Journal of Circuits, Systems, and Computers, Vol. 11, No. 4 (2002) 333–350

BOOTSTRAP MONTE CARLO WITH ADAPTIVE STRATIFICATION FOR POWER ESTIMATION

HENG-LIANG HUANG* and JING-YANG JOU†

Department of Electronic Engineering, National Chiao-Tung University, 1001 Da-Hsueh Road, HsinChu, Taiwan, 30050, ROC

**cooper@ee.nctu.edu.tw*

†*jyjou@bestmap.ee.nctu.edu.tw*

Received 15 April 2002
Accepted 27 June 2002

Monte Carlo approach for power estimation is based on the assumption that the samples of power are *Normally* distributed. However, the power distribution of a circuit is not always *Normal* in the real world. In this paper, the *Bootstrap* method is adopted to adjust the confidence interval and redeem the deficiency of the conventional *Monte Carlo* method. Besides, a new input sequence stratification technique for power estimation is proposed. The proposed technique utilizes a multiple regression method to compute the coefficient matrix of the indicator function for stratification. This new stratification technique can adaptively update the coefficient matrix and keep the population of input vectors in a better stratification status. The experimental results demonstrate that the proposed *Bootstrap Monte Carlo method with adaptive stratification* can effectively reduce the simulation time and meet the user-specified confidence level and error level.

Keywords: Power characterization; adaptive stratification; *Monte Carlo*; *Bootstrap.*

1. Introduction

With the increasing size of design blocks, the number of input vectors required for estimating the power consumption of a circuit is growing exponentially. In the meantime, the time needed for simulating each input vector increases rapidly with the growing complexity of circuits. In previous literatures, methods for shortening the time required for power estimation can be classified into two categories. One is to generate a shorter input sequence, and the other is to sample a small portion of the input vectors from the original sequence. To regenerate an input sequence that has the similar average power as that of the original input sequence, some features of the original sequence need to be preserved while regenerating the shorter one. These features include preserving the pattern transition probabilities,[1] preserving the spatial-temporal correlations for all inputs,[2] and preserving the significant correlations between the clustered inputs.[3] Regenerating a compact input sequence sounds easy. However, the compact input sequence can only be generated according

to a user-specified compaction ratio, which users usually do not know the proper value.

The *Monte Carlo* approach for power estimation is proposed by F. Najm.[4] The method estimates the average power by sampling the input vectors with certain length l from the original sequence and fed them into the simulator to derive a sample value of the average power. The average power consumption can be estimated with the average of several sample values. From *Central Limit Theorem* (*CLT*), the sample values can be presumed as a *Normal* distribution when l approaches infinity. The probability that the estimated mean is within a certain error range of the real mean can also be derived under the assumption. However, the required l to preserve the normality of $\mathbf{x}$ is not discussed. If $\mathbf{x}$ is far from a *Normal* distribution, the basis of the *Monte Carlo* method fails and the estimated power may have a larger error level than expected.

Bootstrap theory is a resampling technique that will generate *Bootstrap Samples* by picking the sample data with replacement and report a *Bootstrap* confidence interval without assuming any parameter of the distribution.[5] This is also known as nonparametric *Bootstrap* resampling. By adopting the *Bootstrap* technique, this paper develops a way to calculate a more accurate confidence level to assure that the user specified confidence level would not be violated in *Monte Carlo* simulation.

Although the *Monte Carlo* method can achieve acceptable input sequence compaction ratio generally, it suffers severe degradation as dealing with power histograms like bi-modal or multi-modal.[4] For *Monte Carlo* approach, large sample variance means large number of samples required for the estimation to converge to the real value. The stratification method on the original input sequence is proposed to minimize the sample variance and the probability of generating the pre-matured samples.[7] According to the method, a gate level power model is required for roughly estimating the power consumption of the original input sequence on a zero-delay logic simulator. The zero-delay gate-level power consumption is used as an indicator of the circuit-level power consumption. With this indicator, the original input sequence can be partitioned into strata, within which the input vectors are with similar power consumptions such that the samples sampled from these strata can have a smaller sample variance. However, the gate level net-list sometimes needs to be concealed especially when they are the intellectual property (IP). In this paper, a novel input sequence stratification technique is proposed. It utilizes the multiple regression method on the sampled input vectors to find the weighting of each input transition, which can be used in the power indicator function for stratification. The proposed technique can restratify the original input sequence according to the updated samples, and keep the sample variance the smallest.

The following parts of this paper are organized as follows. In Sec. 2, some essential definitions and basis for this paper are introduced. Section 3 details the concepts of the *Bootstrap Monte Carlo* method and demonstrates its efficiency. In Sec. 4, the proposed adaptive stratification technique with multiple regression method is presented. The flow of the *Bootstrap Monte Carlo* method combined with the adaptive

stratification technique is shown and evaluated in Sec. 4 also. Section 5 concludes this paper.

