MeshWorks: A Comprehensive Framework for Optimized Clock Mesh Network Synthesis

Anand Rajaram, *Member, IEEE,* and David Z. Pan, *Senior Member, IEEE*

***Abstract*—Clock mesh networks are well known for their variation tolerance. But their usage is limited to high-end designs due to the significantly high resource requirements compared to clock trees and the lack of automatic mesh synthesis tools. Most existing works on clock mesh networks either deal with semi-custom design or perform optimizations on a given clock mesh. However, the problem of obtaining a good initial clock mesh has not been addressed. Also, the problem of achieving a smooth tradeoff between variation tolerance and resource requirements has not been addressed adequately. In this paper, we present our *MeshWorks* framework, the first comprehensive automated framework for planning, synthesis, and optimization of clock mesh networks that addresses the above issues. Experimental results suggest that our algorithms can achieve an additional reduction of 31% in buffer area, 21% in wirelength, and 23% in power, compared to the best previous work, with similar worst case maximum frequency. We also demonstrate the effectiveness of our framework under several practical issues such as blockages, multiple clocks, uneven load distribution, and electromigration violations.**

***Index Terms*—Clock mesh optimization, clock mesh synthesis, robust clock networks, variation tolerant clock network synthesis.**

I. Introduction and Motivation

As the VLSI technology continues to 65 nm and below, the effects of manufacturing variation, power supply noise, temperature variations, and so on, on clock skew are becoming more significant. Clock network is especially sensitive to such variation effects, resulting in unwanted skews. Since higher skews directly reduce the maximum frequency of the circuit, reducing the clock skew variation can improve timing yield. Some of the approaches proposed to reduce clock skew variation are variation aware buffer/wire sizing [11], variation aware routing [12], link insertion in clock trees [13], and leaf-level meshes [1], [3]–[7]. Among different methods suggested for skew variation reduction, the leaf-level mesh with a top-level tree has been shown to be very effective in reducing skew variation in several commercial chips as noted in [1] and [8]. The variation tolerance of a leaf-level mesh is a direct result of its high redundancy, with multiple source to sink paths for every sink. Fig. 1 shows an example of a clock network with top-level tree and leaf-level mesh.

The high resource requirements of a leaf-level clock mesh have historically restricted its use to a few high-end products such as microprocessors [1], [3]–[6]. However, with variation becoming a bigger issue at 65 nm technology and below, even non-microprocessor chips might consider the use of a clock mesh to improve yield. Nevertheless, most non-microprocessor chips still cannot use a leaf-level mesh because of two reasons. First, as noted above, the resource requirements (wirelength, buffers, and power) might be prohibitively high. Second, there is a lack of automatic mesh planning/synthesis and optimization tools to help achieve the design objectives without manual effort [2]. Since application specific integrated circuits (ASICs) typically have much shorter turn-around times than microprocessor chips, they cannot afford to have a manually planned and optimized clock mesh. In fact, the lack of research on automated clock mesh synthesis was noted as early as in 2001 [10]. However, no comprehensive work has been done on this important topic in the literature, to our best knowledge. Even in the recent tutorial on clock distribution networks [8], no systematic method has been presented for mesh planning[1] or optimization. Thus, to make clock mesh a viable option for non-microprocessor chips, a fully automated framework for mesh planning, synthesis, and optimization is needed. Such a framework can enable chip teams to achieve a smooth tradeoff between performance (skew) and power (area).

It may be noted that fully automated clock mesh planning/synthesis and optimization will be very useful to microprocessor chips as well. For example, automated mesh planning/synthesis can be used to get the preliminary clock mesh after which finer adjustments can be made manually. Similarly, mesh optimization can be performed on the individual *grid zones*[2] [8] to reduce power/resources used. The potential difference on the use of such automated methods between microprocessor and other chips lies in their respective *resource vs. skew* tradeoff. While microprocessors might opt for maximum power reduction with a strict skew requirement, other chips might opt for minimum skew with a strict power/resource target.

Review of existing works: Next, we briefly review the existing works on clock mesh. The works of [1] and [3]–[5]

Manuscript received March 23, 2010; revised June 25, 2010; accepted June 25, 2010. Date of current version November 19, 2010. This work was supported in part by the National Science Foundation, in part by the SRC, and in part by the IBM Faculty Award. This paper was recommended by Associate Editor I. Markov.

A. Rajaram is with Magma Design Automation, Austin, TX 78731 USA (e-mail: anandrajaram@ieee.org).

D. Z. Pan is with the ECE Department, University of Texas at Austin, Austin, TX 78712 USA (e-mail: dpan@ece.utexas.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2061130

[1] It is called *grid floor-planning* in [8].

[2] The individual sub-grids driving small zones of a chip.

