Received February 7, 2014, accepted February 23, 2014, date of current version March 12, 2014.

Digital Object Identifier 10.1109/ACCESS.2014.2308922

Hybrid Gate-Level Leakage Model for Monte Carlo Analysis on Multiple GPUs

JINWOOK KIM[1], (Student Member, IEEE), AND YOUNG HWAN KIM[1,2], (Member, IEEE)
[1]Pohang University of Science and Technology, Gyeongbuk 790-784, Korea
[2]University of California, Berkeley, CA 94720, USA

Corresponding author: Y. H. Kim (youngk@postech.ac.kr)

This work was supported by SK Hynix Semiconductor Company, Ltd.

ABSTRACT This paper proposes a hybrid gate-level leakage model for the use with the Monte Carlo (MC) analysis approach, which combines a lookup table (LUT) model with a first-order exponential-polynomial model (first-order model, herein). For the process parameters having strong nonlinear relationships with the logarithm of leakage current, the proposed model uses the LUT approach for the sake of modeling accuracy. For the other process parameters, it uses the first-order model for increased efficiency. During the library characterization for each type of logic gates, the proposed approach determines the process parameters for which it will use the LUT model. And, it determines the number of LUT data points, which can maximize analysis efficiency with acceptable accuracy, based on the user-defined threshold. The proposed model was implemented for gate-level MC leakage analysis using three graphic processing units. In experiments, the proposed approach exhibited the average errors of <5% in both mean and standard deviation with reference to SPICE-level MC leakage analysis. In comparison, MC analysis with the first-order model exhibited more than 90% errors. In CPU times, the proposed hybrid approach took only two to five times longer runtimes. In comparison with the full LUT model, the proposed hybrid model was up to one hundred times faster while increasing the average errors by only 3%. Finally, the proposed approach completed a leakage analysis of an OpenSparc T2 core of 4.5 million gates with a runtime of <5 min.

INDEX TERMS Graphic processing unit, leakage current, monte carlo methods, process variation, statistical analysis, statistical leakage estimation.

I. INTRODUCTION

Owing to the rapid growth of mobile application markets, power consumption has become one of the cardinal factors for VLSI chip designs. In particular, leakage power consumption, or static power consumption, accounts for almost one-half of the overall system power consumption. As the leakage power is consumed even while a chip is in an idle state, it directly affects the battery life of mobile devices. Thus, it is very important to reduce the leakage power of mobile VLSI chips, and, therefore, accurate leakage estimation has become one of the most important steps of mobile chip designs.

As the semiconductor process technology scales down, the process uncertainty increases and the variation in process parameters increases. To deal with the variation of process parameters, a worst-case corner analysis [1] has been widely used. This approach can be used successfully when the process variation is not too serious and the die-to-die (D2D) variation is the main contributor [2]. However, the continuous scaling down of semiconductor process technology has worsened the process parameter variation to a serious level, and leakage variation also has become considerably severe in the state-of-the-art process technology. In addition, the within-die (WID) variation has continuously increased and became comparable with the D2D variation in size [2], [3]. Therefore, a worst-case corner analysis of leakage has become overly pessimistic in many designs, resulting in an invalidity of the analysis results [2], [4]–[6], particularly during the gate-level design.

To address the problem of overly pessimistic leakage analysis, a first-order exponential-polynomial leakage current model (the first-order model) [7], which represents the leakage power under process variations, has been proposed for use with gate-level designs. The first-order model represents the logarithmic value of the leakage current as a linear function of the process parameters, such as channel length, oxide thickness, and threshold voltage. In addition, many approaches to a leakage analysis of gate-level VLSI designs, which use the first-order model, have been proposed. These include analytic gate-level statistical leakage estimation (SLE) methods [4], [5], [7]–[16] and Monte-Carlo (MC) simulation-based gate-level leakage estimation methods [17], [18].

The first-order model is very efficient, and has been successfully used to analyze the leakage current of a VLSI design at the gate-level under process variations. However, as the process technology scaled down to below 65 nm [19], the nonlinear relationship between certain process parameters and the logarithmic value of the leakage current became much stronger. Consequently, the accuracy of the first-order model was strongly questioned for use with state-of-the-art technology [5], [6], [19], [20], and it exhibited significant errors compared to the results of circuit level analyses, obtained using the BSIM4 transistor model [19], [21].

To overcome the weak point of the first-order leakage model, which became clear with the deep submicron process technology, a higher-order exponential-polynomial model [6] and a look-up table (LUT)-based gate leakage model [20] have been proposed. These models are intended for MC-based gate-level leakage analysis. The approach in [6] models the logarithm value of the leakage current as a polynomial of gate channel length and threshold voltage; it uses a first-order expression for threshold voltage and a high-order expression for gate channel length. However, unfortunately, [6] did not deal with other important process variations such as oxide thickness [4], [5]. Thus, this model is more accurate than the first-order model when there are only two variation sources, gate channel length and threshold voltage variations. Compared to the higher-order exponential-polynomial model, the LUT-based model [20] represents the nonlinearity more accurately by representing the logarithmic value of the leakage current using a table. However, in this model, the leakage current needs to be estimated by interpolating the values in the LUT during analysis, and the number of computations for the interpolation increases at the rate of 2^n for piecewise linear interpolation, where n is the number of process parameters. This model is therefore computationally too complex to be applied to a large number of process parameters. Because the leakage current model is repeatedly evaluated for each logic gate during leakage current analysis, high computational complexity of this model can be a big burden for today's large VLSI designs, which have hundreds of millions of logic gates. In addition, it is necessary to divide the ranges of the process parameters into many regions to maintain accuracy, which may increase the LUT to a burdensome size.

This paper proposes a hybrid leakage current model, developed for the gate-level MC analysis of VLSI designs. The proposed model combines the LUT-based model [20] with the first-order model, and takes the advantages of both models. For accuracy, the proposed approach treats the process parameters of strong nonlinear relationships with the logarithm of the leakage current as nonlinear process parameters. For these process parameters, it uses an LUT approach. Other process parameters are regarded as linear process parameters, and the proposed approach uses a first-order model for efficiency for them.

The accuracy and the efficiency of the proposed model depend on the number of parameters which are considered nonlinear. In addition, the number of LUT data points used for the nonlinear parameters affects the accuracy and the efficiency of the proposed model. Thus, it is very important to determine the nonlinear parameters appropriately and the number of data points for the LUT of those parameters. When characterizing the leakage of each logic gate type, the proposed approach adaptively selects the process parameters which should be handled as nonlinear and it determines the number of their LUT data points for each input condition, based on the user-defined error threshold, so that it can obtain the maximum efficiency while maintaining acceptable accuracy.

The remainder of this paper is organized as follows. Section II briefly describes the existing leakage current models. Section III describes the proposed leakage current model and its characterization method in detail. Section IV presents the implementation of the MC simulation with the proposed model on a multiple NVIDIA GPU environment using a CUDA programming environment [22]. Section V presents the performance evaluation of the proposed approach in terms of accuracy and efficiency, and a comparison with existing well-known approaches. Finally, Section VI summarizes the paper and provides some concluding remarks.

II. BACKGROUND

A. FIRST-ORDER EXPONENTIAL-POLYNOMIAL LEAKAGE CURRENT MODEL

State-of-the-art SLE methods use the first-order exponential-polynomial model (the first-order model) [7], which is based on the assumption that major leakage mechanisms such as sub-threshold and gate tunneling leakages are exponentially affected by the process parameter variations. The first-order model of the leakage current approximates the polynomial exponents as a first-order linear model, as shown in (1).

$$I_{leak} = \exp\left(a_0 + \sum_{i=1}^{n} a_i X_i + a_{n+1} R_{n+1}\right), \tag{1}$$

where a_0 through a_{n+1} denote the fitting coefficients determined through characterization, X_i indicates the global source of variation that have correlations with other parameters in other leakage terms [5]. X_i includes the D2D variation parameters and WID variation parameters having correlation. R_{n+1} is the sum of the varying process parameters not correlated with the other leakage parameters. The state-of-the-art SLE methods assume that all X_i's and R_{n+1} are all standard normal random variables. Because the exponent in (1) depends on a normal random variable, the existing SLE methods model the variation in leakage current as a lognormal random variable.

B. LUT-BASED EXPONENT MODEL

The authors of [20] proposed an LUT-based model for the exponent in (1) and demonstrated the accuracy of their model using two interpolation methods: piecewise linear (PWL) and cubic spline interpolations. The LUT stores the natural-logarithmic values of the leakage currents for the given sampling points and the values in the LUT are pre-calculated using a characterization process. Intermediate points between the sampling points can be calculated from the LUT using interpolation.

