Fast Statistical Timing Analysis By Probabilistic Event Propagation*

Jing-Jia Liou, Kwang-Ting Cheng, Sandip Kundu†, and Angela Krstić
Electrical and Computer Engineering Department,
University of California, Santa Barbara
†Intel Corporation, Austin

ABSTRACT

We propose a new statistical timing analysis algorithm, which produces arrival-time random variables for all internal signals and primary outputs for cell-based designs with all cell delays modeled as random variables. Our algorithm propagates probabilistic timing events through the circuit and obtains final probabilistic events (distributions) at all nodes. The new algorithm is deterministic and flexible in controlling run time and accuracy. However, the algorithm has exponential time complexity for circuits with reconvergent fanouts. In order to solve this problem, we further propose a fast approximate algorithm. Experiments show that this approximate algorithm speeds up the statistical timing analysis by at least an order of magnitude and produces results with small errors when compared with Monte Carlo methods.

1. INTRODUCTION

Process variations, manufacturing defects and noise are major factors in determining the timing characteristics of deep sub-micron designs. Process variations often result in a wide range of possible device parameters, making circuit performance hard to estimate. Delay faults caused by interconnect defects and noise sources are also unpredictable in terms of size of induced delay. All these factors are statistical in nature and are best modeled using statistical models. Therefore, the use of statistical methods for timing analysis to incorporate statistical timing deviations caused by these sources seems to be inevitable.

For statistical timing analysis, the delays of cells/interconnects are modeled as correlated random variables with known probability density functions (pdf's). Given these cell/interconnect delays, the cell level netlist and the clock period, statistical timing analysis can derive the probability density functions of the signal arrival times at internal signals and primary outputs. For large designs with a large number of delay random variables, determining closed forms for the probability density functions of the arrival times at the primary outputs is computationally expensive and impractical. Therefore, the popular Monte Carlo based technique is often used to approximate the probability density functions of the signal arrival times at the internal signals and primary outputs. Each run of Monte Carlo simulation consists of two steps: sampling and analysis. In the sampling step, a single value is chosen from each random variable according to the law of probability. The analysis step utilizes these sampled values to derive the arrival times at all signals for the given circuit instance. The stop (convergence) criteria are decided based on the desired accuracy of results or the confidence level. Once the mean or variance converges within the desired precision range, the procedure terminates. The main drawback of the Monte Carlo based method is that a large number of runs is required to achieve a high confidence level. Also, a large number of these runs concentrates on values near the nominal value. Statistical methods for timing analysis have been proposed in [1, 2, 3]. However, due to their high computational complexity, these methods are rarely used in practice. A more practical Monte Carlo-based statistical timing analysis framework applicable to larger designs has been proposed in [4]. This paper also considers the effects of output capacitance loads and input transition times to improve the accuracy.

In this paper, we propose a new, fast statistical timing analysis algorithm. The algorithm targets cell-based designs and all cell delays are modeled using random variables. The goal is to produce arrival-time random variables for all internal signals and primary outputs. The algorithm can be applied for vectorless static analysis as well as for dynamic simulation with given input vectors. The flow of the algorithm is similar to the compiled-code simulation, where each cell is evaluated after all the values at its fanins have become available. At the beginning, the simulation queue contains only the initial events at primary inputs. After that, the algorithm enters a loop in which cells are processed in a levelized order to produce the arrival-time random variables at their outputs. The process continues until all the cells have been evaluated.

The most important characteristic of the new algorithm is that it is deterministic. The final results produced by the algorithm can be determined completely by inputs , i.e., the same inputs will produce the same results, as opposed to the random process used by Monte Carlo methods. Another special feature of the algorithm is that it uses discrete delay random variables to model cell delays. Therefore, it is possible to control the behavior of the algorithm by controlling the discretization of pdf's of random variables. The smaller the number of samples of discrete random variables is, the less accurate the results are and the faster the algorithm runs. We will describe each of these features in detail in later sections. Applying the algorithm to circuits with reconvergent fanouts could result in exponential time complexity. Therefore, we propose a fast approximate algorithm for these circuits. Experiments show that this approximate algorithm speeds up the process by at least an order of magnitude and produces results with small errors when compared with Monte Carlo methods.

2. SIGNAL ARRIVAL TIME EVALUATION FOR A CELL USING PROBABILISTIC EVENTS

The new algorithm takes a cell-level netlist and the pin-to-pin and wire cell delays (as random variables) as inputs and produces

*This work was supported in part by the MARCO/DARPA Gigascale Silicon Research Center (http://www.gigascale.org) and NSF Grant CCR-9901099. Their support is gratefully acknowledged.