2. Preliminary

2.1. *Normal distribution and Gaussian distribution*

2.1.1. *Normal random variable*

A random variable (RV) $\mathbf{x}$ is *Normally* distributed with mean μ_x, and standard deviation σ_x if its probability density function (denoted as *p.d.f.*) equals:

$$f(x) = \frac{1}{\sigma_x\sqrt{2\pi}}\, e^{-(x-\mu_x)^2/2\sigma_x^2}\,. \tag{1}$$

2.1.2. *Gaussian* (*Standard Normal*) *random variable*

A RV $\mathbf{y}$ is *Gaussian* if its *p.d.f.* equals:

$$g(y) = \frac{1}{\sqrt{2\pi}}\, e^{-y^2/2}\,. \tag{2}$$

2.1.3. *Cumulative distribution function* (*c.d.f.*) *of Gaussian*

The probability of a *Gaussian* RV $\mathbf{y}$ smaller than an arbitrary value y is defined as:

$$G(y) = P\{\mathbf{y} \le y\} = \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi}}\, e^{-\xi^2/2}\, d\xi\,. \tag{3}$$

2.1.4. *α-percentile of Gaussian*

The α-percentile of *Gaussian* is denoted as z_α and expressed as:

$$z_\alpha = G^{-1}(\alpha)\,, \qquad 0 \le \alpha \le 1\,. \tag{4}$$

Note that the *p.d.f.* of the *Gaussian* RV is an even function, therefore $z_\alpha = -z_{1-\alpha}$, and $z_{0.5} = 0$.

2.1.5. *Sample mean and sample variance*

Let $\{x_i,\, i = 1, 2, \ldots, n\}$ be n randomly sampled elements out of a population with an arbitrary distribution, the *sample mean* is defined as the arithmetic average of these n samples

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i\,. \tag{5}$$

The sample variance s^2 is defined as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\,. \tag{6}$$

2.2. *Monte Carlo method*

The power consumption of a CMOS circuit is dominated by charging and discharging of the load capacitances at each gate output. The average power consumption can thus be defined as a function in terms of *successive input patterns*:

$$\mu_x = \sum_{j=0}^{N-1} \frac{Power(V^j)}{N}\,, \tag{7}$$

where μ_x is the average power consumption, N is the number of input patterns, and $Power(V^j)$ is the power measured when the primary inputs are transiting from the the jth pattern to the $(j+1)$th pattern. The definition of V^j will be detailed in Sec. 4.

Let **pwr** be the RV defined on a sample space containing all $Power(V^j)$. The average of l values of **pwr** is called a *random sample*, x, whose sample mean approaches the desired average power, μ_x, and can be expressed as:

$$x = \frac{1}{l}\sum_{i=1}^{l} pwr_i\,, \tag{8}$$

where pwr_i is a value of the RV **pwr**. According to the *Central Limit Theorem* (*CLT*), the RV **x** has a distribution close to *Normal* for large l.[8]

To estimate the μ_x in Eq. (7) without simulating all input vectors, the *Monte Carlo* approach for power estimation can help. Let $\bar{x}$ and s^2 be the sample mean and sample variance of **x**, respectively. From Eqs. (4)–(6), the following results can be derived:

$$P\left\{\left|\frac{\mu_x - \bar{x}}{\bar{x}}\right| \le rel\_err\right\} = 1 - 2\alpha\,, \qquad 0 \le \alpha \le 0.5\,, \tag{9}$$

where

$$rel\_err = \frac{z_{1-\alpha^{\mathrm{S}}}}{\bar{x}\sqrt{n}}\,.$$

The *rel_err* stands for the related error level, α is the confidence level, and n is the number of samples of **x**. Equation (9) means that the user can have a confidence level of $1-2\alpha$ about the claim that the error between the real mean μ_x and the sample mean $\bar{x}$ is smaller than the related error level. If the related error level *rel_err* is larger than the user specified error level ε, one or more samples of **x** should be picked and the sample mean and *rel_err* are evaluated again. The procedure is iterated until the user-specified error level ε is satisfied.

2.3. *Bootstrap*[6]

2.3.1. *Bootstrap Replication*

Let **x** be the RV defined as the samples from an arbitrary distribution:

$$\mathbf{x} = \{x_i | 1 \le i \le n\}\,. \tag{10}$$

Let $\mathbf{x}^*$ be the RV defined as the random samples of $\mathbf{x}$ with replacement for each x_i with equal probability, $1/n$:

$$\mathbf{x}^* = \{x_i^* | 1 \le i \le n\,,\ x_i^* \in \mathbf{x}\}\,. \tag{11}$$

The *Bootstrap Replication* $\mathbf{b}$ of $\bar{x}$ is defined as the mean of $\mathbf{x}^*$:

$$\mathbf{b} = \left\{ b_k \middle| b_k = \frac{1}{n}\sum_{i=1}^{n} x_i^*\,,\ 1 \le k \le nb \right\}, \tag{12}$$

where nb is the number of *Bootstrap Replications*.

2.3.2. *Sorted Bootstrap Replications*

Let the RV $\mathbf{B}$ stands for the sorted *Bootstrap Replications* and defined as:

$$\mathbf{B} = \{B_k | B_k \in \mathbf{b}\,;\ B_i \le B_j \text{ if } i < j\,;\ 1 \le i, j, k \le nb\}\,. \tag{13}$$

2.3.3. *Cumulative distribution function (c.d.f.) of* $\boldsymbol{B}$

The probability that the RV $\mathbf{B}$ is smaller than an arbitrary value b is defined as:

$$GB(b) = P\{\mathbf{B} \le b\} = \left\{ \frac{k_b}{nb} \middle| B_i \le b\,,\ 1 \le i \le k_b \right\}. \tag{14}$$

2.3.4. α*-percentile of Bootstrap*

The α-*Bootstrap* percentile is defined as:

$$\theta_\alpha = GB^{-1}(\alpha)\,, \qquad 0 \le \alpha \le 1\,. \tag{15}$$

2.3.5. *Percentile confidence interval of Bootstrap*

There are several ways of calculating confidence intervals of the *Bootstrap Replications*. The most straightforward one for the $1 - 2\alpha$ *Bootstrap* confidence interval is the percentile *Bootstrap* confidence interval, and is defined as the interval that can cover $(1 - 2\alpha)^* nb$ *Bootstrap Replications*:

$$[\theta_{\%,\mathrm{lo}},\, \theta_{\%,\mathrm{up}}] = [GB^{-1}(\alpha),\, GB^{-1}(1-\alpha)]\,, \qquad 0 \le \alpha \le 0.5\,. \tag{16}$$