The PWL interpolation calculates the exponent value of an intermediate point on a straight line between both adjacent sampling points of the intermediate point. Cubic spline interpolation uses a special type of third-order piecewise polynomial interpolant called a spline, and it matches the first and second derivatives of two adjacent splines on the intersection point. Fig. 1 shows the first-order model and the PWL.

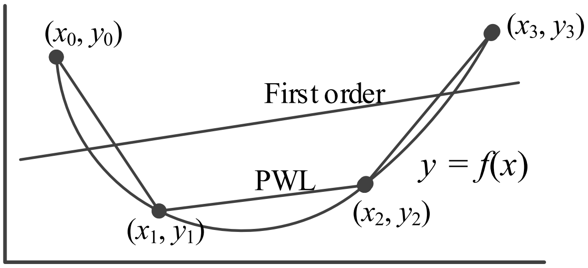


FIGURE 1. Examples of first-order and PWL interpolations.

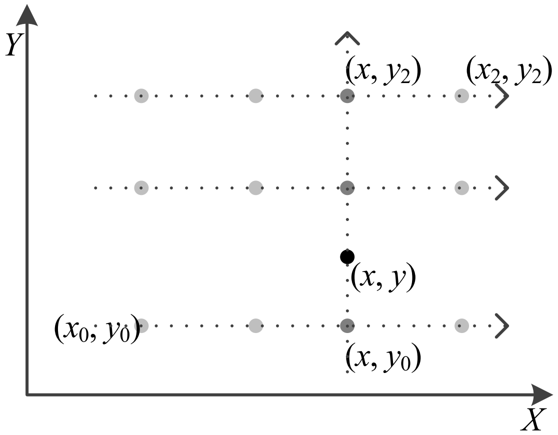


FIGURE 2. Multi-dimensional interpolation.

Multi-dimensional interpolation can be performed through recursive iteration. Fig. 2 shows an example of a two-dimensional interpolation. We assume that we have two process parameters, X and Y, and an LUT with 3×3 sample points (x_i, y_j) for $i, j = 0$, 1, and 2; we want to calculate a value for point (x, y) . First, interpolation with respect to X is performed, and we obtain three interpolated points (x, y_i) for $i = 0$, 1, and 2. Subsequently, interpolation over parameter Y is performed, and we obtain an interpolated value for(x, y) .

Clearly, the LUT-based leakage model is much more accurate than the first-order model even when PWL interpolation is applied. However, the LUT model is incompatible with conventional SLE methods, and only the MC-based leakage analysis can currently handle the LUT model.

III. PROPOSED LEAKAGE MODEL AND ITS CHARACTERIZATION

Although the LUT-based leakage current model [20] is reasonably accurate, its computational complexity can be a significant burden for the MC simulation when a number of varying process parameters are present. In practice, not all process parameters have a nonlinear relationship with the logarithmic value of the leakage current.

In this section, we present a novel leakage current model that combines the LUT-based leakage and first-order models.

A. HYBRID LEAKAGE CURRENT MODEL

The amount of leakage current in a cell is a function of the input state of the cell [23]. The proposed model expresses the leakage current of cell l as (2) when $n+m$ process parameters are considered.

$$I_l = \sum_{\forall state(i)} P_i I_l^i = \sum_{\forall state(i)} P_i \exp\left\{f_l^i\left(X_1, \ldots, X_{n+m}\right)\right\}, \quad (2)$$

where P_i is the probability of input state i of cell l, and I_l^i is the leakage current value of cell l for input state i. Similar to the first-order model, the proposed model is based on the assumption that the major leakage mechanisms are exponentially affected by the varying process parameters [16]. In addition, $f_l^i\left(X_1, \ldots, X_{n+m}\right)$ is modeled instead of I_l^i, which is the logarithmic value of I_l^i.

Let X_k indicate the varying process parameters, which are normalized to a zero mean for $k = 1, \ldots, n+m$. Among them, let $X_1, \ldots, X_n$ be n varying process parameters that have a strong nonlinear relationship with the logarithmic value of the leakage current (nonlinear process parameters), and let $X_{n+1}, \ldots, X_{n+m}$ be m varying process parameters with high linearity (linear process parameters).

The proposed model approximates $f_l^i\left(X_1, \ldots, X_{n+m}\right)$ with an LUT of the first-order models. It combines an n-dimensional LUT for nonlinear process parameters $X_1, \ldots, X_n$, and the first-order model for linear process parameters $X_{n+1}, \ldots, X_{n+m}$ to approximate $f_l^i\left(X_1, \ldots, X_{n+m}\right)$ for an i-th input state of cell l in (2), and uses the PWL interpolation (extrapolation) to obtain the value of the point of interest.

Let $\left\{y_i^j\right\}$ for $j = 1, \ldots, d_i$ be the set of LUT index values for nonlinear process parameter X_i for $i = 1, \ldots, n$, where d_i is the number of data points in an LUT for X_i. For each LUT index $\left(y_1^{j_i}, \ldots, y_n^{j_n}\right)$, the proposed model approximates $f_l^i(X_1, \ldots, X_{n+m})$ using the first-order model of linear process parameters, which is pre-characterized while nonlinear process parameters X_i are set to their index values $y_i^{j_i}$. Therefore, the proposed model contains $\prod_{i=1}^{n} d_i$ first-order models of linear process parameters. For each required data point for interpolation, we can obtain the desired logarithm value for $\left(y_1^{j_1}, \ldots, y_n^{j_n}, x_{n+1}, \ldots, x_{n+m}\right)$ by substituting linear process parameters X_i for $i = n+1, \ldots, n+m$ with linear process parameter values x_i.

TABLE 1. Example look-up table of the proposed model.

X_2 X_1	-0.13	-0.07	0.13
-0.11	$a_0^{-0.11,-0.13}+\sum_{k=3}^{m} a_k^{-0.11,-0.13} X_k$	$a_0^{-0.11,-0.07}+\sum_{k=3}^{m} a_k^{-0.11,-0.07} X_k$	$a_0^{-0.11,0.13}+\sum_{k=3}^{m} a_k^{-0.11,0.13} X_k$
0.02	$a_0^{0.02,-0.13}+\sum_{k=3}^{m} a_k^{0.02,-0.13} X_k$	$a_0^{0.02,-0.07}+\sum_{k=3}^{m} a_k^{0.02,-0.07} X_k$	$a_0^{0.02,0.13}+\sum_{k=3}^{m} a_k^{0.02,0.13} X_k$
0.07	$a_0^{0.07,-0.13}+\sum_{k=3}^{m} a_k^{0.07,-0.13} X_k$	$a_0^{0.07,-0.07}+\sum_{k=3}^{m} a_k^{0.07,-0.07} X_k$	$a_0^{0.07,0.13}+\sum_{k=3}^{m} a_k^{0.07,0.13} X_k$

Table 1 shows an example LUT of the proposed model when n is given as 2, and a 3×3 table is used for X_1 and X_2. As shown in the table, the LUT index values for X_1 are $-0.11, 0.02$, and 0.07, and the index values for X_2 are -0.13, -0.07, and 0.13. The proposed model has nine first-order polynomial equations for $X_3, \ldots, X_{2+m}$.

For example, the $(1, 1)$ value, corresponding to the index value $\left(y_1^1 = -0.11, y_2^1 = -0.13\right)$, of the LUT in Table 1 is obtained by computing the first-order model equation for varying process parameters $X_3, \ldots, X_{2+m}$, which is pre-characterized with $X_1 = -0.11$ and $X_2 = -0.13$. The corresponding first-order model equation is expressed as an equation of the $(1, 1)$ element in Table 1. Here, $a_0^{-0.11,-0.13}$ represents the mean value of $\log\left(I_l^i\right)\big|_{X_1=-0.11,X_2=-0.13}$ for $X_3, \ldots, X_{2+m}$ when X_1 and X_2 are fixed at -0.11 and -0.13, respectively, and $a_k^{-0.11,-0.13}$ indicates the fitting coefficients for X_k for $k = 3, \ldots, 2 + m$ when $X_1 = -0.11$ and $X_2 = -0.13$.

If process parameter values for X_1 and X_2 are given as -0.05 and 0.1,respectively, four LUT values located in $(1, 2)$, $(1, 3)$, $(2, 2)$, and $(2, 3)$ are needed for interpolation and these values can be obtained by evaluating the first-order models in those cells. The desired $\log\left(I_l^i\right)$ value can be obtained by performing two-dimensional PWL interpolation using these four values, as described in Section II-B, which can be easily implemented using a recursive function call.

