

A Practical Framework for Statistical Leakage Estimation

Kyung-Tae Do, Jung Yun Choi, Seokhoon Kim, Hyein Lee, Hyo-Sig Won and Kyu-Myung Choi
{kyungtae.do, jungyun74.choi, fuhaha99.kim, hyein23.lee, hs.won, kmchoi}@samsung.com

Design Technology Team
System LSI Division
Samsung Electronics Co., Ltd.

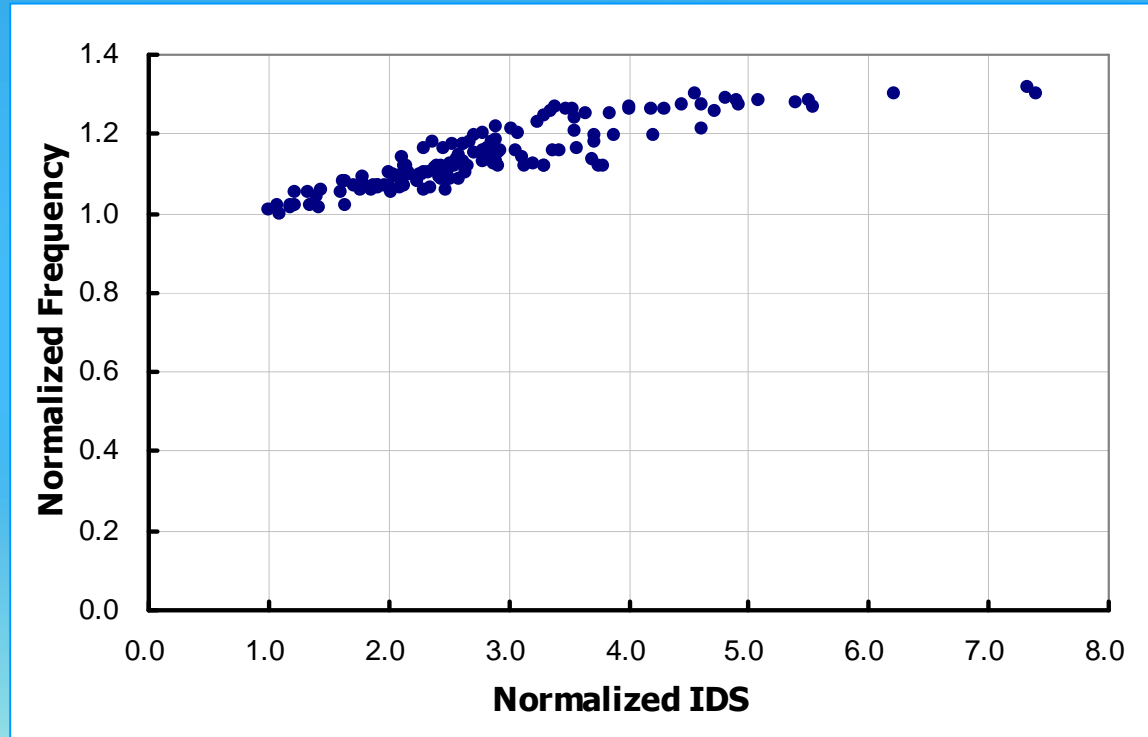


Outline

- ◆ Motivation
- ◆ Preliminaries
- ◆ Framework for Statistical Leakage Estimation
 - Library Characterization
 - Silicon Characterization
 - Leakage Calculation
- ◆ Pilot Test Result
- ◆ Summary

Motivation

- ◆ Variability at 45nm and below nodes is more significant than ever before
 - We see the leakage variation **up to 600%** in 45nm node due to its exponential dependence on the process variation



• Normalized Leakage Vs. Frequency Plot (Mobile CPU Core)

- Traditional corner-based leakage estimation tends to **over-/under-**estimate the leakage under this large process variation due to its inherent limitation

