Scalable Sampling Methodology for Logic Simulation: Reduced-Ordered Monte Carlo

Chien-Chih Yu, Armin Alaghi and John P. Hayes
Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI, 48109, USA
{ccyu, alaghi, jhayes}@eecs.umich.edu

ABSTRACT

Monte Carlo (MC) simulation plays a key role in EDA as the gold standard against which heuristics are measured. It is also an important stand-alone technique for statistics-based tasks like power estimation and reliability analysis. Accurate simulation requires large sample sets and long runtimes, which can be hard to achieve with conventional MC. We propose an approach called Reduced-Ordered Monte Carlo (ROMC), which improves simulation efficiency, while still producing accurate results. ROMC takes advantage of the (partial) redundancy inherent in digital signals. It prioritizes input signals based on their observability at the outputs, and combines inputs based on a compatibility property that enables them to share samples. Experimental results are presented which demonstrate that the ROMC methodology can decrease simulation runtime by several orders of magnitude.

Keywords

Logic simulation, Monte Carlo simulation, redundancy, sampling methods, signal probability.

1. Introduction

Monte Carlo (MC) simulation is a widely-used technique in electronic design automation (EDA) that serves several purposes. It is used as an evaluation method when analytical approaches are infeasible, and it serves to validate heuristic problem-solving techniques. For example, Sauer et al. present a SAT-based timing analysis technique whose results are "validated by comparison to an exact (Monte Carlo) simulation approach" [15]. Other EDA tasks commonly validated by MC include testability and power estimation [13], circuit synthesis [3], and design verification [18]. In some cases, MC is central to the main solution methodology. As a statistical technique, it is naturally suited to inherently uncertain applications such as probabilistic circuit simulation [11] and variability analysis. For example, Chen et al. [3] observe that "Monte Carlo simulation must be used if one wants to consider the impact of component variability and uncertainty when designing a circuit."

To be effective, MC must achieve a proper balance between computational effort and accuracy. Too few samples provide insufficient accuracy, while too many lead to excessive runtime. In fact, the sample size for MC grows rapidly with increasing accuracy levels [7]. To enhance the scalability of MC, we explore the use of signal redundancy to reduce simulation runtime complexity for a given accuracy target. We do so for a generic EDA application, namely, signal probability estimation in digital circuits. Informally, the *signal probability* (SP) of a logic signal s is the probability of s being 1, and is denoted by $p(s)$. SP carries information that can be used for a variety of purposes, ranging from power analysis [13] to reliability estimation [4].

Consider, for instance, the task of calculating the dynamic power consumption P_{dyn} of a gate [13][19]. This can be estimated from the equation $P_{dyn} = 1/2 C_L V^2 f \alpha$, where C_L, V, f and α are the gate's output capacitance, applied voltage, working frequency, and switching activity, respectively. Of these, the switching activity α is the most difficult to determine. Suppose, however, that all inputs are independent and G's output signal is s. We can then use the variance of $p(s)$, which is $p(s)(1 - p(s))$, to represent α, and hence calculate P_{dyn} [13] [19].

Signal probability is determined experimentally by applying N input vectors to an n-input circuit C, either physically or by simulation, and counting the number k of 1s produced on s, in which case, $p(s) = k/N$. This number is considered to be *exact* if it equals $p^*(s)$, where $p^*(s)$ is defined as the SP obtained when $N = 2^n$ and the applied vectors are all different, i.e., when simulation is exhaustive. Clearly, $p^*(s)$ is the fraction of 1s in the truth table for s. In Figure 1, $p(z) = 0.5$ since four of z's eight output values are 1. As explained later, $p^*(z)$ is also 0.5 for this circuit, so the implied simulation of z by sampling half its 16 input vectors is exact. In general, non-exhaustive circuit simulation is inexact, either because the input sample set is inadequate, or logical correlations exist among signals due to reconvergent fanout.

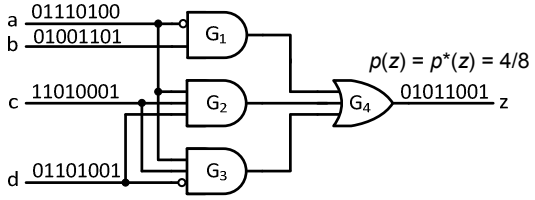

Figure 1. Circuit illustrating SP estimation via simulation.

A straightforward way to calculate SPs analytically was introduced by Parker and McCluskey for testability analysis [12]. Their method accounts for all correlations among logic signals and is exact. However, like exhaustive simulation, its complexity grows dramatically with circuit size, making it infeasible in practice. Various practical methods for approximating SP have been developed [4][5] some of which are based on simulation with MC-style statistical sampling. Recently, several interesting ways to improve sample accuracy have been proposed in the case of analog circuits [17]. For example, Singhee and Rutenbar apply

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA.

a quasi-Monte Carlo technique [17] to simulating process variation effects. However, their algorithms involve continuous functions and cannot be directly applied to digital circuits.

This paper proposes ways to exploit a circuit's redundancy properties to speed up SP estimation. To illustrate, consider again the circuit in Figure 1, which realizes $z(a,b,c,d) = a'b + ac$. Input d is *redundant* in the usual sense that z is independent of d, a fact that can be exploited to reduce sampling effort. While exhaustive simulation of a 4-input circuit requires 16 samples, the *full* redundancy of d reduces the circuit to a 3-input one that can be exhaustively simulated with only 8 samples. The sample input vectors shown apply all possible combinations to the non-redundant inputs a, b and c, so the simulated value of $p(z)$ is exact.