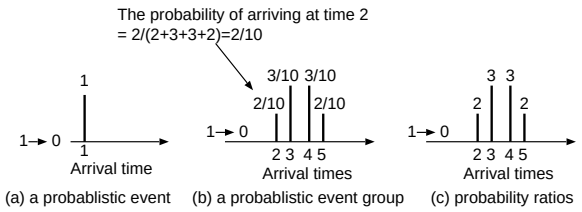

Figure 1: Probabilistic events.

signal arrival times for every node and wire. The basic operation is processing of an individual cell. This requires evaluating the probabilistic events at the output of the cell given the events at the cell's inputs. Starting from a description of the probabilistic events, the following sections will explain how to obtain signal arrival times at the output of a cell.

2.1 Probabilistic events

A *probabilistic event*, which is described by a triple (s,t,p), is a signal s scheduled with an arrival time t and the probability p that the signal will arrive at this time. Figure 1 illustrates the concept of the probabilistic events. Figure 1(a) shows a single probabilistic event for a signal which is scheduled at time 1 and has a probability of 1, i.e., this is a deterministic event. A signal could have more than one event associated with it. All the events at the same signal together form an *event group*. Figure 1(b) shows an event group with four events. The signal's arrival time has 20% of probability to be 2, 30% probability to be 3, etc. The sum of the probabilities in an event group has to be 1. In the rest of the paper for simplicity, the events will be indicated by the numerators of their probability fractions (which are integers) rather than the real probability values (Figure 1(c)). We denote such integers as probability ratios.

In statistical timing analysis, the cell delays are random variables. We apply the "fixed time unit" concept to discretize the delay random variables which can tremendously reduce the complexity of statistical timing analysis. Based on a chosen time unit, all pdf's of random variables are discretized and represented in discrete forms, in which any two adjacent delay data points are spaced by the chosen time unit. This discretization process is described in the next section.

2.2 Discretization of delay random variables

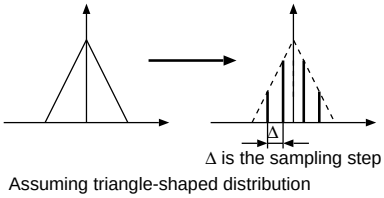

Figure 2: Discretization of random variable.

The discretization of delay random variables is used to generate discrete probability distributions based on a time unit (also called a sampling step). The sampling step is a user specified fixed time unit to be used for discretizing all random variables. The sampling step is also used as the time unit for the signal arrival time evaluations during simulation or timing analysis. Figure 2 illustrates the discretization process with a sampling step Δ for a random variable having a triangle-shaped distribution. A smaller sampling step will result in more data points in the discrete distribution. Therefore, the sampling step controls the resolution and the run time of the algorithm.

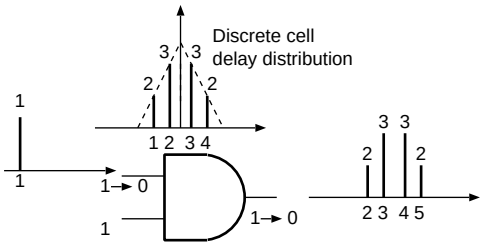

Figure 3: Propagating a single event.

2.3 Signal Arrival Time Evaluation for A Cell

In this section, we describe the propagation process for the probabilistic events and the signal arrival time evaluation for a cell. We start by describing propagation of a single event. Then, we describe the propagation of an event group and finally, propagation of multiple event groups.

Propagating a single event. Propagation of probabilistic events through the circuit can best be illustrated by examples. Figure 3 shows the case of propagating a single event through an AND gate. In this example, the event is a falling transition arriving at time $t = 1$ to the input of the gate. The discrete random variable of the delay for the AND gate is also shown above the gate in the figure. The final events at the output of the AND gate are obtained by shifting the cell delay by one time unit since the deterministic input event arrives at time $t = 1$. Since this is a deterministic event propagation (the probability is 1), the four events in the event group at the output of the AND gate will have the same probability values as the corresponding events in the discrete distribution of the cell delay.

