At-Speed Test Considering Deep Submicron Effects

D. M. H. Walker
Dept. of Computer Science
Texas A&M University
walker@cs.tamu.edu
Life as a DFT Engineer

Quality

Yield

Test

Cost
Outline

- Introduction
- KLPG
  - Results on Silicon
- Supply Noise & Power
  - Model
  - Results on Silicon
- Conclusions
Test Cost Must Fall Fast

- Test cost/transistor must follow Moore’s Law

Need 100x transistor test cost reduction!
But …

- Test cell cost not following Moore’s Law
  - Already using DFT tester or old ATE
  - Handler and probe card cost not scaling
  - High-speed I/Os cost more

⇒ *Must reduce test time*

- Parallel testing running out of gas

⇒ *Must reduce test time per transistor*
  - Constant test time per chip
Reducing Test Time Per Transistor

- Less time, more transistors ⇒ must wiggle more wires in less time
  
  *Higher power dissipation*

- But… tests/transistor rising to cope with DSM

  *Even higher power dissipation*
But … Max Power/Transistor is Falling

ITRS2005, Cost-Perf MPU
Future Digital Test All About Power?

• Fraction of chip that can be fired up at one time is decreasing
  – Mission-mode power constraints
  – Test supply noise
  – Test thermal limits
  – Limits intra-die test parallelism

• How to screen most defects per Joule?
Eliminate Wasted Energy

• Useless transitions
  – Scan power
  – Unnecessary capture power
  – Much research/commercial activity

• Low-odds test patterns
  – Luck – tails of BIST, WRP
  – Shotgun blasts – N-detect, DOREME, TARO, …
Squeeze Chip Harder Instead

- IDDQ
- MINVDD
- ...
- Small Delay Defect
  - KLPG
Our Delay Test Research

• Defect-Based Delay test ATPG considering:
  – Resistive shorts and opens
  – Process variation
  – Capacitive crosstalk
  – Temperature gradients
  – Power supply noise
  – Power dissipation
Kitchen Sink Fault Model

Have we forgotten anything?

Functional Failure
Local Delay Fault
Combined Delay Fault
Global Delay Fault
Reliability Hazard

Spot Defect
Crosstalk
Noise
Process Variation
Supply
Temp
Die-to-Die
Litho
Intra-Die

Supply
Temp
Crosstalk
Noise
Process Variation

Spot Defect
Target Realistic Defects

Resistive Short
Stanojevic et al

Resistive Open
Madge et al
But...

- Fault population too large
- Limited fault model accuracy
- Limited fab data
- Limited calibration time, cost
- **Fault model must be abstract enough for fortuitous detection of unmodeled faults**
  - “The vectors do the work, not the fault model” – J. H. Patel
Our Approach

• Test K longest rising/falling paths through each gate/line (KLPG)
  – Targets resistive opens
  – Targets resistive shorts
    • Sensitize opposing lines
    • Few bridges per line with largest critical area [Tripp]
  – Larger K deals with delay uncertainty
    • Supply noise
    • Process variation
    • Delay modeling errors
    • Crosstalk
    • Analogy to N-detect
Outline

• Introduction
• KLPG
  – Results on Silicon
• Supply Noise & Power
  – Model
  – Results on Silicon
• Conclusions
K Longest Paths Per Gate (KLPG)

- CodGen ATPG developed at Texas A&M
  - CodSim fault simulator
- Tests K longest paths through each gate/line
- Detects small defects on each gate/line
  - Covers all transition faults
- Needs SDF
- May produce more patterns than TF test
Test Generation Algorithm

Scan cells

Search space

Scan cells

Constraints from outside search space
KLPG Test Generation Flow

Start

- Extend the partial path with longest potential delay
- Apply side inputs and perform direct implications

Conflict?

- N
- Y

Complete path?

