Convergence-Provable Statistical Timing Analysis with Level-Sensitive Latches and Feedback Loops

Lizheng Zhang, Jengliang Tsai, Weijen Chen, Yuhen Hu, Charlie Chung-Ping Chen
ECE Department, University of Wisconsin, Madison, WI53706-1691, USA
Email:{lizhengz,weijen,jltsai}@cae.wisc.edu, {hu,chen}@engr.wisc.edu

ABSTRACT

Statistical timing analysis has been widely applied to predict the timing yield of VLSI circuits when process variations become significant. Existing statistical latch timing methods are either having exponential complexity or unable to treat the random variable's *self-dependence* caused by the coexistence of level-sensitive latches and feedback loops.

In this paper, an efficient iterative statistical timing algorithm with provable convergence is proposed for latch-based circuits with feedback loops. Based on a new notion of *iteration mean*, we prove that the algorithm converges unconditionally. Moreover, we show that the converged value of iteration mean can be used to predict the circuit yield during design time. Tested by ISCAS'89 benchmark circuits, the proposed algorithm shows an error of 1.1% and speedup of 303× on average when compared with the Monte Carlo simulation.

1. INTRODUCTION

With ever decreasing feature size of nano-scale integrated circuits, the variation of manufacturing parameters becomes more and more significant and must be considered during design. [1] Classical corner-based timing analysis produces timing predictions that are often too pessimistic and grossly conservative because we have only few chance to have parameters of all gates working on their corner values. Statistical static timing analysis (SSTA) that characterizes time variables as statistical random variables offers a better approach for more accurate and realistic timing prediction.

Correlated time variables due to spatial correlation or reconvergence fanout present themselves as a major challenge when applying SSTA to complicated circuits. Such correlations have been studied extensively in literatures [2–9]. In particular, an *extended canonical time model* has been proposed in [6] to represent these correlations in a compact form to be preserved during propagation of the timing random variables.

However, most of the existing SSTA methods only compute the time variable distribution for combinational circuits. Although with some minor modifications, the existing SSTA method may be extended to deal with flip flop based sequential circuits, we have yet to find an effective SSTA method for sequential circuits consisting of level sensitive latches and feed-back loops. Specifically, with the presence of feedback loop, the timing-wise transparent level-sensitive latches may cause timing random variables to be *self-dependent*. In other words, a timing random variable will be dependent on a time variable with the same name, but instantiated in the previous iteration. Such a *self-dependence* presents itself as a new type of correlation that is caused by the coexistence of latches and feedback loops. An example of self-dependence is illustrated in figure 1.

Previously, a SSTA method for latch-based pipeline design have been proposed [10]. However, the issue of self-dependence is not addressed. In [11], a structural method is proposed to deal with the feedback loops by applying graph sorting algorithms. However, the computation complexity of these algorithms may grow exponentially [11].

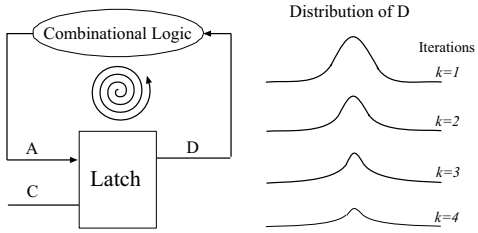


Figure 1: A simple latch circuit with a feedback loop and the possible divergence of the departure time distribution.

In classical deterministic timing analysis, an iterative latch timing algorithm, i.e. the SMO algorithm, has been proposed [12–14] to deal with the self dependence. In order to generalize the SMO algorithm to handle random time variables, one faces a convergence problem that can be briefly explained as follows: In deterministic timing analysis, each time variable assumes a deterministic value. Convergence is guaranteed if each time variable is bounded within a predefined, finite range. However, with SSTA, each time variable is modeled with a mean value and a standard deviation. Even the mean value can be bounded, the corresponding variance may still diverge. This is illustrated in the figure 1.