Propagating an event group. Propagating an event group through a cell requires two operations: shift with scaling and group. *Shift with scaling* shifts the cell delay according to each input event and scales the cell delay probability distribution by multiplying it with the probability ratio assigned to this event. The shift with scaling operation is illustrated in Figure 4. It results in 16 events at the output of the AND gate. *Group* operation adds the probabilities of events at the same arrival time and forms a single event for each arrival time. We use "+" sign to denote a group operation in the following discussion. After grouping the events at the output of the AND gate in Figure 4, the number of events in the event group is reduced from 16 to 7. Note that we can use these two propagation rules to propagate arrival time events through wires/interconnects

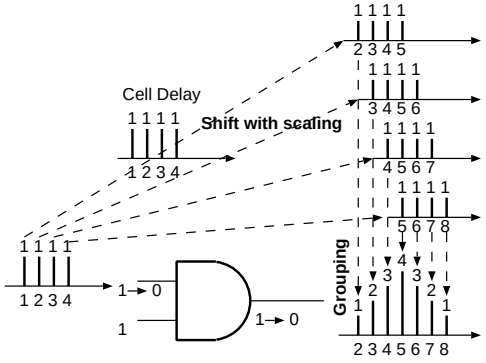

Figure 4: Propagating one event group.

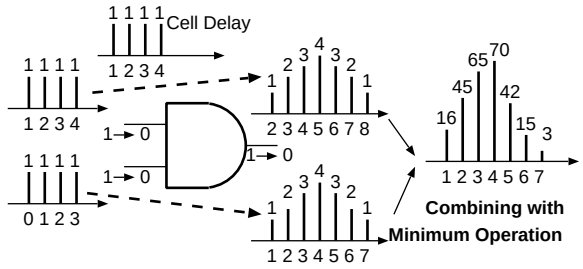

Figure 5: Propagating two groups.

whose delays are also given as random variables.

Propagating multiple event groups. When two or more event groups appear at the inputs of a cell, we first generate the output events for each fanin signal using shift with scaling and group operations. Then, we combine the output events into a single event group at the output. To combine the output events from different inputs, we use either a *minimum*(min) or *maximum*(max) operation depending on the transition type (rising or falling) and the cell type. Figure 5 illustrates the process of combining the events at the output of an AND gate. In this case, we use the *minimum* operation since the output of the AND gate has a falling transition and the earliest event should dominate the result (the dominating events define the final transition of the signal at the output if there are multiple transition types at the inputs). The *minimum* operation compares all possible pairs of events at the output and produces the earliest arriving events. The probability of the event at the output is the product of the two probabilities associated with the pair of events which are compared to form the final event. For example, the event arriving at time $t = 1$ in the lower group shown at the output of the AND gate in Figure 5 is compared with all the events in the upper group. Since this event dominates all other events (its arrival time is earlier than all events in the upper group), the final event at time $t = 1$ will have a probability ratio 16=1+2+3+4+3+2+1. Next, the two events arriving at time $t = 2$ are combined together with a probability ratio $45 = 1\times(3+4+3+2+1)+2\times(2+3+4+3+2+1)+1\times 2$. The term $1\times(3+4+3+2+1)$ represents the probability ratio of the output when the event at time 2 in the upper group (with a probability ratio 1) dominates (i.e., its arrival time is earlier than events at the lower input). The number in parenthesis $(3+4+3+2+1)$ reflects the probability of this assumption being true (by counting the probabilities of the events in the lower group whose arrival times are later than 2). The term $2\times(2+3+4+3+2+1)$ is obtained by assuming the event at time 2 in the lower group dominates. The last item 1×2 represents the output event obtained by assuming the events at both groups arrive at the same time (time 2). The process continues until all the events have been processed. The maximum operation in done in a similar way.

3. STATISTICAL TIMING ANALYSIS ALGORITHM

The previously discussed evaluation process for a single cell can be extended to handle circuits of tree-like structure by levelized simulation, where each cell is evaluated after all its fanin cells have been evaluated. However, for circuits with reconvergent fanouts, events propagated from the same fanout stem will converge at the same cell. When converging, the events at different inputs of the cell are not independent. Therefore, they require a special handling during the combining process (different from the *minimum* or *maximum* operation). In the following sections, we will give details of the signal reconvergency problem and the solution by circuit partitioning.

3.1 Signal Reconvergency and Supergates

If a circuit has reconvergent fanouts, then propagating the event group at the stem forwards to its fanout cone using a *minimum* or *maximum* operation at each cell would result in mixing unrelated events. To illustrate, consider the circuit in Figure 6. Consider gate $SG1$ with inputs a and b and assume there are two events ($e1$ and $e2$) at stem $S1$. Let $e1$ produce an event group $E1_a$ at a and event group $E1_b$ at b. Similarly, $e2$ produces $E2_a$ and $E2_b$. The correct events at $SG1$ should be $\max(E1_a, E1_b) + \max(E2_a, E2_b)$. However, simply propagating $e1 + e2$ would lead to an incorrect result, $\max(E1_a + E2_a, E1_b + E2_b)$. In the latter equation, $E1_a$ should not be compared with $E2_b$ because they are produced by two events $e1$ and $e2$ which do not happen at the same time.