Motivation

- ◆ Statistical leakage estimation is promising approach, but
 - It requires a new extensive leakage library which is largely dependent on the model used for representing the distribution of the leakage
 - Library preparation for the statistical leakage estimation usually requires sensitivity calculation for variability parameters and Monte-Carlo simulation
 - It is not applicable to the characterization of large macros
 - It is hard to introduce scalability for the operating condition (supply voltage, temperature) to statistical library
- ◆ Presented statistical leakage estimation framework
 - : a framework that can be standardized and plugged into conventional flow
 - **Library Aspect:**
 - Generalized representation for cell-level leakage
 - Scalability to operating condition different from characterization condition
 - Handling of practically non-characterizable IP/macros
 - **Estimation Aspect:**
 - Simple and hierarchical calculation for large designs needing high capacity
 - Consideration of real silicon behavior in the analysis

Preliminaries

◆ Statistical leakage estimation methods

● Monte-Carlo simulation-based method

- Create random sequences modeling the process variation and repeatedly estimate the full-chip leakage current using the sequences to calculate the distribution of the full-chip leakage current
- **Computationally expensive** due to excessive simulation

● Analytic method

- Based on the exponential nature of leakage current on process variation, cell leakage currents are modeled in lognormal random variables (ie. e^X , $X \sim N(a, b)$)
- The sum of lognormal random variables, full-chip leakage current, is approximated in a single lognormal random variable using various method (but, mostly with Wilkinson's method(1st and 2nd moment matching)*)

$$I_{leak, full-chip} = e^{X_1} + e^{X_2} + e^{X_3} + \dots + e^{X_N} \approx e^Z$$

- Presented framework is based on the analytic method of using Wilkinson's method while suppressing its complexity using hierarchical partitioning

* S. C. Schwartz and Y. S. Yeh, "On the distribution function and moments of power sums with lognormal components," Bell Syst. Tech. J., vol. 61, pp. 1441–1462, Sept. 1982.

SEC Leakage Estimation Framework

Library Characterization

- Monte-Carlo based cell leakage characterization
- Scaling factor for different operating condition characterization
- Correlation between different input states characterization
- Non-characterizable cells handling using conventional leakage libraries

Silicon Characterization

- D2D and WID variation decomposition
- Spatial correlation characterization for different TEG patterns

Leakage Calculation

- Cell-level leakage calculation for desired operating condition based on Wilkinson's method and input state probability
- Full-chip leakage calculation using Wilkinson's method and silicon characterization result with hierarchical partitioning of the design for runtime reduction

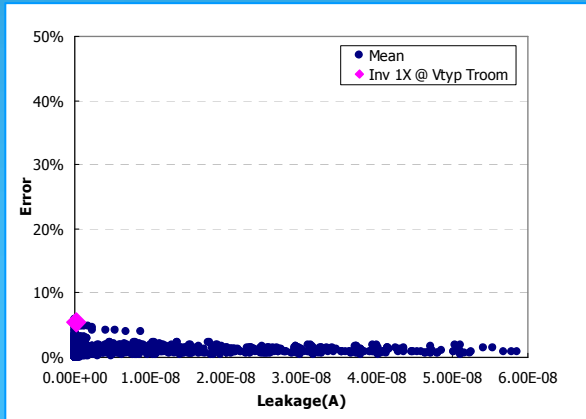
Library Characterization

- ◆ Cell leakage characterization – basic leakage values extraction
 - Unlike most analytic statistical leakage estimation methods, not the sensitivities for variability parameters but the mean and standard deviation of the leakage current are characterized from HSPICE Monte-Carlo simulation
 - Sensitivity-based methods may also require Monte-Carlo simulation to calibrate the inherent error from their low complexity leakage model (mostly, exponential polynomial model)
 - From the Monte-Carlo simulation results, the correlation between leakage values of different input states is also characterized
 - To manage long runtime of Monte-Carlo simulation based characterization, the following techniques are applied
 - The number of min. required Monte-Carlo iteration is calculated from 95% confidence interval range (range < 10% of calculated mean or std dev.)
 - Quasi Monte-Carlo simulation can also be applied
 - Leakage simulation is done in DC analysis mode
 - If internal node initialization is required (ie. F/F, Latch), the initial condition from normal condition transient simulation is applied