- N
- Y

Final justification

Insert into the partial path store

Apply heuristics to avoid false paths

End
Apply KLPG to Industrial Designs

---

**TetraMAX/FastScan**

dofile/procfile/library

---

**Hierarchical Verilog Design**

Same inputs as TF test generation

---

**sdf**

---

**KLPG Test Generator**

---

**CPU Time:**
- ~3x TF test generation
- Memory:
  - 400 MB/1M gates

---

**Test Sequence**
- Load pattern
- Pulse clock

---

**Test Data**
- 010001
- 100001

---

**Tester**

Only 0’s & 1’s are different from TF ATPG outputs
Chips are Slower Using KLPG Test

Transition fault test
10 ns

KLPG-1 test
11 ns

180 nm / 40k gates / full scan / 2.3k scan cells
Cleaner Shmoo

Transition Fault

KLPG
KLPG Silicon Experiment

- TI ASIC design
  - 738K gates (597K gates in 250 MHz clock domain)
  - 130 nm technology
  - 5 clock domains (highest 250 MHz)
  - 8 scan chains, 14,963 muxed D flip-flops in 250 MHz domain

- 24 devices marginally pass regular TF test
<table>
<thead>
<tr>
<th>Test</th>
<th># Patterns</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Delay Test</td>
<td>744</td>
<td>Tests 2 137 critical paths</td>
</tr>
<tr>
<td>Regular TF</td>
<td>1 445</td>
<td>Dynamic compaction</td>
</tr>
<tr>
<td>Randomized TF</td>
<td>1 471</td>
<td>Dynamic compaction</td>
</tr>
<tr>
<td>KLPG-1</td>
<td>12 579</td>
<td>Static compaction</td>
</tr>
</tbody>
</table>
KLPG Test Results

Up to 3% delay decrease seen in KLPG-1

KLPG unique detects
KLPG with Bridge Faults

- KLPG-1 targets resistive opens
- SAF, N-detect, KLPG-1 tests have good coverage of resistive shorts
  - Sar-Dessai and Walker, ITC99
  - Qiu, Walker et al, TECHCON03, VTS04
- Sensitization much easier than propagation
  - Propagate first, then sensitize
- Ignore input-dependent gate strength
- Ignore opposing transition
Bridge Fault ATPG Approach

- Generate longest path through bridge site
- Set DC bits to sensitize opposing value on bridged line (e.g. 0 opposing ↑)
  - No extra uncompacted patterns, since need to test resistive opens
- Else, set opposing value first, then generate path
  - “Top-off” patterns, but may compact
## Bridge Fault Robust LOC Results

<table>
<thead>
<tr>
<th>Circuit</th>
<th># Lines</th>
<th># Shorts</th>
<th>Robust KLPG-1 with Shorts</th>
<th>Robust KLPG-1 w/o Shorts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td># Test Patterns</td>
<td>CPU (m:s)</td>
</tr>
<tr>
<td>s13207</td>
<td>13 207</td>
<td>26 414</td>
<td>1 006</td>
<td>2:30</td>
</tr>
<tr>
<td>s15850</td>
<td>15 850</td>
<td>31 700</td>
<td>520</td>
<td>2:45</td>
</tr>
<tr>
<td>s35932</td>
<td>35 932</td>
<td>71 864</td>
<td>43</td>
<td>15:51</td>
</tr>
<tr>
<td>s38417</td>
<td>38 417</td>
<td>76 834</td>
<td>1 061</td>
<td>15:03</td>
</tr>
<tr>
<td>s38584</td>
<td>38 584</td>
<td>77 168</td>
<td>589</td>
<td>12:00</td>
</tr>
</tbody>
</table>
Bridge ATPG Results

• Modest cost increase
  – Pattern count increases 4.7-19.4%
  – ATPG time increases <9.2%

• Expect less impact on large designs, due to lower care bit density
KLPG Improvements

- Compaction
- Coverage metric
- Crosstalk
Dynamic Compaction

- Test Set Compaction
  - Static Compaction
    - Performed after test generation
  - Dynamic Compaction
    - Performed during test generation
    - Classic method good for stuck-at tests but not suitable for path delay tests

- Develop dynamic compaction for KLPG tests
Dynamic Compaction Approach

Vector pair and NAs (circles) for Path1

Vector pair and NAs (Xs) for Path2

Vector pair and NAs for Path1 & 2
Dynamic Compaction Algorithm

• Definitions:
  – vector: output for ATE
  – pattern: a set of necessary assignments associated with one or more paths
  – POOL: a data structure to save patterns