Because the first-order model for $X_{n+1}, \ldots, X_{n+m}$ (which consists of m+1 terms) is necessary to compute each LUT value, the size of the model is given as (3).

$$size = (m + 1) \times \prod_{i=1}^{n} d_i \tag{3}$$

In general, the cubic spline interpolation discussed in [20] shows more accurate results than the PWL interpolation for the same number of data points. However, if n data points are used in the interpolation, we should compute the derivatives of all n points for cubic spline interpolation. On the other hand, PWL interpolation only requires two data points, which are neighbors of the point of interest. The proposed method uses PWL interpolation to minimize the computational overhead resulting from the interpolation.

A one-dimensional interpolation requires two data points, and therefore, an n-dimensional interpolation requires 2^n data points. Each required data point is obtained by computing the first-order model for the linear parameters. This is clearly a significant burden for an MC simulation of a large n value. In a practical case, however, the number of nonlinear varying process parameters is not sufficiently large to make the MC simulation impractical. Moreover, even though only one or two process parameters, which have high nonlinear relationships with $f_l^i\left(X_1, \ldots, X_{n+m}\right)$ in (2), are considered as interpolating parameters, the proposed model improves the accuracy without a large burden in terms of the computation time.

B. GATE LIBRARY CHARACTERIZATION

Unfortunately, similar to the conventional LUT-based models, it is difficult to estimate the maximum error bound and determine the appropriate number of data points (LUT index values) for each LUT parameter. Moreover, it is difficult to choose the LUT index values that minimize the error and determine which process parameters should be considered for the LUT. In particular, it is nearly impossible to find a global optimum solution for the overall parameter space.

Instead of finding a global optimum solution, the proposed characterization method inevitably finds a local optimum solution using a given error threshold on the sample space. The proposed method estimates the leakage current values while one process parameter varies in the sampling range and the other parameters are fixed at their nominal values (without a process variation for the other process parameters). The error of the first-order model is then estimated, and the LUT index values are determined through optimization. This characterization is performed for each input state of a gate. The detailed procedure is described as follows:

For each process parameter X_i, 1) let S be a set of sampling points s_k for $k = 1, \ldots, l$, where l is the number of sampling points, 2) estimate the reference leakage current values $I\left(s_k\right)$ for each sampling point s_k, 3) find a_0 and a_1 for the first-order model $a_0 + a_1 X_i$ using the sampling data $I\left(s_k\right)$, 4) estimate the maximum absolute relative error as (4), i.e.,

$$\text{Error} = \max_{s_k \in S}\left(abs\left(\frac{I\left(s_k\right) - \exp\left(a_0 + a_1 s_k\right)}{I\left(s_k\right)}\right)\right), \tag{4}$$

and 5) if the error in (4) is greater than the user-defined error threshold, mark X_i as an interpolating process parameter; otherwise, X_i is marked as a linear parameter.

For each interpolating process parameter X_i, the proposed method finds the minimum subset P of S as a set of LUT index values under the constraint that the error is less than the given error threshold using a conventional optimization algorithm such as a genetic optimization and full search algorithm. The optimization problem can be formulated as (5),

$$\begin{aligned} &\text{minimize}\, d_i = \sum_{k=1}^{l} x_k \\ &s.t. \qquad x_k = 0 \text{ or } 1 \text{ for } k = 1, \ldots, l \\ &\qquad\qquad P = \{s_k \text{ such that } x_k = 1\} \\ &\qquad\qquad V = \{\log I\,(s_k) \text{ for } s_k \in P\} \\ &\max_{s_k \in S} \left(abs \left(\frac{I(s_k) - \exp(F(P, V, s_k))}{I(s_k)} \right) \right) \le \text{Error}_{threshold} \end{aligned} \tag{5}$$

where $F\,(P, V, s_k)$ is a function that performing interpolation to find the value of s_k, which is the value of the underlying function $V = f\,(P)$ at the query point s_k. In addition, x_i is a decision variable in which a sampling point s_i is chosen as the LUT index value when $x_i = 1$. Set P is a set of LUT index values, and set V is a set of the corresponding logarithmic values of the reference values. In the above procedure, we simply use the maximum absolute relative error for the error metric. Other error metrics such as the squared sum of errors (SSE) can also be used.

After the optimization, n interpolating parameters, i.e., X_i for $i = 1, \ldots, n$, and the LUT index values for each interpolating parameter X_i, are determined. The corresponding LUT is n-dimensional, and the number of elements in the LUT is given by $\prod_{i=1}^{n} d_i$. For each LUT element, the first-order model is characterized for $X_{n+1}, \ldots, X_{n+m}$.

IV. IMPLEMENTATION OF THE PROPOSED MODEL FOR MC ANALYSIS ON A MULTIPLE GPU ENVIRONMENT USING CUDA

We implemented the proposed hybrid gate leakage model for the MC analysis under a multiple GPU environment using the NVIDIA CUDA platform [22]. The overall flow of the implemented MC analysis (Fig. 3) is similar to that of [18], but we modified the flow of [18] to consider the spatial correlation and utilize multiple GPUs. In Fig. 3, the white boxes represent the host-side processes (processes on a CPU), and the gray boxes represent the GPU-side processes. 1) A grid number is assigned for each cell in the netlist. 2) A principle component analysis (PCA) is performed on the CPU to determine a transformation matrix for transforming independent random values into correlated random values; eigenvalue decomposition is performed on each covariance matrix of the spatially correlated process parameter. 3) D2D random values and 4) spatial correlated random values are generated on a single GPU. 5) The generated D2D and correlated random values are copied to all GPUs. 5) MC simulation is performed on the GPUs. 6) One GPU accumulates the leakage current values of all partitions for each MC sample.

In this procedure, to utilize multiple GPUs effectively, the netlist is partitioned with grid units. Fig. 4 shows an example of partitioning and multi-GPU assignment. The grids are equally divided with respect to the number of available GPUs. Fig. 4 shows that six grids exist and three GPUs are available. The six grids are divided uniformly into three partitions for the three GPUs. The leakage current in each partition is computed by the thread on each GPU, and therefore, three corresponding threads are executed on the three GPUs to compute the leakage current values of three partitions. Then, one GPU accumulates the leakage current values of three partitions to obtain the leakage current of a circuit.

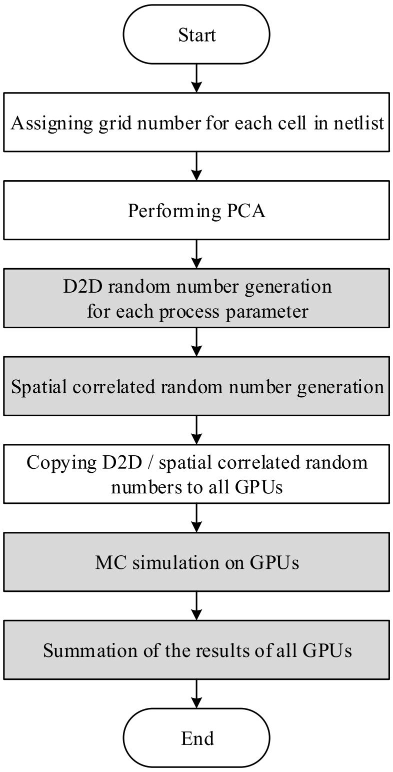

Start
Assigning grid number for each cell in netlist
Performing PCA
D2D random number generation for each process parameter
Spatial correlated random number generation
Copying D2D / spatial correlated random numbers to all GPUs
MC simulation on GPUs
Summation of the results of all GPUs
End

FIGURE 3. Overall flow of the proposed method (blank box: process on a CPU, grayed box: process on a GPU).

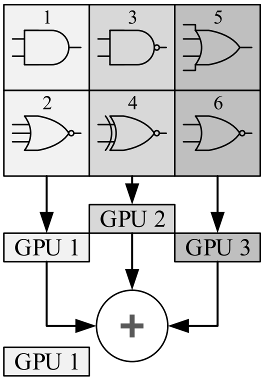


FIGURE 4. Partitioning and multi-GPUs.

