Analyzing Timing Uncertainty in Mesh-based Clock Architectures

Subodh M. Reddy  
Fujitsu Laboratories of America, Inc.  
California, USA  
subodh@fla.fujitsu.com

Gustavo R. Wilke*  
UFRGS  
Porto Alegre, Brazil  
wilke@inf.ufrgs.br

Rajeev Murgai  
Fujitsu Laboratories of America, Inc.  
California, USA  
murgai@fla.fujitsu.com

Abstract

Mesh architectures are used to distribute critical global signals on a chip, such as clock and power/ground. Redundancy created by mesh loops smooths out undesirable variations between signal nodes spatially distributed over the chip. However, one problem with the mesh architectures is the difficulty in accurately analyzing large instances. Furthermore, variations in process and temperature, supply noise and crosstalk noise cause uncertainty in the delay from clock source to flip-flops. In this paper, we study the problem of analyzing timing uncertainty in mesh-based clock architectures. We propose solutions for both pure mesh and (mesh + global-tree) architectures. The solutions can handle large design and mesh instances. The maximum error in uncertainty values reported by our solutions is 1-3ps with respect to the golden Monte Carlo simulations, which is at most 0.5% of the nominal clock latency of about 600ps.

1 Introduction

Mesh or grid architectures are popular for distributing critical global signals on a chip, such as clock and power/ground. The mesh architecture uses inherent redundancy created by loops to smooth out undesirable variations between signal nodes spatially distributed over the chip. These variations can be due to non-uniform switching activity in the design, within-die process variations, or asymmetric distribution of circuit elements (such as flip-flops). For power/ground, mesh can help reduce voltage variations at different nodes in the network due to non-uniform switching activities. For the clock signal, a mesh (Figure 1) has been shown to achieve very low skew in microprocessor designs, e.g., Digital Alpha [2]; IBM G5 S/390 [6], Power4 and PowerPC [12]; SUN Sparc V9 [13]. Mesh also has excellent jitter mitigation properties.

However, one major problem that has limited the applicability of mesh architectures is the difficulty in analyzing them with sufficient accuracy. The main reasons are the huge number of circuit nodes needed to accurately model a fine mesh in a large design and a large number of metal loops present in the mesh structure. As a result, circuit simulators such as SPICE either require inordinate amount of memory or run-time. In fact, HSPICE and HSIM (Synopsys) failed to analyze even coarse meshes for an industrial design [4].

An added degree of complication is brought forth by variations in parameters that affect clock latency [18, 14, 3, 10]. Examples of such parameters are process (channel length, oxide thickness, interconnect width and thickness, etc), supply voltage, temperature and crosstalk noise. Variations in these parameters cause variations or uncertainty in delay from the clock root to flip-flops, both die-to-die and clock cycle-to-clock cycle [17, 8]. With technology scaling, the magnitude of parameter variations and the sensitivity of clock latency towards variations are increasing.

The focus of this paper is to analyze the timing uncertainty of mesh-based clock architectures in the presence of parameter variations. We believe this is the first work that addresses this problem. We propose solutions for both pure mesh and (mesh + global-tree) architectures. The solutions can handle large design and mesh instances. We show that uncertainty values reported by our solutions are within 1-3ps of those obtained from the golden Monte Carlo simulations (e.g., 35ps vs. 33ps), where the nominal clock latency is about 600ps. Another major benefit of our scheme is that it is easily amenable to distributed- or grid-computing.

The paper is organized as follows. Section 2 gives preliminaries. Section 3 describes previous work on clock mesh analysis. An overview of our methodology for uncertainty analysis of clock meshes under parameter variations is presented in Section 4. The details of our methodology are presented along with experimental results in Section 5. We conclude and give directions for future work in Section 6.