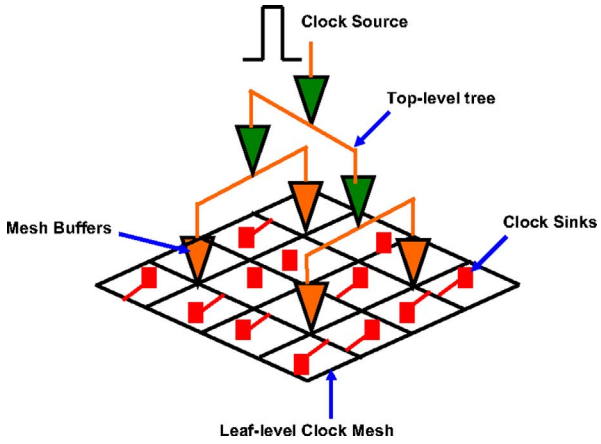

Fig. 1. Clock network with top-level tree driving a leaf-level mesh. The clock sinks are attached to the mesh.

deal with custom/semi-custom mesh design and do not address the problem of automatic mesh synthesis/optimization. The works of [6] and [7] perform optimizations on a *given* clock mesh. However, the problem of obtaining a good initial clock mesh that can be optimized has not been addressed. The work of [9] deals with the synthesis of hybrid clock network with a top-level mesh and bottom-level tree. Since the size of the bottom-level trees are not negligible, they will still have considerable skew variation. The work of [6] performs clock mesh sizing considering only the nominal skew targets and ignores variation. The works in [14] and [15] present efficient methods for clock mesh analysis and they do not deal with clock mesh synthesis.

To our best knowledge, [7] is the first work that aims to achieve a *variation tolerance vs. wirelength* tradeoff in a clock mesh. Given a clock mesh and buffer library, [7] uses a set-cover formulation to obtain the minimum buffer resource to drive the mesh under slew constraints. Using this buffered mesh, [7] applies network survivability theory (used in computer networks) to remove some of the mesh segments without significantly affecting variation tolerance. The heuristic in [7] makes sure that every sink has at least a certain number of paths from certain number of buffers within a given distance in the clock mesh. The edges not present in short paths of any sink are removed, resulting in an optimized clock network. Though the work of [7] is efficient, it has a few key drawbacks as summarized below.

1) It does not consider the problem of initial mesh planning/synthesis and relies on manually selected mesh for performing optimizations.
2) The network survivability formulation requires a fixed number of buffers to be connected to each sink through a fixed number of non-overlapping paths. For example, it does not differentiate between the presence of a lightly loaded or heavily loaded buffer at a given point. Hence, the effect of differential loading on buffer delays is not modeled by the mesh optimization formulation.
3) Electrical characteristics of mesh buffers are ignored and all buffers are treated identically during mesh optimization. Also, the interaction between mesh reduction and buffering is ignored.
4) Electromigration (EM) issue is totally ignored. The work of [6] considers EM requirements as a constraint during mesh sizing. In the context of mesh optimization by removing mesh segments, the inverse problem of solving existing EM violation with minimum additional wire-area is needed.

In this paper, we attempt to address all these drawbacks. The key components of our MeshWorks framework are the following.

1) *Mesh planning and synthesis:* a simple yet effective method for planning and synthesis of a buffered clock mesh for a given set of design constraints. This method can choose a good initial mesh, which can be further optimized for power/resource reduction.
2) *Mesh optimization:* an efficient algorithm using *network sensitivity theory* to remove mesh edges with little impact on skew variability. This formulation is more accurate than the work of [7] because the mesh delay sensitivities are directly considered during optimization.
3) *Buffer modeling for mesh optimization:* an efficient buffer modeling method that is especially suitable for use during clock mesh optimization.
4) *Wire sizing for reliability:* an effective heuristic that sizes relatively few mesh segments to meet the EM constraints of the optimized mesh.

The above contributions make MeshWorks the first comprehensive framework for complete automation of clock mesh networks synthesis and optimization.

II. Mesh Planning and Synthesis

The mesh planning and synthesis problem can be stated as follows.

1) *Given*: sink locations and load capacitance, buffer library, interconnect parameters, variation models, nominal/variational skew targets.
2) *Problem*: obtain an initial clock mesh with minimum routing and buffering resources such that the given design constraints are likely to be satisfied. It shall be noted that our objective is not to get a final clock mesh, but to *quickly* get a good mesh that can further be optimized using the algorithm presented in Section III.

A. Terms and Definitions

Here, we define a few common terms to facilitate our discussions.

1) $S = \{s_1, s_2, ...s_N\}$ is the set of all N clock sinks, where s_i denotes the ith sink.
2) $B = \{b_1, b_2, ...b_T\}$ is the set of all T buffer sizes in the library with the *buffers numbered in non-decreasing order of size/drive strength*. For each buffer size b_p, the maximum load that can be driven under a given max-slew constraint Max_Slew is denoted by CL_p^{max}.
3) Let $D_q(Cap)$ be the delay at the output of a buffer of size q $(1 \leq q \leq T)$ when it drives a load cap of value Cap.
4) Let $IntDel(l, C)$ denote the delay when an interconnect of length l drives a load capacitance of value C.

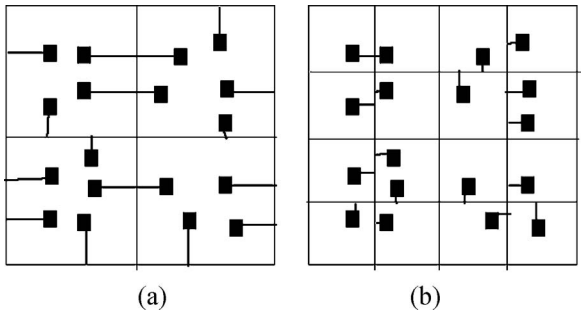

Fig. 2. Examples of (a) sparse and (b) dense clock meshes. A dense mesh is likely to have shorter stubs.