In this paper, we present a solution to such a convergence problem. Conceptually, we argue that the notion of *circuit convergence* should be differentiated from the overall *algo-*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ASP-DAC 2005, Jan. 24-27, 2006, Yokohama, Japan
Copyright 2006 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

0-7803-9451-8/06/$20.00 ©2006 IEEE.

rithm convergence. Therefore, even the actual data arrival time in the circuit may diverge, the convergence of the algorithm will not necessarily be affected. We proposed a novel SSTA algorithm, *StatITA*, for latch-based circuits with feedback loops based on a quantity of *iteration mean* which is the average latest data arrival time per iteration. We prove that *StatITA* converges unconditionally after sufficient number of iterations. Moreover, we show that the converged value of the iteration mean can be used to predict the circuit yield.

The rest of the paper is organized as following: Section 2 presents preliminary of the latch timing analysis and the graph model of the circuit with feedback loops; Section 3 introduces the theory of our iterative timing method; Section 4 summarizes the *StatITA* algorithm; Section 4 presents the C/C++ implementation and testing results; Section 5 gives the conclusions.

2. LATCH TIMING PRELIMINARY

$T = T_i^0 + T_i^1$: clock cycle time
T_i^0: clock low time at latch i
T_i^1: clock high time at latch i
H_i: hold time of latch i
S_i: setup time of latch i
C_i: rising clock edge arrival time at i^{th} latch
a_i: earliest data arrival time at i^{th} latch
A_i: latest data arrival time at i^{th} latch
d_i: earliest data departure time at i^{th} latch
D_i: latest data departure time at i^{th} latch
δ_{ij}: minimum combinational delay from latch i to j
Δ_{ij}: maximum combinational delay from latch i to j
$\lambda_{ij} = \delta_{ij} - T$: adjusted minimum delay
$\Lambda_{ij} = \Delta_{ij} - T$: adjusted maximum delay
Y: total circuit yield
N: total number of latches in the circuit
s_i: setup time violation at i^{th} latch
h_i: hold time violation i^{th} latch
s_c: critical setup time violation of the circuit
h_c: critical hold time violation of the circuit
p_m: number of latches in the feedback loop m
G_m: cycle mean of the feedback loop m
G_c: critical cycle mean of the circuit
O_i^k: iteration mean of latch i at k^{th} iteration

Figure 2: Notation used in this work

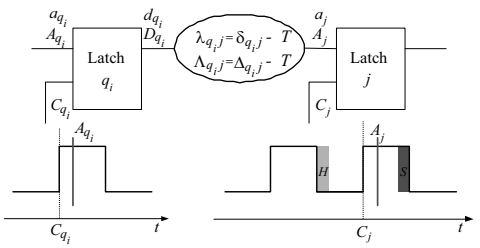


Figure 3: Latch Timing Diagram

It is common for high-end VLSI circuits to have feedback loops. If level-sensitive latches are used as the sequential elements, iterative methods will be applied for circuit timing due to the possible self-dependence issue.

Figure 3 shows a latch j and one of its input latch q_i that has combinational output paths to latch j. All latches are assumed to be active-high, but no generality is lost since no restriction is posted on clocks. So the data departure time of latch q_i at the k^{th} iteration will be:

$$d_{q_i}^k = max(a_{q_i}^k, C_{q_i}) \quad (1)$$
$$D_{q_i}^k = max(A_{q_i}^k, C_{q_i}) \quad (2)$$

On the other hand, the data arrival time of latch j will be decided by its all input latches $q_1, q_2, ...$ as:

$$a_j^{k+1} = min(d_{q_1}^k + \lambda_{q_1 j}, d_{q_2}^k + \lambda_{q_2 j}, ...) \quad (3)$$
$$A_j^{k+1} = max(D_{q_1}^k + \Lambda_{q_1 j}, D_{q_2}^k + \Lambda_{q_2 j}, ...) \quad (4)$$

To make the circuit free from delay faults, the setup and hold time constraints must be satisfied at any latch $j = 1, 2, ..., N$ after sufficient iterations:

$$h_j^\infty = (C_j - T_j^0 + H_j) - a_j^\infty \leq 0 \quad (5)$$
$$s_j^\infty = A_j^\infty - (C_j + T_j^1 - S_j) \leq 0 \quad (6)$$

where s_j^∞ and h_j^∞ are the setup and hold time violations for latch j after sufficient iterations. It is more convenient to define the *critical setup time violation*, s_c^∞ and *critical hold time violation*, h_c^∞ as:

$$h_c^\infty = max(h_1^\infty, h_2^\infty, ..., h_N^\infty) \quad (7)$$
$$s_c^\infty = max(s_1^\infty, s_2^\infty, ..., s_N^\infty) \quad (8)$$

with which the setup and hold time constraints can be expressed compactly as:

$$h_c^\infty \leq 0 \quad and \quad s_c^\infty \leq 0 \quad (9)$$

The above discussion, although it is intended to deal with the deterministic timing analysis, is also applicable when process variations are considered except that all time variables involved will become random variables.