Of course, fully redundant inputs such as d in the above example rarely occur in practice. We address a new type of *partial redundancy* where independence of input variables is conditional on specific values applied to other variables. For instance, z in Figure 1 becomes independent of b and c, when a = 1 and 0, respectively. Hence, b and c can be considered partially redundant with respect to a. Most of a circuit's primary inputs (PIs) are partially redundant; however the degree of their redundancy varies widely. Some inputs become redundant much more often than others. We will show how such partial redundancies can be exploited to speed up simulation.

This work proposes an extension to the MC approach we call Reduced-Ordered Monte Carlo (ROMC), which has better performance than conventional MC. It incorporates two main techniques to improve the quality of circuit simulation. The first technique orders and prioritizes the PIs according to their *observability*. For instance, consider the problem of estimating the SP of z_1 in Figure 2. Some PIs such as x_{10} are more observable at z_1 than others, and some, such as x_2, are difficult to observe. Low-observability PIs tend to become partially redundant due to other PIs' assignments, while high-observability PIs are unlikely to become redundant. The concepts of observability and redundancy discussed here are related to the influence property of Boolean functions [9], and observability as used in the testing context [2]. We employ a modified version of the SCOAP testability measurement heuristic [2] to quickly find the most observable PIs.

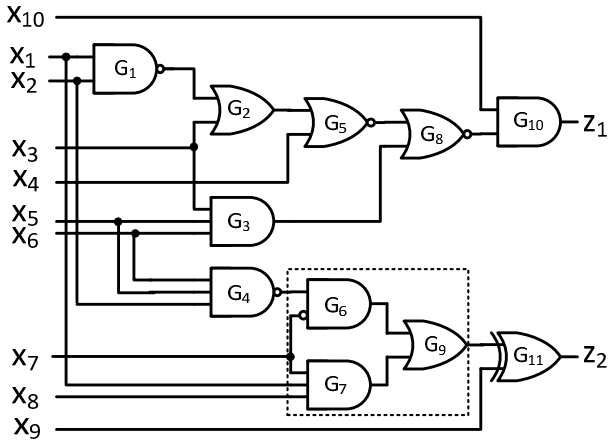

Figure 2. Ten-input two-output circuit; a multiplexer-like structure formed by G_6, G_7 and G_9 is highlighted.

The second technique involves finding *compatible* PIs, which have the property that only one of the PIs can affect certain primary outputs (POs) at any time. This relation among input variables can also be interpreted as a kind of partial redundancy. For example, x_4 and x_5 in Figure 2 are compatible PIs with respect to PO z_1. If x_3 = 1(0), then z_1 becomes independent of x_4 (x_5). So both x_4 and x_5 are partially redundant PIs associated with x_3, and they do not affect output z_1 at the same time. Such behavior resembles that of a multiplexer, whose control (select) signals ensure that only one data input D_i is connected to the multiplexer's output at a time; the others are effectively masked. The data inputs $\{D_i\}$ are viewed as compatible. More generally, we find compatibility relations among the inputs of multiplexer-like structures. One such structure is marked by dashed lines in Figure 2, where x_1 and x_8 are compatible with respect to x_2, x_5 and x_6. We will show that compatible PIs can share resources, such as input samples or randomness sources that drive the samples. This can lead to large efficiency gains in SP estimation.

Figure 3 illustrates the accuracy improvement from applying ROMC to the circuit in Figure 2. Here, accuracy is measured in terms of standard error [7], and refers to the difference between the exact probability $p^*(z)$ and the probability $p(z)$ estimated by the simulation for various sample sizes; the errors are averaged over the two outputs z_1 and z_2. For a given accuracy level, ROMC is significantly faster than MC because it achieves the required accuracy with fewer samples. Furthermore, in this case, ROMC can produce exact signal probabilities with only 32 samples.

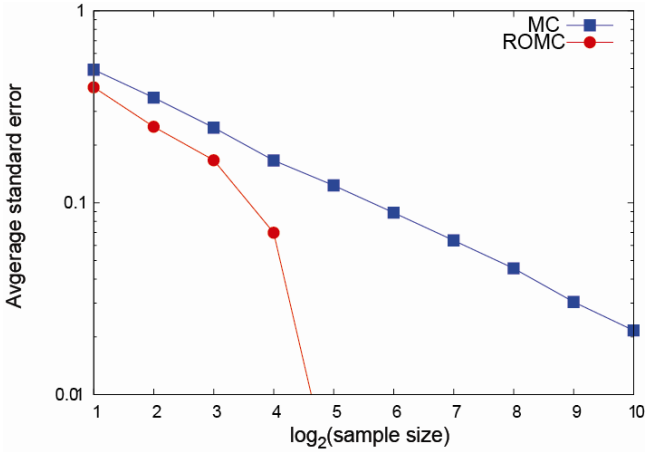

Figure 3. Accuracy comparison between ROMC and a typical MC calculation of the average output SP for the circuit in Figure 2.

The main contributions of this paper are: 1) Removing unneeded samples by identifying compatible PIs; 2) Improving sample accuracy by exploiting PI observability; and 3) Implementing and validating an MC algorithm for circuit simulation based on the foregoing concepts.

2. Sampling Concepts

This section reviews some basic concepts that are needed later. First, we formalize the SP estimation problem introduced in the previous section, and then discuss some basic properties of MC simulation. Lastly, we review the Boole-Shannon expansion.