2 Preliminaries

2.1 Mesh-based Clock Architecture

Figure 1 shows a typical mesh architecture used for distributing the clock signal from the PLL or root buffer to sequential elements such as flip-flops (FFs) and latches on the chip. It has three main components: 1) a (uniform) mesh, 2) a global buffered tree that drives the mesh, and 3) local interconnect, which connects the clock inputs of FFs directly to the nearest point on the mesh. The mesh is a uniform rectangular grid of wires spanning the entire chip area, driven by the mesh buffers and propagating the clock to the FFs. An m x n mesh or grid has m rows (horizontal wires) and n columns (vertical wires). The size of the mesh is m x n. For a given chip size, the greater the mesh size, the more fine-grain the mesh is. A mesh node (or grid node) is the point where each row is connected to each column. As shown in Figure 1, the global (H-)tree delivers the clock signal to the mesh nodes via buffers called mesh buffers. We assume a uniform array of k x l mesh buffers. In Figure 1, k = m = 4 and l = n = 4. The mesh wire between two adjacent mesh nodes is called a mesh segment, and represents one grid unit.
2.1.1 Clock Network Model

Each buffer (mesh buffer and tree buffer) is modeled using the BSIM3 transistor models for NMOS and PMOS. Since the mesh is largely composed of wires, it is important to have an accurate wire model. To model wires smaller than 100\(\mu\)m, a single-\(\pi\) model, which has two capacitors, a resistor and an inductor, is used (Figure 2). For longer wires, a 3-\(\pi\) model is used, as shown in Figure 3. Our study on Fujitsu’s 0.11\(\mu\)m technology showed that this scheme is delay-accurate within 0.5\% of 4-\(\pi\) and 5-\(\pi\) models [16]. It helps reduce the number of nodes in the SPICE model. The same rule is used to model wires that connect FFs to the mesh and wires on the global tree. The clock pin of a FF is modeled as an equivalent capacitance.

2.2 Clock Timing and Uncertainty

In any clock distribution scheme, one of the most important concerns is to accurately compute the clock arrival time \(a\) (also called clock delay or latency) at the clock input pin of each FF. Assume we have a path \(P\) in a design whose start and end gates are FFs \(F_{\text{ pred}}\) and \(F_{\text{ succ}}\). Let clock arrival times at these FFs be \(a_{\text{ pred}}\) and \(a_{\text{ succ}}\) respectively. The maximum delay \(d_{\text{ max}}\) allowed on \(P\) is a function of \((a_{\text{ succ}} - a_{\text{ pred}})\), the difference in clock arrival times at the two FFs.

\[
d_{\text{ max}} \leq a_{\text{ pred}} - a_{\text{ succ}} + \tau - t_{\text{ set-up}},
\]

where \(\tau\) is the clock cycle and \(t_{\text{ set-up}}\) is the set-up time for \(F_{\text{ pred}}\). \(a_{\text{ pred}} - a_{\text{ succ}}\) is known as the skew between \(F_{\text{ pred}}\) and \(F_{\text{ succ}}\). By comparing the arrival times among all FFs, we can compute the worst relevant clock skew in the design. This is the maximum negative difference in arrival times at two FFs that are connected by a data path. For a fixed clock cycle, the worst skew limits the maximum delay in the data path. Thus, it has a direct impact on the design turnaround time. Alternatively, for a given design, the skew impacts the maximum clock frequency for which the design will function correctly.

In practice, at a given flip-flop on a chip, two consecutive clock rising (or falling) edges may not be \(\tau\) time units apart. Moreover, for the same corresponding flip-flop on two chips, the clock latencies from the clock source may be different. Clock timing uncertainty denotes the deviation of the timing of the clock edge from its expected value. Uncertainty affects \(a_{\text{ succ}}\) and \(a_{\text{ pred}}\) in (1) and hence \(d_{\text{ max}}\) or \(\tau\), as discussed above. Uncertainty in clock timing can be due to several factors.

1. Supply \((V)\) noise: This is caused by different sets of gates switching in different clock cycles. Since gate delay depends on the value of supply voltage, any change in the supply voltage of a clock buffer changes the clock arrival time at the FF.

2. Temperature \((T)\) variation: This variation arises due to different switching activities on the chip (both spatial and temporal) and because power and temperature are strongly coupled to each other, especially for leakage-dominant technologies. A block with higher switching activity dissipates higher dynamic power, leading to higher local temperatures. That, in turn, increases the leakage power dissipation, further increasing the total power. A gate operating at a higher temperature exhibits higher delay due to reduced carrier mobility.

3. Process variations (within die and die-to-die) \(P\): Examples of process variations include intrinsic variations such as random dopant fluctuations in a MOSFET channel and extrinsic variations such as channel length and oxide thickness variations. In a chemical mechanical planarization (CMP) process, interconnect width, thickness, spacing and height may vary significantly from the intended values. These variations cause gate and wire delays to deviate from their desired values. It is difficult to predict the precise magnitude of variations and hence the exact values of wire and gate delays after manufacturing.