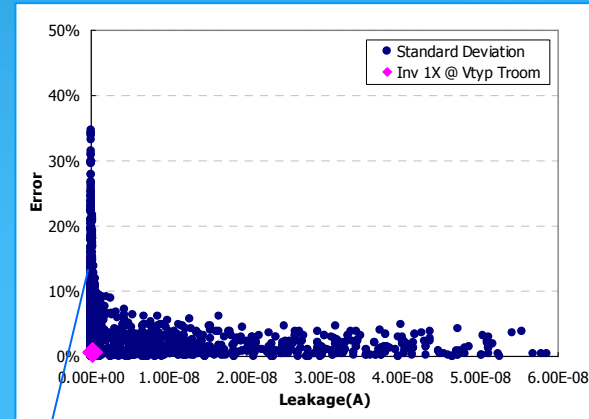
Library Characterization

◆ Cell leakage characterization – basic leakage values extraction

● Error plot of Monte-Carlo iteration number control method

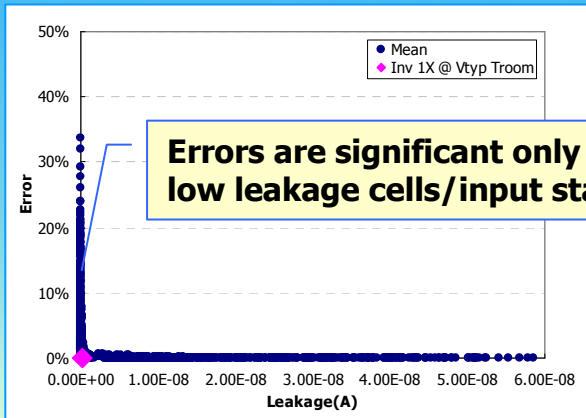


< Error plot of the leakage mean >

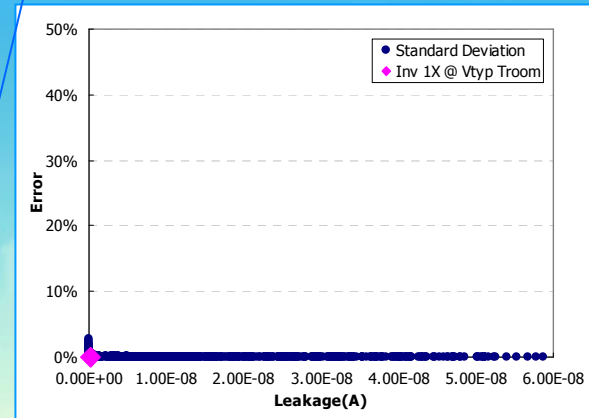


< Error plot of the leakage standard dev. >

● Error plot of DC analysis based Monte-Carlo simulation



< Error plot of the leakage mean >



