Validation and Debugging of Statistical Analysis Key to Robustness of Cadence SSTA solution

Cadence Design Systems
Sachin Shrivastava

Canvas Conversations

cādence™

Presented at

cadence designer network

cdn™ LIVE

Silicon Valley 2007

Validation and Debugging of Statistical Analysis

Key to Robustness of Cadence SSTA solution

1. Abstract

Statistical Timing Analysis is a methodology to create a robust and tractable framework to analyze timing of a design in the presence of process variations. A key challenge that lies before customers is the creation of a methodology for validation of results. We need to create testcases that specifically target the effects of various types of process variations, and additionally to create independent validation scenarios that allow us to gauge the accuracy of the analysis. We look at the challenges involved in this, and describe how to create solutions to address the challenges. One standard technique that is used is Monte-Carlo spice. We discuss the infrastructure we created for extracting paths from design for Monte-Carlo simulations. Since there are multiple new points of analysis involved in SSTA, debugging SSTA accuracy issues are extremely difficult, in this paper we look at the techniques developed for debugging accuracy issues. We also look at techniques to isolate the effect of different effects - this helps in debugging any differences in results of SSTA and Spice.

2. Introduction

With shrinking process node sizes, the inherent effect of process variations is playing a larger factor in defining the behavior of a circuit. Conventional Static Timing Analysis (STA) using best case/worst case analysis is overly pessimistic, and could be optimistic also in some cases. This has resulted in the promotion of Statistical Static Timing Analysis (SSTA) as a method for estimating yield of a circuit in terms of timing activities.

There is significant literature available which talks about methods of performing statistical analysis [1]. All these methods approximate statistical yield calculation using a simplified method which can be computed in real time. Since the methods used in SSTA are very different from the golden Monte-Carlo, it becomes very difficult to debug any accuracy issues between two different types of methos.

In following sections, we talk about methods for validation and debugging accuracy of statistical analysis.

In section 3, we explain methods of generating spice netlists which can be used for running monte-carlo simulation, in order to generate golden results.

Output of SSTA tool is probability density function which is represented using mean and standard deviation. Hence for correctness of the results, we need to ensure that both mean and standard deviation of SSTA correlates with Monte-Carlo results.

In section 4, we explain method for debugging mean value mismatches. In section 5 we explain debugging techniques for standard deviation accuracy issues.

3. Creating Monte-Carlo Spice setup

Creating Spice netlist

For a small circuits, DSPF can be directly used, and if DSPF is not available, SPEF can also be converted to spice format using quick scripts.
For big designs, where simulating full netlist in spice is not possible, a path specific spice deck can be extracted. We use Celtic-NDC for extracting spice deck of a path [2].
Executing following commands in Celtic-NDC dumps, path specific spice deck including excitations.

```
set_pathsim_mode -spiceout -results_dir directory
report_timing  -from_<rise/fall>  from -
through_<rise/fall> through1 –<rise/fall>  -to endPoint
```

Adding Monte-Carlo analysis statements

Desired variations can be applied in Spectre using following statements [3],

```
statistics {
 process { // process: generate random number once per MC
run
     vary pg1 dist=gauss std=12 percent=yes
     vary pg2 dist=gauss std=pg2_std // pg2_std is a
parameter
     ...
 }
 mismatch { // mismatch: generate a random number per
instance
     vary pr1 dist=gauss std=2
     vary pr2 dist=gauss std=0.5
 }
}
```

Then monte-carlo analysis can be performed using following statements,

```
alias measurement dcmeas {
    export real delay
    run tran(stop=40n)
    real in=cross(sig=V(in), thresh=0.6, dir='fall, n=1)
    real out=cross(sig=V(out), thresh=0.6, dir='rise,
n=1)
    delay=out-in
 }

run montecarlo (scalarfile="mc.dat",donominal='yes’,
variations='mismatch’, firstrun=1, numruns=1000 )
{
run dcmeas
}
```

4. Debugging Mean Accuracy Issues

In following section, we explain step by step procedure which is used for debugging any issue in accuracy of the mean value.