2.1 Circuit Convergence

The major concern for an iterative latch timing method is the *circuit convergence*:

Definition 1. *A latch-based sequential circuit with feedback loops is said to* **converge** *during timing iterations if and only if both the latest and earliest data arrival times at the input of every latch are finite values after infinite number of iterations.*

According to the monotonicity of time variables involved in the iterative timing analysis, [13], it is impossible to have data arrival times, either the latest or the earliest, to be finite but oscillating among several values. So if a circuit converges as defined above, all of its time variables will converge to a fixed value or, in statistical case, to a fixed distribution.

Following theorem can simplify our convergence discussion by just focusing on the latest data arrival time only:

Theorem 1. *A latch-based circuit with feedback loops will converge during timing iterations if and only if every latest data arrival time in the circuit has an upper bound.*

PROOF. According to equations (1) and (2), both latest and earliest data departure time of any latch q_i will be lower bounded by the clock arrival time: $d_{q_i}^k \geq C_{q_i}$ and $D_{q_i}^k \geq C_{q_i}$.

So the latest and earliest data arrival times latch j will also have a finite lower bound:

$$\begin{aligned} a_j^{k+1} &\geq min(\lambda_{q1j} + C_{q_1}, \lambda_{q2j} + C_{q_2}, ...) \\ A_j^{k+1} &\geq max(\Lambda_{q1j} + C_{q_1}, \Lambda_{q2j} + C_{q_2}, ...) \end{aligned}$$

Obviously, the earliest data arrival time of every latch is always upper bounded by the latest data arrival time of the same latch. So if all the latest arrival times in the circuit have upper bound, then circuit will converge as defined above. This proves the sufficiency. The necessity is trivial according to the definition of the convergence. □

2.2 Reduced Timing Graph

Timing iterations will be done at each latch in the circuit. To illustrate such iterations graphically, the entire circuit is partitioned into two parts: latches and combinational feedback sub-circuit. A *reduced timing graph*, $\{\boldsymbol{V}, \boldsymbol{E}\}$, is then constructed to represent the original circuit: latches are modeled by nodes of $n_i \in \boldsymbol{V}$ and the combinational feedback sub-circuit is abstracted as directed edges of $e_{ij} \in \boldsymbol{E}$ with weight of the adjusted maximum delay Λ_{ij}.

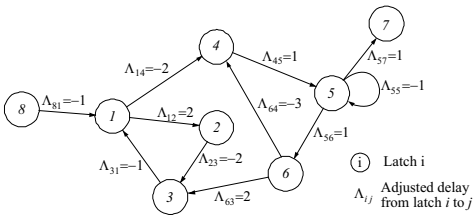


Figure 4: The reduced timing graph of an example circuit

A simple example of such reduced timing graph is shown in figure 4 where a circuit with 8 latches are modeled.

3. ITERATIVE TIMING THEORY

The main difficulty for iterative latch timing, as mentioned before, is the existence of feedback loops in the reduced timing graph. Every loop m with p_m latch nodes in it will have a *cycle mean*(G_m) defined as the average edge weight in the loop:

$$G_m = \frac{1}{p_m} \sum_{e_{ij} \in m} \Lambda_{ij} = \frac{1}{p_m} \sum_{e_{ij} \in m} \Delta_{ij} - T$$

and p_m is usually called *cycle length*.

For example, in the reduced timing graph shown in figure 4, latch nodes $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ will form a loop with length of 3, and the cycle mean of this loop is $G = (\Lambda_{12} + \Lambda_{23} + \Lambda_{31})/3 = -1/3$.