4. Crosstalk noise \(X\): Delay of a clock wire \(v\) can change if there is an aggressor \(a\) that is physically close to \(v\) and is switching. Since the aggressor’s switching behavior can change from one cycle to the next, it can lead to timing variation on the victim. Clock is one of the most important signals in the design. \(V_{\text{sat}}/V_{\text{sh}}\) shielding is typically done on both sides of the clock to eliminate such crosstalk impact. Shielding, however, does not prevent crosstalk from the top and bottom layers, when a wide bus is going over the clock line.

5. PLL jitter: Clock generated from the PLL has an inherent jitter. Some of these parameters (such as process) have random unknown values, but the other parameter variations are deterministic. They depend on the state of the design and the last & current signal values, and have to be computed for each cycle. Examples are supply and crosstalk noise. Their exact computation typically requires prohibitive CPU and memory resources and may be infeasible in practice. Nevertheless, both kinds of parameter variations cause uncertainty in the timing of the clock edge at a flip-flop from its expected value.

Let \(D\) denote the latency (path delay) from the clock root to a flip-flop. In general, \(D\) is a function of supply voltage \(V_i\) at each clock buffer \(B_i\) on the path, the temperature \(T_i\) at each clock buffer and wire, the set of process parameters \(P\), and crosstalk noise \(X\). In short, we write \(D(\bar{V}, \bar{T}, \bar{P}, \bar{X})\), where \(\bar{V}\) denotes the vector of all buffer voltages \(\{V_i\}\). In the presence of parameter variations, \(D\) is a distribution with mean \(\mu\) and standard deviation \(\sigma\). We define uncertainty in \(D\), denoted \(U(D)\), as \(k\sigma\). In this paper, we use \(k = 3\).

Problem Statement: Given a mesh-based clock network and VTPX parameter variations for each component of the clock network (i.e., clock buffers and wires), determine the timing uncertainty \(U(D)\) in the clock latency \(D_i\) from the clock root to each flip-flop FF_i.

3 Previous Work

If the clock network is a tree, uncertainty analysis can be carried out using gate-level statistical static timing analysis [9, 15, 1]. However, such an approach is not directly applicable for a mesh-based clock network due to metal loops (cycles) present in the mesh. We are not aware of any work on clock mesh uncertainty analysis. The only known solution is that if the mesh model fits in the memory, we can run Monte Carlo simulations (MCSs) [7] assuming some distribution for parameter variations and obtain a delay distribution at each FF, from which timing uncertainties at FFs could be derived. However, this is possible only for small design and mesh instances.

Not much has been published on the problem of clock mesh latency analysis. [12, 5] present a scheme that breaks the clock mesh into a tree and apply a smoothing algorithm to redistribute the mesh loads. The tree is analyzed for latency. However, no accuracy results are shown. In [2], the clock mesh is verified in two steps. First, an AWE-based reduction [11] is performed on the mesh to simplify the mesh elements. Then, the simplified circuits are simulated using SPICE. The accuracy and efficiency of this method depend on the accuracy and stability of the moment matching technique.

Recently, a sliding window scheme (SWS) was proposed for latency analysis of clock meshes [4]. Since uncertainty analysis derives its basic idea from SWS, we describe it next.

3.1 Sliding Window Scheme for Mesh Latency

In SWS, the mesh is modeled with two different resolutions: a detailed circuit model is used for the mesh elements geometrically close
Figure 4: The sliding window scheme