Fig. 5 shows the program flow of each thread on a GPU for the implemented MC analysis. As mentioned previously, the netlist is divided into L partitions for the number of

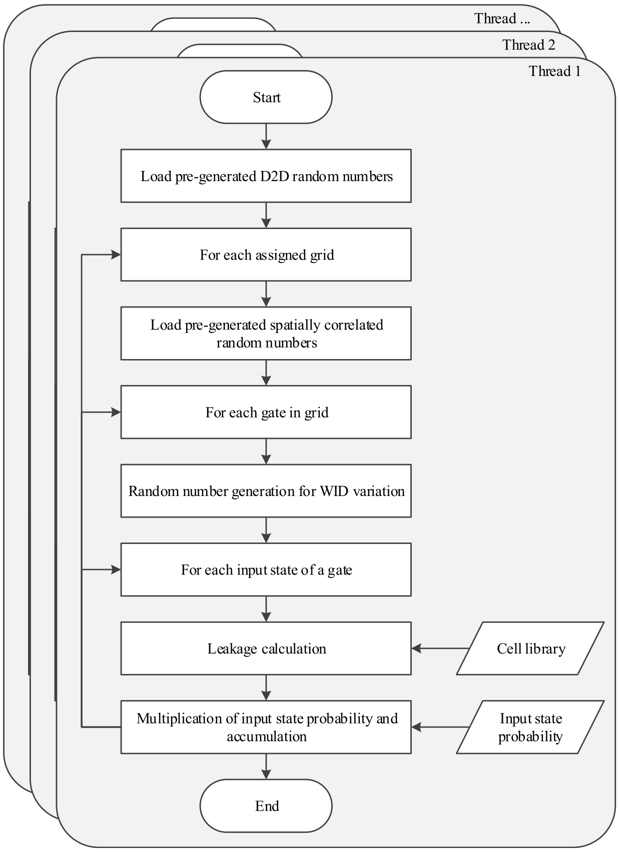

Thread ...
Thread 2
Thread 1
Start
Load pre-generated D2D random numbers
For each assigned grid
Load pre-generated spatially correlated random numbers
For each gate in grid
Random number generation for WID variation
For each input state of a gate
Leakage calculation
Cell library
Multiplication of input state probability and accumulation
Input state probability
End

FIGURE 5. **MC simulation flow on a GPU.**

available GPUs. The MC simulation for the ith MC sample is conducted by L threads on L GPUs whose thread indexes are i. Each thread on a GPU is executed for an assigned partition of the netlist, as shown in Figs. 4 and 5.

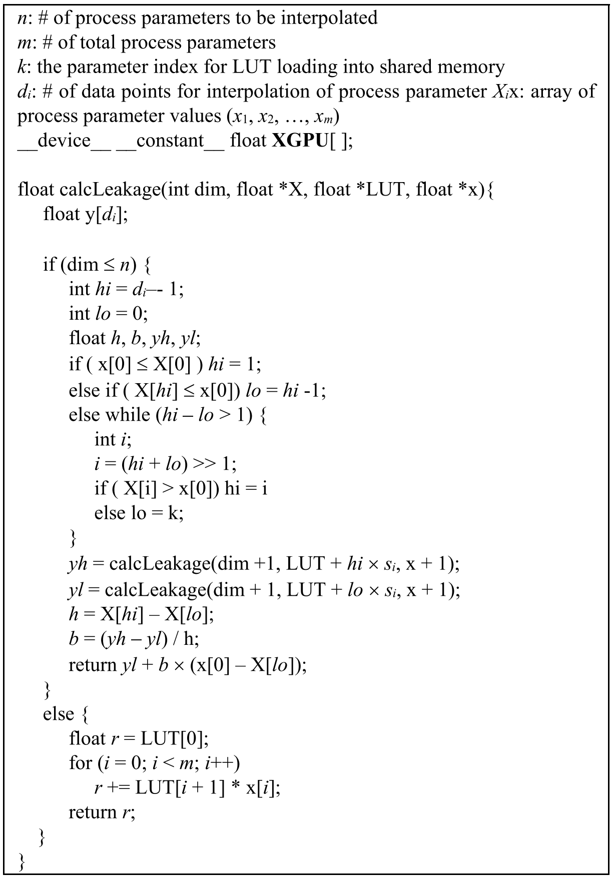
A. LUT DATA STRUCTURE

The LUT index values are stored in a variable **XGPU** in constant memory to reduce the access latency of the LUT index values. Because only a one-dimensional array is allowed in the constant memory of the CUDA, **XGPU** is declared as a one-dimensional array. Let c be the number of characterized cells and v_i be the number of input states of cell i. Then, the length of the **XGPU** is given as $M \times \sum_{i=1}^{c} v_i$, where M is the maximum value of $\sum_{i=1}^{n} d_i$ for all input states and characterized cells. Therefore, the LUT index values of process parameter X_i for the k-th input state of cell j are located in $X_{i-1}\ [d_{i-1}]$ in a zero-based array ordering, where X_i is a pointer for the start address of the LUT index values of process parameter X_i for the k-th input state of cell j, which is given as

$$\mathbf{X}_i = \&\mathbf{XGPU}\left[M\sum_{l=1}^{j-1} v_l + M(k-1) + \sum_{l=1}^{i-1} d_l\right], \quad (6)$$

where the "&" symbol represents the address of the operand in C language. Here, $M\sum_{i=1}^{j-1} v_i$ for each cell is pre-calculated and stored in another array to easily point the proper LUT index values.

The LUT values, which are first-order polynomials for $X_{n+1}, \ldots, X_{n+m}$, are stored in a three-dimensional array. The first index of the LUT signifies a cell, and the second index signifies the input state of a cell. Thus, LUT[i][j] is a one-dimensional array that stores n-dimensional LUT elements for the input state j of cell I, and each element consists of $(m+ 1)$ floating point values for the corresponding first-order model equation. The LUT value of LUT index $(\tilde{x}_1, \ldots, \tilde{x}_n)$ for input state j of cell i is stored in LUT[i][j][$x_1 \times s_1 + \cdots + x_n \times s_n$], where $s_i = s_{i+1} \times d_{i+1}$ and $s_n = m + 1$.



```
n: # of process parameters to be interpolated
m: # of total process parameters
k: the parameter index for LUT loading into shared memory
di: # of data points for interpolation of process parameter Xix: array of
process parameter values (x1, x2, …, xm)
__device__ __constant__ float XGPU[ ];

float calcLeakage(int dim, float *X, float *LUT, float *x){
    float y[di];

    if (dim ≤ n) {
        int hi = di–- 1;
        int lo = 0;
        float h, b, yh, yl;
        if ( x[0] ≤ X[0] ) hi = 1;
        else if ( X[hi] ≤ x[0]) lo = hi -1;
        else while (hi – lo > 1) {
            int i;
            i = (hi + lo) >> 1;
            if ( X[i] > x[0]) hi = i
            else lo = k;
        }
        yh = calcLeakage(dim +1, LUT + hi × si, x + 1);
        yl = calcLeakage(dim + 1, LUT + lo × si, x + 1);
        h = X[hi] – X[lo];
        b = (yh – yl) / h;
        return yl + b × (x[0] – X[lo]);
    }
    else {
        float r = LUT[0];
        for (i = 0; i < m; i++)
            r += LUT[i + 1] * x[i];
        return r;
    }
}
```

FIGURE 6. **CUDA implementation of leakage calculation on LUT.**

B. CELL LEAKAGE CALCULATION

Fig. 6 shows the CUDA implementation of a cell leakage calculation. For process parameter X_i, where $i \leq n$, the proposed method finds lo and hi such that X[lo] and X[hi] are the nearest adjacent points to the corresponding random value x[0]. By utilizing a binary search procedure, $\log_2 d_i$ operations are required to find the two adjacent points. The proposed method then computes the exponent values corresponding to X[lo] and X[hi] by calling itself recursively, and returns the interpolated value. If $i > n$, the proposed method computes the first-order model and returns it.

V. EXPERIMENTAL RESULTS

A. EXPERIMENTAL ENVIRONMENT

In the experiments, we evaluated the performance of the proposed approach in terms of its efficiency and accuracy. We used ISCAS-85 circuits and an OpenSparc T2 processor [24] as the benchmark circuits. A Synopsys Design Compiler and Astro were used for synthesis and placement, respectively. We used two transistor models: an industrial 32-nm transistor model and the Predictive Technology Model (PTM) [25] with a high-performance 22-nm process based on the BSIM4 model.

The process parameter variations considered for the PTM were the gate channel length, gate oxide thickness, and both nMOS/pMOS doping concentration-dependent threshold voltage variations. Seven process parameter variations, which include the gate channel length and the gate oxide thickness variations, were considered for the 32-nm industrial transistor model. We assumed that these process variations were normally distributed, and that the D2D and WID variations in these process variations had the same parts. In addition, the three-sigma (3σ) values of the D2D and WID variations for each process parameter were determined to be 10% of their mean value (i.e., $3\sigma/\mu = 0.1$). Because the D2D and WID variations are independent, the 3σ values of the total process variations were approximately 14.14% of their mean values.

We employed an analytic SLE and conventional MC-based leakage estimation methods as the benchmark methods. More specifically, we used a Wilkinson's method (WM)-based estimation (which uses Wilkinson's method to sum all of the leakage components modeled using the first-order model) [7], Chang's hybrid method (HC) [4], and VCA [5] as benchmark analytic SLE methods. The GPU-based MC method using the first-order model (F-MC) [18] was used as the benchmark MC-based leakage estimation method. Although the F-MC has been proposed without considering the spatial correlation and multiple GPUs, we applied the same implementation scheme of the spatial correlation and multiple GPUs, described in Section IV, to the F-MC for experimental purposes. In addition, although the authors of [20] did not present a GPU-based MC method using their model, we implemented one in the same MC implementation flow for the proposed model, and used it as a benchmark method (L-MC). Higher-order exponential-polynomial model [6] was not employed because it cannot handle process parameters considered in experiments except for both gate channel length and threshold voltage variations.