To solve the problem, we propose a circuit partitioning algorithm with a sampling technique for events at fanout stems. We first simplify the problem by partitioning the circuit into a set of supergates [5]. A *supergate* is a single-output sub-circuit with all inputs being independent from each other. Therefore, to obtain the arrival time for the output cell of a supergate, it is sufficient to solve the problem on the subcircuit defined by the supergate. In other words, to obtain the signal arrival times for a cell, we have to first detect if the current cell is a reconvergent gate (i.e., the output cell of a supergate) and then derive the arrival time for the supergate using the algorithm described in the next section. For the example circuit in Figure 6, there are two supergates, $SG1$ and $SG2$, in the circuit. $SG1$ is defined as the intersection of fanin cone of $SG1$ and fanout cones of $S1$ and $S2$. Inside the region defined by the supergate $SG1$, there are two other stems: $S3$ and $S4$. Therefore, four fanout stems $S1$, $S2$, $S3$ and $S4$ are contained inside the supergate $SG1$. Likewise, supergate $SG2$ contains three stems $S2$, $S3$ and $S4$. Please note that supergates could overlap with each other (e.g., $SG1$ overlaps with $SG2$).

3.2 The Exact Algorithm

To derive the events at the output of a supergate, several techniques are needed. Next, we illustrate these techniques using first a supergate with only one stem, two stems, and finally a general case of multiple stems.

Sampling-evaluation process for a stem. For a supergate with only one stem, we process the events at the stem one-by-one. Each time we take only one event from the group of events at the stem (*sampling*) and propagate it forward to the output of the supergate. Next, since there is no reconvergent problem, events produced by this single event at fanins of the reconvergent gate are combined by applying a *minimum* or *maximum* operation. Finally, the probabilities of the combined event group are scaled with the probability of the sampled stem event. This *sampling-evaluation* process is repeated for each event at the stem and the resulting event group is continuously accumulated by applying the *group* operation. When all events at the stem have been processed, the accumulated event group represents the signal arrival time of the supergate.

Sampling-evaluation process for two stems. In this case, there are two possible configurations: (1) no stem is in the fanout cone of the other stem ($S1$ and $S2$ in Figure 6 for $SG1$), and (2) one stem is in the fanout cone of the other stem ($S2$ and $S3$ in Figure 6 for $SG1$).

For the first case, the sampling-evaluation process is similar to the single stem case. The only difference is that the process starts with a sampled event pair. The sampled event pair is formed by taking one event from each event group. For a complete sampling-

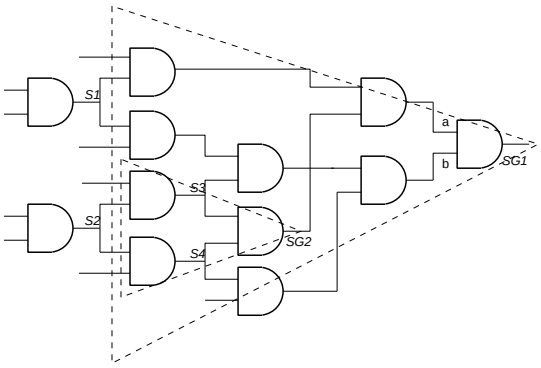

Figure 6: Multiple stems in supergates.

evaluation process, all possible pairs of events at the two stems should be considered. For example, assume there are two initial events at $S1$ ($e1_{S1}, e2_{S1}$) and also two events at $S2$ ($e1_{S2}, e2_{S2}$). Therefore, there are four possible pairs: ($e1_{S1}, e1_{S2}$), ($e1_{S1}, e2_{S2}$), ($e2_{S1}, e1_{S2}$) and ($e2_{S1}, e2_{S2}$) and the sampling-evaluation process will be done for each pair. At the end of each process, the probabilities of the resulting event group are scaled with the probability of the sampled event pair, which is the product of probabilities of two sampled events. We use the term *cross-product sampling-evaluation* to name this process.

