The Count of Monte Carlo

Lou Scheffer
Introduction

ï In the Dumas novel, *The Count of Monte Cristo*, the hero is accused of treason and sent to prison

ï However,
  - The charges were trumped up
  - The accusers, though successful and popular, had even worse sins to their name

ï The Count spends the entire (very long) novel restoring his name and discrediting his enemies

ï In the end he triumphs and they are revealed as imposters, and ruined!
Introduction

Likewise, Monte Carlo is accused of many crimes:
- Wasting computational resources
- Missing errors
- Inability to provide constructive feedback

But, as in the novel, these charges are false

The alternative techniques have many flaws too

Monte Carlo is not only practical itself, it may be the only practical method!
Accusations against Monte Carlo

- The following are “well known” problems with Monte Carlo:
  - It is slow
  - It needs a pruned path list: if the pruning is wrong it can miss errors.
  - Even if it works, the results simply tell you what the situation is, not offer any constructive criticism.

Each of these is based in fact, but not the whole story
Monte Carlo is slow

- Based on two misconceptions:
  - If you repeat the whole analysis (say) 10,000 times, then it will take 10,000 times as long
  - Other methods are faster in practice

- But Monte Carlo can be drastically sped up

- Other methods will slow down with real (not toy) problems
  - Big problem is that problem space is very high dimensionality.
Assumptions

- Extraction with sensitivities
- Reduction with sensitivities
- Willing to use a server farm
  - 100 cheap processors (Linux/Intel?)
  - No fast communication
The chip-to-chip variation problem

- First, consider only chip-to-chip variation
- 4 dimensions for each routing layer
  - Width, thickness, ILD thickness, via resistance
- At least 3 dimensions for cells
  - P(rocess), V(oltage), T(emperature)
  - In practice more since N and P can vary independently
- So now we have at least 43 dimensions
- And this is without any accounting for on-chip variations, a major goal of statistical analysis
Extending statistical to on-chip variation

- Simplest form of on-chip variation just adds gradients to each variable
- This turns each variable (such as $V_T$) into 3 variables
  - Mean value, X gradient, Y gradient
- So now we have at least 140 variables for a modern process.
Global variation across the wafer

- Slow variation across the wafer can be modeled as chip-chip variation + (almost) linear variation across the chip.
140 variables ñ thatís not too bad

But is on-chip variation described well enough by linear variation?

Unfortunately, noÖ.
Intra-chip variation is not quite linear

From Stine, Boning and Chung, 1997
Graph is somewhat misleading

- Includes both predictable and statistical fluctuations.
  - Some (probably most) of what you see is caused by predictable factors such as local density
  - Critical to remove these predictable effects first. Don’t use statistics to model these variations!
  - But what’s left is still not linear Ö.

- How many variables are required?
  - Look at design rules ñ different (tighter) device matching if two devices are within 200 microns
  - 20 mm chip implies 100 by 100 grid per basic variable

- Hierarchical methods (Agarwal) used 6 levels
  - 1 + 4 + 16 + 64 + 256 + 1024 = 1365 variables/basic variable
Is this good enough?

- Gridded representations have some limitations
- Correlation of two items in a square is overestimated
- Two nearby items (but in different squares) are underestimated
- Softer (wavelet like) and overlapping squares can reduce this problem, but not eliminate it
- So let's ask what the correct-as-possible, physics based, model looks like:
An accurate conceptual model

- Suppose your chip has 5000 paths that might become critical
- Say each path has 4 gates
- Now you have 20,000 locations on the chip.
- Each must have its own random variable, since each (for example) $V_T$ is different, but all are correlated.
- So there are 20,000 $V_T$s and correlations form a 20,000 by 20,000 *completely dense* matrix
- And this is for a design with few and small paths!
On-chip variation continued

- But $V_T$ is only one of the variables
- There are easily 40-50 others, so the true (physics based) situation is:
**Structure of the correlation problem**

Each metal layer

<table>
<thead>
<tr>
<th>M1t</th>
<th>M1w</th>
<th>Ild1-2</th>
<th>Via1-2</th>
<th>M2-10</th>
<th>Vt</th>
<th>Δl</th>
<th>Δw</th>
<th>μ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
</tr>
<tr>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
</tr>
<tr>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
</tr>
<tr>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
<td>Ö</td>
</tr>
</tbody>
</table>

If 5000 paths, 4 gates/nets per path, then each colored square represents a dense 20K by 20K matrix.
So many dimensions! What’s your point?

- In theory this is no problem
- But some methods don’t work well in practice with this many dimensions
- For example, (Chang & Sapatnechar, ICCAD 2003) requires decomposing the correlated variables into independent random variables
  - But this decomposition is $O(N^3)$
  - 512,000,000,000,000 operations in this example
  - And you thought Monte Carlo was slow!
Another example from IBM DAC 03 presentation.

Note that all Monte Carlo variants are faster than the other proposed methods!
But Is Monte Carlo any Faster?