The gate channel length was considered to be a spatially correlated parameter, and to consider the spatial correlation, we used the grid model in which the area of one region was 10 μm × 10 μm, which was adjusted from 40 μm × 40 μm in the 90-nm process [5] in consideration of the shrinkage achieved through advanced process technology. The amount of spatial correlation between regions was assumed to linearly decrease as the distance between the regions increased in generating the spatial-correlation matrix [26]. The spatial-correlation matrix was adjusted to be a positive semi-definite matrix using the method described in [27].

The proposed method (MC analysis using the proposed hybrid gate leakage model, H-MC), L-MC, and F-MC were implemented using an Nvidia CUDA programming environment with CUBLAS [28], which is a CUDA version of BLAS [29]. HC was implemented on Mathworks MATLAB [30], and both WM and VCA were implemented using the C programming language. In the experiments, we used a Linux machine consisting of an Intel Xeon E5-2690 octa-core CPU with a 2.9-GHz clock frequency, 64 GB of memory, and three NVIDIA GeForce GTX 680 graphics cards [31].

In the characterization, the leakage current values were sampled for each process parameter in the range of -19% to 19% of their mean value in 1% increments using Synopsys HSPICE [32]. As mentioned before, the 3σ values of the total amount of process variations were set at 14.14%. We estimated the error to be in the plus-minus four-sigma ($\pm 4\sigma$) range. As a result, we obtained 39 sample points for each process parameter. Using these 39 samples, the interpolating parameters and the LUT index values were determined. A Mathworks MATLAB [30] optimization toolbox was used to determine the interpolating parameters, the LUT index values, and the fitting coefficients of the first-order model. Five sampling points, namely, -4σ, -2σ, 0, 2σ, and 4σ were used to compute the first-order model of the linear process parameters of each LUT element.

For the LUT model [20], the number of LUT index values of each process parameter was set to the maximum number of LUT index values of the characterized cell leakage library for the proposed hybrid gate leakage model (thirteen points for PTM, and eight points for the industrial TR model), and the LUT index values were determined to minimize the errors (4).

B. ACCURACY EVALUATION FOR THE LEAKAGE MODEL

In this experiment, we used ten ISCAS-85 benchmark circuits. For each circuit, we generated 200 random MC samples. The random values for the D2D variations in the process parameters in each sample were randomly generated as normal distributions with 10% of the $3\sigma/\mu$ variation, as mentioned earlier. The random values of the WID variations in the process parameters of each cell in each MC sample were also randomly generated with the same amount of variations as the D2D variations.

The leakage current of each MC sample was evaluated using the proposed hybrid gate leakage current model, the first-order model, and a LUT-based model [20]. We used the leakage current value of each MC sample obtained using Synopsys HSPICE as the reference value and estimated the absolute relative errors. In the ''Leakage calculation'' step shown in Fig. 5, Synopsys HSPICE was used to compute the leakage current value of the current input state of the current gate for the given process parameter values.

TABLE 2. Accuracy evaluation results of the proposed leakage current model for 200 random samples for each ISCAS-85 benchmark circuit. (R^2: the minimum value of the coefficient of determination among 10 circuits.

		PTM 22nm model			Industrial 32nm model		
		Relative error		R^2	Relative error		R^2
		Max.	Avg.		Max.	Avg.	
First-order model		98.5%	82.1%	-0.277	75.6%	21.1%	0.477
LUT model		5.1%	1.4%	0.999	8.8%	2.5%	0.998
Hybrid model	3%	19.1%	3.1%	0.997	11.2%	3.6%	0.996
	5%	26.8%	4.8%	0.993	21.3%	6.1%	0.987
	10%	38.2%	7.5%	0.981	44.0%	14.2%	0.926
	20%	96.7%	17.3%	0.974	61.4%	22.1%	0.844

Table 2 shows the results of this experiment. The maximum relative error represents the maximum value among the absolute values of relative errors of all MC samples for all ISCAS-85 benchmark circuits. The average relative error represents the average value of the absolute values of relative errors of all MC samples for all ISCAS-85 benchmark circuits. The coefficient of determination R^2 is the minimum value of R^2 for all ISCAS-85 benchmark circuits. The first-order model showed very inaccurate results, which were 98.5% and 75.6% of the maximum relative errors for the 22-nm PTM and the industrial 32-nm transistor model, respectively. In particular, in terms of R^2, which provides a measure of how well observed the outcomes were replicated by the model, the first-order model scored -0.277 and 0.477 for the PTM and industrial transistor model, respectively. On the other hand, the proposed model showed high R^2 scores of greater than 0.97 for the PTM, and 0.84 for the industrial model. Although the maximum error of the proposed model for the 20% error threshold was comparable with that of the first-order model, the proposed model showed a much lower average error and a much higher R^2 score for the PTM, and a comparable average error and much higher R^2 score for the industrial model. Although the LUT model showed more accurate results than the proposed model, the proposed model also showed quite accurate results with average errors of less than 4% when the error threshold was set to 3%, which are still within an acceptable range. Although the error threshold was unable to limit the errors, the average errors were not far from the error threshold, and the accuracy of the proposed model increased as the error threshold decreased. The errors were somewhat controllable even not precise.

Fig. 7 shows the leakage current estimation results for 200 random samples of the c6288 ISCAS 85 benchmark circuit when a 22-nm PTM transistor model was used. The x-axis represents the reference value, and the y-axis represents the corresponding estimated value. The black solid line represents the reference values obtained using Synopsys HSPICE. As this figure shows, the results of the first-order model remained very far from the reference values. On the other hand, the results of the proposed and LUT models were relatively close to the reference values. In addition, the first-order model showed a tendency toward underestimation. In the characterization, the relative error was the target to be optimally minimized. Because an underestimation always yields a maximum error of less than 100%, we obtained the fitting coefficients of the first-order model to underestimate the values through optimization.

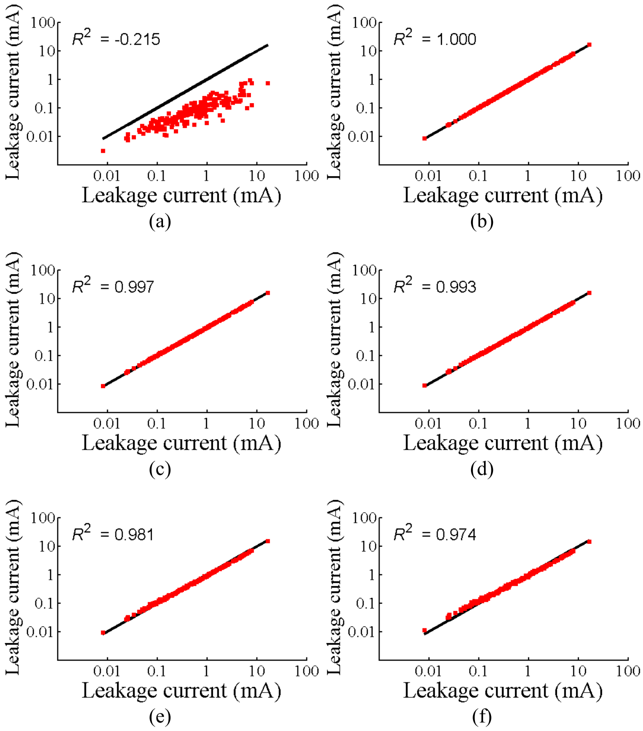


FIGURE 7. Leakage current estimation results for 200 random samples of c6288 and PTM 22nm TR model. (a) First-order model. (b) LUT model. (c) Hybrid model: 3%. (d) Hybrid model: 5%. (e) Hybrid model: 10%. (f) Hybrid model: 20%.

C. PERFORMANCE COMPARISON OF THE LEAKAGE ESTIMATION METHODS

In this experiment, we evaluated the leakage current distributions of the benchmark circuits using the proposed and benchmark methods. To obtain the reference leakage current distribution, we used Synopsys HSPICE. Similar to the experiment presented in Section V-*B*, Synopsys HSPICE was called in the leakage calculation step shown in Fig. 5, and was used to compute the leakage current value of a cell (SPICE-MC).