Figure 5: Statistical simulation model for a buffer driving a wire

to the nodes whose latency we are measuring and a simplified model is used for the mesh elements far from the nodes being measured. The simplification is with respect to the local FF connections. Given a mesh of size \( m \times n \), define a rectangular window \( W \) of size \( r \times s \), where \( r < m \) and \( s < n \). Expand \( W \) by some border to obtain \( \tilde{W} \) (Figure 4, in which the border is 1 grid unit). If the lower left corner of \( W' \) is fixed to a point on the mesh, \( W' \) covers some fixed region of the mesh (Figure 4). The connection of a FF within \( W' \) to the nearest mesh segment is modeled accurately by an appropriate \( \pi \) model, as described in Section 2.1.1 (single-\( \pi \) or 3-\( \pi \), depending on the length of the connection). The clock input pin of the FF is modeled as a capacitance. FFs that lie outside \( W' \) and their connections to the mesh are modeled approximately. The wire connecting such a FF to the mesh is replaced by an equivalent single capacitance; the wire resistance is ignored. Given a mesh node \( \alpha \) outside \( W' \), the region covered by \( \alpha \) is the unit rectangle shown in Figure 4. Let \( C_{\alpha} \) be the sum of the clock input pin capacitances of all the FFs in this region along with the capacitances of the wires connecting them to the mesh. Then, \( C_{\alpha} \) is lumped as a single capacitance at \( \alpha \). The mesh segments outside \( W' \) are still modeled with appropriate \( \pi \) models. The SPICE file corresponding to this model for the window location is generated and simulated. The clock latencies at all FFs in the inner window \( W \) are measured.\(^1\) Next, the window \( W \) is slid horizontally or vertically so as to overlap with the previous locations. Once again, a SPICE model is created and run. The entire mesh simulation is broken down into multiple independent window-based simulations. In fact, \( \left( \frac{1}{r} \right) \times \left( \frac{1}{s} \right) \) SPICE simulations are needed to cover the entire mesh and all the FFs in the design.

SWS is a divide-and-conquer partitioning technique. Approximating each FF saves either 7 nodes (if the wire is longer than 100\( \mu \)m) or 3 nodes (otherwise). In a typical design, where there are hundreds of thousands of FFs, reduction in the SPICE model size can be huge. It was shown in [4] that HSPICE could not finish on a 65x65 mesh with 100K FFs. It needed more than 2GB of memory, whereas SWS could complete in less than 1.5 hours within 1GB memory using four machines. The latencies computed by SWS, using a border of 1 grid unit, are almost always within 1% of the latencies computed from SPICE simulation of the complete mesh. It was also shown that using no border (i.e., a border of 0 grid units) does not yield accurate results; errors of up to 30% were seen. By increasing the border beyond 1 grid, the accuracy does not improve much. However, the runtime increases significantly. In short, empirically a border of 1 grid unit was found to be optimum. Also, window size was shown to have very little impact on accuracy. However, smaller window size means smaller model and hence better chances for large designs to fit in the memory. But smaller window also implies more simulations.

4  Clock Clock Mesh Mesh Uncertainty Analysis

4.1  Modeling Sources of Uncertainty

We model various sources of uncertainty as follows. Refer to Figure 5, where inverting buffer 1 drives inverting buffer 2 through a wire.

1. Supply Noise \( \text{supply}_1 \): Supply noise is modeled by supplying independent power supplies to each clock buffer, and allowing them to vary randomly according to a noise model. The amount of variation is controlled by a user input parameter, \( \text{supply}_1 \).

2. Temperature Variation \( T \): Rising temperature causes CMOS circuits to operate more slowly, and wiring resistances to increase. Temperature variation of transistors is modeled by specifying an underlying temperature for the entire chip and then applying random local temperature variations on each clock buffer and interconnect. The variation to apply is given by a user input parameter, \( \text{max}_\text{deltemp} \).

3. Process Variation \( \text{process} \): As shown in Figure 5, process variation of transistors is modeled using only channel length \( (L_{\pi}) \) and \( (L_{\pi}) \), for PMOS and NMOS transistors respectively) and threshold voltage (\( \text{vth}_{\text{PMOS}} \) and \( \text{vth}_{\text{NMOS}} \)). Other variations, such as oxide thickness and dopant concentration, have the overall effect of varying the threshold voltage and hence are indirectly included in our model. The variations of threshold voltage and channel lengths are passed into each instance of the buffer sub-circuit models. Process variation of wiring is modeled by applying random process factors \( p_{\text{f}} \) and \( p_{\text{w}} \) to the wiring capacitance and resistance respectively in the wire models.

4. Crosstalk Noise \( X \): Crosstalk noise is modeled by attaching external noise sources to the wire model (Figure 5) and by applying random inputs at these sources based on some probability distribution. The crosstalk factor associated with the instances must also be defined whenever a wire is instantiated. The crosstalk factor is a unique property of each design, and is supplied by the user through the parameter \( \text{xfactor} \).