< Error plot of the leakage standard dev. >

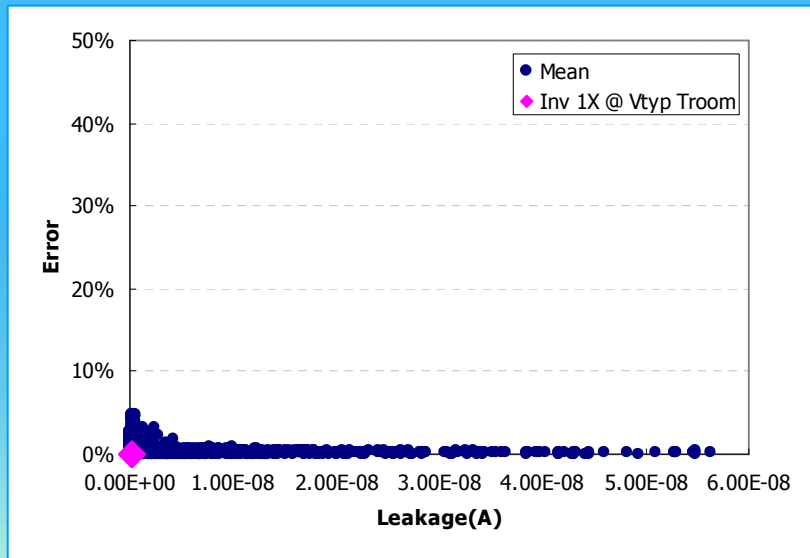
Library Characterization

◆ Cell leakage characterization – operating condition dependency

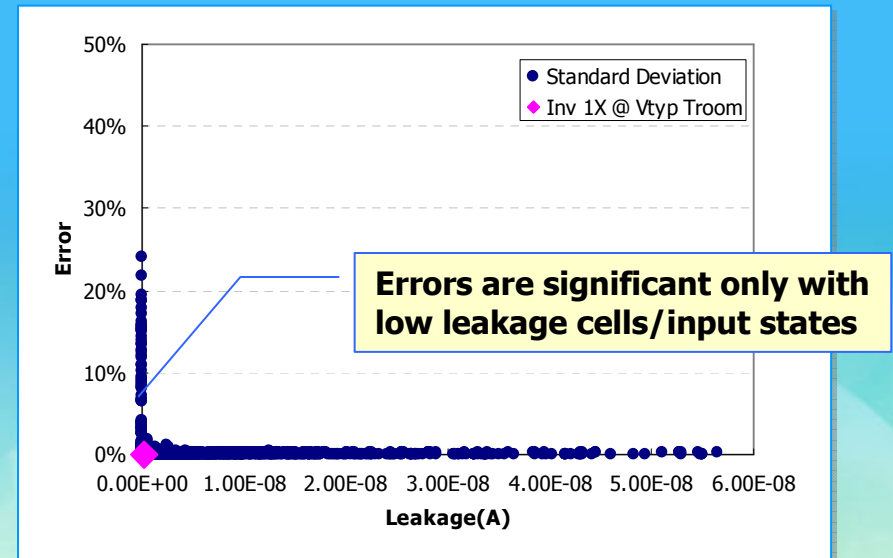
- Operating condition (supply voltage, temperature) dependency of the mean and standard deviation of leakage currents is characterized in a scaling factor with the following equation ($a \sim f$: coeff., V : supply voltage, T : temperature)