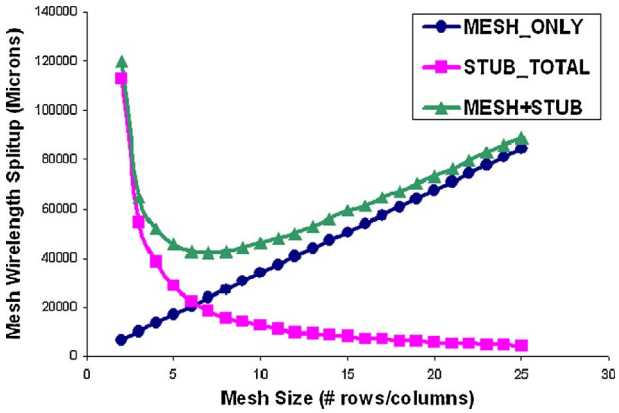

Fig. 3. Determining the right mesh size.

5) The leaf-level mesh, by definition, covers the entire chip area spanned by all the sinks. The X,Y dimensions of the chip area are given by X_{bound},Y_{bound}. Mesh size is defined by the number of horizontal, vertical segments denoted by m,n. Such a mesh will have $m * n$ nodes, numbered sequentially from 1 to $m * n$.
6) Each clock sink s_i is attached to the nearest mesh node using an interconnect called *stub* of length $L^i{}_{stub}$.
7) The buffers directly driving the mesh are called mesh buffers.

B. Total Wirelength as a Function of Mesh Size

The total wirelength of the clock mesh along with the stubs can be written as the following:

$$L_{tot} = L_{mesh} + L_{stub} = m * X_{bound} + n * Y_{bound} + \sum_{i=1}^{N} L^i{}_{stub}. \quad (1)$$

The wirelength of the mesh itself is a linear function of mesh size. Let us now consider the effect of increasing the mesh size on the sum of wirelengths of all the stubs. As either m or n increases, a randomly chosen sink is more likely to have closer horizontal or vertical mesh segment. Since the maximum number of stubs is constant, it is very likely that the total stub length decreases. In a sparse mesh, the mesh wirelength is less when compared to the dense mesh. However, the total stub wirelength is likely to be more for a sparse mesh than a dense mesh because each sink needs to be connected to the nearby mesh point using a longer interconnect. Fig. 2 illustrates this fact with a simple example.

Fig. 3 shows how the wirelength changes as a function of mesh size in one of our testcases. The key point to be noted is that it is easy to get a plot similar to Fig. 3 for a given set of sinks even though the shape of wire-length function might differ. From such a plot, choosing an appropriate mesh size or size range that fits our "wirelength vs. mesh-size" tradeoff requirement is trivial.

C. Skew as a Function of Mesh Size

Skew variation is typically a decreasing function of mesh size because of two factors. First, the mesh itself becomes more dense, resulting in more redundancy, making it more tolerant to variations. Second, the length of the stub also decreases, resulting in reduction of the maximum possible uncontrolled delay variation. In general, skew in a given mesh can be expressed as a sum of three components as follows:

$$\begin{aligned} Sk_{bound} = {} & [Max(D_p(CL_p^{max})) - Min(D_q(CL_{q-1}^{max}))] \\ & + Delay(D_{max}) + IntDel(L_{stub}^{max}, C_L^{max}) \\ & \forall p, q : 1 \le p, q \le T \end{aligned} \quad (2)$$

where $L_{stub}^{max} = Min(\frac{X_{bound}}{2n}, \frac{Y_{bound}}{2m})$ gives the maximum length of any stub when the chip area of dimension X_{bound}, Y_{bound} is divided equally into m rows and n columns, C_L^{max} gives the maximum value of sink load capacitance for the given set of sinks and D_{max} is the maximum distance between a sink and the nearest mesh buffer.

The first component in (2) is the skew due to the differential loading/sizing of the mesh buffers. This is the difference between the maximum delay of any buffer in the library under its maximum loading condition and the minimum delay of any buffer in the library under the maximum loading condition of the *previous sized buffer* $(q-1)$. A previous sized buffer for a given buffer is the nearest smaller buffer when all buffers are sorted in ascending order of drive strength. We can consider a load of CL_{q-1}^{max} to be a lower bound of the load for buffer b_q because we assume the smaller sized b_{q-1} will be used whenever the load is smaller than what buffer b_{q-1} can drive to save buffer area. Please refer to the end of Section II-C on assumptions made in this regard. Thus, this term gives a tight upper bound for the maximum skew that can be introduced in the mesh due to differential buffer loading. As stated in [1], uneven buffer loading is one of the most important reasons for the skew in a clock mesh and is typically the most dominant part of the skew.

The second component in (2) is because of the difference in proximity of each sink to the buffer that is closest to it. Due to the redundancy of the mesh, this component will be usually small for a well-driven mesh satisfying the slew requirements. If D_{max} is the maximum distance for a given buffered clock mesh, then maximum skew is equal to the delay in the segment itself. This corresponds to the worst case situation where a sink is located right next to a mesh buffer, while another is located at a distance of D_{max} from the same buffer with all other components being identical.

The third component in (2) is due to the difference in the stub lengths and load capacitance. This component can be

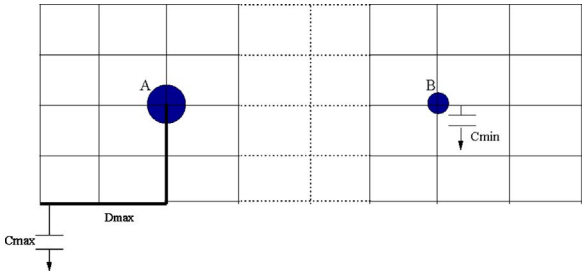

Fig. 4. Three dominant skew components in a mesh: skew due to buffer delay imbalance, skew due to difference in distance from closest buffer, and skew due to different stub length and load capacitance.