2.3.6. *Bias-Corrected and accelerated (BCa) confidence interval*

The *BCa* confidence intervals are complicated to describe but are as easy to use as the percentile confidence interval[5]:

$$[\theta_{BCa,\mathrm{lo}},\, \theta_{BCa,\mathrm{up}}] = [GB^{-1}(\alpha_1),\, GB^{-1}(\alpha_2)]\,,$$

$$\alpha_1 = G\left(\hat{z}_0 + \frac{\hat{z}_0 + z_\alpha}{1 - \hat{a}(\hat{z}_0 + z_\alpha)}\right), \tag{17}$$

$$\alpha_2 = G\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha}}{1 - \hat{a}(\hat{z}_0 + z_{1-\alpha})}\right).$$

The $\hat{z}_0$ is designated as the *Bias-Correction* coefficient. It is simply derived from the portion of *Bootstrap Replications* that are smaller than $\bar{x}$ (the sample mean of $\mathbf{x}$):

$$\hat{z}_0 = G^{-1}\left(\frac{\#\{b < \bar{x}\}}{nb}\right), \tag{18}$$

where the $\#\{b < \bar{x}\}$ is the number of *Bootstrap Replications* that are smaller than $\bar{x}$. The $\hat{a}$ is designated as the *Acceleration* coefficient. Before defining $\hat{a}$, the definition of the *Jacknife* value, $J_{(i)}$, of $\bar{x}$ is defined as:

$$\mathbf{J}_{(i)} = \left\{ J_{(i)} \middle| J_{(i)} = \frac{1}{n-1}\left[\left(\sum_{j=1}^{n} x_j\right) - x_i\right],\ 1 \le i \le n \right\}. \tag{19}$$

The mean of $J_{(i)}$ designate as $J_{(\cdot)}$ is defined as:

$$J_{(\cdot)} = \frac{1}{n}\sum_{j=1}^{n} J_{(i)}\,. \tag{20}$$

The *Acceleration* coefficient is then defined as:

$$\hat{a} = \frac{\sum_{i=1}^{n}(J_{(\cdot)} - J_{(i)})^3}{6 \times \left\{\sum_{i=1}^{n}(J_{(\cdot)} - J_{(i)})^2\right\}^{3/2}}. \tag{21}$$

The *BCa Bootstrap* estimation of μ_x is defined as the 0.5-percentile of the distribution of *BCa Bootstrap Replications*:

$$\begin{aligned}\bar{x}_{BCa} &= GB^{-1}\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{0.5}}{1 - \hat{a}(\hat{z}_0 + z_{0.5})}\right) \\ &= GB^{-1}\left(\hat{z}_0 + \frac{\hat{z}_0}{1 - \hat{a}\hat{z}_0}\right).\end{aligned} \tag{22}$$

The *Bias-Correction* coefficient $\hat{z}_0$ is designed to compensate the difference between $\bar{x}$ and the *BCa* mean $\bar{x}_{BCa}$. If the difference between them equals zero, $\hat{z}_0$ equals zero also. As for the *Acceleration* coefficient, it refers to the rate of change of the standard error of $\bar{x}$ with respect to $\bar{x}$, measured on a normalized scale.[6] The larger it is, the wider is the confidence interval. Detail discussion about how this acceleration coefficient works is referred to the references.[5] For a *Normally* distributed $\bar{x}$, the $\hat{z}_0$ and $\hat{a}$ are both zero, and $\alpha_1 = \alpha_2 = \alpha$. The *BCa* confidence interval is exactly the same as standard confidence interval.

3. Bootstrap Monte Carlo Simulation

3.1. *Bootstrap confidence level*

For conventional *Monte Carlo*, the confidence interval and the *rel_err* are calculated with Eq. (9), in which the *rel_err* totally relies on the assumption that the distribution of $\bar{x}$ is *Normal*. The possibility that the distribution of $\bar{x}$ might be skewed

or platykurtic is ignored. *Bootstrap* technique, can be used to adjust the confidence level when the normality of the population is poor.

For a given error level, ε, the acceptable range for the real mean μ_x is defined as:

$$[A_{\text{lo}},\, A_{\text{up}}] = [(1-\varepsilon)\text{MAX}(\bar{x},\, \bar{x}_{BCa}),\ (1+\varepsilon)\text{MIN}(\bar{x},\, \bar{x}_{BCa})]\,. \tag{23}$$

The acceptable range covers the safe range into which the real mean μ_x can land without violating the user specified error level ε, with respect to either $\bar{x}$ or $\bar{x}_{BCa}$. And then, the *Bootstrap Confidence Level* is defined with:

$$\alpha_{BCa} = GB(A_{\text{lo}}) + 1 - GB(A_{\text{up}})\,. \tag{24}$$

3.2. *Bootstrap Monte Carlo method*

With the α_{BCa} from Eq. (24), whether the user specified confidence level is guaranteed or not, can be easily determined. The proposed *Bootstrap Monte Carlo* (*BMC*) method is demonstrated with the pseudo-codes in Fig. 1.

```
Bootstrap Monte Carlo ( )
Pwr, ε, α; /* Conventional Monte Carlo parameters */
nb; /* Bootstrap parameter */
{
  nSamples = 1; rel_err = 1; #Boot = 0;
  z_α = G^{-1}(α);
  while (rel_err ≥ ε) {
    nSamples^{++};
    get new sample x_n from Pwr;
    update sample mean x̄ and sample variance s^2;
    update rel_err;
    if (rel_err ≤ ε) {
      Generate nb Bootstrap Replications b from x;
      Calculate α_BCa from b;
      #Boot^{++};
      if (α_BCa > 2*α) {
        rel_err = 1;
      }
    }
  }
  return nSamples, nBoot, and x̄;
}
```

Fig. 1. Pseudo code of *BMC*.