There will usually many loops existing in the reduced timing graph. Among them, the loop with the most positive cycle mean is with the most importance:

Definition 2. *The* **critical cycle mean** *of the reduce timing graph, G_c, is the larger value between 0 and the largest cycle mean among all possible loops:*

$$G_c = max(0, G_1, G_2, ...) \geq 0 \tag{10}$$

where $G_1, G_2, ...$ are cycle means for all loops $1, 2, ...$ in the reduced timing graph.

For example, there are totally 4 loops in the example circuit shown in figure 4: loop $1(1 \rightarrow 2 \rightarrow 3 \rightarrow 1)$, loop $2(1 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 1)$, loop $3(4 \rightarrow 5 \rightarrow 6 \rightarrow 4)$ and loop $4(5 \rightarrow 5)$. The cycle means of these loops are $G_1 = -1/3$, $G_2 = 1/5$, $G_3 - 1/3$ and $G_4 = -1$. So the critical cycle mean is $G_c = max(0, -1/3, 1/5, -1/3, -1) = 1/5$.

The critical cycle mean, G_c, revealed in later sections, is one of the most important circuit parameters determining the yield of the circuit with feedback loops. It is noticeable the similarity between the critical cycle mean G_c and the well-known concept of *maximum cycle mean(MCM)* in general graph theory. In deterministic cases, efficient algorithms are available to compute the MCM [15,16]. But these algorithms can not be directly applied when G_c becomes a random variable because of process variations. We here, instead, propose to compute G_c with an iterative method using the key idea of *iteration mean*:

Definition 3. *At every iteration k, each latch node i in the reduced timing graph will have an* **iteration mean** *defined as the latch's average latest data arrival time per iteration:*

$$O_i^k = \frac{A_i^k}{k+1} \tag{11}$$

3.1 Graphical Imitation of Iterative Timing

Graphically, the update of the latest data arrival times at all latches, equations (2) and (4), can be imitated by one step of simultaneous "*hop*" of time variables at all nodes along all edges in the reduce timing graph with the following rules:

1. If a time variable hops along an edge, the edge weight is added.
2. If a time variable hops into a node, it will continue hopping only if it arrives later than the node's clock. Otherwise, it will "*die*" and a new time variable with the value of the clock arrival time at the node starts hopping.
3. If a time variable hops out of a node with multiple output edges, then multiple "*clones*" of the time variable will hop along all output edges.
4. If multiple time variables hop towards a node simultaneously, only the one with largest value will hop into the node.

It is clear that a time variable may not always hop along a given loop m in the reduce timing graph because it could possibly die. So it is meaningful to pickup those loops that are actually being followed by time variables and call them *timing loops*.

Definition 4. *If a time variable starts hopping at a node and it comes back to the same node later, then the loop through which the time variable passes is called a* **timing loop**.

The relationship between a latch node and a timing loop can be one of the following three cases: (1) A node is **within** *a timing loop if it is a node member of the timing loop; (2) node is* **dominated** *by a timing loop if it doesn't belong to the timing loop but the time variable hopping into it originates in the timing loop; (3) a node is* **independent** *on a timing loop if it is neither within nor dominated by the timing loop.*

For example, in figure 4, if the loop $1 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 1$ becomes a timing loop, then node 8 will be independent to the timing loop and node 7 will be dominated by the timing loop.

3.2 Compute G_c from Iteration Mean

To prove the convergence of the iterative mean, we first prove a theorem which is valid for any loop in the reduce timing graph.

Theorem 2. *If there is a loop, m, with cycle mean of G_m and length of p_m, in the reduced timing graph, then for any iteration index of $k \geq p_m$ and any node $m_i (i = 1, 2, ..., p_m)$ in the loop, the latest data arrival time will satisfy the following inequality:*

$$A^k_{m_i} \quad \geq \quad A^{k-p_m}_{m_i} + p_m G_m \tag{12}$$

where the equality holds if m becomes a timing loop.