$$Scaler(V, T) = \exp(aV^2 + bT^2 + cV + dT + eVT + f)$$

● Error plot of the scaling method



< Error plot of the leakage mean >

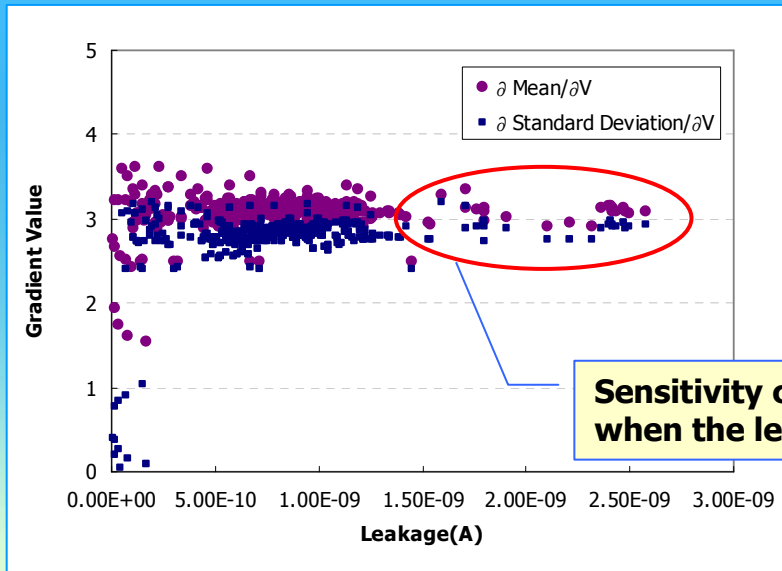


< Error plot of the leakage standard dev. >

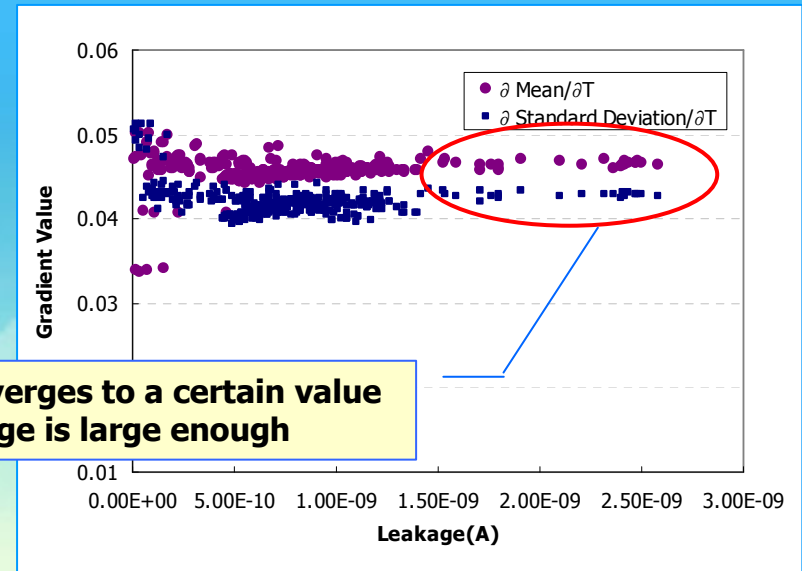
Library Characterization

◆ Cell leakage characterization – non-characterizable cell handling

- For large macros like SRAM, as Monte-Carlo simulation is practically not possible, the mean and standard deviation of the leakage is derived from corner libraries (combination of NN/SS/FF condition library)
 - SS, FF corner leakage values are considered as -3 and 3 sigma leakage values
- For non-characterizable cells, the generalized operating condition dependency is characterized from collected characterization results and the gradient of scaling equation for operating condition
 - Gradient converges to a certain value when the leakage is large enough



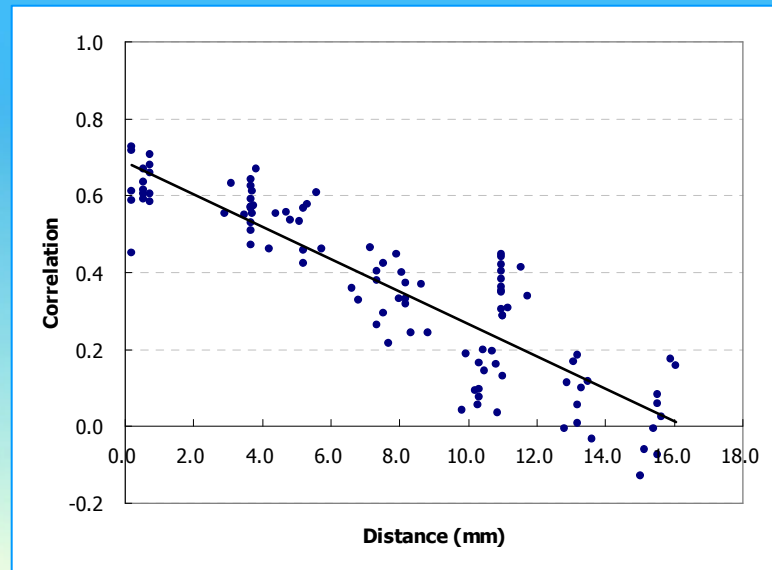
Sensitivity converges to a certain value when the leakage is large enough