• Check the compatibility between necessary assignments of new path against a pattern in the POOL

• Generation of final test vector is postponed until test generation is finished
Dynamic Compaction Flow

1. Start with new pattern $F$
2. $POOL$ empty?
   - Yes: Insert $F$ into $POOL$
   - No: Set $P$ to the first pattern in $POOL$
3. Conflict check between $F$ and $P$
   - Conflict?
     - Yes: Set $P$ to the next pattern in $POOL$
     - No: Combine necessary assignments of $F$ and $P$ & Do Final Justification
4. Pass Justification?
   - Yes: Insert $F$ into $POOL$
   - No: Update $P$ with $F$, Reorder $P$ in $POOL$
Dynamic Compaction Experiments

- K Longest robustly testable path generation through each line (K=1)
  - Launch-on-shift/capture
  - Compare to static compaction
- POOL size influence on vector count
- KLPG-1 vs Transition Fault Test
Dynamic Compaction Algorithm

- Definitions:
  - \textit{vector} : output for ATE
  - \textit{pattern} : a set of necessary assignments associated with one or more paths
  - \textit{POOL} : a data structure to save patterns
- Check the compatibility between necessary assignments of new path against a \textit{pattern} in the \textit{POOL}
- Generation of final test \textit{vector} is postponed until test generation is finished
Circuits

• ISCAS 89 benchmark circuits
  – Full scan
  – Unit delay model

• Chip1 (44K Gates)
  – Partial scan
  – Embedded memories
  – Unit delay model

• Chip2a (22K Gates)
  – Partial scan
  – Embedded memories
  – SDF delay
Robust Test (launch-on-capture)

% Vector Reduction Rate

Vector Count

- Static
- Dynamic

37% 23% 26% 43% 48% 21% 39% 55% 53% 60% 60%
POOL Size Influence (LOC robust)
A long transition fault test tests longer paths than a regular transition fault test.
## Test Size (KLPG-1 vs. Transition)

<table>
<thead>
<tr>
<th></th>
<th>Robust</th>
<th>Non-robust</th>
<th>Long TF</th>
<th>Total</th>
<th>Comm. TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>s15850</td>
<td>289</td>
<td>6</td>
<td>7</td>
<td>302</td>
<td>231</td>
</tr>
<tr>
<td>s35932</td>
<td>24</td>
<td>4</td>
<td>0</td>
<td>28</td>
<td>68</td>
</tr>
<tr>
<td>s38417</td>
<td>425</td>
<td>41</td>
<td>1</td>
<td>467</td>
<td>365</td>
</tr>
<tr>
<td>s38584</td>
<td>249</td>
<td>134</td>
<td>70</td>
<td>453</td>
<td>528</td>
</tr>
<tr>
<td>chip1</td>
<td>1192</td>
<td>452</td>
<td>103</td>
<td>1747</td>
<td>1900</td>
</tr>
<tr>
<td>chip2a</td>
<td>619</td>
<td>687</td>
<td>493</td>
<td>1799</td>
<td>2537</td>
</tr>
<tr>
<td>chip3</td>
<td>4406</td>
<td>1688</td>
<td>550</td>
<td>6644</td>
<td>1445</td>
</tr>
</tbody>
</table>
Dynamic Compaction Results

- Dynamic Compaction for KLPG tests
  - Up to 3x reduction in vector count
  - ~2x CPU time increase
  - Small additional memory consumption
  - KLPG-1 test size comparable to commercial transition fault test
Dynamic Compaction Future Work

- Heuristics to accelerate dynamic compaction
- Advanced algorithms for more optimal results
- Dynamic compaction for more complicated industrial designs
- Constraints for power supply noise and temperature
Delay Fault Coverage Metric

- VTS04 metric not constructive for delay test quality
  - Need longest path through each line to accurately compute it – must run KLPG
    - SDQM has same problem
- Die-to-die and intra-die process variation
  - Die-to-die now done as post-process – wasteful
  - Simple bounds on when to stop path generation
    - coverage vs. pattern count
Fault Coverage vs. K

