Fail Net Density Map

Stacked Wafer Level Fail Net Density Map

Stacked die map of fails

CAA based yield model

Leininger, et al., Infineon
State of the Art

- Statistical diagnosis of logic fails is a rapidly emerging design/defect learning technology
  - Learning design-specific, process-specific or product-process issues from product
  - Statistical relevance with large number of samples
- Complement/enhance existing fab-oriented yield management systems
  - Add intra-chip resolution and visibility
- Complement/enhance DFM and yield modeling
  - Provide feedback and calibration
- Most existing solutions are home-grown at IDMs
  - Challenging data security/access issue for fabless/foundry
  - No integrated commercial solution yet (that I know of)
Diagnosis & Changing Role Of Test

- Test may be the first real opportunity to uncover the statistical impact of new catastrophic and parametric defect sources
- Tests can be crafted based on potential defects

“A Rapid Yield Learning Flow …”, M. Keim et al., ITC 2006
Identifying Systematic Defect Candidates

- Minimum diffusion enclosure contacts are susceptible to high resistance failure
- Kim et al., ITC 07

- Non-redundant p-diffusion contact
  - Tighten the guideline by slightly increasing the minimum dimension
  - Minimum diffusion enclosure contacts will be found
Single via location decides the affected nets

- Slow-to-rise and slow-to-fall transition on R1 and R2 paths
- Kim et al., ITC '07
Design Feedback

- Test provides feedback
- Prioritization of detectable faults
- Design fixes
- Kim et al., ATS ‘08
Debugging Design Margins

Design margins are barriers to efficiency

Variations:
Voltage fluctuates between 0.9V and 1.1V
PLL can jitter by 5ps
Silicon can be fast and leaky or slow

Uncertainties:
When exactly is my voltage 0.9V?
When exactly will I get jitter?
When will I get slow or fast silicon?

Uncertainty → margins
Excess margins are wasteful!
Test and Parametric Yield

EDA Model

Parameter

Process Variability

SI

Probability

Statistical Spec

Worst Case Spec

Risk of Parametric Defects
Margins and Test

- At-speed functional test is the classic approach
  - best (but not perfect) correlation with product failures
  - well understood
  - conventional tester architectures based around it
- Becoming increasingly difficult to implement
  - multiple, uncorrelated clock domains
  - expensive to generate vectors
    - typically start with simulation verification vectors
  - expensive and time consuming to make simulation vectors work on the tester
    - need to cyclize
    - need to account for tester loads
    - need to work across process variants
Binning Challenges

- Inventory/demand management
- Over-testing
  - Parts put in low frequency bins = reduced revenue
- Under-testing
  - Parts put in too high frequency bins = DPPM, customer returns
- Test environment ≠ Operating environment
  - Binning all about correlation

Pant et al, Intel, ITC 2010
Correlation and Power Supply

- Managing power supply during test is critical
  - Too dirty $\rightarrow$ unwanted fails
  - Too clean $\rightarrow$ unwanted passes

- Clock after scan load is a massive power impulse
  - Leads to non-intuitive frequency behavior
  - Pessimistic at low F, optimistic at high F
Scan Test Power Environment Details

Due to power supply degradation, operating point changes, thus reducing path slacks

Power supply degradation during capture

Yilmaz et al, AMD, ITC 2010
Correlation is Challenging

- AC scan, ring oscillator, functional test, system all different
- Need to identify tests that correlate well (lots of empirical data)

Bienek et al, AMD ITC 2011
Correlation and “System Level Test”

- Unique system level test failures don’t correlate to other tests
- Correlate strongly to customer failures/performance levels
- Likely to be marginalities that are poorly detected by other means

Data source: Sounil Biswas, nVidia, Ind. Test Challenges, 2011
“Small Delay” Fails

- Baseline testing for high speed parts may not be enough for defect levels in the 100s of PPM
  - Need to augment with poorly understood models (see picture) or “system level test” (plug part into board and see if it works)

Hapke et al, AMD/Mentor ITC 2011
What Causes Small Delay Failures?

- Highly resistive shorts
- Partial opens
- Partial on/off on multi-finger devices
- Extreme cases of device variability

Hapke et al, AMD/Mentor ITC 2011
At-Speed Scan Test for SDDs

- Key is to identify an appropriate WS test frequency for at-speed scan tests
- Static timing analyzer not adequate at predicting true silicon timing
  - STA not predicting order or paths

STA slack

Non-Timing Critical

Timing Critical

M. Mateja, AMD, ITC 2011
Wafer sort data: AC Scan FMAX Histogram

Maximum frequency search results
Greater than 180K fully functional six-core devices
Normal distribution

M. Mateja, AMD, ITC 2011
Wafer sort data: AC Scan FMAX Histogram

Zooming in shows a long tail of slower devices

M. Mateja, AMD, ITC 2011
Overall Economic Impact of Diagnosis and Debug

- Yield ➤
- Manufacturing costs ➤
- System debug time ➤
- Field-diagnostic and repair costs ➤
- Down-time ➤

➤ Accurate and efficient diagnosis has a tremendous positive impact on time-to-market and life-time cost of ownership of a product

➤ Continued innovation needed to meet challenges