We formulate SP computation in general terms, so that it applies to many different EDA situations. The *SP estimation problem* may be stated as follows. Given a combinational circuit C (including the pseudo-combinational equivalent of a sequential circuit) with n PI and m PO signals, as well as a set of N sample input sequences, calculate the (average) SPs of the POs. Figure 1 illustrates this for n = 4, m = 1, and N = 8. We make the usual

assumption [4] that each PI x_i has SP $p(x_i) = 0.5$ and is independent of all other PIs. This means that their joint probability distribution is the product of their individual distributions, and captures the informal notion of "random" sampling. It is possible to extend our methods to arbitrary input distributions, but that is beyond the scope of this paper.

Consider a n-input circuit C that implements the Boolean function $z(x_1,x_2,\ldots,x_n)$, where the PIs have the same SP $p(x_i) = 0.5$ and are all independent. The SP estimation problem for C is to determine $p^*(z)$. Since it is usually not feasible to find the exact value of $p^*(z)$ if C is large, we aim to find an *estimator* probability $p(z)$ that provides a good estimate of $p^*(z)$.

For a given estimator $p(z)$, a bias β and a variance $var[p(z)]$ are defined as follows [7]:

$$\beta = E[p(z) - p^*(z)] \quad (1)$$

$$var[p(z)] = E[(p(z) - p^*(z) - \beta)^2] \quad (2)$$

where $E[\boldsymbol{X}]$ denotes the expected value of $\boldsymbol{X}$. The square root of the variance of an estimator is its *standard error* (SE). It is usually desirable for an estimator to have $\beta = 0$, in which case it is called *unbiased*, and to have a small variance [7]. An unbiased estimator with variance zero is exact, that is, $p(z) = p^*(z)$.

MC is a method of estimating $p^*(z)$ by applying N uniform and independent random samples to the PIs and collecting N samples at the output z. Let $z(i)$ denote the i^{th} value of z associated with the i^{th} sample. The estimator $p(z)$ can be defined as

$$p(z) = (1/N)\textstyle\sum_i z(i)$$

It can be shown that $p(z)$ in this case is unbiased [7] and that the variance of the estimation is

$$var[p(z)] = (1/N)\, p^*(z)\,(1 - p^*(z)) \quad (3)$$

So the variance and the SE of an MC simulation can be reduced by increasing the sample size N. However, reducing the SE by a factor of k requires the sample size to be increased by factor of k^2 making it impractical if a the desired error level is low. Consequently, many techniques for variance reduction without increasing N have been proposed [7][14]. The method proposed in this paper can be seen as another variance-reduction technique.

Although the input samples for MC are usually assumed to be independent, which allows samples to be repeated, it is also possible to use MC with non-repeating samples [14]. We call the former MC with sample replacement (MCW) and the latter MC without sample replacement (MCWO). Like MCW, MCWO is an unbiased estimation method, but its variance follows that of a hypergeometric distribution [14]. For an MCWO estimator $p(z)$

$$var[p(z)] = (1/N)p^*(z)(1 - p^*(z))\,(2^n - N)/(2^n - 1)$$

For a fixed sample size N, if the number of PIs n in C is reduced by some number, then the variance will become smaller. The change can be significant if N is comparable to 2^n. This is not true in the case of MCW since its variance is independent of n. This fact turns out to be important in our proposed method since we aim at reducing the sample space of the MC, and can benefit from it only if we use non-repeating samples.

Finally, we summarize the Boole-Shannon expansion and provide a compact version for later use. An n-variable Boolean function $F(X)$ can be expanded around any variable x_i as follows [8]:

$$F(x_1,x_2,..,x_i,..,x_n) = x_i'.F_{xi'} + x_i.F_{xi}$$

where $F_{xi'} = F(x_1,x_2,..,0,..,x_n)$ and $F_{xi} = F(x_1, x_2,..,1,..,x_n)$ denote the negative and positive *cofactors* of F, respectively, for x_i.

For example, consider the function generated by G_8 in Figure 2. It can be expanded around x_3 thus:

$$G_8(x_1,x_2,x_3,x_4,x_5,x_6) = x_3'((x_1x_2)' + x_4) + x_3(x_5x_6)'$$

The negative and positive cofactors are $(x_1x_2)' + x_4$ and $(x_5x_6)'$, respectively. The variable x_4 (x_5) does not appear in the positive (negative) cofactor due to partial redundancy; the value of x_3 always blocks the propagation path from x_4 (x_5) to G_8. As this example suggests, Boole-Shannon expansions and cofactors provide a tool for describing signal compatibility.

We can also express signal probabilities in terms of cofactors:

$$p^*(F) = \tfrac{1}{2^k}\textstyle\sum_0^{2^k-1} p^*(F_i) \quad (4)$$

It is also useful to define the Boole-Shannon expansion with respect to a product of k variables $X_k \subseteq X$. In this case, we are dealing with cube cofactors [8] such as $F_{x1'x2'\ldots xk'}$ and $F_{x1x2\ldots xk}$. With this notation, cofactors are awkward to write, so we denote $F_{x1'x2'\ldots xk'}$ by F_0, $F_{x1'x2'\ldots xk}$ by F_1, and $F_{x1x2\ldots xk}$ by F_2^{k-1}. The cube expansion around X_k can then be expressed compactly as

$$F(X) = x_1'x_2'\ldots x_k'F_0 + x_1'x_2'\ldots x_kF_1 + \ldots + x_1x_2\ldots x_k\,F_2^{k-1} \quad (5)$$

3. Variable Ordering

As discussed in Section 1, fully redundant PIs can be exploited to reduce the sample size for simulation simply by not wasting samples on them. Similarly, the partially redundant, or less observable, PIs that become redundant frequently can be ignored in favor of more observable ones. (Of course, we cannot ignore them completely like fully redundant PIs.)