As mentioned previously, the 3σ values of the D2D and WID process parameters were set to 10% of their mean values (i.e., $3\sigma/\mu = 10\%$), and therefore, the 3σ values of the total process variations were approximately 14.14% of their mean values. All MC-based methods, including the MC simulation using Synopsys HSPICE, were performed on 16,384 samples for each ISCAS-85 benchmark circuit. This sample number corresponds to a 95% confidence level with ±3% and ±2.5% sampling errors for the mean values of the PTM and industrial

TABLE 3. Estimation results for leakage current distribution for 10 ISCAS-85 benchmark circuits. The average values and the maximum values of absolute relative errors for ISCAS-85 benchmark circuits are represented in %. (HC: H. Chang's method [4] , WM: Wilkinson's method [7], VCA: Virtual cell approximation method [5], F-MC: Monte-carlo simulation using the first-order model [18], L-MC: Monte-carlo simulation using LUT-based model [20], and H-MC: Monte-carlo simulation using the proposed hybrid gate leakage model.

		PTM 22nm TR model										Industrial 32nm TR model									
		Average errors (%)					Maximum errors (%)					Average errors (%)					Maximum errors (%)				
		μ	σ	Percentile points			μ	σ	Percentile points			μ	σ	Percentile points			μ	σ	Percentile points		
				75	95	99			75	95	99			75	95	99			75	95	99
HC		90.2	93.0	89.2	92.4	93.2	91.5	93.9	90.9	93.3	94.2	23.9	53.3	18.6	36.3	50.4	33.3	64.7	39.5	49.5	62.5
WM		90.9	93.5	89.9	92.9	93.7	91.5	94.0	90.9	93.3	94.1	26.1	57.9	19.1	41.7	55.4	33.2	65.7	26.6	49.5	62.5
VCA		90.9	93.5	89.9	92.9	93.7	91.5	94.0	90.9	93.3	94.1	26.1	56.6	19.2	40.9	54.2	33.2	65.2	26.6	49.5	62.3
F-MC		90.9	93.4	90.0	92.9	93.5	91.7	94.1	90.8	93.5	94.0	25.9	57.9	19.9	42.4	55.0	33.4	65.6	26.9	50.0	62.8
L-MC		1.2	1.6	2.2	5.1	2.8	2.8	5.3	3.8	12.6	1.4	1.6	1.8	1.5	3.2	2.7	2.6	4.9	3.6	6.5	1.2
H-MC	3%	2.3	2.0	3.5	2.9	5.2	4.1	3.8	5.8	7.9	19.1	2.7	2.0	3.1	3.1	2.0	3.9	4.3	5.4	5.6	4.6
	5%	4.1	4.2	5.3	7.1	6.1	7.2	6.7	8.6	12.0	19.2	5.1	3.5	6.1	4.1	3.4	7.6	6.5	8.8	6.8	7.6
	10%	5.5	6.0	5.7	8.4	8.2	11.5	10.4	13.0	16.7	21.7	11.1	5.8	13.3	9.4	6.6	19.0	9.8	23.0	18.4	13.9
	20%	5.2	10.7	4.1	11.0	12.2	11.4	16.6	11.8	17.8	23.8	16.6	7.7	20.4	12.7	8.0	27.2	18.4	32.4	22.4	20.0

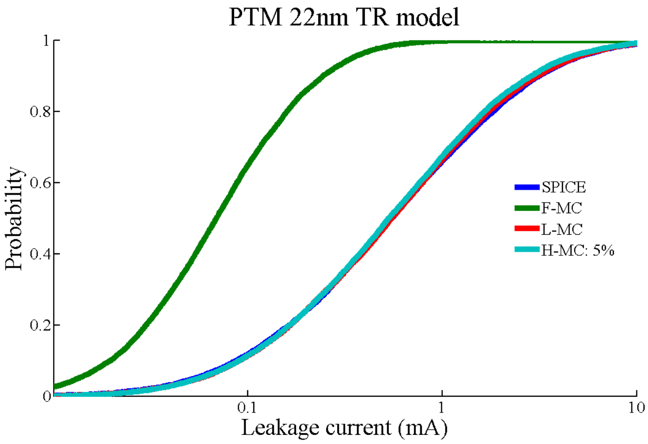


FIGURE 8. CDF comparison between the results of the proposed method, the first-order MC, MC using LUT model and SPICE MC simulation for c6288 benchmark circuit. (PTM 22nm TR model and 5% error threshold for the proposed method.)

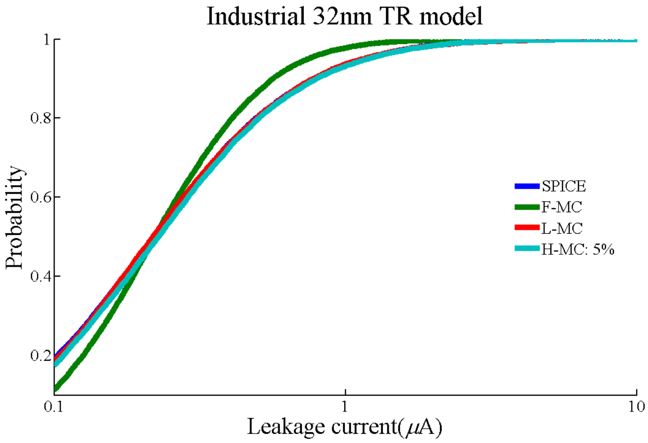


FIGURE 9. CDF comparison between the results of the proposed method, the first-order MC, MC using LUT model and SPICE MC simulation for c6288 benchmark circuit. (Industrial 32nm TR model and 5% error threshold for the proposed method.)

transistor model, respectively, and ±2.5% sampling errors for the standard deviation values of both the PTM and industrial transistor model [33].

Table 3 presents the average and maximum values of the absolute relative errors of the benchmark circuits. The percentile points of the analytic SLE methods were calculated from the inverse lognormal cumulative distribution function (CDF) using the estimated mean and standard deviation values. These percentile point errors were presented to evaluate the accuracy of the benchmark methods at the tail of the distribution.

The proposed method (MC analysis with the proposed hybrid gate leakage model, H-MC) clearly showed improved accuracy for all statistics (mean, standard deviation, and percentile points) of both the PTM and industrial transistor model as compared with all analytic SLE methods and F-MC. For the PTM, the benchmark methods based on the first-order model showed inaccurate results of approximately 90% on average and maximum errors for both the mean and standard deviation values. On the other hand, H-MC showed relatively accurate results with approximately 7% maximum errors for both the mean and standard deviation values when the error threshold was set to 5%. Although the L-MC showed the best accuracy among all methods, the differences between L-MC and the proposed H-MC for the 5% error threshold were only less than 3% in both mean and standard deviation values on average.

Similar results were obtained for the industrial 32-nm transistor model. The first-order model based methods (analytic SLEs and F-MC) exhibited very inaccurate results with average errors of greater than 23% for the mean value and 50% for the standard deviation value. Among all methods considered, L-MC showed the best accuracy, as it did for the PTM, but there was only a small difference of less than 4% on average between the results of L-MC and H-MC for an error threshold of 5% in terms of both the mean and standard deviation values.

Figs. 8 and 9 show the CDFs for the c6288 ISCAS 85 benchmark circuit of the PTM and industrial transistor model, respectively. Similar to the results presented in Table 3,

TABLE 4. Runtime comparison of the MC with the proposed model and benchmark methods. (WM: Wilkinson's method [7], VCA: virtual cell approximation method [5], F-MC: the first-order exponential-polynomial based MC simulation [18], L-MC: MC simulation using LUT-based model [20], and H-MC: MC analysis with the proposed model.

	# of cells	PTM 22nm TR model									Industrial 32nm TR model								
		WM	VCA	F-MC	L-MC	H-MC (ms)				SPICE	WM	VCA	F-MC	L-MC	H-MC (ms)				SPICE
		(ms)	(ms)	(ms)	(ms)	3%	5%	10%	20%	(hr)	(ms)	(ms)	(ms)	(ms)	3%	5%	10%	20%	(hr)
c432	141	260	1.4	2.5	16.5	10.0	8.7	6.4	5.7	1.49	236	2.0	3.8	873.7	17.5	7.5	5.1	5.0	0.97
c499	184	260	1.5	2.6	16.4	11.7	10.0	8.0	5.5	1.86	276	2.0	4.0	836.7	39.1	12.9	6.3	6.0	1.12
c880	250	296	2.2	3.3	25.3	15.4	13.1	9.8	8.1	1.9	296	3.1	5.4	1,457	26.3	10.9	7.4	7.3	1.44
c1355	232	268	1.8	2.9	19.5	13.3	11.3	8.9	6.4	2.12	280	2.4	4.6	1,024	41.2	14.1	7.1	6.8	1.83
c1908	224	276	1.9	3.0	21.1	13.7	11.7	9.1	6.9	2.04	288	2.6	4.8	1,152	36.2	13.1	7.2	6.9	1.91
c2670	402	448	3.8	4.7	43.1	26.5	22.7	16.6	13.3	3.98	580	5.4	8.4	2,551	50.5	20.2	12.3	12.0	4.37
c3540	786	1,030	7.2	7.8	80.0	46.0	39.8	28.3	24.1	8.41	1,460	10.1	14.7	4,626	92.7	38.0	22.1	21.5	7.71
c5315	1,234	1,750	9.8	10.7	109.5	62.3	53.4	38.8	32.2	11.85	2,550	13.9	20.3	6,212	129.4	54.0	30.7	29.9	10.67
c6288	2,426	5,600	18.7	19.1	208.3	114.0	94.0	70.2	60.7	12.71	8,510	26.4	37.5	13,643	95.1	60.5	51.1	50.3	16.43
c7552	1,308	1,690	10.1	10.8	108.0	63.5	53.4	39.8	31.5	9.82	2,550	13.7	20.5	5,942	130.9	54.1	31.0	30.0	7.82