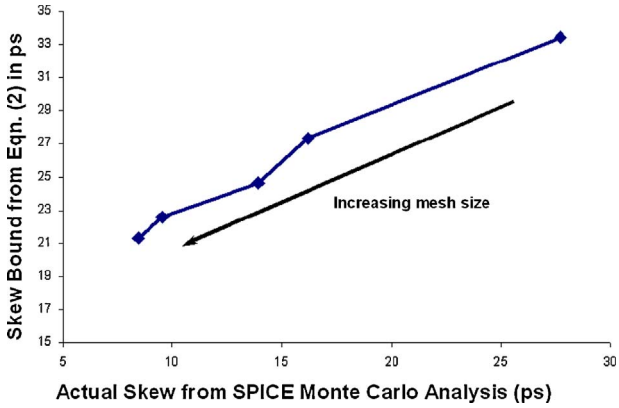

Fig. 5. Plot showing the fidelity of the skew bound (2). Though skew bound is not perfectly linear of actual worst case skew, it is monotonic.

significant because it is uncontrolled by the redundancy of the mesh. It represents the worst case skew that can be caused when one of the sinks is located on the mesh itself and the other sink with maximum load capacitance is connected to the mesh using a stub of maximum length. Fig. 4 illustrates the situation in which all the three factors discussed above might combine, resulting in maximum skew between two sinks shown. For the first case, a big buffer drives a big load capacitance that is located at a distance D_{max} from the buffer. For the second case, a small buffer drives a small capacitance located right next to it.

Among the skew components, the first component depends only on the buffer library and sets a practical limit on the skew obtainable using the given set of library buffers. The third component depends only on the mesh size and, hence, can be obtained for a given mesh size once we get the plots in Fig. 3. However, to accurately evaluate the second skew component, the precise location of mesh buffers should be known. But buffer locations cannot be known unless we choose the mesh size. Thus, there is a chicken and egg problem in accurate estimation of the second component.

For a given set of library buffers and slew requirements, as the mesh is made denser, there will be addition of more mesh buffers to satisfy the slew requirements. Thus, for a randomly selected sink, the location of the nearest buffer is likely to be proportionately closer as we increase the mesh density. Another useful observation is that the value of L_{stub}^{max} scales in the same general way as the value of D_{max} as the size of the mesh is increased. Thus, we can approximate the value of D_{max} by a scaled factor of L_{stub}^{max}, where the scaling factor is a function of the buffer library and the mesh buffer placement/sizing algorithm. The value of scaling factor can be estimated based on a few experiments and used for estimating the skew bound subsequently. Though this approach is an approximation and we can find corner cases where this observation need not be true, our experiments on several benchmark circuits show that this assumption is valid in practice. Also, the choice of buffer placement/sizing algorithm influences the accuracy of this approximation. For example, if the buffer placement/sizing is done in such a way that buffers are placed close to sinks, then the second factor can even be neglected from skew bound analysis. Our buffer placement/sizing algorithm, discussed in Section II-D, enables us to achieve that.

Fig. 5 shows the plot between the skew bound estimated using the above approximation and the accurate skew obtained by running SPICE Monte-Carlo analysis on one of our benchmark circuits. As we can see, the skew bound, though not perfectly linear, is still monotonic w.r.t. the changes in the actual worst case skew and, hence, has high fidelity. We observe similar curves for all our other benchmark circuits.

Procedure: *Obtain_Initial_Mesh_Size*
Input: $L_{max}, L_{min}, S_{max} \Rightarrow$ Min, Max Wirelength & Skew target.
1. Get m, n such that total wirelength from (1) is $\simeq L_{min}$.
2. Using the current m, n, obtain L_{tot} using (1).
3. If ($L_{tot} \geq L_{max}$)
   Print "Relax design constraints"
   Quit.
4. Obtain value of Sk_{bound} from (2)
5. If ($Sk_{bound} \leq S_{max}$)
   Return m, n. Stop.
   Else
   Increment values of m, n by 1 and go to step 2.

Fig. 6. Top-level algorithm of selecting the initial mesh size.

Thus, (2) can be used to get a high fidelity estimation of skew bound for a mesh of given size. Because of the closed form nature of this equation, skew bounds for a given mesh size can be estimated quickly under the assumptions discussed above. The steps to obtain the size of the initial mesh are summarized in Fig. 6. We start off with mesh sizes near minimum total wirelength in Step 1 of Fig. 6 because it enables us to reach a mesh size with minimum size that also satisfies the skew limits. Also, in Step 5 (else part), we increment both the values of m and n by 1. This assumes that the density of sinks in both horizontal and vertical directions are similar. If this is not the case, then m and n can be incremented independently depending on whether the current mesh segment density is lower in the horizontal or vertical directions.

In practice, the value of S_{max} parameter used as input to Fig. 6 should be chosen such that it is not too tight. This is because the value of Sk_{bound} obtained from (2) is always pessimistic since it is a bound for the worst possible skew for a given mesh size.