Basic Checks

- Is slew in SDC in accordance with library
  (e.g. using 30-70% slews in SDC, with 20-80% slew library)
- Is library characterized with pre-driver
- Check voltage, temperature, spice models in spice run same as char condition

Next Level Checks

- Do report timing in STA mode, with following options

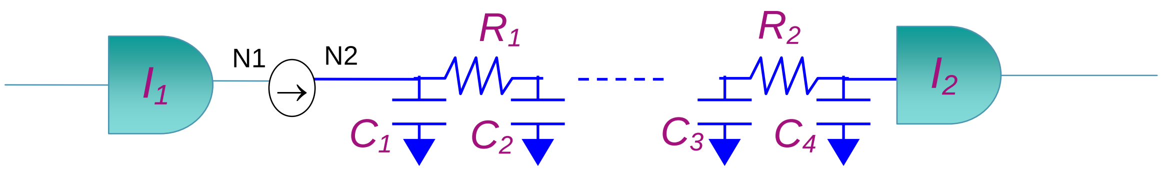
```
report_timing –format {instance arc delay slew load} -net
```

- Perform stage by stage delay/slew comparison
- Pick the first instance which shows difference in delay
  - Is this first instance, connected to input port
    - Check if slews used are same
    (SDC has slews specified in library threshold, while PWL in spice will have 0-100% slew)
    - If library is characterized with pre-driver, and spice is excited with linear ramp
      – First instance is expected to have some error
      – Use non-linear PWL in spice also

  - Is this multi input cell
      - Check if condition used at other pins are same in spice and SSTA
      - To check conditions used by CTE use report_cell_instance_timing
      - Ensure that library has conditional delays
      - Ensure, Library does not have a default arc delay (which is MAX of all arcs)

- Using NLDM
    - Run the tool in debug mode, where it prints Ceff (C effective)
    - Input slew to an instance is available from report_timing
    - Check delay tables in library
    - Extrapolation/Intrapolation issue ?
        - Re-char the library with more points
    - Does library look-up delay matches spice results ?
        - Yes, Its delay calculation issue
    - Else, It’s a library issue or C-eff issue
    - Validate library by running spice (not covered here)
        - If errors, report to characterization team
    - Else, its could be C-eff issue
    -

Measure C-eff in spice



Sample spice deck

```
vtmp N2 N1 0
.meas tran time0 when V(N1)=0.01*vdd
.meas tran time1 when V(N1)=0.5*vdd

.meas tran charge integ I(tmp) from = 'time0' to 'time1'
.measure capacitance param = 'charge/((0.5 - 0.01)*vdd)’
```

```
vtmp N2 N1 0
.meas tran time0 when V(N1)=0.01*vdd
.meas tran time1 when V(N1)=0.5*vdd

.meas tran charge integ I(tmp) from = 'time0' to 'time1'
.measure capacitance param = 'charge/((0.5 - 0.01)*vdd)’
```

- Using ECSM
    - With, ECSM perform normal NLDM debugging, which will make sure that results are not very off

(ECSM ensures last 5-10% accuracy)

- ECSM debugging, requires knowing proprietary SgS information which can not be shown here
- But following can be checked

- ECSM waveforms are monotonic
- Run spice experiments, and check if ECSM waveforms are correct
- Remove ECSM tables from library and re-run

If it improves the results, report it to R&D

5. Debugging Standard Deviation Accuracy Issues

Before debugging STD issues, ensure mean is correlating. Fixing mean issues, will most likely fix most of the STD issues also.

Global Variations

Check if sensitivities are matching or not ?