PROOF. Assuming time variable $A^{k-p_m}_{m_1}$ of node m_1 at iteration $k - p_m$ tries to hop to node m_2, from iteration equations (2) and (4), we have:

$$\begin{aligned} A^{k-p_m+1}_{m_2} &\geq max(A^{k-p_m}_{m_1}, C_{m_1}) + \Lambda_{m_1,m_2} \\ &\geq A^{k-p_m}_{m_1} + \Lambda_{m_1,m_2} \end{aligned}$$

where the equality holds if $A^{k-p_m}_{m_1}$ survives the hop from node m_1 to node m_2. Iteratively making such hops for p_m times along the loop of m, we will return back to node m_1 and

$$\begin{aligned} A^k_{m_1} &\geq A^{k-p_m}_{m_1} + \Lambda_{m_1,m_2} + \Lambda_{m_2,m_3}, ..., +\Lambda_{m_{p_m},m_1} \\ &= A^{k-p_m}_{m_1} + p_m G_m \end{aligned}$$

where the equality holds only when the time variable survives every step of hopping which means the loop is a timing loop. □

With this theorem in hand, we are then ready to present our first major contribution:

Theorem 3 (Convergence of Iteration Mean). *No matter what is the initial state of the reduced timing graph, the sequence of the iteration mean $O^0_j, O^1_j, O^2_j, ..., O^k_j, ...$ for any node j will always converge to one of the following two values after sufficient number of iterations:*

1. *$O^\infty_j = G_m$ if node j is within or dominated by a timing loop m with cycle mean G_m.*
2. *$O^\infty_j = 0$ if node j is independent on any timing loop.*

PROOF. In case 1, if node j is within the timing loop m whose length is p_m after r iterations, then for any index $k \geq r + p_m$ equality holds in theorem 2. Since there will exist an iteration index $r \leq t \leq r + p_m$ and an integer $n = 0, 1, 2, ...$ such that $k = np_m + t$, after applying theorem 2 for n times, the latest data arrival time of node j at iteration k will be $A^k_j = A^t_j + np_m G_m = A^t_j + (k - t)G_m$. So the iteration mean:

$$O^\infty_j = \lim_{k\to\infty} \frac{A^k_j}{k+1} = \lim_{k\to\infty} \frac{(k-t)G_m}{k+1} = G_m$$

If node j is not within but dominated by the timing loop m, then we define two constants Σ^+ and Σ^- as:

$$\Sigma^+ = \sum_{\Lambda_{i,j}>0} \Lambda_{i,j} \quad and \quad \Sigma^- = \sum_{\Lambda_{i,j}<0} \Lambda_{i,j}$$

which are clearly finite values.

With these two constants, if the time variable of node j is x hops away from node m_i in the timing loop m, then the latest data arrival time of node j at k^{th} iteration will be:

$$A^{k-x}_{m_i} + \Sigma^- \leq A^k_j \leq A^{k-x}_{m_i} + \Sigma^+$$

Since Σ^+ and Σ^- are finite constants, then

$$O^\infty_j = \lim_{k\to\infty} \frac{A^{k-x}_{m_i}}{k+1} = G_m$$

which proves the first case of the theorem.

For case 2, all time variables hopping into node j must start from a finite value such as clock arrival times or primary input arrival times. So the latest data arrival time at node j must be finite for any iteration index k. So the iteration mean of the node obviously converges to zero. □

For the example graph shown in figure 4, the convergence of the iteration mean of some latch nodes is graphically illustrated in figure 5:

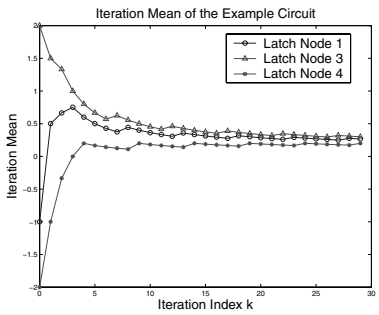


Figure 5: The iteration mean of three nodes in the example circuit of figure 4 and their convergence trend.

In [10], authors proposed to use the graph sorting algorithm which has exponential complexity in the worst cases. By substituting theorem 3 into equation (10), the critical cycle mean can be computed much simpler as:

$$G_c = max(O^\infty_1, O^\infty_2, ..., O^\infty_N) \tag{13}$$

where $O^\infty_1, O^\infty_2, ..., O^\infty_N$ are converged iteration means for all N latches in the circuit.