5. PLL jitter: We will assume a maximum PLL jitter of \( 3\sigma \text{PLL} \). \( \text{PLL} \).\n
4.2 Computing Uncertainty: Basic Idea

The basic idea is simple: we use SWS for analyzing timing uncertainty of a mesh. We attach variation parameters with each buffer and wire on the clock network, as illustrated in Figure 5. For each window \( W' \) of SWS, a SPICE model of the mesh is created (just as in [4]) and Monte Carlo simulations (MCSs) are carried out. In each run of the MCSs, the values of VTPX parameters for each component of the clock network are determined from their respective distributions, and the latency \( D_{\text{c}} \) of each flip-flop FF is computed. The uncertainty \( U(D_{\text{c}}) = 3\sigma(D_{\text{c}}) \) is then computed from this distribution. Finally, \( U(D_{\text{c}}) \) is collected from all windows \( W' \) to yield uncertainties at all the FFs in the design.

In this paper, we do not use large design and mesh instances. The feasibility of SWS for those has already been shown [4]. Our focus is on small design and mesh instances. The feasibility of SWS for those has already been shown [4]. Our focus is on small design and mesh instances. The feasibility of SWS for those has already been shown [4].
to determine if SWS can be used for accurate uncertainty analysis of both pure mesh and (mesh + global tree) architectures, and if so, to derive a practical and usable methodology. The next section presents detailed results of our study.

5 Results

5.1 Experimental Set-up & Definitions

All our experiments were conducted in Fujitsu’s 0.11μ technology. The 3σ variations for various parameters are shown in Table 1.

<table>
<thead>
<tr>
<th>parameters</th>
<th>3σ variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMOS/PMOS tunnel length</td>
<td>0.50μ</td>
</tr>
<tr>
<td>NMOS/PMOS threshold voltage</td>
<td>20mV</td>
</tr>
<tr>
<td>interconnect resistance</td>
<td>20.00%</td>
</tr>
<tr>
<td>interconnect capacitance</td>
<td>5.00%</td>
</tr>
<tr>
<td>temperature</td>
<td>20°C</td>
</tr>
<tr>
<td>(V_{dd})</td>
<td>10.00%</td>
</tr>
<tr>
<td>crosstalk switching probability</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1: 3σ variations for different parameters

In the following, we will compare FF uncertainties obtained from a methodology \(M\) against those from a golden reference methodology \(G\). For instance, \(M\) may correspond to the SWS-based uncertainty analysis, and \(G\), to running MCSs on the flat single model of the mesh-based clock network. To evaluate the quality of uncertainty results, we use two metrics: 1) error in the maximum uncertainty, \(E-\text{UMAX}\), and 2) the maximum uncertainty-error at a FF, \(\text{MAXE-FF}\). \(E-\text{UMAX}\) is obtained by first computing \(\text{UMAX}\), the maximum over uncertainties at all the flip-flop clock pins (i.e., \(\text{UMAX} = \max_{FF} \{U(D_i)\}\)), using \(M\) and then comparing it with the \(\text{UMAX}\) computed by \(G\). \(\text{MAXE-FF}\) is calculated by first computing the percentage error in uncertainty at each FF under \(M\) with respect to the golden uncertainty value at that FF and then picking the maximum percentage error value over all the FFs. Note \(\text{MAXE-FF}\) \(\geq E-\text{UMAX}\).

Since we use Monte Carlo simulations to compute timing uncertainties, the accuracy of results depends on the number of simulations. More simulations usually mean higher accuracy. Since it was not feasible for us to run a large number of simulations due to limited CPU resources, we did an experiment to determine the number of simulations that yield uncertainty values within 10% accuracy, where the golden result used 800 simulations. It turned out that running 400 simulations resulted in \(\text{MAXE-FF}\) of about 5.5% with respect to the golden result, whereas with 100 simulations, we obtained \(\text{MAXE-FF}\) of about 16%. So we use 400 simulations in all our MCS runs (unless stated otherwise).

We present results for two architectures: pure mesh with no global tree, and complete clock network with mesh and global tree.

5.2 Pure Mesh

First, we study effectiveness of clock mesh in mitigating uncertainty. Then, we investigate accuracy of SWS-based uncertainty analysis methodology. In both experiments, only the mesh along with mesh buffers was modeled and simulated. The global tree was not explicitly included in the model.

5.2.1 Effectiveness in Uncertainty Mitigation

Although the global tree was not explicitly included in the model, different values for maximum skew and uncertainty were used on the inputs of the mesh buffers. These model the skew and uncertainty due to the global tree driving the mesh. The mesh buffer inputs were assumed to be independent Gaussian distributions with mean clock arrival times satisfying the maximum skew and standard deviation \(\sigma\) related to uncertainty.