- Drop fault when UB/LB coverage falloff
- Most sites need only a few paths
Ideal K in C5315 with Die-to-Die

- Most sites need 1 or 2 paths
- Most paths in many-path sites are ~same length
  - Can drop most w/o much coverage loss
Capacitive Crosstalk

- Crosstalk affects near-critical paths
  - Don’t worry about near-critical due to spot defect – probability dominated by defect
  - Consider case (b)
Capacitive Crosstalk

• Filter out couplings based on arrival time
• Use simple greedy algorithm
  – Couplings in order of delay increase
  – Sensitize opposing transition one at a time
  – May miss many little coupling case
  – Worry about timing alignment?
    • Probabilistic
• Compaction impact? – More care bits
Crosstalk Alignment

- Need path from PI to crosstalk site to have correct timing
- KLPG ATPG algorithm uses min/max delay constraints
- Targets are opposing transition in timing window
- Constraints narrow as path is built
- If potential alignment/transition is not realized, drop target
- Update timing with each crosstalk site, since could set other crosstalk sites to help or oppose
Outline

• Introduction
• KLPG
  – Results on Silicon
• Supply Noise & Power
  – Model
  – Results on Silicon
• Conclusions
Supply Noise

• Supply noise significantly impacts the timing performance of DSM designs
  – Frequency
  – Gate Density
  – Power Density
  – Supply Voltage
  – Delay sensitivity to voltage

• Excessive supply noise may come from:
  – Random fill of don’t care bits
  – Test pattern compaction

• Noise ⇒ longer delay ⇒ Overkill
Concept: Effective Region

- Circuit extracted as RC network
- Effective Region for a device: RC time constant < Clock cycle
- Assumption: all caps in region are equally effective

No action in current cycle, irrelevant
Discharge in current cycle, effective
Find Effective Region for a Device

- Current Algorithm: search region radius from small to maximum
- Practical improvement: binary search
- Perform only once for one design
Concept: Grid

- Grid is the smallest unit for analysis
  - RC time small enough compared with clock
  - Uniform voltage level

Each Grid Contains:
1) Decoupling capacitance
2) Parasitic capacitance
3) Switching devices

Each Effective Region consists of a set of grids
Grid Noise Model

- Assumption: Off-chip current ignored during the launch cycle
- Switching charge is equally provided by all grids in its effective region
Grid Noise Model

\[ \Delta V_{\text{max}} = \left( \sum (\alpha_i \cdot Q_i) \right) / (C_d + C_p) \]

- **Grid i**: a grid whose effective region covers current grid
- **Q_i**: switching charge of Grid i
- **\alpha_i**: fraction of Q_i provided by current grid
Switching Current Model

- **Dynamic Charging Current:**
  - Look-up table by simulation
  - \( \text{Charge: } Q = 0.5 \cdot I_{\text{peak}} \cdot (t_{\text{end}} - t_{\text{begin}}) \)

- **Short Circuit Current:**
  - empirical function (Saturation Current, wire and device capacitance)
Delay Model

- Look-up table at nominal voltage
  - By simulation
  - Delay = f(t_{in}, C_{out})
  - Out\_slew = g(t_{in}, C_{out})

- Delay/slew is linear to supply voltage
  - linear factor by simulation
Supply Noise Analysis Flow

- Complexity: $O(\text{cell}_\text{count} + \text{grid}_\text{count}^2)$
- Typically $\text{grid}_\text{count}^2 < \text{cell}_\text{count}$
Experimental Design

- Experiments on NXP design
  - 130nm DSP-like design (1M+ transistors)
- LOC path delay patterns with “X” bits
  - statically sensitized paths \(\Rightarrow\) ensures transitions propagate on the target path
- Filling strategy: randomly set “X” bits to 1 with a specified rate
- Generate filled patterns using various fill rates
Experimental Measurements

- Path delay by analysis is correlated with measurement

\[ y = 0.5622x + 1.8262 \]
\[ R^2 = 0.8317 \]
Experimental Measurements

- Noisy patterns cause significant delay increase
- Measured offset due to delay model characterization
Supply Noise Future Work