For the second case, when one stem is in the fanout cone of the other stem ($S3$ is in the fanout cone of $S2$ in Figure 6), the sampling-evaluation process starts from the stem closer to the primary inputs (i.e., stem $S2$). First, one event at $S2$ is sampled, evaluated and propagated until $S3$ is reached. Then, the event group of $S3$ is also sampled and propagated to the output of the supergate repeatedly until all events at $S3$ are processed. After the simulation of events at $S3$ is done, the remaining events at $S2$ are re-visited for another round of sampling-evaluation phase. This process continues until there is no event left un-processed at $S2$. Each time events reach $SG1$, the probabilities of these events are scaled with probabilities of the two sampled events of $S2$ and $S3$, and accumulated at temporary storage at $SG1$. At the end of the process, final results are obtained at $SG1$. This recursion-like process is named as *recursive sampling-evaluation.*

Sampling-evaluation process for a general case. For a supergate with more than two stems, the sampling-evaluation should be further generalized. The evaluation sequence is the combination of two processes: cross-product and recursive sampling-evaluations. All stems of a supergate are first levelized according to their evaluation dependency, i.e., we check if one stem is in the fanout cone of the other stem. Stems are put in the same level if they do not depend on each other in the sampling-evaluation process. A cross-product sampling-evaluation is applied to these stems with a sampled event tuple formed by taking one event from each event group. Whenever another level of stems are reached by a previous sampling-evaluation process, a new cross-product sampling-evaluation phase starts for the new level. The process continues until all possible pairs of events are evaluated for the current level. Then, the sampling-evaluation returns to the previous level (recursive sampling-evaluation). For example, for the four stems $S1$, $S2$, $S3$ and $S4$ of supergate $SG1$ in Figure 6, two stems $S1$ and $S2$ are in the first level and the other two stems are in the next level. Assume there is a total of three initial events: one at $S1$ ($e1_{S1}$) and two at $S2$ ($e1_{S2}, e2_{S2}$). Also assume that $e1_{S2}$ produces two events, $e1_{S3,e1_{S2}}$ and $e2_{S3,e1_{S2}}$, at $S3$ and similarly another two events, $e1_{S4,e1_{S2}}$ and $e2_{S4,e1_{S2}}$, at $S4$. The complete sequence of computations to obtain the event group representing the signal arrival times at $SG1$ is:

1. cross-product sampling-evaluate for $S1$ and $S2$ with ($e1_{S1}, e1_{S2}$)
2. cross-product sampling-evaluate for $S3$ and $S4$ with ($e1_{S3,e1_{S2}}, e1_{S4,e1_{S2}}$); accumulate the event group at $SG1$
3. cross-product sampling-evaluate for $S3$ and $S4$ with ($e1_{S3,e1_{S2}}, e2_{S4,e1_{S2}}$); accumulate the event group at $SG1$
4. cross-product sampling-evaluate for $S3$ and $S4$ with ($e2_{S3,e1_{S2}}, e1_{S4,e1_{S2}}$); accumulate the event group at $SG1$
5. cross-product sampling-evaluate for $S3$ and $S4$ with ($e2_{S3,e1_{S2}}, e2_{S4,e1_{S2}}$); accumulate the event group at $SG1$
6. repeat steps 1-5 with replacing $e1_{S2}$ by $e2_{S2}$

As it can be seen, the time complexity increases rapidly with increasing number of stems. The estimated run time is proportional to $O(N_e^{N_s})$ (N_e is the number of events and N_s is the number of stems), which is apparently not feasible for practical applications. In order to reduce the time complexity, we propose an approximate algorithm in the next section.

3.3 An Approximate Algorithm

The approximate algorithm combines several techniques to jointly improve the run time. They are described in the following several paragraphs.

Dropping low probability events. In the process of event propagation, it is possible to produce events with very low probabilities. These events will only produce events with even lower probabilities. Therefore, it is desirable to set a minimum probability to screen out these events as early as possible. The events with probabilities lower than the set minimum probability are dropped from the event group whenever they are propagated to the output of a cell.

Filtering out unnecessary stems. Although some stems produce reconvergent events, the arrival times of the events caused by them are so early that they will never affect the arrival time at the output of the supergate. By some simple analysis, we can identify such stems and eliminate them from the sampling-evaluation process and thus speed up the algorithm. Our method for identifying such stems is through the use of the simple event group propagation (Section 2.3) for each stem while assuming there is no event group at other stems. In this way, the range of arrival times of events (generated by the stem under consideration) at the output of the supergate can be estimated with low computing resources. A stem is removed from consideration in the sampling-evaluation process if the estimated range of the arrival time at the supergate output caused by events on this stem does not overlap with the range of the arrival times at the supergate caused by the events from other stems.