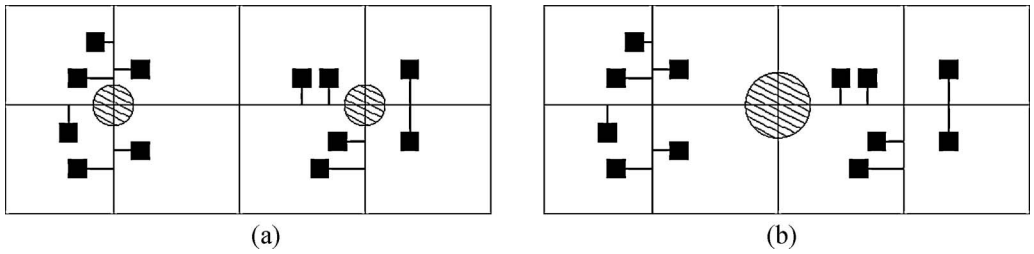

Fig. 7. Example where the buffer insertion algorithm of [7] might not take the better choice. The shaded circles represent buffers of proportional size. (a) New buffering solution. (b) Solution of [7].

A note on (2): It may be noted that (2) inherently makes the following assumptions.

1) Several buffers of incrementally different sizes/drive strengths are available to make the target skew physically possible. As noted in [17], most practical libraries will have hundreds of different buffer sizes to choose from. Hence, this assumption is valid in practice.
2) The buffer placement/sizing is done such that the smallest buffer that can drive a given set of loads will be used. In other words, we assume that all the buffers in the library have a valid capacitance range, which is used to choose the smallest buffer for a given load. This assumption is also valid in practice as power/area reduction is a key objective of any clock network synthesis algorithm.

D. Mesh Optimization Friendly Buffer Placement/Sizing

The buffer insertion heuristic of [7] has two main drawbacks. First, the potential impact of buffer insertion on mesh optimization is not considered. This might result in buffer insertion at nodes that could have been optimized away if the buffer were not present. Second, the cost function used in the set-cover formulation of [7] ignores the low-pass filter characteristics of an RC mesh [14], [15]. For an RC mesh, the attenuation of a ramp signal applied at a given node increases exponentially as a function of distance from the node. This attenuation is constant for a given clock frequency. Hence, inserting several small buffers distributed throughout the clock mesh instead of fewer big buffers might result in lesser buffer area and improve slew at the clock sinks. This is illustrated in Fig. 7. The solution in Fig. 7(a) uses two smaller buffers to drive the same amount of load instead of one big buffer in Fig. 7(b). Considering the attenuation characteristics of an RC mesh, the solution in Fig. 7(a) will result in lesser slew rate for a given buffer area. In other words, for a given slew requirement at the clock sinks, the solution in Fig. 7(a) will result in lesser buffer area. However, the work of [7] might randomly pick one of them.

To address the above drawbacks, we propose the following cost function for the greedy set-cover algorithm of [7]. The cost of inserting a buffer of size p at node i of the clock mesh is given as follows:

$$Cost_i^p = \frac{{b_p}^2}{{b_T}^2} * \frac{1}{N_{uncov}} * \frac{1}{C_{Load}^i} \tag{3}$$

where b_T is the biggest buffer in the given library, N_{uncov} is the number of uncovered nodes that can be covered by the buffer under consideration, and C_{Load}^i is the value of capacitance at the mesh node i, including the capacitance of all the sink nodes attached to it. The advantages of using the new cost function are the following.

1) Use of $\frac{{b_p}^2}{{b_T}^2}$ term instead of b_p term of [7] forces the cost of several small buffers to be less than the cost of one big buffer even if the two solutions have the same total area. Thus, this cost function indirectly considers the RC attenuation effect of the mesh.
2) The $\frac{1}{C_{Load}^i}$ term lowers the cost of adding a given buffer closer to the sinks even if coverage can be done from a farther node. This reduces the RC attenuation by placing the buffers closer to the sinks. Also, this makes the buffer locations *optimization friendly* as the edges connected to buffers are less likely to be removed because of the close proximity to the sinks. Section III-D has more details on this.
3) Similar to the work of [7], the cost function is inversely proportional to the number of new, uncovered mesh nodes that can be covered by the buffer under consideration.

The other aspects of the set-cover formulation are same as in [7]. It may be noted here that the above cost function can lead to lower slews at the clock sinks compared to the cost function of [7] since more smaller buffers are placed closer to the sinks. This can allow the buffering algorithm to aim for a relaxed slew target to get a mesh comparable to [7] in terms of the maximum slews at the clock sinks. For example, let us assume that by using the cost function of [7], we aim to get 50 ps slew in the buffering algorithm and we get 50 ps actual slew based on SPICE analysis. Then, we may be able to aim for a higher slew target (say 60 ps) and still get a clock mesh with all sinks below 50 ps slew when we use the new cost function.

Impact of mesh buffer placement on top-level clock tree: The increased number of mesh buffer from the above buffer placement method might increase the wirelength of the top-level clock tree. However, this effect is compensated by two opposite effects that are explained next. Comparing the situations in Fig. 7(a) and (b), case (b) will have fewer mesh buffers and, hence, lesser top-level wirelength. However, the capacitance of the single end point will be high because of bigger buffer. Also, the mesh optimization cannot remove the edges that connect the big buffer to the two clusters of sinks on either side, increasing the wirelength of the mesh. In contrast, the situation in case (a) has more end points

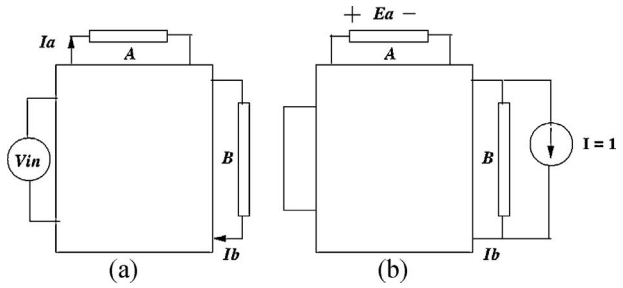

Fig. 8. Network sensitivity theory can be applied for clock mesh optimization. (a) Original network. (b) Auxiliary network.