• Supply noise model refinement
  – Off-chip $dl/dt$ current
  – Array-bond chips
  – Ground bounce

• Better activity estimation
  – Focus effort on noisy patterns
  – Incremental estimation for ATPG
  – Avoid logic simulation
Constant Power Dissipation

- Constant power \(\Rightarrow\) linear temperature rise
- Easy to characterize
  - Know temperature for each pattern
- Adjust capture clock timing
  - Longer delay as temperature rises
  - 35-55% delay increase for 100°C rise in 65 nm
- Reorder patterns for constant power dissipation
- Consider groups of 10 patterns
  - Takes 1-10 ms for \(\sim\)1°C rise
  - 200 bit scan chain @ 100 MHz \(\Rightarrow\) 2 \(\mu\)s/pattern
  - 10 patterns = 20 \(\mu\)s \(<\) 1 ms
Constant Power Flow

- **Issues**
  - Need fast power model
  - Patterns not independent
    - Power due to both scan-in and scan-out switching

**Dynamic Compaction**

**Mentor Preferred Fill**
- Reduce capture power

**Adjacent Fill**
- Reduce average power

**Reorder Patterns**
- Minimize power variation
Power Modeling

- Prior work by Touba et al indicated WSA proportional to scan chain switching
- Improved using scan chain WSA
  - Scan cell feeding more gates likely to cause more circuit switching
  - Most circuit switching during scan happens in first few levels of logic
- Experiments showed almost no difference in pattern reordering results using model vs. simulation (exact) results
Power Model Results

Power Correlation for s38417 per Pattern

\[ y = 8.3539x + 1 \times 10^6 \]

\[ R^2 = 0.9929 \]
Constant Power Algorithm

Compute shift power of each pattern; /* power model */
Group patterns in order using specified group size;
Compute total power $P[i]$ of each group $i$;
Compute average power $ave$ of all groups;
while iteration count not exceeded, do
  for each group $i$, do
    if $P[i] > (1+pvb)*ave$ /* $pvb =$ power variance bound = 0.05 here */
      Find pattern $P_n$ with lowest power in group $j$ with lowest total power
      Select pattern $P_m$ with highest power in group $i$ and swap with $P_n$
    else if $P[i] < (1-pvb)*ave$
      Find pattern $P_n$ with highest power in group $j$ with highest total power
      Select pattern $P_m$ with lowest power in group $i$ and swap with $P_n$
    else
      continue to next group;
  Re-compute shift power for $P_{n-1}$, $P_n$, $P_{m-1}$, $P_m$ /* power model */
  Re-compute total shift power for group $i$, $j$
  Update $ave$;
s38417 Results

Diagram showing the relationship between Group # and Shift Power, comparing Initial Power and Final Power.
Constant Power Results

- Fast - < 1 minute on ISCAS89
- Std. Dev./Average drops by 2.5-6x
  - ~3% on ISCAS89 circuits
- Remaining variation mostly due to high-power patterns
  - Solution: veto high-power patterns during compaction
Conclusions

- Demonstrated KLPG on industrial designs
  - Modest test data volume increase
  - Affordable ATPG time increase
- Demonstrated noise model on industrial design
- Demonstrated constant power reordering
Future Work

- Demonstrate on industrial data
- Fault Coverage Metric
  - Drop faults detected with high probability
  - Exploit spatial and structural correlation
- Maximize coupling capacitance
- Use supply noise model in compaction and filling
- $\frac{dl}{dt}$ model and multi-cycle launch
Acknowledgements

• Current Students
  – Zheng Wang
  – Zhongwei Jiang
  – Shiva Ganesan

• Former Students
  – Jing Wang (AMD)
  – Lei Wu (TI)
  – Wangqi Qiu (Pextra)

• Colleagues at TI, NXP

• Sponsors – NSF, SRC
Needs

• SRC task 1618 liaison
• Design and test data
More Information

- http://faculty.cs.tamu.edu/walker
- http://research.cs.tamu.edu/eda
- walker@cs.tamu.edu
Questions?