Choosing effective stems. Stems do not produce equally significant reconvergent events. Therefore, we can find the more important stems using the results obtained by sampling-evaluating each stem. The method compares the results of the sampling-evaluation process for each stem with those without considering any stems. Thus we can estimate how sensitive the signal reconvergency is to an event group produced by a stem. We propose to choose one or two most sensitive stems for each supergate to estimate the final event group at the output of the supergate (single-stem or two-stem estimation). This single-stem or two-stem technique is the most effective method to improve the efficiency of the algorithm with minimum loss of accuracy.

Limiting the circuit depth of supergates. The size of a supergate is an important factor in the run time of the algorithm. This is

because there is generally fewer stems in a supergate with a smaller number of gates and it takes less time to apply sampling-evaluation for each stem. This observation motivates the concept of limiting the circuit depth of supergates, i.e., limiting the number of logic levels between stems and the output of the supergate when building the supergate structure. In this way, the size of supergates can be reduced. However, if we limit the logic level of a supergate, the inputs of the limited-level supergate will no longer be independent. Thus, the results will no longer be accurate. However, the effects caused by signal correlations are weaker if the reconvergent gate is farther from the stem source (the distance is measured by the number of logic levels between the source and the reconvergent gate) [6]. Therefore, the error caused by this heuristic can be minimized if the limit of supergate depth is not too small (say, larger than 10). In the next section, we will present results of experiments conducted in order to observe how accuracy changes by varying the depth of supergates.

4. EXPERIMENTAL RESULTS

In the following, we demonstrate that by using the approximate techniques discussed above it is possible to speed up the arrival time estimation process by at least an order of magnitude and at the same time maintain small error percentages as compared with a Monte Carlo-based static timing analyzer. All experiments utilize the techniques for "filtering unnecessary stems" and "single-stem estimation".

First, we demonstrate the effectiveness of the "dropping low probability events" heuristic. Next, we show that by varying the number of data samples of each random variable, it is possible to select an optimal parameter for discretizing random variables to balance between run time and accuracy. Similarly, it is also possible to find the optimal logic depth limit for constructing the supergates. Since all these techniques are orthogonal to each other, we can apply all of them for a fast statistical timing analysis.

We use the combinational parts of ISCAS89 benchmark circuits for our experiments. These circuits were first optimized for performance by Synopsys Design Compiler [7]. The means of all cell delays are assumed to be a function of the number of inputs/outputs of the cells. The standard deviation (σ) is in the range of (4%, 10%) of the mean (the value of σ is fixed for each cell).

In the following experiments, a Monte Carlo process for traditional static timing analysis with 10^5 runs is used as a comparison target. The number of runs, 10^5, is selected to balance the accuracy and the run time of the Monte Carlo method. The error percentage of the sample mean obtained by Monte Carlo methods is bounded by $c * s/(\sqrt{n} * m)$ [8], where c is the solution of the equation, $T(c) = (1+\gamma)/2$ ($T(c)$ is the *Student t* distribution with a parameter c and γ is the confidence level), s^2 is the sample variance, m is the sample mean and n is the number of samples (runs). Therefore, we can estimate the error percentage with 10^5 runs as bounded by $c * s/(\sqrt{n} * m) = 3.0 * 0.10/\sqrt{10^5} = 0.095\%$ with $c = 3.0$, $\gamma = 0.99$ and $s/m = 0.10$.

Before comparing the new algorithm with the Monte Carlo method, we need to select three parameters for the approximate techniques: the minimum probability of events (P_m) for filtering, the number of data samples for discretizing cell delay random variables (N_s) and the depth limit of supergates (D). Note that only D is treated as a circuit-dependent parameter, while the other two parameters are kept the same for all circuits.

Figure 7 shows the effects of dropping low probability events on the error percentages for arrival time mean and variance and run time for circuit s15850, which is chosen for demonstration because the size of the circuit is appropriate for experimenting various configurations and it actually has the worst performance among the tested benchmarks (Table 1). The error percentages are obtained by comparing the results of using different P_m for filtering against the results obtained without dropping low probability events. As it can be seen, with the increasing P_m, the errors for the mean and variance increase and the run time decreases. Using the plot in Figure 7, we choose $P_m = 10^{-5}$ which gives reasonably low errors (0.114% for mean and 3.24% for variance) while it has a fast run time (174). Note that all error percentages used in this paper are $|M_e| + 3 * \sigma_e$, where M_e and ${\sigma_e}^2$ are the mean and the variance of error percentages of signal arrival times of all signal nodes in the circuit. This error percentage bound can cover more than 99% of all cases by its 3σ range.