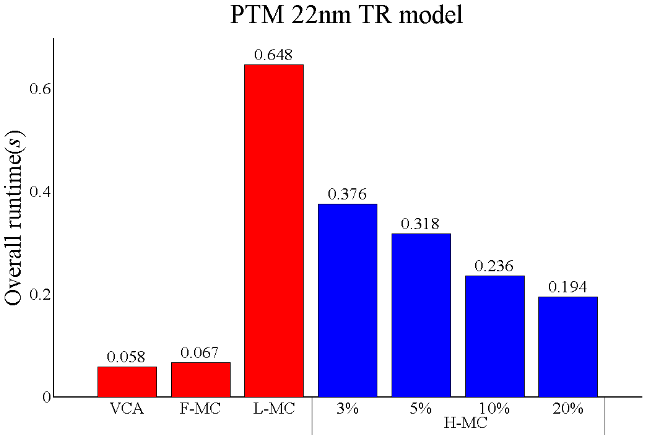


FIGURE 10. Overall runtimes to analyze ten ISCAS-85 benchmark circuits for PTM 22nm TR model.

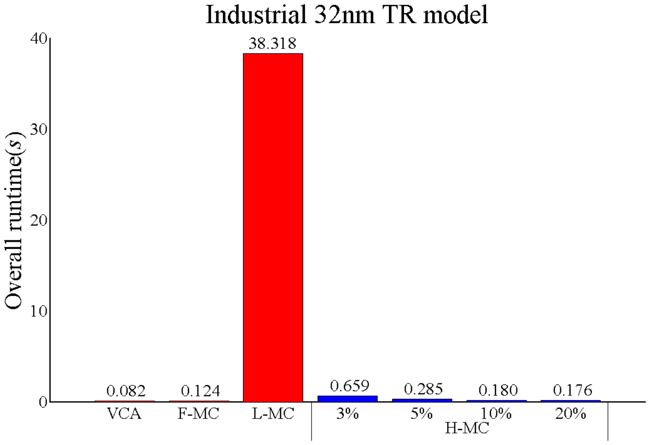


FIGURE 11. Overall runtimes to analyze ten ISCAS-85 benchmark circuits for industrial 32nm TR model.

we can easily ascertain that H-MC is more accurate than the first-order model based benchmark methods.

Table 4 and Figs. 10 and 11 compare the runtimes of the proposed and benchmark methods when one GPU was used to perform the MC methods. Figs. 10 and 11 show the overall runtime used to analyze ten ISCAS-85 benchmark circuits for the PTM and industrial TR models, respectively, and Table 4 details the results of the experiment. Note that the HC method was implemented using Mathworks MATLAB [30], and thus a direct comparison between HC and the other methods is unfair; we therefore omitted the HC runtime results. In [5], the runtimes of HC, WM, and VCA were compared, and the runtime of HC was shown to be comparable with that of VCA.

As shown in the table, VCA showed the fastest results. VCA was at least 2.5 times faster than the proposed H-MC. F-MC ranked second. When compared with F-MC, the proposed method (H-MC) had at least a 50% longer runtime. Although H-MC was much slower than both VCA and F-MC, it showed much more accurate results. In addition, by applying multi-processing for multiple GPUs, the proposed method can improve the computation speed. When compared with the L-MC, the proposed H-MC reduced the runtimes by 40–70% for the PTM, and over 90% for the industrial model. The computational complexity of L-MC increases at the rate of 2^n, where n is the number of process parameters. Therefore, the efficiency improvement of H-MC over L-MC highly depends on the number of considered process parameters.

Table 5 shows the comparison results of the runtimes of the proposed method and the benchmark methods for a combinational logic of the OpenSparc T2 core (4.5 million gates). As shown in Table 5, VCA was still the fastest among all methods tested. However, the runtime of the proposed H-MC was significantly reduced as the number of GPUs increased. When three GPUs were used for the computation, the runtime was reduced to one-third of the runtime of a single GPU.

Fig. 12 shows the runtimes with respect to the number of GPUs used. In this experiment, we estimated the runtimes for the parallelizing overhead and the MC process for four to eight GPUs through a simulation. The overhead includes the data loading and a summation of the results from all GPUs. Fig. 12 shows that the parallelized overhead increases as the

TABLE 5. Runtime comparison of the proposed method and benchmark methods for combinational logic parts of OpenSparcT2 core (4.5 million gates and runtime for VCA: 40s for PTM and 91s for industrial model.

		PTM 22nm TR model (s)			Industrial 32nm TR model (s)		
Number of GPUs		1	2	3	1	2	3
F-MC		125	64	42	222	112	75
L-MC		1,709	852	567	37,453	19,022	13,581
Proposed	3%	983	484	322	728	367	242
	5%	609	306	204	690	348	235
	10%	423	212	143	355	179	120
	20%	346	174	121	332	166	112

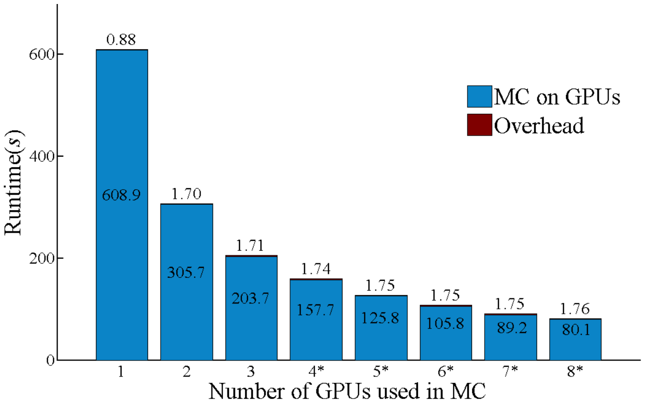


FIGURE 12. Runtime for OpenSparc T2 core of the proposed method for 5% error threshold for PTM 22 nm TR model. (Results for 4 8 GPUs are estimated values).

number of GPUs used increases. However, the runtimes of the overhead were small enough to be neglected, as compared with those of the MC simulation, because the data size to be transferred to each GPU, given as $LM \times (n+1) \times$ the unit size of the floating point variable for L GPUs, M threads, and n process parameters, is negligible compared with the 4 GB/sec bandwidth (8-lane PCI-express 3.0 bus) between the DRAM for a GPU and DRAM for a CPU. In addition, the amount of computations for accumulation of MC results for all partitions is also negligible compared with the amount of computations for MC simulation. Thus, the runtime of the H-MC decreased in accordance with an increase in the number of GPUs used.

When a single GPU was used, the runtime of H-MC for a 5% error threshold was only approximately 10 min for 4.5 million gates; if more GPUs are available, the runtime of H-MC will be further decreased. We therefore claim that the computational overhead of MC using the proposed hybrid gate leakage current model is not a major problem in practical applications.

VI. CONCLUSION

This paper proposed an accurate gate leakage current model that combines an LUT and the first-order exponential-polynomial model, along with its characterization method. The proposed hybrid gate leakage model uses an LUT for varying process parameters having strong nonlinear relationship with the logarithm of leakage current and uses the first-order model for other varying process parameters as each table element in LUT. By combining an LUT and the first-order exponential-polynomial model, the proposed model is more accurate than the first-order model and more efficient than the LUT-based model.

In the accuracy evaluation of the proposed hybrid gate leakage current model, the proposed model obtained high R^2 scores of close to 1. Although the error threshold could not bound the maximum and average relative errors, the average errors were close to the error threshold, and the proposed model showed comparable accuracy with that of the LUT model.