3.3. *Bootstrap Monte Carlo versus conventional Monte Carlo*

The proposed *BMC* method is tested with estimating the average powers of the *ISCAS-85* benchmark circuits. There are 10 000 input vectors in the input sequence for each circuit. The input sequence is a compound of three segments: a counter

sequence, a LFSR sequence and a sequence of pseudo random numbers. Note that half of the input vectors in the counter sequence have only single input change, therefore consumes less power. The LFSR sequence represents the input vectors with temporal correlations. The pseudo random input vectors, on the other hand, are spatially and temporally independent. The arrangement of the input sequence is to give the estimator a tough situation because the power histogram of such an input sequence is most likely to be skewed, long tailed and platykurtic at the same time. Besides the *ISCAS-85* benchmarks, an additional circuit, *add_mpr*, is included. It is a circuit with a mode controlling input that controls the function of the circuit to be an adder or a multiplier. The power histogram of it is a typical bimodal distribution. The experimental results are listed in Tables 1 to 3.

Table 1. MC versus BMC with $\alpha = 0.05$.

Circuit	*MC*		*BMC*		
	viol_r	*nVecs*	*viol_r*	*nVecs*	*#Boot*
C432	0.1235	1538.51	0.0949	1786.97	14.4926
C499	0.1164	585.86	0.1050	629.34	2.9860
C880	0.1218	616.24	0.1010	663.19	3.1880
C1355	0.1145	633.05	0.1005	682.92	3.2847
C1908	0.1194	908.35	0.1004	996.04	5.4834
C3540	0.1586	154.90	0.1563	156.16	1.0405
C6288	0.1259	1344.78	0.1014	1532.66	11.0338
add_mpr	0.1384	281.35	0.1288	288.27	1.2202
Max	0.1586	1538.51	0.1563	1786.97	14.4926
Avg	0.1273	757.88	0.1110	841.94	5.3412

Table 2. MC versus BMC with $\alpha = 0.025$.

Circuit	*MC*		*BMC*		
	viol_r	*nVecs*	*viol_r*	*nVecs*	*#Boot*
C432	0.0748	2055.63	0.0487	2504.36	25.7884
C499	0.0591	820.55	0.0497	886.94	4.4027
C880	0.0611	862.70	0.0497	935.03	4.7865
C1355	0.0594	884.39	0.0499	961.19	4.9675
C1908	0.0653	1248.65	0.0525	1397.90	9.0968
C3540	0.0832	230.54	0.0810	232.59	1.0764
C6288	0.0710	1812.33	0.0485	2150.83	19.6258
add_mpr	0.0688	408.67	0.0627	421.19	1.5073
Max	0.0832	2055.63	0.0810	2504.36	25.7884
Avg	0.0678	1040.43	0.0553	1186.25	8.9064

Table 3. MC versus BMC with $\alpha = 0.005$.

Circuit	*MC*		*BMC*		
	viol_r	*nVecs*	*viol_r*	*nVecs*	*#Boot*
C432	0.0266	3089.78	0.0092	4145.60	59.5692
C499	0.0163	1345.06	0.0115	1480.03	8.3435
C880	0.0179	1408.56	0.0111	1558.86	9.2150
C1355	0.0180	1442.10	0.0138	1600.27	9.6379
C1908	0.0198	1980.65	0.0121	2322.61	19.8780
C3540	0.0160	409.03	0.0146	416.39	1.3426
C6288	0.0238	2768.35	0.0105	3561.91	44.9796
add_mpr	0.0130	700.62	0.0111	728.20	2.9168
Max	0.0266	3089.78	0.0146	4145.6	59.5692
Avg	0.0189	1643.02	0.0117	1976.73	19.4853

Tables 1 to 3 are the results for α equals 0.05, 0.025 and 0.005, respectively. The corresponding confidence levels are 90%, 95% and 99%. For each α, there are results for every circuit with both conventional Monte Carlo method (*MC*) and the proposed *Bootstrap Monte Carlo* method (*BMC*). Each method is performed 10 000 times for each circuit to estimate their average power consumption. The error level ε is set to 0.05. If the error percentage of the estimated power exceed ε, the number of violations is increased by 1. The *viol_r* columns are the violation ratio defined as the number of violations divided by the number of *Monte Carlo* simulations:

$$viol\_r = \frac{\#\left\{\dfrac{\bar{x} - \mu_x}{\mu_x} > \varepsilon\right\}}{10\,000}. \tag{25}$$

The *nVecs* columns contain the numbers of input vectors sampled by the corresponding estimation methods. For a good estimation method, the *viol_r* should be close to and always smaller than $2^*\alpha$. For two estimation methods with the same *viol_r*, the one with smaller *nVecs* is the better one. There is one additional *#Boot* column for *BMC*. It is the average number of times that *Bootstrap* process being invoked for a *BMC* estimation. It is roughly in proportion to the number of additional samples required for *BMC*. With a larger *#Boot*, the overhead of using *Bootstrap* is greater. For the circuits that the *viol_r* of *MC* exceed $2^*\alpha$, the *#Boot* for *BMC* is supposed to be larger to keep the *viol_r* of *BMC* within $2^*\alpha$ safe range.