in the top-level, which can increase the top-level wirelength. However, the total pin capacitance is lower than in the first case because, to achieve comparable slew rates at the sinks, case (a) can use smaller buffers with lesser total area. Also, several extra mesh edges can be optimized away, resulting in lesser mesh wirelength. Thus, case (a) reduces both the top-level tree pin capacitance and the mesh wirelength at the cost of increasing the top-level tree wirelength. In many cases, the increase in top-level wirelength might be compensated by the potential new mesh edges that can be removed. For example, let us assume that each big mesh buffer is replaced by two smaller buffers due to the new cost function as in Fig. 7. In this case, the increase in top-level tree wirelength is bounded by the distance connecting the old mesh buffer location to the locations of the two new buffers. This is because any old tree driving the original mesh buffers can be extended along the mesh edges to reach the new mesh buffers. Thus, for the example in Fig. 7, the maximum increase in top-level tree wirelength is two edges, one each to connect the old buffer location in Fig. 7(b) to the two new buffers in Fig. 7(a). However, the potential number of new optimization that can done on this mesh due to the new buffer locations is clearly more than two edges. Thus, the increase in top-level tree wirelength can be reduced back in mesh optimization, if required, to obtain a mesh that is more comparable to the original mesh in terms of wirelength.

III. Network Sensitivity Based Mesh Optimization

In this section, we will first review the concepts of network sensitivity theory that is at the root of our mesh optimization approach. Next, we present our efficient buffer model that is used during the mesh optimization. Finally, using these concepts, we present our network sensitivity based mesh optimization algorithm.

A. Network Sensitivity Theory

Given a RC network, network sensitivity theory aims to efficiently evaluate sensitivities of a given output parameter (voltage or current) to changes in the circuit parameters. A straight forward and inefficient method to obtain the sensitivities is to perturb each circuit parameter and observe the changes in the output. However, in the case of RC networks with no active elements, the sensitivities of a given output can be obtained w.r.t. *every* parameter in the network using the method of [18] without perturbing any circuit parameters.

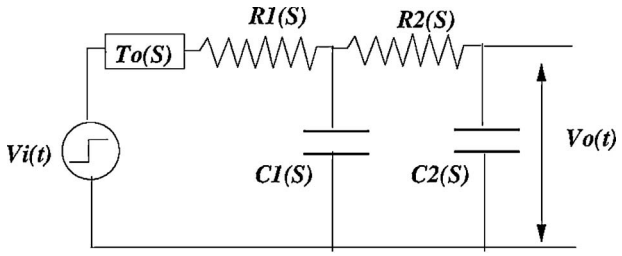

Fig. 9. Accurate buffer used for clock mesh optimization. S is the size of a given buffer.

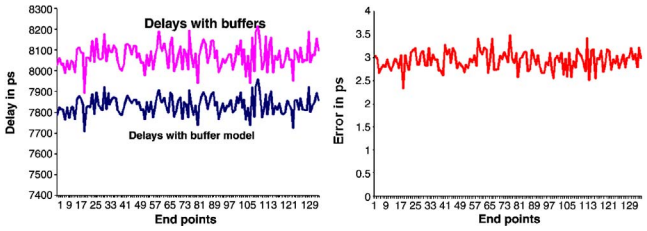

Fig. 10. Comparison of sink delays in SPICE obtained using buffers and the buffer model for a clock mesh testcase.

Consider [18, Fig. 8], which shows a generic electrical network with three identified elements for illustrative purposes. The elements can be any of the passive components such as R, C, and L. Let I_A and I_B be the currents through these elements in the nominal circuit. The element V_{in} represents all the sources in the RC network. Let the voltage across element B be considered as the output of this network. According to [18], to obtain the sensitivities of the output voltage w.r.t. all the parameters of the circuit, *irrespective of the number of circuit parameters*, we need to construct an auxiliary network for the original network as follows: all the independent current sources are opened, voltage sources are shorted, a unit current source is applied across the element B, and the voltages across all the components in the network are measured. According to [18], the relationship between the currents of the original network, element values, and voltages in the auxiliary network is given as follows:

$$E_a = \frac{\partial E_b}{\partial A}\frac{A}{I_A} \tag{4}$$

where E_a is the voltage across any element in the auxiliary network, $\frac{\partial E_b}{\partial A}$ is of sensitivity of the output voltage E_b w.r.t. parameter A (the required value), and I_A is the current flowing through the element A in the original network. Thus, using only two simulations, the sensitivities of a given output w.r.t. all the network parameters can be obtained, irrespective of the number of parameters. Though the method of [18] is efficient when compared to the perturbation method, it still requires one simulation for each output. Thus, a direct application of this method is not practical for multioutput networks, such as clock networks. Our method to overcome this drawback is explained in Section III-C.