If Yes,

Either delay varies non-linearly with XL,VTH

Or, linear combination of XL and VTH is not correct

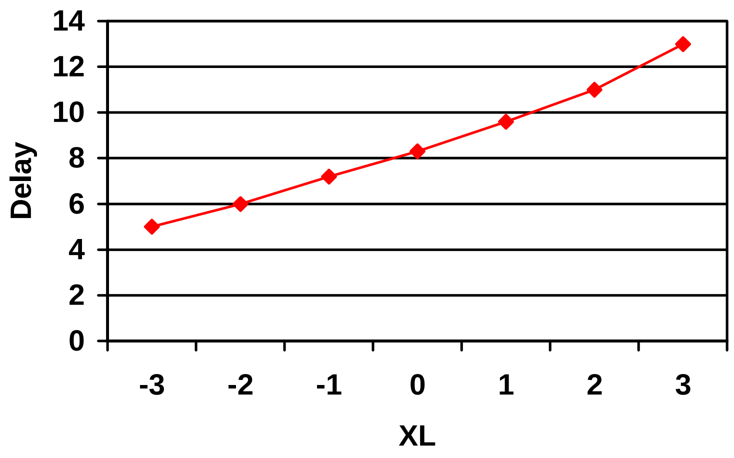
If No,

SSTA, sensitivity calculation is not correct

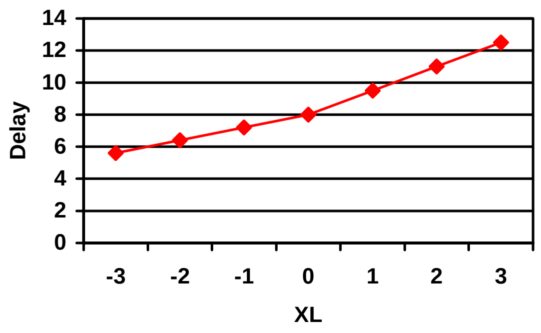
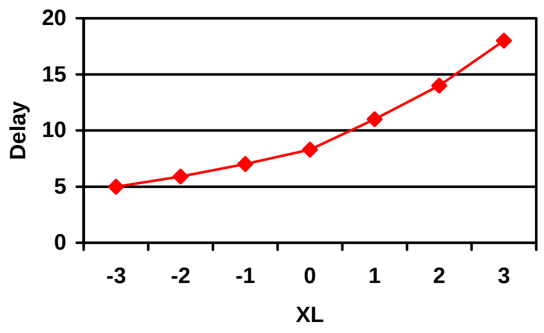
Checking non-linearity

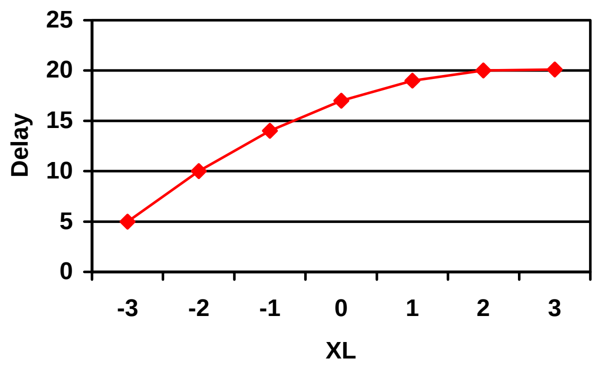
In spice sweep process parameter from -3*sigma to +3*sigma, and observe the linearity

```
<Existing spice deck>
.alter
.param XL=-3*sigma
.alter
.param XL=-2.5*sigma
..
..
.alter
.param XL=0
..
..
.alter
.param XL=3*sigma
```



If plot is found to be linear, try changing characterization points.

If plot is not symmetric around mean, try –sigma to +sigma	
If plot is concave and results are optimistic, try 0 to 3*simga characterization	

If plot is convex and results are optimistic, use 1*sigma (if using 3*sigma)	

Checking linear-combination error

```
<Existing spice deck>
.alter
.param XL=3
.param VTH=0

.alter
.param XL=0
.param VTH=0.7

.alter
.param XL=3
.param VTH=0.7
```

Check
(D1-D0)+(D2-D0) = (D3-D0)

If this check fails, there is interdependence. Select parameters with uncorrelated delay dependencies.