Accordingly, we propose the following sampling approach. Suppose we want to use $N = 2^k$ samples to estimate the SP $p(F)$ of function F with an n-member input set X. Select k variables $X_k = x_1,x_2,\ldots,x_k \subseteq X$ that are more observable than the others. Generate N samples that include every possible value of X_k exactly once, and assign random values to the remaining $n - k$ variables. The following samples illustrate this for $k = 3$ and $n = 9$.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
0	0	0	0	1	0	1	1	0
0	0	1	1	0	1	1	0	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	1	1	0	0
1	0	0	1	1	1	0	1	0
1	0	1	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1
1	1	1	1	1	0	1	0	0

Equation (5) enables us to express the above sampling scheme in the following way. Estimate the SP of each of the cofactors $F_0,F_1,\ldots,F_{N-1}$ with one sample and average the estimates. In practice, estimating the cofactors with one sample is sufficient. For instance, with a good variable ordering the cofactors will be close to 1 or 0, so a single sample provides a very good estimate.

$$p(F) = (1/N)(\, p(F_0) + p(F_1) + \cdots + p(F_{N-1})\,) \quad (6)$$

The average estimate $p(F)$ is unbiased. Its variance is therefore

$$var[p(F)] = (1/N^2)(var[p(F_0)] + var[p(F_1)] + \ldots + var[p(F_{N-1})]) \quad (7)$$

The variance of each cofactor estimate $p(F_i)$ is that of conventional MC with one sample, so

$$var[p(F_i)] = p^*(F_i)(1 - p^*(F_i))$$

Now consider two exemplary cases. First, suppose that $p^*(F_i) = 0.5$ for $i = 0, 1, 2,..., N - 1$; then $var[p(F_i)] = 1/4$. In this case, according to (7) we have $var[p(F)] = 0.25N$, which means that the proposed approach is no better than conventional MC. Next, suppose that $p^*(F_i) = 0$ for $i = 0, 1, 2,..., (N/2) - 1$ and $p^*(F_i) = 1$ for $i = N/2, (N/2) + 1, (N/2) + 2,... , N - 1$. This time we have $var[p(F_i)] = 0$, so $var[p(F)] = 0$ and the estimate is exact.

The difference between the two cases sketched above is that in the first one, $x_1, x_2,..., x_k$ have high redundancy and low observability at the output F. This is why the corresponding cofactor has probability 0.5. In the second case, however, $x_1, x_2,\ldots, x_k$ have high observability at F, and after assigning a value to them, completely determine the value of F.

Thus, for a given function F, we need to look for k variables that minimize $var[p(F)]$. This means we have to consider the possible choices, calculate the probability of the corresponding cofactors, and calculate $var[p(F)]$. If all choices must be considered, this problem is more difficult than exhaustive simulation, and hence is not feasible. We therefore use heuristic observability evaluation methods to find the most observable variables.

Consider the circuit in Figure 4. The output z has exact probability $p^*(z) = 0.5$. If we try to estimate this value with a 4-sample simulation, the MC method yields a variance of 1/16 = 0.063 according to (3). Now apply the proposed technique to this example. Since we have a total of four samples, we need to find the two most observable variables, which are obviously a and b. The variance $var[p(F)]$ of the estimation is given by (7) thus:

$$(1/16)(var[p(F_{a'b'})] + var[p(F_{a'b})] + var[p(F_{ab'})] + var[p(F_{ab})])$$
$$= (1/16)(\ var[p(cd)] + var[p(1)] + var[p(c'+d')] + var[p(0)])$$
$$= 6/256 = 0.023$$

which implies that this technique yields a better estimate.

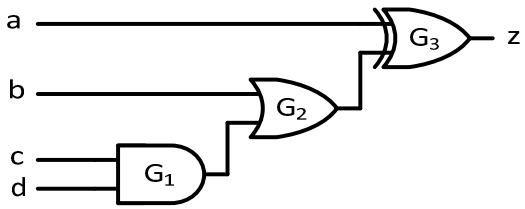

Figure 4. Variable ordering; *a* and *b* are more observable than *c* and *d*.

4. Sample Space Reduction

Next, we introduce a technique to shrink a circuit's sample space by converting it to another circuit with fewer PIs but the same output probabilities. As discussed in Section 2, this sample space reduction results in better SP estimates without the need to increase the sample size.

Consider again the function $z(a,b,c,d) = a'b + acd + acd'$ of Figure 1 The redundant PI d allows the sample space to be reduced from 16 to 8. Such redundant PIs can be detected by their Boolean difference. The *Boolean difference* of $z(x_1, x_2,\cdots, x_i, \cdots, x_n)$ with respect to input x_i is defined as

$$\frac{dz}{dx_i} = z(x_1, x_2,..,0,..,x_n) \oplus z(x_1, x_2,..,1,..,x_n)$$

where $\oplus$ denotes XOR. It follows immediately [1] that x_i is redundant if and only if

$$\frac{dz}{dx_i} = 0 \qquad (8)$$

Partially redundant PIs also reduce the sample size, and can be detected in a similar way. Consider again the partially redundant inputs in Figure 1. Input b (c) becomes redundant if $a = 1$ (0), so at least one of them is always redundant in the sense that b and c do not affect the output at the same time. Knowing this, we can further reduce z's sample space by tying b and c together to form a single PI for simulation purposes. This reduces the number of samples for exhaustive simulation from eight to four, namely, $abcd$ = 0000, 0110, 1000, 1110, while z's SP remains unchanged. So just by obtaining the output values for these few samples, we can determine the exact SP $p^*(z)$.