B. Accurate Buffer Modeling for Mesh Optimization

The sensitivity calculation method of [18] can be applied only for a passive network. To apply the concepts of sensitivity theory to a clock mesh, all the clock buffer must be modeled using a combination of voltage/current sources

and passive elements such as resistors and capacitors. The typical switch resistance modeling of buffers is becoming increasingly inadequate to approximate the buffers in the sub 100 nm technologies. This inadequacy is compounded by the inherent difficulty in modeling the effective capacitance of a mesh because of its multiple paths and multiple drivers that can possibly interact in highly non-linear fashion. Thus, we need a buffer model that is accurate, independent of the load and also captures the non-linear behavior of the buffers. The buffer model proposed in [19] satisfies all these requirements. The basic idea behind the work of [19] is the use of a two-pole approximation for modeling a buffer instead of the single pole approximation of a switched resistance modeling. As a result, it can be characterized almost independent of the load that it drives and captures the non-linear behavior of the buffers. An example of this model is shown in Fig. 9 where S is the size of the buffer. The values of $R1$, $R2$, $C1$, $C2$ are obtained by using the OPTIMIZE function in HSPICE [20] to approximate the delay characteristics of a given buffer. The parameter T_o is the constant delay that might be added to model the delay of the buffer. In this paper, we adapt the work of [19] to make it suitable for the problem approximating a library of buffers.

The work of [19] concentrates on modeling a single buffer. Though this can be trivially extended for a library of buffers, the values of the parameters $R1$, $R2$, $C1$, $C2$ can differ drastically based on the initial values used in the OPTIMIZE function of HSPICE. Ideally, we would want monotonic changes in the values of $R1$, $R2$, $C1$, $C2$ for monotonic changes in buffer sizes. This requirement is feasible under the assumption that a bigger buffer is used to drive proportionally bigger load under a given slew target. The monotonic property of $R1$, $R2$, $C1$, $C2$ parameters ensures that the any buffer resizing done with these models will be accurate. The monotonic characteristic of the RC parameters can be guaranteed by first obtaining a good approximation for either the smallest or the biggest buffer size in the library using large search space. For all the other buffers, the approximations are obtained by constraining the maximum or minimum values of $R1$, $R2$, $C1$, $C2$ to the values of the previous or next sized buffers using the OPTIMIZE function in HSPICE. From our experiments, we observed that this always preserves the monotonic nature of the parameters while resulting in accurate approximations.

Accuracy of the buffer model: Fig. 10 compares the clock sink delays for one of our mesh testcases with original buffers and the buffer models. As seen from this figure, the two delay curves track well across the clock sinks. We observe similar results for all our testcases. For all the testcases, the error because of our buffer models is around 4% for delays and 1% for skews. Thus, any optimization done using our buffer models is likely to be accurate.

C. Mesh Optimization Algorithm

To minimize the mesh wirelength without significantly affecting the variation tolerance, the mesh segments that are not critical for variation tolerance should be removed. Consider Fig. 11 where a mesh drives several clock sinks. In this case, the edges shown in dashed lines can be safely removed without significantly affecting the skew characteristics of the original mesh because they are far away from all the clock sinks. Similarly, we would like to have a dense mesh in places where the clock sink distribution is high and a sparse mesh in locations where the density is much lower.

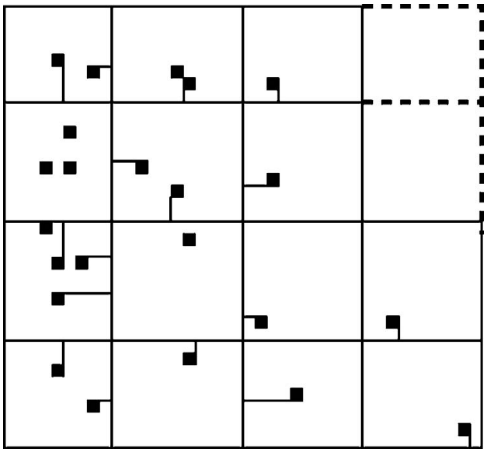

Fig. 11. Simple example of network sensitivity based mesh optimization.

In this paper, we attempt to achieve the above objectives using network sensitivity theory as explained below. Let Del_i denote the delay of a sink s_i and let a mesh segment connected between mesh nodes p and q be denoted by $Seg(p, q)$. The delay sensitivity for the sink s_i w.r.t. width $W(p, q)$ of the mesh segment $Seg(p, q)$ can be expressed as follows:

$$\frac{\partial Del_i}{\partial W(p,q)} = \frac{\partial Del_i}{\partial R(p,q)} \frac{\partial R(p,q)}{\partial W p,q} + \frac{\partial Del_i}{\partial C(p,q)} \frac{\partial C(p,q)}{\partial W p,q}. \quad (5)$$

In the above equation, the terms $\frac{\partial Del_i}{\partial R(p,q)}$ and $\frac{\partial Del_i}{\partial C(p,q)}$ are the values of delay sensitivity w.r.t. the resistance and capacitance of the mesh segment. There is no closed form expression for evaluating these terms. The terms $\frac{\partial R(p,q)}{\partial W p,q}$ and $\frac{\partial C(p,q)}{\partial W p,q}$ are the changes in resistance and capacitance values of the mesh segment as a function of width. These terms can be easily obtained though the relationship between interconnect width and resistance/capacitance values. For the simple case of $R(p, q) = \frac{R0(p,q)}{W}$, and $C(p, q) = C0(p, q) * W$, these expressions are $-\frac{R0(p,q)}{W^2}$ and $C0(p, q)$ respectively. In order to select mesh edges for removal, we first quantify the effect of removing each edge by defining the following cost function for each mesh segment $Seg(p, q)$

$$Cost(p,q) = Max\left(\frac{\partial Del_j}{\partial W(p,q)} - \frac{\partial Del_k}{\partial W(p,q)}\right) W(p,q) \quad (6)$$

$$\forall j, k \in sinks\ S.$$

The above cost function approximates the maximum change in skew in the entire mesh when a given segment is removed. The criticality of each mesh segment w.r.t. variation tolerance will be proportional to the value of cost function. The basic idea of our approach is to remove the segments that have a low cost function, resulting in an optimized mesh. However, the following sub-problems must be solved for efficient application of network sensitivity toward solving mesh optimization problem.