As demonstrated in the tables, the *viol_r* for conventional *Monte Carlo* method exceeds $2^*\alpha$ in all 24 cases. On the other hand, with the *Bootstrap* method monitoring confidence level in *BMC*, the *viol_r* are either under or close to the $2^*\alpha$, except for two cases: *C3540* and *add_mpr*. One thing needs to be emphasized is that the *nVecs* for *C3540* and *add_mpr* are the two smallest ones among all circuits. If the number of samples is too small to represent the original population, e.g. pre-matured samples, *BMC* might sometimes fail to keep the *viol_r* within $2^*\alpha$. This is because the

Bootstrap method produces its *Bootstrap Replications* from the samples of *Monte Carlo*. This drawback will be discussed and eliminated with the newly proposed method *Bootstrap Monte Carlo with Adaptive Stratification* method (*BMCAS*) in the following section. Regardless of this deficiency, the proposed *Bootstrap Monte Carlo* method is more trustable than conventional *Monte Carlo* method with about 10% increasing in *nVecs*.

4. Adaptive Stratified Random Sampling

Stratification is a technique to divide the sample space into subspaces to reduce the sample variance so that the *Monte Carlo* can converge sooner with smaller number of samples, n, and achieve better compaction ratio. An indicator function for stratification is a function that returns a value closely related to or even equals the power consumption of input vectors. To build an indicator function, the multiple regression method is adopted.

4.1. *Single variable linear regression*[10]

Given a collection of n data points of two variables $\mathbf{x}$ and $\mathbf{y}$:

$$(\mathbf{x}, \mathbf{y}) = \{(x_i, y_i) | 1 \leq i \leq n\} . \tag{26}$$

The best line describing the relation between $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$\begin{aligned} \hat{\mathbf{y}} &= a\mathbf{x} + b , \\ a &= \frac{E[\mathbf{xy}] - E[\mathbf{x}]E[\mathbf{y}]}{E[\mathbf{xx}] - E[\mathbf{x}]E[\mathbf{x}]} , \\ b &= E[\mathbf{y}] - aE[\mathbf{x}] , \end{aligned} \tag{27}$$

where $E[\ ]$ is the function of expected value, and $\hat{\mathbf{y}}$ is the predictor of $\mathbf{y}$. This predictor is the one with zero bias and minimum RMS error.

4.2. *Multiple regression*[10]

Multiple regression is simply an extension of the single variable linear regression. Given n data points of $m + 1$ variables:

$$\begin{aligned} (\mathbf{X}, \mathbf{y}) &= \{[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m], \mathbf{y}\} \\ &= \{\{[x_{1,i}, x_{2,i}, \ldots, x_{m,i}], y_i\} | 1 \leq i \leq n\} . \end{aligned} \tag{28}$$

Note that $\mathbf{X}$ is a row matrix stands for the $\mathbf{x}$ variables. The best function describing the relation between $\mathbf{X}$ and $\mathbf{y}$ is defined as:

$$\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}\mathbf{A} + b\,, \\
\mathbf{C} &= E[\mathbf{X}\mathbf{y}] - E[\mathbf{X}]E[\mathbf{y}]\,, \\
\mathbf{D} &= E[\mathbf{X}^T\mathbf{X}] - E[\mathbf{X}]^T E[\mathbf{X}]\,, \\
\mathbf{A} &= \mathbf{D}^{-1}\mathbf{C}^T\,, \\
b &= E[\mathbf{y}] - E[\mathbf{X}]\mathbf{A}\,,
\end{aligned} \tag{29}$$

where $\mathbf{A}$ is a column matrix that contains the coefficients for each $\mathbf{x}_1, \ldots, \mathbf{x}_m$. Note that the order of $\mathbf{X}$ and $\mathbf{A}$ are switched to make the product of them a scalar.

4.3. *Variable selection*

As we can see in Eq. (29) that deriving the coefficient matrix $\mathbf{A}$ includes a matrix inverse operation on matrix $\mathbf{D}$. If matrix $\mathbf{D}$ were singular, the equation of measuring the coefficient matrix $\mathbf{A}$ would fail. To prevent this, the basic assumptions of *Multiple Regression* need to be taken into consideration while selecting the $\mathbf{x}$ variables.

Assumption 1: The relation between $\mathbf{y}$ and $\mathbf{x}_i$ is linear.

The $\mathbf{y}$ variable in power estimation is the power consumption of each input vector. The power consumption of CMOS circuits is dominated by the dynamic power. The dynamic power is consumed when the inputs of the circuits are switching. In other words, more input switching implies that more dynamic power is consumed. This makes the switching conditions of the primary inputs good candidates for the $\mathbf{x}$ variables.

Assumption 2: The $\mathbf{x}_i$ variables are mutually independent.

For a primary input, there are four possible transitions between two consecutive clock cycles: 0 to 0, 0 to 1, 1 to 0, and 1 to 1.