The proposed hybrid gate leakage model was implemented for an MC analysis under a multiple GPU environment. In the MC simulation, the proposed method (MC with the proposed model) was slower than VCA, which is the fastest analytic SLE method among the benchmark methods used. However, the first-order model-based benchmark methods showed large relative errors of approximately 90% for the 22-nm transistor PTM in all statistics, and 50% for the industrial 32-nm transistor model in standard deviation value and at the tail of the distribution. On the other hand, the proposed method showed results that are more efficient with comparable accuracy to that of the LUT-based MC method.

Although the proposed method was slower than VCA, the runtime of the proposed method can be reduced by utilizing multiple GPUs. When only one GPU is used, the runtime is less than 10 min for a commercial design with 4.5 million gates, and if more GPUs are available, this runtime can be reduced further. The computational complexity of the proposed method therefore poses no problem.

In conclusion, we expect that an MC analysis using the proposed hybrid gate leakage model can provide accurate leakage analysis results within a practical runtime.

REFERENCES

[1] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Reading, MA, USA: Addison-Wesley, 2010, p. 864.

[2] S. Nassif, "Design for variability in DSM technologies [deep submicron technologies]," in *Proc. IEEE 1st ISQED*, Mar. 2000, pp. 451–454.

[3] B. Doyle, R. Arghavani, D. Barlage, and S. Datta, "Transistor elements for 30nm physical gate lengths and beyond," *Intel Technol. J.*, vol. 6, no. 2, pp. 42–54, 2002.

[4] H. Chang and S. S. Sapatnekar, "Prediction of leakage power under process uncertainties," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 2, p. 12, Apr. 2007.

[5] W. Kim, K. T. Do, and Y. H. Kim, "Statistical leakage estimation based on sequential addition of cell leakage currents," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 4, pp. 602–615, Apr. 2010.

[6] O. Assare, M. Momtazpour, and M. Goudarzi, "Accurate estimation of leakage power variability in sub-micrometer CMOS circuits," in *Proc. 15th Euromicro Conf. Diginal Syst. Des.*, Sep. 2012, pp. 18–25.

[7] H. Chang and S. S. Sapatnekar, "Full-chip analysis of leakage power under process variations, including spatial correlations," in *Proc. 42nd Des. Autom. Conf.*, Jun. 2005, pp. 523–528.

[8] S. Narendra, V. De, S. Borkar, D. Antoniadis, and A. Chandrakasan, "Full-chip sub-threshold leakage power prediction model for sub-0.18 μm CMOS," in *Proc. ISLPED*, Aug. 2002, pp. 19–23.

[9] A. Srivastava, R. Bai, D. Blaauw, and D. Sylvester, "Modeling and analysis of leakage power considering within-die process variations," in *Proc. ISLPED*, Aug. 2002, pp. 64–67.

[10] S. Mukhopadhyay and K. Roy, "Modeling and estimation of total leakage current in nano-scaled CMOS devices considering the effect of parameter variation," in *Proc. ISLPED*, Aug. 2003, pp. 172–175.

[11] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical analysis of subthreshold leakage current for VLSI circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 2, pp. 131–139, Feb. 2004.

[12] R. R. Rao, A. Devgan, D. Blaauw, and D. Sylvester, "Parametric yield estimation considering leakage variability," in *Proc. 41st DAC*, Jul. 2004, pp. 442–447.

[13] H. F. Dadgour, S.-C. Lin, and K. Banerjee, "A statistical framework for estimation of full-chip leakage-power distribution under parameter variations," *IEEE Trans. Electron Devices*, vol. 54, no. 11, pp. 2930–2945, Nov. 2007.

[14] K. Heloue, N. Azizi, and F. Najm, "Full-chip model for leakage-current estimation considering within-die correlation," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 28, no. 6, pp. 874–887, Jun. 2009.

[15] R. Shen, S. X.-D. Tan, and J. Xiong, "A linear algorithm for full-chip statistical leakage power analysis considering weak spatial correlation," in *Proc. 47th DAC*, 2010, pp. 481–486.

[16] M. Gao, Z. Ye, Y. Wang, and Z. Yu, "Efficient tail estimation for massive correlated log-normal sums—With applications in statistical leakage analysis," in *Proc. 47th ACM/IEEE DAC*, Jun. 2010, pp. 475–480.

[17] V. Veetil, D. Sylvester, D. Blaauw, S. Shah, and S. Rochel, "Efficient smart sampling based full-chip leakage analysis for intra-die variation considering state dependence," in *Proc. 46th ACM/IEEE DAC*, Jul. 2009, pp. 154–159.

[18] J. Ahn, J. Kim, and Y. H. Kim, "Statistical leakage analysis by parallel Monte-Carlo programming on a CUDA platform," in *Proc. 3rd ASQED*, Jul. 2011, pp. 146–150.

[19] J. Kim, W. Kim, and Y. H. Kim, "Evaluation of the state-of-the art statistical leakage estimation methods using the BSIM4 transistor model," in *Proc. 12th ISIC*, Dec. 2009, pp. 409–412.

[20] J. Ahn, J. Kim, and Y. H. Kim, "Leakage current modeling for GPGPU systems," in *Proc. Int. SoC Des. Conf.*, Nov. 2011, pp. 175–178.

[21] X. J. Xi *et al.*, "BSIM4.3.0 MOSFET model-userŠs manual," Dept. Electr. Eng., Univ. California, Berkeley, CA, USA, 2003.

[22] (2013, May 4). *NVIDIA CUDA—Parallel Programming and Computing Platform* [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[23] E. Acar, A. Devgan, R. Rao, Y. Liu, S. Nassif, and J. Burns, "Leakage and leakage sensitivity computation for combinational circuits," in *Proc. ISLPED*, Aug. 2003, pp. 96–99.

[24] (2013, May 9). *OpenSPARC T2* [Online]. Available: http://www.oracle.com/technetwork/systems/opensparc/opensparc-t2-page-1446157.html

[25] (2013, May 9). *The Predictive Technology Model (PTM)* [Online]. Available: http://ptm.asu.edu/introduction.html

[26] A. Srivastava, S. Shah, K. Agarwal, D. Sylvester, D. Blaauw, and S. Director, "Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance," in *Proc. 42nd DAC*, Jun. 2005, pp. 535–540.

[27] L. R. Schaeffer, "Modification of negative eigenvalues to create positive definite matrices and approximation of standard errors of correlation estimates," Ph.D. dissertation, Dept. Animal Poultry Sci., Univ. Guelph, Guelph, ON, Canada, 2010.

[28] (2013, May 4). *CUBLAS—NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS)* [Online]. Available: https://developer.nvidia.com/cublas

[29] (2013, May 4). *BLAS (Basic Linear Algebra Subprograms* [Online]. Available: http://www.netlib.org/blas/

[30] (2013, Jul. 22). *MATLAB—The Language of Technical Computing* [Online]. Available: http://www.mathworks.com/products/matlab/

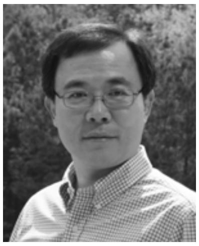
[31] (2013, Jul. 22). *GeForce GTX 680 | Specifications | GeForce* [Online]. Available: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-680/specifications

[32] (2012, May 17). *Synopsys Hspice—Accurate Circuit Simulation* [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx.

[33] D. J. Hyun, K. T. Do, and Y. H. Kim, "Monte Carlo simulation for a SoC design using the convergence condition of confidence interval," in *Proc. ITC/CSCC*, Jul. 2007, pp. 1205–1206.

JINWOOK KIM (S'07) was born in Seoul, Korea. He received the B.E. degree in electrical engineering from the Pohang University of Science and Technology, Pohang, Korea, in 2007, where he is currently pursuing the Ph.D. degree.

His current research interests include DRAM power network modeling and simulation with emphasis in model order reduction, gate-level statistical analysis, and parallel CAD algorithms on GPGPU system.



YOUNG HWAN KIM (S'86–M'89) received the B.E. degree in electronics from the Kyungpook National University, Korea, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, USA, in 1977, 1985, and 1988, respectively.

From 1977 to 1982, he was with the Agency for Defense Development, Korea, where he was involved in various military research projects, including the development of autopilot guidance and control systems. From 1983 to 1988, he was a Post-Graduate Researcher, developing VLSI CAD programs with the Electronic Research Laboratory, University of California, Berkeley.

Since 1988, he has been with the Division of Electronic and Computer Engineering, Pohang University of Science and Technology, Korea, where he is currently a Professor. His research interests include the design of plasma and liquid crystal display systems, MPSoC and GPGPU system design for display and computer vision applications, statistical analysis and design technology for deep-submicron semiconductor devices, and power noise analysis.

Dr. Kim has served as an Editor of the *Journal of the Institute of Electronics Engineers of Korea*, and as a General Chair and a Committee Member of various International and Korean domestic technical conferences, including International SoC Design Conference and the IEEE ISCAS 2012.