In general, two partially redundant inputs x_i and x_j of a function z that never affect z together have the following property:

$$\frac{dz}{dx_i}.\frac{dz}{dx_j} = 0 \qquad (9)$$

We call such inputs *compatible*. For example, inputs b and c of $z(a,b,c,d) = a'b + acd + acd'$ are compatible because

$$\frac{dz}{db}.\frac{dz}{dc} = a'.a = 0$$

Theorem: If two inputs x_i and x_j of an n-input function $f(x_1,x_2,.., x_i,..,x_j,..,x_n)$ are compatible, then the $(n - 1)$-input function $g = f(x_1,x_2,.., x_i,..,x_i,..,x_n)$ obtained by equating x_j and x_i in f has the same signal probability as f, that is, $p^*(g) = p^*(f)$.

Proof: Equation (9) implies that for all the possible values of the variables other than x_i and x_j, we have either $\frac{df}{dx_i} = 0$ or $\frac{df}{dx_j} = 0$. Hence, $f_{x'_i} = f_{x_i}$ or $f_{x'_j} = f_{x_j}$. Applying Boole-Shannon expansion to these equations, yields $f_{x'_ix_j} = f_{x_ix_j}$ and $f_{x'_ix'_j} = f_{x_ix'_j}$, or $f_{x_ix'_j} = f_{x_ix_j}$ and $f_{x'_ix'_j} = f_{x'_ix_j}$. Therefore,

$$p^*\left(f_{x_ix_j}\right) + p^*\left(f_{x'_ix'_j}\right) = p^*\left(f_{x_ix'_j}\right) + p^*\left(f_{x'_ix_j}\right)$$

Also, according to (4) we have:

$$p^*(f) = \frac{1}{4}\left[p^*\left(f_{x'_ix'_j}\right) + p^*\left(f_{x'_ix_j}\right) + p^*\left(f_{x_ix'_j}\right) + p^*(f_{x_ix_j})\right]$$
$$= \tfrac{1}{2}\left[p^*\left(f_{x'_ix'_j}\right) + p^*(f_{x_ix_j})\right]$$

Since g is the same as f with x_i and x_j connected, we can write $g_{x_i} = f_{x_ix_j}$ and $g_{x'_i} = f_{x'_ix'_j}$. Consequently,

$$p^*(g) = \tfrac{1}{2}[p^*(g_{x'_i}) + p^*(g_{x_i})] = \tfrac{1}{2}\left[p^*\left(f_{x'_ix'_j}\right) + p^*(f_{x_ix_j})\right] = p^*(f) \quad \square$$

So tying together a pair of compatible PIs yields a circuit with fewer inputs but the same SP. For example, tying the compatible PI pairs of Figure 2 yields the 5-input circuit of Figure 5 with the same SPs. More generally, an input set $\mathbb{X}$ of a function forms a *compatible set*, if all x_i-x_j pairs in $\mathbb{X}$ are compatible. All the members of $\mathbb{X}$ can be tied together without altering SP's. This is an instance of a general compatibility relation [8], and so is reflexive, symmetric but not necessarily transitive. It can be shown that the ten PIs of Figure 2 include 18 compatible pairs and four compatible triples, but no larger ones.

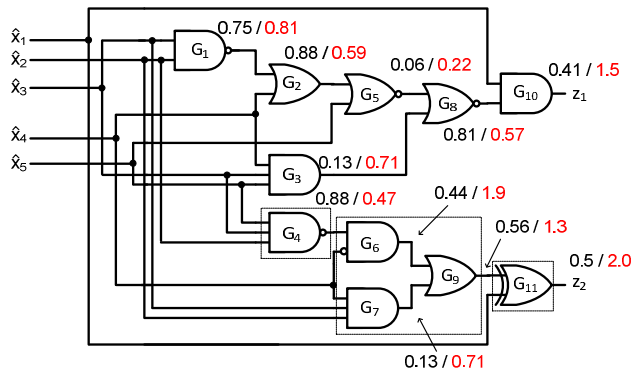

Figure 5. Five-input circuit obtained from Figure 2 by connecting compatible PIs; the supergates of z_2 are marked by dotted lines; *p*/Pdyn* denote the SP and dynamic power at each gate.

Determining compatible sets with special properties is a difficult task that arises often in EDA, for example, in state minimization for finite state machines, and resource scheduling for high-level synthesis [8][10]. These problems can be modeled by *compatibility graphs* (or their complements, *conflict graphs*). The problem of interest here is finding a minimal number of compatible sets to cover all PIs and minimize the relevant sample space, where the chosen sets must form a disjoint *cover* (partition) of the PIs. This problem is related to the clique partitioning problem in graph theory [10].

There is usually more than one optimum solution to our PI partitioning problem, and they all reduce the sample space equally. However, some partitions provide better simulation accuracy due to differences in their observability properties. For the circuit of Figure 2, ROMC selects the compatible sets $\{x_1, x_5\}, \{x_2, x_8\}, \{x_3, x_7\}, \{x_4, x_6\}, \{x_9, x_{10}\}$, which leads to a 5-input circuit with the same output SPs as the 10-input original; see Figure 5. Other nearly equivalent solutions exist such as: $\{x_1, x_5\}$, $\{x_2, x_6, x_8\}$, $\{x_3, x_7\}$, $\{x_4, x_9\}$, $\{x_{10}\}$.