Silicon Characterization

◆ Silicon characteristics extraction

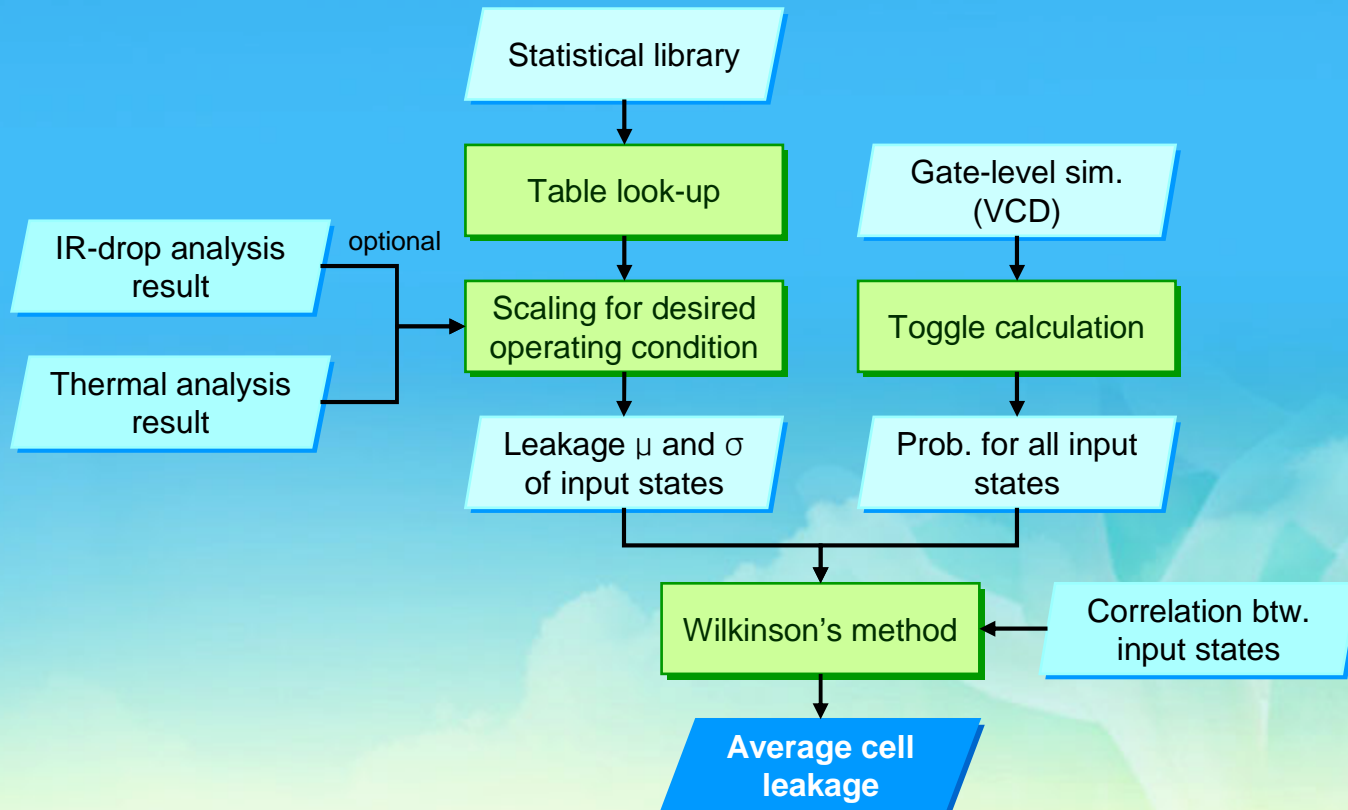
- D2D and WID variations are decomposed analytically
- Spatial correlation inside the chip is characterized based on the TEG result
 - The same test patterns are put inside a chip and the spatial correlation is calculated based on their location
 - Various splits of test patterns (V_{th} , size, and etc.) can be utilized for silicon characterization
- Example of characterized spatial correlation
 - High coefficient of determination : $R^2 \sim 0.8$