3.3 Circuit Yield Prediction

In section 3, we defined a circuit parameter of *critical cycle mean* and claim that it is closely relating the circuit yield. We will establish this relationship solidly in this section.

Theorem 4 (Circuit Yield Computation). *A circuit will converge to a finite state if and only if its reduced timing graph has zero critical cycle mean: $G_c = 0$.*

PROOF. If $G_c > 0$, then according to the definition, there will be at least one loop whose cycle mean is G_c. Then applying equation 12 for enough times, the latest data arrival time in this loop can be arbitrarily large. Using theorem 1, the circuit will not converge in this case. Since $G_c \geq 0$ from definition, the necessity of the first assertion is proved.

If $G_c = 0$ then we have two cases. The first case is that there will be no timing loop in the reduced timing graph after sufficient number of iterations; This means that eventually all time variables hopping in the graph will come from

finite values and so that all latest data arrival times in the graph will be finite values. So circuit will converge by definition in section 2.1.

The second case is that there are some timing loops existing in the reduced timing graph after sufficient number of iterations. In this case, for any of such timing loop m, its cycle mean, G_m, must be non-positive since $G_m \leq G_c \leq 0$. So for any node m_i in m, applying theorem 2 for sufficiently large iteration index k:

$$A^k_{m_i} = A^q_{m_i} + (k-q)G_m \leq A^q_{m_i}$$

where q is a large but finite iteration index satisfies $k = q + np_m$ with loop length p_m and integer n. Since $A^q_{m_i}$ is obviously finite, the latest data arrival time must be upper bounded and the circuit converges in this case too. This proves the sufficiency. □

Circuit will fail if it diverges since the latest data arrival time of some latches will go to infinity after sufficient number of iterations. But the circuit is not guaranteed to be functional even it converges in iterations. The setup and hold time constraints have to be additionally satisfied in order to be free of delay faults. So the overall timing yield of the circuit will be:

$$\begin{aligned} Y &= Pr\{G_c = 0 \cap s^\infty_c \leq 0 \cap h^\infty_c \leq 0\} \\ &= Pr\{max(G_c, s^\infty_c, h^\infty_c) = 0\} \end{aligned} \quad (14)$$

4. ITERATIVE TIMING ALGORITHM

The iterative latch timing algorithm, *StatITA*, is shown in figure 6 based on the theory in section 3.

StatITA takes the circuit as an input and compute the yield of the circuit at a given clock cycle time. The key iteration part of the algorithm is the repeat block from line 7 to line 19 where the convergence check is done in line 14.

StatITA has been implemented in C/C++ and tested on the ISCAS'89 benchmark circuits. The SSTA core is implemented based on the work from [6].

4.1 Accuracy of StatITA

All known tasks of statistical timing analysis can equivalently be accomplished by Monte Carlo simulations. The iterative timing analysis for latch-based circuits with feedback loops is not an exception either. So an iterative Monte Carlo timing analysis with 10,000 repetitions is also implemented in C/C++, *MontITA*, in parallel with *StatITA*.

Figure 7 shows the distributions of the critical cycle mean G_c computed from both *StatITA* and *MontITA* for circuit s526 at a clock cycle of 400*ps*. The close match between *StatITA* and *MontITA* clearly shows the accuracy of the proposed iterative statistical latch timing algorithm.

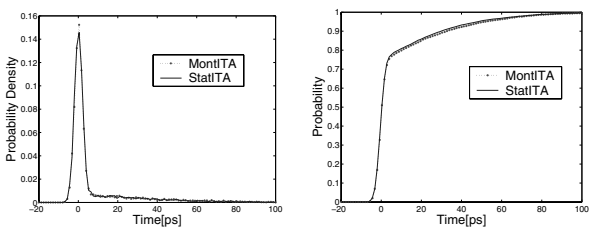
The first application of a fast and accurate statistical latch timing algorithm is to predict the minimum clock cycle time at which the circuit will meet a yield goal. For this purpose, we define *the value of* T_{97} *as the minimum clock cycle time at which a given circuit will have a* 97% *timing yield.*

Table 1 shows the numerical T_{97} comparison between *MontITA* and *StatITA*. The average prediction error for the tested circuits is 1.1% which again demonstrates the accuracy of the proposed algorithm.