Sensitivity Calculation
Measure delay/slew sensitivity of each stage in spice

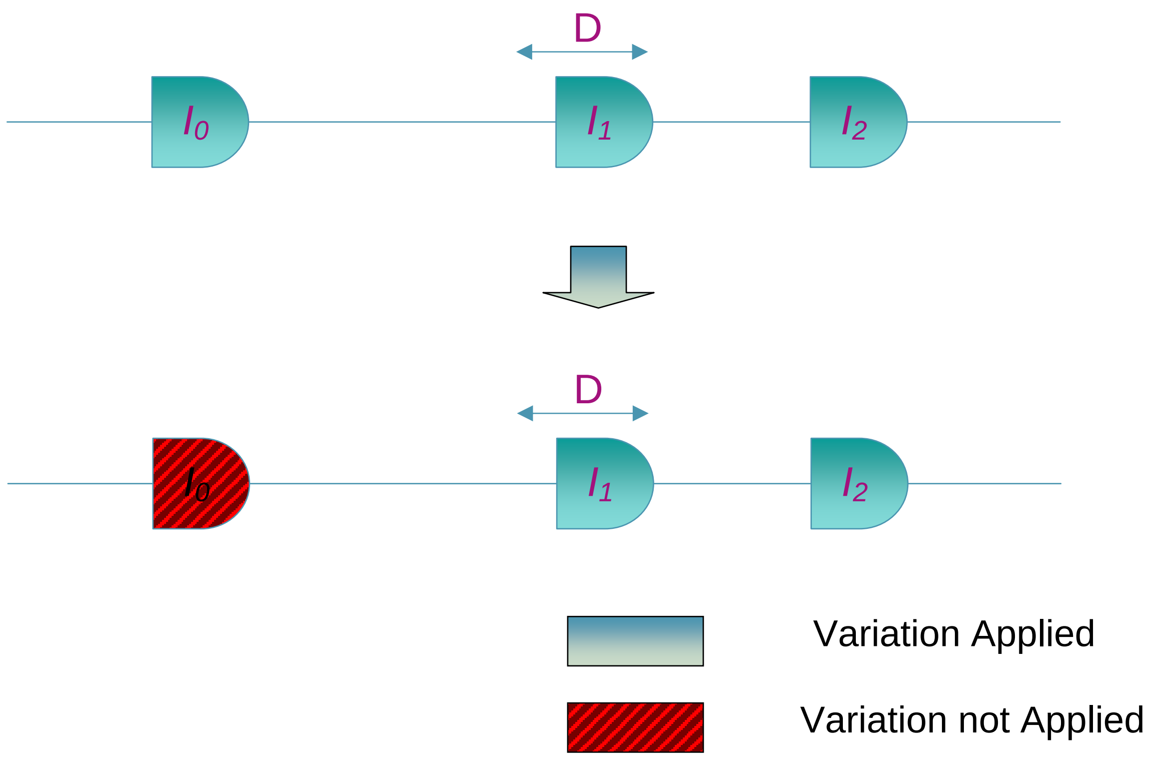
```
<Existing spice deck>

.meas tran delay trig V(IN1) val=0.5 cross=1  targ
    V(I3_A) val=0.5 cross=1
.meas tran wire0 trig V(IN1) val=0.5 rise=1  targ V(I1_A)
    val=0.5 rise=1
.meas tran cell1 trig V(I1_A) val=0.5 rise=1  targ
    V(I1_YB) val=0.5 fall=1
.meas tran wire1 trig V(I1_YB) val=0.5 fall=1  targ
    V(I2_A) val=0.5 fall=1
.meas tran cell2 trig V(I2_A) val=0.5 fall=1  targ
    V(I2_YB) val=0.5 rise=1
.meas tran wire2 trig V(I2_YB) val=0.5 rise=1  targ
    V(I3_A) val=0.5 rise=1

.alter
...
...
```

Compare the SSTA sensitivities with spice sensitivities. Pick the first instance, where sensitivities do not correlate

Now stage sensitivities can be debugged using following,

- Run SSTA by disabling following features (one by one)
  - Slew Sensitivity (Effect of slew change on delay)
  - Capacitance Sensitivity (Effect of Ceff change on delay)
- From above run calculate following sensitivities
  - Primitive Delay Sensitivity
  - Delay Sensitivities (Due to slew sensitivity)
  - Delay Sensitivities (Due to capacitance sensitivity)

- Spice runs, do not have above sensitivities separated !
- Separating Slew sensitivity components in spice



- If delay sensitivity (due to slew sens) is not matching, check following
  - Slew sensitivity of previous stage
  - Delay sensitivity to input slew (for current state)
- If primitive delay sensitivity is not matching
  - Follow steps similar to Mean debugging
    - C-eff
    - Library Look-up
    - Library Validation with spice experiments

Random (WID) Variations

For random variations, most of global debug methodology can be reused, the only difference is that, generating random sensitivities in spice requires MC runs in mismatch mode (instead of two simple runs for global). And resultant STD of MC is actually random sensitivity.

6. Conclusion

A complete methodology was presented for validating statistical timing analysis results, and we also presented methods for debugging if there are any accuracy issues.

7. Reference

[1] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, "First-Order Incremental Block-Based Statistical Timing Analysis", DAC 2004, pp 331-336.

[2] Celtic-NDC User Guide

[3] Spectre User Guide