Let the Boolean value for the ith input in the jth clock cycle be denoted as b_i^j, the input pattern in the jth clock cycle is defined as:

$$Pat^j = \{b_i^j | b_i^j \in \{0, 1\},\ 1 \le i \le n\_in,\ 0 \le j \le N\}\,, \tag{30}$$

where n_in stands for the number of primary inputs, and N stands for the total number of input vectors. The jth input vector is defined as two consecutive input patterns:

$$V^j = \{(Pat^j,\ Pat^{j+1}) | 0 \le j \le N - 1\}\,. \tag{31}$$

There are four transition variables designated for the transition behavior of each input:

$$\begin{aligned}
&\mathbf{T}(i, V^j) = \{T_k(i, V^j) | 0 \le k \le 3,\ 1 \le i \le n_in,\ 0 \le j \le N - 1\}\,, \\
&\text{where} \qquad T_k(i, V^j) = \begin{cases} 1, & \text{if } k = 2 \times b_i^j + b_i^{j+1}\,, \\ 0, & \text{otherwise}\,. \end{cases}
\end{aligned} \tag{32}$$

For an input i and input vector V^j, one and only one of the four transitions can take place. This makes the four transition variables mutually dependent:

$$T_0(i, V^j) + T_1(i, V^j) + T_2(i, V^j) + T_3(i, V^j) = 1\,. \tag{33}$$

Therefore, at most three out of the four transition variables need to be chosen as the $\mathbf{x}$ variables for multiple regression. In this paper, T_0, T_1 and T_2 are selected. Note that, any combination of three chosen transition variables will not lose their generality because the removed one can always be derived from the others.

After choosing the $\mathbf{X}$ variables for Eq. (29), it can be rewritten in the form of constructing the predictor for the power consumption of each input vector. Let the power consumption of each input vector indicated by the indicator function be:

$$\begin{aligned} \hat{P}ower(\mathbf{V}) &= \mathbf{X}(\mathbf{V})\mathbf{A} + b\,, \\ \mathbf{X}(\mathbf{V}) &= \{[T_0(i, \mathbf{V}),\ T_1(i, \mathbf{V}),\ T_2(i, \mathbf{V})] | 1 \le i \le n\_in\}\,, \end{aligned} \tag{34}$$

where $\mathbf{V}$ is the set of all input vectors. In the same manner, the coefficient matrix $\mathbf{A}$ can be derived from the multiple regression equations:

$$\begin{aligned} \mathbf{C} &= E[\mathbf{X}(\mathbf{W})Pwr(\mathbf{W})] - E[\mathbf{X}(\mathbf{W})]E[Pwr(\mathbf{W})]\,, \\ \mathbf{D} &= E[\mathbf{X}(\mathbf{W})^T\mathbf{X}(\mathbf{W})] - E[\mathbf{X}(\mathbf{W})]^T E[\mathbf{X}(\mathbf{W})]\,, \\ \mathbf{A} &= \mathbf{D}^{-1}\mathbf{C}^T\,, \\ b &= E[Pwr(\mathbf{W})] - E[\mathbf{X}(\mathbf{W})]\mathbf{A}\,, \end{aligned} \tag{35}$$

where $\mathbf{W}$ is the set of sampled input vectors and $Pwr(\mathbf{W})$ are their corresponding power consumption measured with simulator.

4.4. *BMC with adaptive stratification* (*BMCAS*)

With the coefficient matrix $\mathbf{A}$ and Eq. (34), the population can be stratified into a certain number of strata. Initially, the population is not stratified, and the first few samples $\mathbf{x}_i$'s are randomly sampled. After the number of sampled input vectors is larger than a predefined threshold, the *Multiple Regression* function is invoked to derive $\mathbf{A}$. The $\hat{P}ower(\mathbf{V})$ is calculated with Eq. (34) and the stratification process starts. Hence, the stratified random sampling process takes over the place of random sampling. After some other new input vectors are sampled, the *Multiple Regression* process is executed again to recalculate a better $\mathbf{A}$ for re-stratification. The pseudo code for the proposed *Bootstrap Monte Carlo with Adaptive Stratification* (*BMCAS*) is depicted in Fig. 2.

The reason of starting multiple regression after the number of $\mathbf{W}$ larger than 9^*n_in is to prevent too many empty elements in matrix $\mathbf{D}$ which might lead to a singular matrix for the matrix inverse operation. The *keep_sampling* variable is an insurance to prevent the *Bootstrap Monte Carlo* failure caused by the premature

```
Bootstrap Monte Carlo with Adaptive Stratification ( )
Pwr, ε, α; /* Conventional Monte Carlo parameters */
nb; /* Bootstrap parameter */
{
  nSamples = 0; rel_err = 1; nRestrat = 0; keep_sampling = 0;
  W = {∅}; S = {∅}; V = {V^i | 1 ≤ i ≤ N}
  z_α = G^{-1}(α);
  while (rel_err ≥ ε)
  {
    nSamples^{++};
    if (stratified)
      Sample input vectors v from V with stratified sampling;
    else
      Sampled input vectors v from V with random sampling;
    S = S ∪ v; W = W ∪ v;
    Get y = Power(v); from simulator;
    y = y ∪ y;
    if (keep_sampling) {
      keep_sampling–; rel_err = 1; continue;
    } else {
      update x̄, s^2, and rel_err;
    }
    if (rel_err ≥ ε) {
      if (#(W) > 9*n_in) {
MR:       X = T(S);
          if (A = Multiple_Regression(X, y) is success) {
            P̂ower(V) = T(V)A + b;
              Restratification(V);
              stratified = TRUE; W = {∅}; nRestrat^{++};
          }
      }
    } else {
      Generate nb Bootstrap Replications b from x;
      Calculate α_BCa from b;
      #Boot^{++};
      if (α_BCa > 2*α) {
        rel_err = 1;
      } else if (stratified) {
        rel_err = 1; keep_sampling = 2; goto MR;
      } else {
        return nSamples, nRestrat, and x̄;
      }
    }
  }
}
```

Fig. 2. Pseudo code of *BMCAS*.

samples as discussed in previous section. The *rel_err* and the *keep_sampling* are given one and two, respectively when *BMCAS* exits before any stratification is performed. With these setting, the *BMCAS* will stratify the population at least once and samples three new samples after the forced stratification get at least three samples that are sampled from the population more uniformly.