Lots of ways to speed up Monte Carlo
- Path pruning (discuss this later)
- Generate response surfaces
- Parallelization
- Simulating many similar cases/paths ñ simulators can take advantage of this.
  - Implies you can only parallelize by paths, but thatís OK
Response surfaces

- Find delay as a simple polynomial (often first order) in parameters
- Say we have a path that depends on 100 parameters
- Instead of 10,000 simulations, we can do 100, compute the coefficients, then do the remainder by simple dot products.
Rough times with parallel processing

- Pick 10,000 paths
  - The Monte Carlo to do this can be done in parallel
- Need 100 simulations each, say 10 sec each, even if we need circuit level analysis
- That’s a total of $10^7$ seconds (1/3 year)
- Use 100 processors
  - Evaluating a path is good match to a small cheap CPU
  - Bank of Linux/Intel boxes fine for this
- Now $10^5$ seconds (about 1 day)
Supposed flaw #2 ñ path pruning

The accusation is:
- Monte Carlo is performed on paths
- There are too many paths for exhaustive evaluation
- Usually look at the N worst paths
  - Perhaps the union of the worst path under the 10 most common corners (2 cell x 5 interconnect)
- But what if there is some path, not in the top N nominal paths, that is highly sensitive and hence often the true critical path?

This is a legitimate concern
Worst case example for path pruning

A huge number of identical paths with narrow distributions

Many independent, slightly faster paths with a wider distribution

Number of paths

Delay

Probability

TAU 2004
Lou Scheffer, Cadence
Path pruning

- Solution ņ donít use the 10 common corners (or at least donít use these alone)
- Need more Monte Carlo, not less
- Generate worst paths in N Monte Carlo cases, take the union of these
- Easy to see why this works ņ if a path can show up in production in any appreciable percentage it should show up in one or more MC cases.
  - For N=100, at most $10^{-5}$ chance of missing a 10% yield effect
  - Actually better than this since we can estimate correlations
Possible flow

- Generate Monte Carlo cases
- Variational extract
  - Variational reduction
    - Delay calc
    - STA
    - Distribution and correlation at endpoints
    - Preliminary timing yield
      - OK?
        - Modify circuit
        - Union of plausible critical paths
          - Path 1
          - Path 2
          - Path N

~100 ~10000
Path pruning

- Also provides a way to answer "How many paths do I need?"
- Given the STA runs, can find distribution of each endpoint
- Given a non-critical endpoint can find the distribution and correlation
- This allows you to find the odds this is really critical
Example

- Suppose, in run 27, path A is critical, B is next
  - A has a delay of 1000 ps, B only 900 ps
- From the STA runs, can find the variance of A and B, and the correlation between A and B
- This allows you to find the odds that B > A
- Then you can decide if you need to include this path
- Similar process gives timing yield estimation
Worst case example for path pruning

A huge number of identical paths with narrow distributions

Many independent, slightly faster paths with a wider distribution
Accusation 3: No constructive criticism

- Methods that run on timing graphs can (in theory) incrementally evaluate changes

- Accusation: Monte Carlo is inherently batch
  - Not true — each step in initial analysis can be incr.
  - Can be almost as incremental as non-statistical flow

- Even if batch, this is the case now for detailed route/extraction/delay calculation, and we’ve made that work (though it is not ideal)

- Furthermore, Monte Carlo results can be mined for lots of useful sensitivity data
Short paths cause little problem

- They need to be fixed no matter how they are found
- The can be fixed without detailed re-examination
  - Fix with good margin usually easy to find
Long paths seem to be the problem

- Rip up old logic that was not good enough
- Replace with new or modified logic
- How to evaluate this new logic?
  - PDF/CDF propagation has natural incremental extent.
  - But Monte Carlo on parallel machines is faster!
- And what values are being propagated? They are crude guesses anyway (no detailed routing/extract/crosstalk during synthesis)
- So maybe a fast but crude estimator can/should be used. More research is called for Ö.
So let’s re-examine the charges

- **Speed:**
  - Not so guilty as charged, and the accusers are not so innocent either

- **Missed paths**
  - Can be fixed with more Monte Carlo
  - With parallel processing, good performance and incremental

- **Lack of constructive feedback**
  - Parallel processing may solve ſ can do incremental MC
  - It’s not clear accurately propagating rough guesses is any improvement.
Advantages of Monte Carlo

- Theoretically sound
  - Commonly used as a golden reference

- Easy to explain; worst cases can be explicitly demonstrated

- Can do gate or circuit level, or a mixture.
  - Circuit level requires no cell characterization, which could be really hard when all effects are included

- Can incorporate other effects easily
  - Crosstalk
  - IR drop
Conclusions

- Monte Carlo is perceived as only suitable for "golden reference" work
- But the drawbacks are not as serious as thought
  - Can be made fast enough for practical work
  - Missed paths can be minimized
  - Can provide good feedback for improvement
- More sophisticated methods will have problems with real world cases, too
- MC has some very compelling advantages
- Monte Carlo will not merely endure, it will prevail!
Maybe some variables are better behaved

- If perfectly correlated, need only one path
- If perfectly uncorrelated, need only ~40 paths
  - 40th is critical if worse than 39th, 38th, ..., 1st
  - Each has odds of less than $\Omega$
  - So overall the 40th path will seldom ($<10^{-12}$) be critical
- But in real chips many thousand of paths need to be considered (see next plot)
- Ugly mathematical territory where no variables can be combined, and none can be dropped
Distribution / # Paths

1M Gate IBM/ASIC Design (130 nm)
5 sources of variation, Monte Carlo analysis

From IBM DAC 03 paper

Critical Slack (ns)
TAU 2004
Lou Scheffer, Cadence