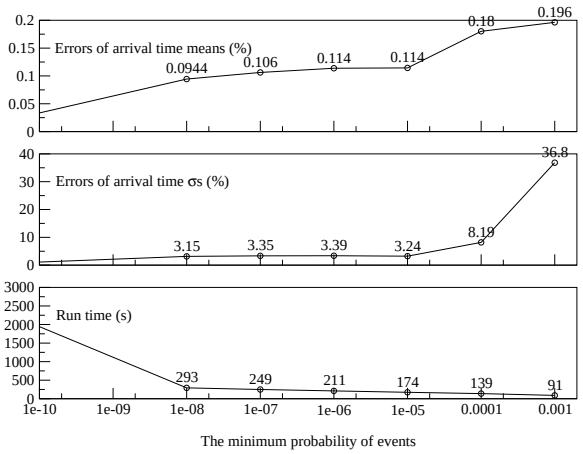

Figure 7: The error percentages and the run time v.s. the minimum probability for s15850.

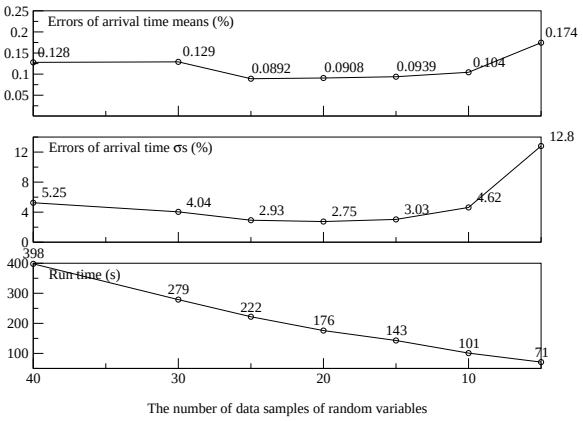

Figure 8: The error percentages and the run time v.s. the number of data samples of cell delay random variables for s15850.

Figure 8 shows the effects of the number of data samples of random variables (N_s) on the error percentages for arrival time mean, variance and run time for circuit s15850 with $P_m = 10^{-5}$. The comparison target in this experiment is the Monte Carlo process. This plot demonstrates an interesting property of varying N_s: a bathtub shape of error graphs. A larger number of samples for discretizing cell delay random variables does not necessarily give lower errors. The reason is because with a fixed P_m more events are filtered out due to the larger number of samples, where events have lower average probabilities than those with a smaller number of samples. Us-

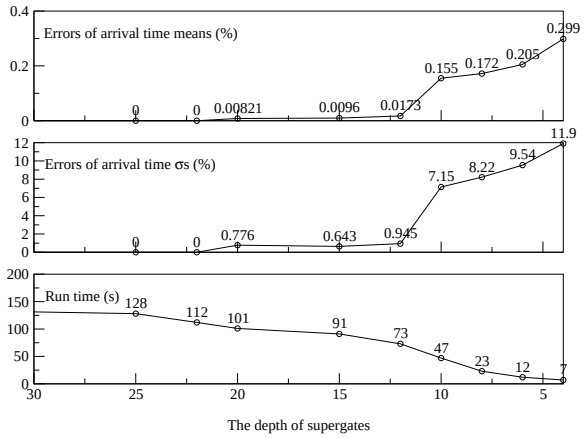

Figure 9: The error percentages and the run time v.s. the depth of supergates for s15850.

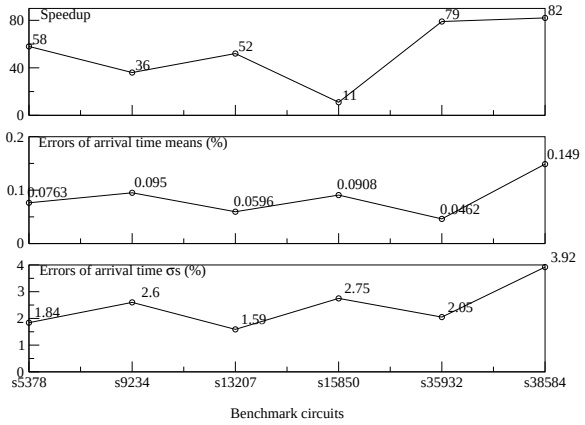

Figure 10: The speedup and the error percentages for benchmark circuits.

ing this plot, we derive that $N_s = 20$ matches best with $P_m = 10^{-5}$.