4.2 Performance of StatITA

```
1: procedure StatITA(ClockCycle T)
2:     for (each latch i) do                          ▷ initialization
3:         C_i = setClockArrivalTimeForLatch(i)
4:         a_i^0 = 0; A_i^0 = 0; O_i^0 = 0;
5:     end for
6:     k = 1;
7:     repeat                                         ▷ iteration starts
8:         {a_i^k, A_i^k}=statistiticalTiming(a_i^{k-1}, A_i^{k-1});
9:         done = true;
10:        for (each latch i ) do
11:            O_i^k = A_i^k/(k+1);
12:            mean = |μ_{O_i^{k-1} − O_i^k}|;
13:            std = σ_{O_i^{k-1} − O_i^k}
14:            if mean ≥ threshold ∪ std ≥ threshold then
15:                done = false;                      ▷ Not converged yet
16:            end if
17:        end for
18:        k = k + 1;
19:    until done                                     ▷ iteration ends
20:    k = k-1;
21:    G_c = 0; s_c^∞ = −∞; h_c^∞ = −∞;
22:    for (each latch i) do
23:        G_c = max(O_i^k, G_c);
24:        s_c = max(A_i^k − (C_i + 0.5T − S), s_c^∞);
25:        h_c = max((C_i − 0.5T + H) − a_i^k, h_c^∞);
26:    end for
27:    Y = Pr{max(G_c, s_c^∞, h_c^∞) = 0};            ▷ circuit yield
28: end procedure
```

Figure 6: Iterative timing analysis algorithm **StatITA**



(a) *p.d.f.* of G_c for s526 (b) *c.d.f.* of G_c for s526

Figure 7: Critical cycle mean G_c for circuit s526 computed from *StatITA* and *MontITA* at clock cycle of 400ps

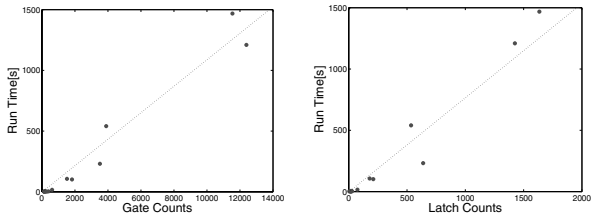
As reported in almost all SSTA works, Mont Carlo simulation is generally used as a "golden" method to evaluate the effect of process variations in the circuit timing. But the problem to directly apply Mont Carlo simulation in large circuit timing is the excessive CPU time it needs. This performance problem will be obviously more severe in iterative timing since each Monte Carlo sample will need many iterations to get the converged timing result.

From table 1, the excessive computation time needed by the MontITA is obvious. The average speedup of StatITA over MontITA is 303×.

The iteration core of statistical timing analysis is linear to the number of gates in the circuit according to [6]. The total number of iterations is determined by the convergence threshold and the circuit topology, not the size of the circuit. So the proposed StatITA algorithm will have linear complexity with respecting to the circuit size. This conclusion is clearly demonstrated in figure 8 where the linear trend of the run time with respecting to the gate number

Circuits	Gates	Latches	T_{97}[ps]			CPU Time[s]		
			StatITA	MontITA	Error	StatITA	MontITA	Speedup
s298	130	14	443	452	2.0%	2.14	320	150x
s526	196	21	465	469	0.9%	5.76	694	120x
s641	173	19	999	998	0.1%	1.17	372	320x
s820	279	5	777	788	1.4%	1.35	692	513x
s953	401	29	862	858	0.5%	3.32	1041	314x
s1423	616	74	2088	2051	1.8%	16.0	2083	130x
s5378	1517	179	764	780	2.1%	106	12372	117x
s9234	1827	211	859	858	0.1%	101	19073	189x
s13207	3516	638	1242	1246	0.3%	231	41571	180x
s15850	3889	534	1189	1199	0.8%	540	61044	113x
s38417	11543	1636	1544	–	–	1468	200hr*	490x*
s38584	12389	1426	1430	–	–	1209	303hr*	903x*
Average	–	–	–	–	**1.1%**	–	–	**303x**

Table 1: 97% yield clock cycle(T_{97}) and CPU time comparison between *StatITA* and *MontITA*.
(*)Estimation is from 100 repetitions and the accuracy of StatITA is not evaluated for these circuits.