5. Implementation Issues

This section presents our circuit sampling algorithm ROMC, which incorporates the variance and sample-space reduction techniques introduced in Sections 3 and 4. Figure 6 shows the pseudo-code for ROMC, as well as some of its main procedures.

Given an n-input m-output circuit C and a sample size $N = 2^k$, ROMC first finds an observability value for each PI via a modified SCOAP heuristic [2]. The modifications arise from differences between observability in the testing context and ROMC's. First, a fanout stem's SCOAP observability is the minimum value among its fanout branches; this is replaced by the average value of the branches' observability. For testing purposes, observing a signal at one PO is enough, but for ROMC, a good observable signal is one that is observable at many POs. A second modification is that SCOAP calculates the worst-case observability for both the 0 and 1 values of a signal, whereas ROMC computes the average of these values.

Next, ROMC determines compatibility relations among the PIs and constructs a PI compatibility graph. The **Compatible_Pair_Detection** procedure searches for multiplexer-like structures in order to detect compatible PIs. These structures are naturally confined to the given circuit C's *supergates* [16], which are subcircuits that encapsulate maximal fanout-reconvergence structures. The procedure partitions C into supergates and recursively searches them for compatible signals. For instance, there are three supergates for PO z_2 in Figure 5, namely: $SG(g_{11}) = \{g_{11}\}$, $SG(g_9) = (g_9, g_7, g_6)$, and $SG(g_4) = \{g_4\}$. $SG(g_9)$ is actually a multiplexer, so some of its inputs are compatible. These compatibility relations are propagated toward the PIs and lead to the conclusion that x_8 is compatible with x_5 and x_6.

After constructing the compatibility graph G for the original PI set X, ROMC performs clique partitioning on G to find a reduced set $\hat{X}$ of $\hat{n} \leq n$ PIs that form a PI cover. Equivalently, one could construct the PI's conflict graph $\bar{G}$, and solve the coloring problem for $\bar{G}$, i.e., find the minimum number of vertex colors such that all adjacent vertices have different colors [10]. There are many fast heuristic algorithms to solve this well-studied problem. Figure 7 shows the conflict graph of the circuit in Figure 2 and its coloring solution. The PIs (vertices) with the same color label are those that are tied together in Figure 5.

Next, ROMC sorts the PIs according to their observability. The observability value of each newly formed PI $\hat{x}_i$ is the sum of the observability values of the corresponding original PIs. For instance, in Figure 5, x_9 and x_{10} of Figure 2 are replaced by a new

```
ROMC(Circuit C, PIs X, Sample size N = 2^k) {
        // Returns estimated signal probabilities of the POs
  Observability Values X_obs. = Observability_Evaluation(C, X)
  Edge List E = Compatible_Pair_Detection(C, X)
  Graph G = G(X, E)   // G is the compatibility graph of C;
        vertices are PIs, edges connect compatible PIs
  Reduced PI Set X̂ = Clique_Partitioning(G)
  Observability Values X̂_obs = Add X_obs. of connected PIs in X̂
  Ordered List L = Sort PIs in X̂ by their observability X̂_obs.
  Sample Set S = All combinations of first k PIs in L, with
        random values assigned to remaining PIs
  return Monte_Carlo(C, S)  // Returns SPs generated by
        applying Monte-Carlo simulation to C with sample set S
}
Observability_Evaluation(Circuit C, PIs X){
        // Returns observability value of each PI
  return Modified_SCOAP(C, X)
}
Compatible_Pair_Detection (Circuit C, PI X){
        // Returns list of compatible PI pairs
  Supergate Partition SGP = Partition_to_Supergates(C)
        // Partitions C into fanout-free network of subcircuits
  if (SGP = C) then   // C cannot be further partitioned
      for each PI pair (x_i, x_j) in X
          if (x_i and x_j are compatible) then
              add (x_i, x_j) to list L
      return L
  else   // Recursive case
      for each subcircuit C' in SGP
          L_i' = Compatible_Pair_Detection(C', X')
              // Recursive call to supergate subcircuits
          for each pair (x_i, x_j) in L_i'
              propagate the relation to the PIs
              add compatible PIs to list L
              // Combine recursive results
  return L
}
```

Figure 6. Pseudo-code for the ROMC simulation algorithm.

PI $\hat{x}_1$, hence $\hat{x}_1$'s observability is the sum of those of x_9 and x_{10}. The new PIs are then ordered by observability; in the case of Figure 5, the ordering is $\hat{x}_1\ \hat{x}_4\ \hat{x}_3\ \hat{x}_2\ \hat{x}_5$.

At this point, ROMC generates samples based on its compatibility and observability figures. For the example (Figure 5), if the sample size $N = 8$, then PIs $\hat{x}_1$, $\hat{x}_4$ and $\hat{x}_3$ are chosen as the most observable. ROMC applies all 8 combinations 000, 001,...,111 to these three PIs, and assign random bit sequences to $\hat{x}_2$ and $\hat{x}_5$. It simulates the circuit with the resulting 8 samples.

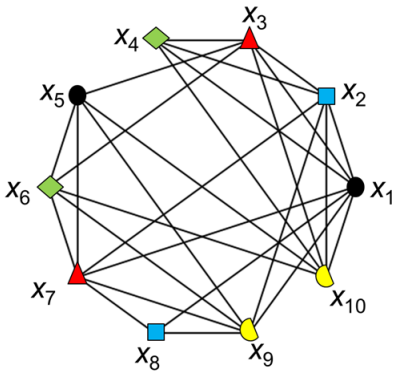

Figure 7. Conflict graph for the circuit in Figure 2; PIs with the same color labels are compatible and can share samples.