To compare uncertainty results, we use two metrics: 1) error in the maximum uncertainty \((E-\text{UMAX})\) and 2) the maximum uncertainty-error at a FF \((\text{MAXE-FF})\). \(E-\text{UMAX}\) is obtained by first computing \(\text{UMAX}\), the maximum over uncertainties at all the flip-flop clock pins (i.e., \(\text{UMAX} = \max_{FF} \{U(D_i)\}\)), using \(M\) and then comparing it with the \(\text{UMAX}\) computed by \(G\). \(\text{MAXE-FF}\) is calculated by first computing the percentage error in uncertainty at each FF under \(M\) with respect to the golden uncertainty value at that FF and then picking the maximum percentage error value over all the FFs. Note \(\text{MAXE-FF} \geq E-\text{UMAX}\).

Since we use Monte Carlo simulations to compute timing uncertainties, the accuracy of results depends on the number of simulations. More simulations usually mean higher accuracy. Since it was not feasible for us to run a large number of simulations due to limited CPU resources, we did an experiment to determine the number of simulations that yield uncertainty values within 10% accuracy, where the golden result used 800 simulations. It turned out that running 400 simulations resulted in \(\text{MAXE-FF}\) of about 5.5% with respect to the golden result, whereas with 100 simulations, we obtained \(\text{MAXE-FF}\) of about 16%. So we use 400 simulations in all our MCS runs (unless stated otherwise).

We present results for two architectures: pure mesh with no global tree, and complete clock network with mesh and global tree.

5.3 Complete Clock Network

Having established that SWS is accurate for analyzing the timing uncertainty of a pure clock mesh, we now investigate if the SWS-based uncertainty analysis can handle the clock network of Figure 1, which includes, in addition to the mesh, a global tree that drives the mesh through mesh buffers.
The impact of border on SWS accuracy for mesh uncertainty analysis is shown in Figure 6. The table below summarizes the results for different mesh sizes and window sizes.

<table>
<thead>
<tr>
<th>Mesh Size</th>
<th>Window Size</th>
<th>E-UMAX (ps)</th>
<th>UMAX (ps)</th>
<th>% Error</th>
<th>E-UMAX (%)</th>
<th>UMAX (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8x8</td>
<td>2</td>
<td>34.29</td>
<td>14.74</td>
<td>57.03</td>
<td>32.22</td>
<td>6.04</td>
</tr>
<tr>
<td>16x16</td>
<td>4</td>
<td>33.3</td>
<td>7.62</td>
<td>77.13</td>
<td>32.49</td>
<td>2.43</td>
</tr>
</tbody>
</table>

Table 3: UMAX & E-UMAX for tree-mesh decoupling

One straightforward way of analyzing uncertainty of the complete clock network with SWS is to include the entire tree for each location of the window in SWS-based Monte Carlo simulations. Though accurate, this scheme is time consuming, memory intensive and wasteful, since it re-analyzes the same tree for each window location. If we can decouple the tree uncertainty analysis from the mesh analysis and carry out the two separately, the complete clock network uncertainty analysis can be sped up, using less memory as well.

To ascertain the validity of decoupling for analyzing timing uncertainty, we carried out the following comparison. The golden methodology comprised of running MCSs on the entire monolithic clock network model (with global tree and mesh together), and measuring the uncertainty at each FF. The methodology corresponded to decoupling the tree and mesh analyses. It comprised of running MCSs on the global tree, deriving mean and standard deviation of the clock arrival time at the input of each mesh buffer, and using them as inputs to the mesh uncertainty analysis. One single simulation model was created for the mesh. The mean and standard deviation of the latency at the input of a mesh buffer are the same as those derived from the global tree analysis. Moreover, the latency variables at the mesh buffer inputs are assumed to be independent Gaussian variables. The mesh uncertainty analysis computes uncertainty at every FF. The comparison of \( M \) and \( G \) results is shown in Tables 3 and 4 for two mesh sizes (8x8 and 16x16) for a 5mm x 5mm chip having 1000 FFs placed with a uniform random distribution. Table 3 shows UMAX, the maximum error in uncertainty (i.e., E-UMAX), and Table 4 shows results for the flip-flop with maximum error in uncertainty (i.e., MAXE-FF). For the 8x8 mesh case, 11.48ps is the uncertainty \( U \) (with the decoupled methodology) of the FF with maximum error, whereas its golden uncertainty is 30.6ps, resulting in a percentage error of 62.5%. MAXE-FF for the 16x16 mesh is even larger: 82.5%.