(a) CPU time v.s. gate number (b) CPU time v.s. latch number

Figure 8: Run time of *StatITA* v.s. circuit size

and latch number in the circuit is shown.

5. CONCLUSIONS

A novel iterative timing statistical algorithm.*StatITA*, for latch-based circuits with feedback loops is proposed and its convergence during iterations is both theoretically proved and experimentally demonstrated. A novel concept of *iteration mean* is proposed to decide the convergence of the algorithm and the relationship between the converged iteration mean and circuit yield under process variations are founded both theoretically and experimentally.

Tested by the ISCAS'89 benchmark circuits, the proposed algorithm shows an error of 1.1% and 303× speedup on average when compared with Monte Carlo simulations.

6. ACKNOWLEDGEMENT

This work was partially funded by TSMC, UMC, Faraday, SpringSoft, National Science Foundation under grants CCR-0093309 & CCR-0204468 and National Science Council of Taiwan, R.O.C. under grant NSC 92-2218-E-002-030. Also great thanks to professor Barry D. Van Veen for the great discussions.

7. REFERENCES

[1] S. Nassif, "Within-chip variability analysis," *Electron Devices Meeting, 1998. IEDM '98 Technical Digest., International*, pp. 283 – 286, Dec 1998.

[2] C. S. Amin, N. Menezes, K. Killpack, F. Dartu, Y. Ismail, U. Choudhury, and N. Hakim, "Statistical static timing analysis: How simple can we get?" *42th Design Automation Conference, DAC'05*, 2005.

[3] J. Le, X. Li, and L. Pileggi, "Stac: statistical timing analysis with correlation," *Design Automation Conference, 2004. Proceedings. 41st*, pp. 343 – 348, June 2004.

[4] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," *ICCAD'03*, pp. 621–625, Nov 2003.

[5] C. Visweswariah, K. Ravindran, and K. Kalafala, "First-order parameterized block-based statistical timing analysis," *TAU'04*, Feb 2004.

[6] L. Zhang, W. Chen, Y. Hu, and C. C. Chen, "Statistical timing analysis with extended pseudo-canonical timing model," *DATE'05*, March 2005.

[7] M. Orshansky, C. Spanos, and C. Hu, "Circuit performance variability decomposition," *4th International Workshop on Statistical Metrology, 1999. IWSM. 1999*, June 1999.

[8] S. Tsukiyama, M. Tanaka, and M. Fukui, "A statistical static timing analysis considering correlations between delays," *Proceedings of the 2001 conference on Asia South Pacific design automation*, January 2001.

[9] F. N. Najm and N. Menezes, "Yield estimation and optimization: Statistical timing analysis based on a timing yield model," *Proceedings of the 41st annual conference on Design automation*, 2004.

[10] M. C.-T. Chao, L.-C. Wang, K.-T. Cheng, and S. Kundu, "Static statistical timing analysis for latch-based pipeline designs," *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004*, pp. 468 – 472, 2004.

[11] R. Chen and H. Zhou, "Clock schedule verification under process variations," *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004*, pp. 619 – 625, Nov 2004.

[12] K. A. Sakallah, T. Mudge, and O. Olukotun, "$checkt_c$ and $mint_c$: timing verification and optimal clocking of synchronous digital circuits," *IEEE International Conference on Computer-Aided Design, 1990. ICCAD-90*, pp. 552 – 555, 1990.

[13] T. Szymanski and N. Shenoy, "Verifying clock schedules," *IEEE/ACM International Conference on Computer-Aided Design, 1992. ICCAD-92*, pp. 124 – 131, 1992.

[14] J.-F. Lee, D. Tang, and C. Wong, "A timing analysis algorithm for circuits with level-sensitive latches," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 535 – 543, May 1996.

[15] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Mathematics*, vol. 23, pp. 309–311, 1978.

[16] S. M. Burns, "Performance analysis and optimization of asynchronous circuits," *PhD Thesis, California Institute of Technology*, 1991.