Leakage Calculation

◆ Cell-level leakage calculation

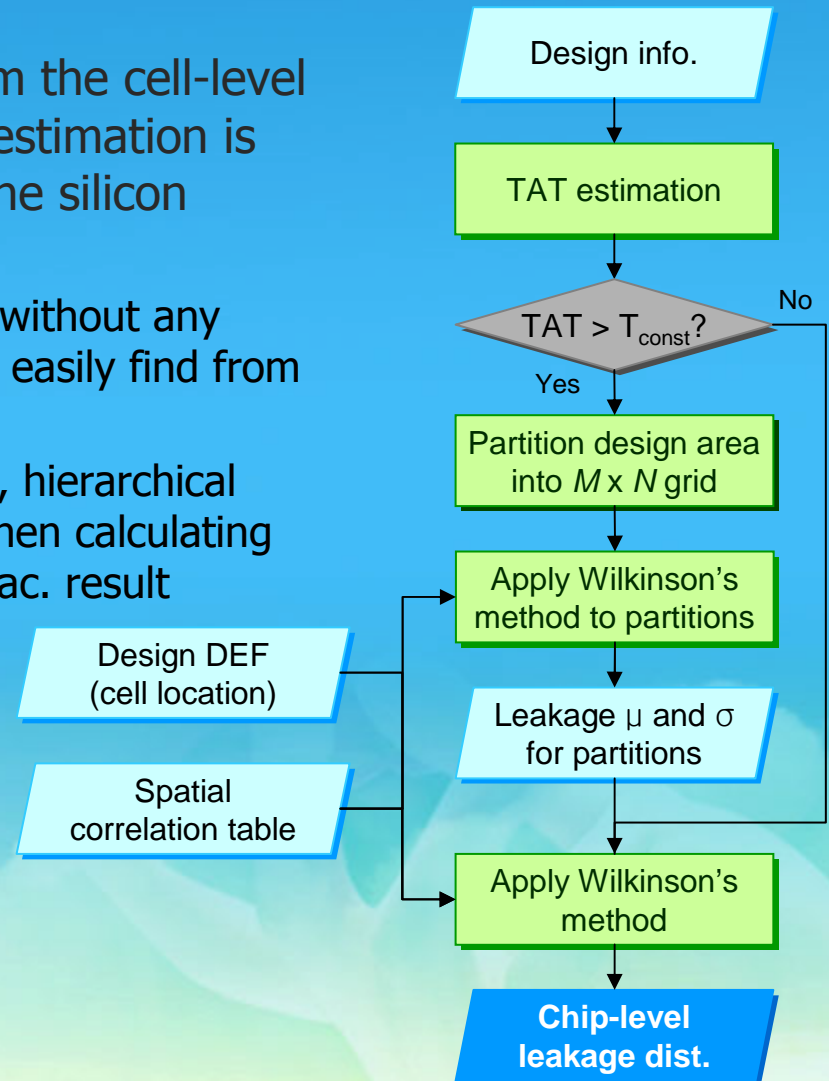
- In cell-level leakage calculation, average leakage calculation as well as scaling for the desired condition is performed
 - Conventional average leakage calculation method is used except that Wilkinson's method is used for the addition of different input state leakages instead



Leakage Calculation

◆ Full-chip leakage calculation

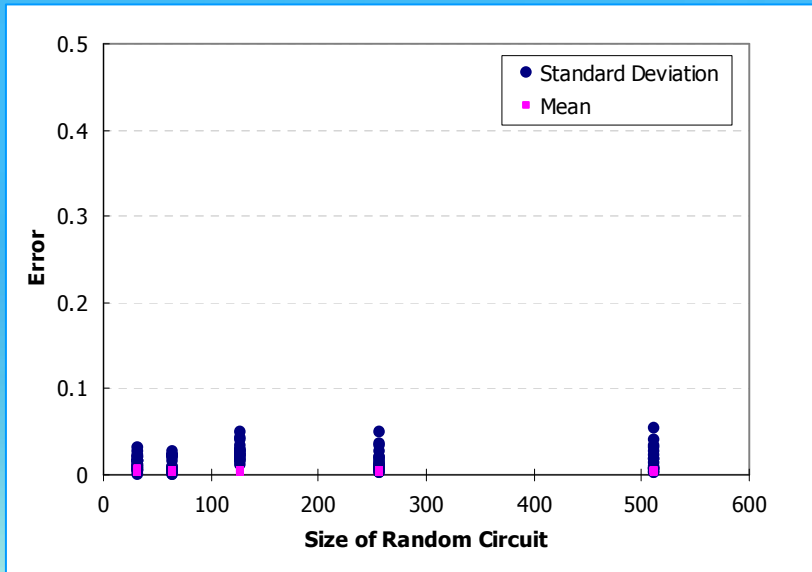
- Using the cell leakage distribution from the cell-level leakage calculation, full-chip leakage estimation is made using Wilkinson's method and the silicon characterization result
 - Wilkinson's method is applied directly without any approximation technique that one can easily find from various literature
 - Instead, for the runtime improvement, hierarchical partitioning of the design is applied when calculating full-chip leakage based on silicon charac. result