The effects of the supergate depth limit on the error percentage, variance and run time are shown in Figure 9 with $P_m = 10^{-5}$ and $N_s = 20$. The experiment shows that with a lower limit of logic level for supergates the run time will be lower, but the error will be larger. For s15850, $D = 22$ seems to be the best choice. The best values of D for different circuits depends on their circuit structure.

Similar results and graphs are obtained for other benchmark circuits. The results for several circuits are plotted in Figure 10 (the compared cpu time for the new algorithm include the time for the initialization, the circuit partition, and the heuristics.) The approximate algorithm has achieved more than one order of magnitude speedup over the Monte Carlo process with the errors of means bounded within 0.095% as compared with the results produced by the Monte Carlo process, except for the circuit s38584. The value 0.095% is the error bound of Monte Carlo process. By tracing the sources of larger errors in s38584, we have found that the larger errors are actually caused by the "single-stem estimation" heuristic. To further increase the accuracy level, we propose to select a few supergates which require more elaborate methods than "single-stem estimation". To handle supergates with multiple stems, it is possible to use a special Monte Carlo process which can directly take samples from the probabilistic events. By applying the Monte Carlo method inside a supergate with the same number of runs for a complete circuit, we can have smaller errors for the supergate since there is a smaller s/m ratio inside a supergate. This leads to a somewhat hybrid approach that combines the new method with the Monte Carlo method. Please note that the new algorithm consumes about ten times the memory required by the Monte Carlo approach, since it has to store the probabilistic events for each signal. However, these probabilistic events actually can be used to construct the waveform of the arrival time distribution, which is a more accurate description of the distribution than just with the mean and variance of the Monte Carlo approach. If the Monte Carlo approach is used to collect the arrival time samples for each signal (for a complete waveform), the memory requirement will be in the same level as the new algorithm. And the unused probabilistic events can be deleted to save the resource when there is no further reference to them.

Ckt	s5378	s9234	s13207	s15850	s35932	s38584
N_g	6.58	6.97	8.16	9.55	3.59	4.42
N_s	1.33	1.20	1.02	1.43	1.27	0.87

N_g: average number of gates per supergate
N_s: average number of fanout stems per supergate

Table 1: The average number of gates and fanout stems of supergates

There is another anomalous circuit (s15850) which has the lowest speedup factor. This circuit has very complex structures of supergates. Both the average numbers of gates and stems in a supergate are the largest among all circuits (Table 1). These two numbers indicate that the average time spent on handling each supergate within s15850 should be the highest among all circuits.

5. CONCLUSIONS

We propose a novel deterministic statistical timing analysis algorithm based on the concept of probabilistic event propagation. Experiments show that this algorithm is significantly faster than Monte Carlo methods and produces results with high accuracy. Therefore, it can be applied to larger circuits. The new method can also be used as a core engine in many applications for which it is important to consider statistical delay models such as: yield estimation and optimization, power/glitch estimation, performance sensitivity analysis and target selection for delay fault testing.

6. REFERENCES

[1] D. R. Tryon, F. M. Armstrong, and M. R. Reiter. Statistical Failure Analysis of System Timing. *IBM Journal of Research and Development*, 28(4):340–355, July 1984.

[2] H.-F. Jyu, S. Malik, S. Devadas, and K. Keutzer. Statistical Timing Analysis of Combinational Logic Circuits. *IEEE Transactions on VLSI Systems*, 1(2):126–137, June 1993.

[3] H.-F. Jyu and S. Malik. Statistical Delay Modeling in Logic Design and Synthesis. *Proc. DAC*, pp. 126–130, June 1994.

[4] J.-J. Liou, A. Krstić, K.-T. Cheng, D. Mukherjee, and S. Kundu. Performance Sensitivity Analysis Using Statistical Methods and Its Applications to Delay Testing. *Proc. ASP-DAC*, pp. 587–592, January 2000.

[5] S. C. Seth, and V. D. Agrawal. A New Model For Computation of Probabilistic Testability in Combinational Circuits. *Integration, The VLSI Journal*, 7(1), pp. 49-75, April 1989.

[6] D.-I. Cheng, K.-T. Cheng, D. C. Wang, and M. Marek-Sadowska. A new hybrid methodology for power estimation. *Proc. DAC*, pp. 439–444, June 1996.

[7] Synopsys. Design Compiler Reference Manual. May, 2000.

[8] A. Papoulis. Probability and Statistics. *Prentice-Hall International, Inc.*, New Jersey, 1990.