6. Experimental Results

To gauge the efficiency and accuracy of our approach, we applied ROMC to SP estimation for representative ISCAS-85 and LGSyn-92 benchmark circuits. Accuracy was measured in the following way: For an n-input m-output circuit C, reference (gold) signal probabilities were generated for all m POs using conventional MC. If $n \leq 31$, then C was exhaustively simulated; otherwise, C was simulated with 2^{31} random samples. This sample size 2^{31} produces results that accurate enough for verifying ROMC's performance considering the relatively small size of the benchmark circuits. Then, for a fixed sample size $N = 2^k$, ROMC and MC sample C 100 times. The accuracy for $N = 2^k$ is measured in terms of the average standard errors of the estimated results [7].

Each circuit was simulated with sample sizes ranging from 2^7 to 2^{24}. We found that ROMC can identify compatible PIs and determine each PI's observability quickly, even in circuits containing over 3,600 gates. The runtime overhead for compatible signal identification and observability estimation was less than 2 seconds on an Intel Quad-Core 2.35GHz, 64-bit PC with 4G RAM machine for all benchmarks.

Figure 8 shows the runtime improvements for the representative benchmark circuits. The improvement at each accuracy level is measured by the ratio between the MC and ROMC sample sizes needed to achieve the required accuracy. From the figure, we see that ROMC can reduce runtimes by one to three orders of magnitude. The average runtime improvement for an estimated error of 10^{-4} is nearly three orders of magnitude. In addition, these simulation results also show that ROMC's runtime improvement grows with increasing accuracy levels. In other words, ROMC can produce very accurate results with far fewer samples than MC. This suggests that ROMC is well-suited to applications that require highly accurate signal probabilities, such as power estimation [13].

We would like to emphasize that a variance reduction method might produce much worse results than the ones produced by MC if it is not designed carefully [7], and developing a variance reduction technique suitable for all kinds of circuits is a challenge. ROMC, however, produces no sample variance higher (worse) than the variances generated by MC, which suggests that it is a very broadly applicable method. As the results show, it is effective for various types of circuits such as the arithmetic and error detection circuits found in the ISCAS-85 and LGSyn-92 benchmark sets [6].

Figure 9 compares MC and ROMC for the c1196 circuit where the standard error of ROMC falls dramatically with increasing sample size. The big accuracy improvement from $N = 18$ to $N = 20$ results from a combination of the compatibility and MCWO features of ROMC. The sample space of c1196 is reduced from 2^{23} to 2^{21} due to the presence two compatible PI pairs. MCWO can reduce the sample variance exponentially when the sample size is close to that of the entire sample space, which is 2^{21} in this case.

Although ROMC does not produce negative results in any of the selected benchmarks, there are a few cases where ROMC makes insignificant runtime improvement. Figure 10 shows the runtime comparison for c499, which is an error-correction circuit [6]. The insignificant improvement is due to the fact that all PIs of such circuits tend to be equally important (of the same low redundancy) for all POs.

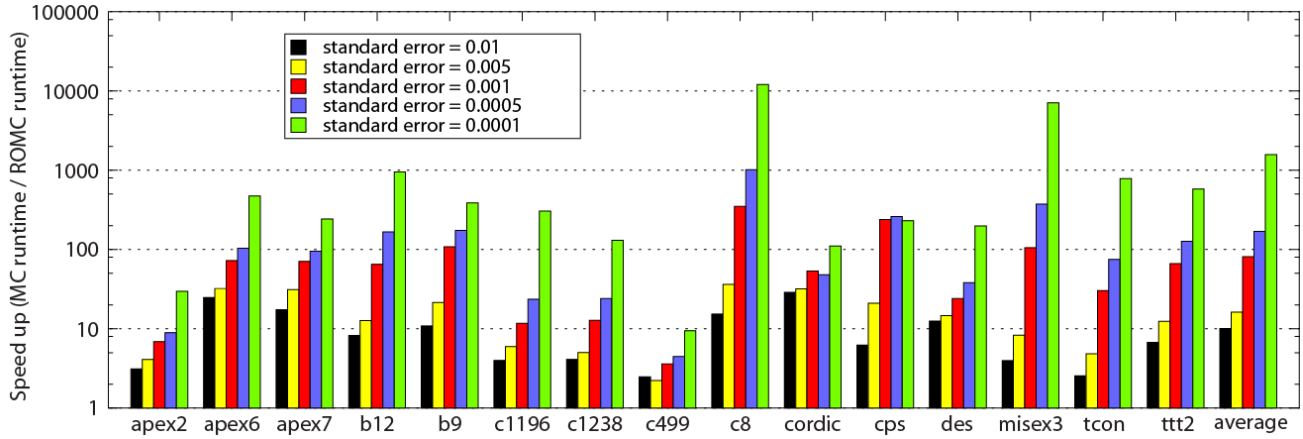

Figure 8. Speed-up of ROMC over MC for the benchmark circuits at five accuracy levels defined by standard error.

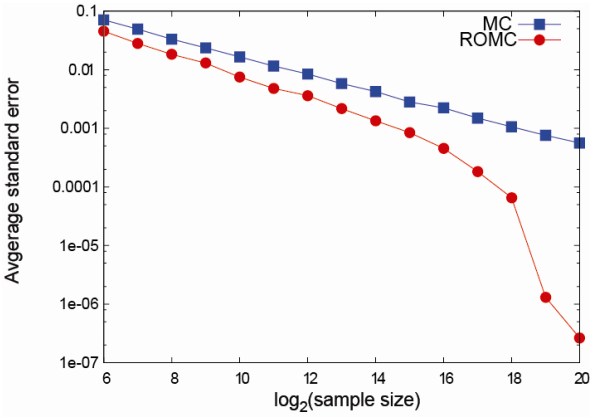

Figure 9. Comparison between MC and ROMC for c1196.