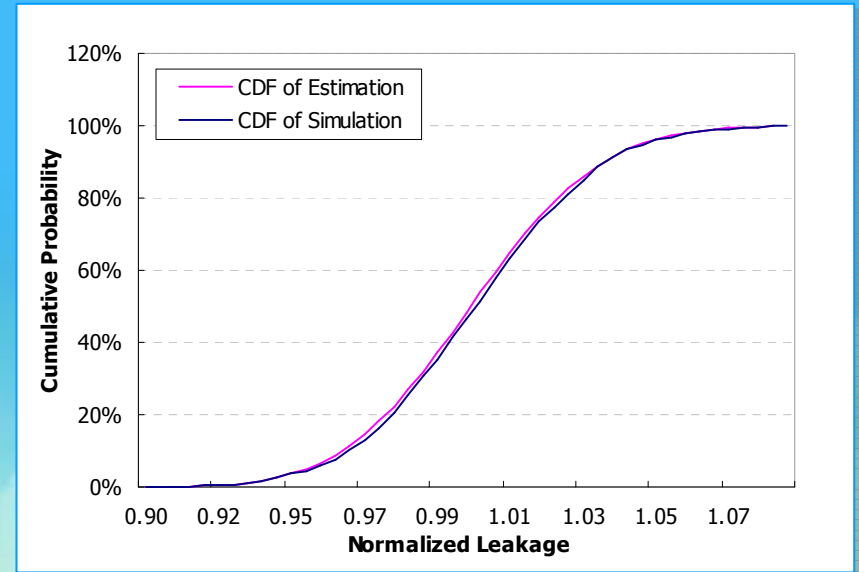
Pilot Test

◆ Comparison with HSPICE Monte-Carlo simulation

- Using 32nm standard cells, random circuit netlist is generated and the comparison between the presented framework and HSPICE Monte-Carlo simulation is made
- The presented framework provided errors under **1%** in the mean comparison and **5%** in the standard deviation comparison



• Error plot of the presented framework



• CDF comparison of 64 instance random circuit

Pilot Test

◆ Analysis result comparison with conventional estimator

- Using 32nm mobile CPU design, the leakage from the presented framework and that from the conventional deterministic leakage estimator are compared
 - As a conventional deterministic leakage estimator, Samsung in-house power estimator, CubicPower is used
- Statistical analysis showed less distribution in the leakage than that of conventional leakage estimator with pessimism introduced for variability consideration

Mobile CPU Design	Conventional Estimator	Proposed Framework	Delta
-3-sigma @ 0.9v, -40c	0.9 uW	1.4 uW	+0.5uW
+3-sigma @ 1.1v, -40c	17 uW	13 uW	-4 uW
+3-sigma @ 1.1v, 125c	7.8 mW	6.3 mW	-1.5mW

Summary

- ◆ A practical framework for statistical leakage estimation is presented
 - The algorithm for the statistical calculation is simplified and does not require any approximation during calculation by using silicon characterization result
 - The scalability to the operating condition is added to meet designer's various request
 - The framework can be easily built by utilizing conventional libraries in the statistical library preparation
 - Especially, it is useful for the large macros which are practically not characterizable for statistical estimation
 - In random circuit experiment, the presented framework showed errors under 5% when compared to HSPICE Monte-Carlo simulation
 - In the comparison with conventional leakage estimator, the presented framework showed less pessimism in the leakage estimation