4.5. *Experimental results*

To demonstrate the performance of the proposed *BMCAS*, one more stratification method called *Hamming Distance Method* (*HDM*) is implemented for comparison. It is a stratification method based on the assumptions that when power consumption increases with the number of inputs transitions. *HDM* method stratifies the input vectors into strata according to the *Hamming Distance* of each input vector. For both methods, the input vectors are stratified into six equal sized strata. The reason is that one can get little sample variance reduction by setting the number of strata larger than six 0.

The results are in Tables 4 to 6. The results from *BMC* method in Sec. 4 are listed in the tables, too. It is designated as *NO_STRAT* because there is no stratification procedure in *BMC*. Similar to Tables 1 to 3, Tables 4 to 6 show the results for confidence level 90%, 95% and 99%, respectively. The first columns of them are the names of the circuits. The numbers in the brackets next to the circuit names are the number of inputs of the corresponding circuit. They are listed as a reference because *BMCAS* re-stratifies the population after 9^*n_in new sampled input vectors. There are three columns of data for each stratification method. The "*nSample*" column shows the average number of samples required for the corresponding stratification method to converge to a value of estimated power. The "*nVecs*" columns are the average numbers of sampled input vectors. The smaller is the *nVecs*, the better is the stratification method. The error level ε is set to 0.05. The *viol_r* column shows the violation ratios with the same definition as Eq. (25), and the #Boot columns are of the same definition as in Tables 1 to 3. The extra *#ReStrat* column for *BMCAS* is the average number of stratification being performed.

Comparing the proposed *BMCAS* and the *BMC*, the *viol_r* of *BMCAS* are closer to $2^*\alpha$ than *BMC* in almost all circuits and all confidence level. Besides, with the proposed adaptive stratification technique, the numbers of sampled input vectors are about 27% smaller than those of *BMC* in average. For some circuits, *BMCAS* even needs only half of the number of input vectors that the *BMC* needs. The *#Restrat* for some circuits, like *C3540*, are equal to exactly 1.0 because all of the *BMC* estimations exit with a number of sampled input vectors smaller than the threshold for starting stratification, and *BMCAS* will perform at least one stratification process before exiting. As shown in Tables 4 to 6, the *viol_r* for *C3540* are safely kept within $2^*\alpha$. As for the results from *HDM* method, although its *nVecs* is the smallest, the violation ratio exceeds $2^*\alpha$ for all cases. This makes the results from *HDM* untrustable and the *nVecs* meaningless. With the safely kept *viol_r* and

Table 4. Comparison of various stratification method, confidence level = 90% ($2\alpha = 0.1$).

Circuit	*NO_STRAT*				*HDM*			*BMCAS*				
	nSample	*nVecs*	*viol_r*	*#Boot*	*nSample*	*nVecs*	*viol_r*	*nSample*	*nVecs*	*viol_r*	*#ReStrat*	*#Boot*
C432 (37)	297.74	1786.47	0.0949	14.4926	48.56	291.36	0.1596	193.99	1163.97	0.0918	3.6702	7.8359
C499 (42)	104.89	629.34	0.1050	2.9860	4.96	29.79	0.1421	90.72	544.30	0.0739	1.9919	3.8312
C880 (61)	110.53	663.19	0.1010	3.1880	13.02	78.12	0.2141	107.92	647.56	0.0870	1.8311	3.7967
C1355 (42)	113.82	682.92	0.1005	3.2847	4.85	29.10	0.1388	94.46	566.80	0.0770	1.9974	3.9666
C1908 (34)	166.00	996.04	0.1004	5.4834	10.60	63.59	0.2060	105.05	630.31	0.0795	2.2278	2.2278
C3540 (51)	26.03	156.16	0.1563	1.0405	7.10	42.61	0.1923	30.81	184.89	0.0982	1.0000	2.0777
C6288 (33)	255.44	1532.66	0.1014	11.0338	10.36	62.16	0.1969	138.80	832.80	0.0831	3.0274	5.2061
add_mpr (36)	48.04	288.27	0.1288	1.2202	43.35	260.07	0.1628	53.22	319.32	0.0911	1.3348	2.4049
Max	297.74	1786.47	0.1563	14.4926	48.56	291.36	0.2141	193.99	1163.97	0.0982	3.6702	7.8359
Avg	140.31	841.88	0.1110	5.3412	17.85	107.1	0.1766	101.87	611.24	0.0852	2.1351	3.9184

Table 5. Comparison of various stratification method, confidence level = 95% ($2\alpha = 0.05$).