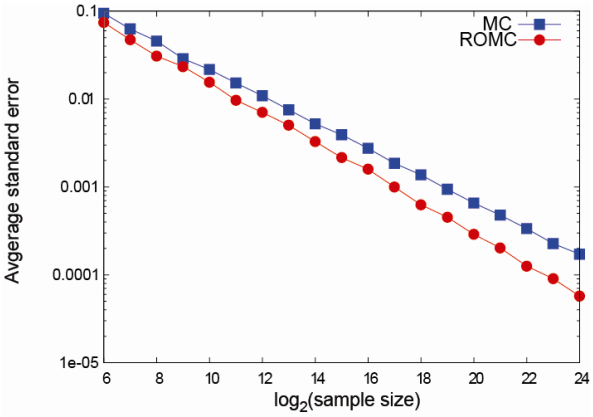

Figure 10. Comparison between MC and ROMC for c499.

Finally, we noted in Section 1 that power estimation is a useful application of SP data. To illustrate, consider again the circuit of Figure 5. By computing the SP of the intermediate lines, we can evaluate the circuit's overall switching activity, and hence its power consumption, using the relation

$$P_{\text{dyn}} = 1/2 C_{\text{L}} V^2 f . p(s)(1 - p(s))$$

In Figure 5, gate output lines are marked with p^*/P_{dyn}, where p^* is the line's exact SP and P_{dyn} is the gate's dynamic power dissipation in microwatts, assuming an IBM 130-nm process with a 1.5 volt supply and a 1 GHz clock frequency. It is worth noting that ROMC estimates the power values with less than 1% error via only 16 samples, while MC has a 12% error with the same number of samples.

7. Conclusions

We have proposed an MC-based algorithm ROMC for digital circuit simulation, which aims to reduce simulation runtime and/or increase simulation accuracy. The simulation goal is SP calculation, which has many EDA applications. ROMC exploits two redundancy properties of logic signals (compatibility and observability) in novel ways to remove unnecessary samples and reduce sample variance. Hence, it is always faster—sometimes dramatically so—than conventional MC for a given accuracy level.

We have also presented experimental results showing that the proposed approach reduces simulation time by as much as three orders of magnitude on benchmark circuits. In addition, it can reduce SP estimation errors for a given runtime budget. These two nice properties make ROMC particularly suitable for probability-related EDA tasks such as switching activity measurement and soft-error analysis, topics we are continuing to explore.

Acknowledgement: This work was supported by Grant CCF-1017142 from the U.S. National Science Foundation.

References

[1] S. B. Akers, Jr., 1959. On a theory of Boolean functions. *Jour. SIAM*, pp. 487-498.

[2] M. L. Bushnell and V.D. Agrawal, 2000. *Essentials of Electron. Testing*. Springer.

[3] C-H. Chen et al., 2001. Efficient approach for Monte Carlo simulation experiments and its applications to circuit systems design. *Proc. Simulation Symp.,* pp. 65-71.

[4] M. R. Choudhury and K. Mohanram, 2007. Accurate and scalable reliability analysis of logic circuits. *Proc. DATE*, pp. 1-6.

[5] S. Ercolani el al., 1989. Estimate of signal probability in combinational logic networks. *Proc. ETC*, pp.132-138.

[6] M. C. Hansen et al., 1999. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *IEEE Design & Test*, vol. 16, no. 3, pp.72-80.

[7] J. M. Hammersley and D. C. Handscomb, 1964. *Monte Carlo Methods*. London: Methuen.

[8] G. D. Hachtel and F. Somenzi, 1996. *Logic Synthesis and Verification* Algorithms. Kluwer Academic Publishers.

[9] J. Kahn et al., 1988. The influence of variables on Boolean functions. *Proc. FOCS,* pp. 68-80.

[10] G. De Micheli, 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill.

[11] A. Paler et al., 2011. Tomographic testing and validation of probabilistic circuits. *Proc. ETS*, 63-68.

[12] K. P. Parker and E. J. McCluskey, 1975. Probabilistic treatment of general combinational networks. *IEEE Trans. Computers*, pp. 668-670.

[13] M. Pedram, 1994. Power estimation and optimization at the logic level. *Jour. High Speed Electron. & Syst*. pp. 179-202.

[14] S. M. Ross, 1990. *A Course in Simulation*. Prentice Hall.

[15] M. Sauer et al., 2011. Estimation of component criticality in early design steps. *Proc. IOLTS,* pp. 104-110.

[16] S. C. Seth and V. D. Agrawal, 1989. A new model for computation of probabilistic testability in combinational circuits. *Jour. VLSI Integration*, pp. 49-75.

[17] A. Singhee and R. A. Rutenbar, 2010. Why Quasi-Monte Carlo is better than Monte Carlo or Latin Hypercube sampling for statistical circuit analysis. *IEEE Trans. CAD*, pp. 1763-1776.

[18] S. Tasiran et al., 2001. A functional validation technique: biased-random simulation guided by observability-based coverage. *Proc. ICCAD*, pp. 82–88.

[19] Q. Wu et al., 1997. A note on the relationship between signal probability and switching activity. *Proc. DAC*, pp. 117-120.