The reason for such huge errors is that the latency variables at the mesh buffer inputs are not all independent: they are correlated to each other. Correlation between the latency variables at two mesh buffers depends on the tree edges shared between the paths from the clock tree root to the two buffers. This is shown in Figure 7, where the paths from the root A to mesh buffers X and Y share edges AB and BC. Each of these edges contributes the same delay to the two paths. This is not considered in the independent variable assumption.

One way to incorporate common path correlations at mesh buffer inputs is as follows. The tree uncertainty analysis generates delay distribution for each stage of the global clock tree (e.g., mean \( \mu_{AB} \) and standard deviation \( \sigma_{AB} \) of the delay of edge AB in Figure 7).

### Decoupling Tree Analysis and Mesh Analysis

#### 5.3.1 Decoupling Tree Analysis and Mesh Analysis

To ascertain the validity of decoupling for analyzing timing uncertainty, we carried out the following comparison. The golden methodology comprised of running MCSs on the entire monolithic clock network model (with global tree and mesh together), and measuring the uncertainty at each FF. The methodology corresponded to decoupling the tree and mesh analyses. It comprised of running MCSs on the global tree, deriving mean and standard deviation of the clock arrival time at the input of each mesh buffer, and using them as inputs to the mesh uncertainty analysis. One single simulation model was created for the mesh. The mean and standard deviation of the latency at the input of a mesh buffer are the same as those derived from the global tree analysis. Moreover, the latency variables at the mesh buffer inputs are assumed to be independent Gaussian variables. The mesh uncertainty analysis computes uncertainty at every FF. The comparison of \( M \) and \( G \) results is shown in Tables 3 and 4 for two mesh sizes (8x8 and 16x16) for a 5mm x 5mm chip having 1000 FFs placed with a uniform random distribution. Table 3 shows UMAX, the maximum error in uncertainty (i.e., E-UMAX), and Table 4 shows results for the flip-flop with maximum error in uncertainty (i.e., MAXE-FF). For the 8x8 mesh case, 11.48ps is the uncertainty \( U \) (with the decoupled methodology) of the FF with maximum error, whereas its golden uncertainty is 30.6ps, resulting in a percentage error of 62.5%. MAXE-FF for the 16x16 mesh is even larger: 82.5%.

The reason for such huge errors is that the latency variables at the mesh buffer inputs are not all independent: they are correlated to each other. Correlation between the latency variables at two mesh buffers depends on the tree edges shared between the paths from the clock tree root to the two buffers. This is shown in Figure 7, where the paths from the root A to mesh buffers X and Y share edges AB and BC. Each of these edges contributes the same delay to the two paths. This is not considered in the independent variable assumption.

One way to incorporate common path correlations at mesh buffer inputs is as follows. The tree uncertainty analysis generates delay distribution for each stage of the global clock tree (e.g., mean \( \mu_{AB} \) and standard deviation \( \sigma_{AB} \) of the delay of edge AB in Figure 7).
E-UMAX is about half of the mesh dimension (e.g., 8x8 window for 16x16 mesh), UMAX and MAX-E-FF values respectively. With window dimension the decoupled tree and mesh analyses with correlations, using SWS 1-2% change in the percentage error with border is only 0.3-0.7ps, mesh and 647ps for the 16x16 mesh. As for the impact of border, the is really small, given nominal clock latencies of 570ps for the 8x8 35ps range. A 12% error in uncertainty translates to about 4ps, which Golden UMAX clock latency (around 570-645ps).

Since our methodology is based on SWS, it is capable of analyzing uncertainty of large meshes and design instances, and is easily amenable to distributed- or grid-computing.

Future work is in the following directions. 1) Running several hundred MCSs on a large design & fine mesh can be time consuming if hundreds of compute-servers are not available. We plan to work on making our methodology faster. 2) For the complete clock network, the decoupling method should handle correlations between consecutive stages on a path. 3) In this work, we modeled variation sources for each wire and buffer independently. However, supply voltage and temperature of components located close to each other are usually correlated. We will extend our model to handle these correlations.

### References