Circuit	*NO_STRAT*				*HDM*			*BMCAS*				
	nSample	*nVecs*	*viol_r*	*#Boot*	*nSample*	*nVecs*	*viol_r*	*nSample*	*nVecs*	*viol_r*	*#ReStrat*	*#Boot*
C432 (37)	417.39	2504.36	0.0487	25.7884	72.63	435.80	0.0703	261.99	1571.98	0.0393	4.6465	12.3182
C499 (42)	147.82	886.94	0.0497	4.4027	6.96	41.74	0.0852	108.05	648.32	0.0392	2.0039	3.8246
C880 (61)	155.84	935.03	0.0497	4.7865	20.14	120.83	0.1193	132.88	797.26	0.0458	1.9994	4.2675
C1355 (42)	160.20	961.19	0.0499	4.9675	6.77	40.66	0.0847	111.59	669.55	0.0400	2.0152	3.8161
C1908 (34)	232.98	1397.90	0.0525	9.0968	16.29	97.75	0.1148	131.98	791.87	0.0430	2.9044	4.7751
C3540 (51)	38.77	232.59	0.0810	1.0764	10.42	62.51	0.1158	43.19	259.15	0.0457	1.0000	2.1437
C6288 (33)	358.47	2150.83	0.0485	19.6258	15.91	95.45	0.1146	188.89	1133.32	0.0395	3.9200	7.7984
add_mpr (36)	70.20	421.19	0.0627	1.5073	64.84	389.06	0.0768	68.21	409.27	0.0443	1.9117	2.7532
Max	417.39	2504.36	0.0810	25.7884	72.63	435.80	0.1193	261.99	1571.98	0.0458	4.6465	12.3182
Avg	197.71	1186.25	0.0553	8.9064	26.75	160.48	0.0977	130.85	785.09	0.0421	2.5501	5.2121

Table 6. Comparison of various stratification method, confidence level = 99% ($2\alpha = 0.01$).

Circuit	*NO_STRAT*				*HDM*			*BMCAS*				
	nSample	*nVecs*	*viol_r*	*#Boot*	*nSample*	*nVecs*	*viol_r*	*nSample*	*nVecs*	*viol_r*	*#ReStrat*	*#Boot*
C432 (37)	690.93	4145.60	0.0092	59.5692	125.95	755.71	0.0135	425.68	2554.10	0.0073	6.8489	26.3906
C499 (42)	246.67	1480.03	0.0115	8.3435	11.88	71.27	0.0269	148.93	893.58	0.0097	2.7605	4.8357
C880 (61)	259.80	1558.86	0.0111	9.2150	37.33	223.95	0.0316	186.26	1117.56	0.0120	2.1826	5.9437
C1355 (42)	266.71	1600.27	0.0138	9.6379	11.53	69.20	0.0275	155.74	934.48	0.0093	2.8516	5.1525
C1908 (34)	387.10	2322.61	0.0121	19.8780	30.19	181.13	0.0330	209.86	1259.15	0.0078	4.2333	8.1702
C3540 (51)	69.34	416.39	0.0146	1.3426	18.92	113.53	0.0367	73.16	438.95	0.0094	1.2434	2.5202
C6288 (33)	593.65	3561.91	0.0105	44.9796	29.90	179.37	0.0325	315.67	1894.00	0.0093	6.0004	16.0834
add_mpr (36)	123.03	728.20	0.0111	2.9168	112.04	672.25	0.0137	93.23	559.35	0.0074	2.0002	3.9304
Max	690.93	4145.60	0.0146	59.5692	125.95	755.71	0.0367	425.68	2554.10	0.0120	6.8489	26.3906
Avg	329.65	1976.73	0.0117	19.4853	47.22	283.30	0.0269	201.07	1206.40	0.0090	3.5151	9.1283

the smaller number of *nVecs*, we can summarize that the proposed *BMCAS* is the most reliable one and can efficiently reduce the required number approach.

5. Conclusion

In this paper, the *Bootstrap* technique is adopted to assure the confidence level when doing *Monte Carlo* estimation. With this technique, the *Monte Carlo* method can be improved since confidence level is monitored during the *Monte Carlo* simulation. Besides, this paper proposed a novel *Adaptive Stratification* technique, with which the population can be dynamically and well stratified to keep the sample variance minimized. The experimental results on the *ISCAS-85* benchmarks show that the proposed *Bootstrap Monte Carlo with Adaptive Stratification* (*BMCAS*) successfully preserves the confidence level while efficiently reducing the required number of input vectors.

References

1. A. Pinar and C. L. Liu, "Power invariant vector sequence compaction", *Proceeding of ICCAD*, 1998, pp. 473–476.
2. R. Marculescu, D. Marculescu, and M. Pedram, "Sequence compaction for power estimation theory and practice", *IEEE Trans. CAD*, July 1999, pp. 973–993.
3. N. Dragone, R. Zafalon, C. Guardiani, and C. Silvano, "Power invariant vector compaction based on bit clustering and temporal partitioning", *Proceeding of ISLPED*, 1998, pp. 118–120.
4. R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation", *IEEE Trans. VLSI*, March 1993, pp. 63–71.
5. B. Efron, "Better bootstrap confidence intervals", *J. American Statistical Association* (1987) 171–200.
6. B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall, 1993.
7. C. S. Ding, Q. Wu, C. T. Hsieh, and M. Pedram, "Stratified random sampling for power estimation", *IEEE Trans. CAD*, June 1998, pp. 465–478.
8. A. Papoulis, *Probability and Statistics*, Prentic-Hall, 1990.
9. H. Stark and J. W. Woods, *Probability, Random Process, and Estimation Theory for Engineers*, Prentice-Hall, 1986.
10. A. Bogliolo, L. Benini, and G. D. Micheli, "Adaptive least square behavioral power modeling", *Proceeding of ED&TC*, 1997, pp. 400–406.
11. S. R. Huang, "Effectiveness of optimum stratified sampling and estimation in Monte Carlo production simulation", *IEEE Trans. Power System*, May 1997, pp. 566–572.