1) As stated in Section III-A, the method of [18] is inefficient for clock network which has many output (sink) nodes.
2) The method of [18] can be used only to obtain voltage sensitivities and cannot be directly used to obtain delay sensitivity.
3) The sensitivities for a given segment assumes that all the other mesh segments are held constant.

The above sub-problems can be easily solved when the following key observations are considered.

1) The RC mesh network behaves as a low-pass filter [14], [15], in which the attenuation of a ramp input signal applied at a given node increases exponentially as a function of distance from the source node. So, the delay sensitivities of closely located sinks will be almost the same w.r.t. most clock segments.
2) According to [21], the effects of most variation can be modeled using linear approximation without any significant effect on the accuracy. As a result, we can obtain the delay sensitivity terms of (5) using Elmore delay without accuracy loss.
3) The Elmore delay sensitivities can be obtained efficiently by evaluating the voltage sensitivities of the DC equivalent network of the original mesh network. The DC equivalent circuit can be obtained by shorting all voltage sources and replacing all the capacitances by current sources of equal magnitude [22]. The node voltages in this circuit represent the Elmore delays of the original mesh networks and the voltage sensitivities are the sensitivities of the Elmore delays.
4) The sensitivities of several output voltages in the DC equivalent circuit can be evaluated efficiently by reusing the results of LU factorization. This is because the only change made in solving the different auxiliary networks for different output nodes is the location of the unit current source [22].
5) The analysis efficiency can be further improved by exploiting the sparse nature of the nodal admittance matrix for most RC mesh networks [22].

Using the above observations, the value of cost function of (6) can be evaluated for each mesh edge efficiently.

Overall mesh optimization algorithm:

1) Identify the different sink clusters such that sinks in each cluster are closely located.
2) Obtain an approximate circuit by merging all the sinks in each cluster into a single *merged* sink with capacitance equal to the total capacitance of all the merged sinks. The resulting mesh will be a good approximation of the initial mesh as far as sensitivity calculations are concerned and will have far fewer end points compared to the original mesh.
3) Replace all the mesh buffers with the accurate buffer model values presented in Section III-B.
4) Obtain Elmore delay sensitivities of every *merged* sink w.r.t. all the mesh segments by efficient reuse of the results of LU factorization and making use of sparse matrix methods. Using the delay sensitivities, obtain the $Cost(p, q)$ for each mesh segment as defined in (6).
5) Sort mesh segments in increasing order of $Cost(p, q)$ value and remove the required number of segments to satisfy the wirelength reduction target. It may be noted here that we can sort the mesh nodes instead of mesh segments using the same framework as above and start by removing the mesh nodes with minimum cost. In this case, the cost of a mesh node can be given as a sum of cost of all mesh edges attached to that node. When a mesh node is removed, all the mesh segments attached to it are simultaneously removed.

D. Buffer-Resizing for Mesh Optimization

A key drawback of [7] is that the optimized mesh uses the same buffer placement/sizing as the initial mesh. This can result in buffer area and power wastage. In this paper, we propose an efficient buffer resizing heuristic to reduce the buffer area/power for a given optimized mesh. The main steps in our approach are as follows.

1) For each clock buffer, obtain the rectangular covering region in the mesh where the total capacitance (including sink capacitance) is less than buffer load limit under the given slew constraint.
2) For each buffer that has an overlap with another buffer, consider resizing to the previous sized buffer such that the total covering region for all clock buffers is maintained.
3) Repeat this process till there exists no buffer that can be sized down without reducing the total coverage.

The amount of buffer area reduction obtained by the above heuristic is proportional to the reduction in mesh wirelength. However, the proportional reduction in power is likely to be less because the redundant buffers in the optimized mesh were driving light loads.

IV. Wire Sizing for Reliability

As noted in [25]–[30], EM is increasingly becoming a significant issue in the deep sub-micrometer IC designs. In general, EM is relevant to clock mesh because of the significant current flowing in it. EM is especially relevant to clock *mesh optimization* because removing any mesh segment can potentially increase the current density in a nearby segment. Thus, we need a systematic approach to address any EM violations that might occur due to mesh optimization. This section proposes an efficient method to address this issue systematically. According to the empirical model developed in [31], the mean time to failure (MTTF) of a wire considering EM issue is given as follows:

$$MTTF = \frac{C}{J^n} * exp(\frac{E_a}{k.T}) \tag{7}$$

where C and n are empirical constants, J is the average current density, E_a is the activation energy for the EM mechanism, k is the Boltzmann constant, and T is the temperature. Thus, the only parameter that can be adjusted during mesh optimization is the current density. The current density can be adjusted either by controlling the mesh edges that are removed during

optimization or by wire-sizing after mesh optimization. Accurate implementation of the first method requires analysis of the mesh for EM violations after removing every mesh segment and hence is very costly in terms of run-time. The second method of wire-sizing is better because only mesh segments with EM issues in the optimized mesh need to be sized. As a result, we choose the second strategy.

The central fact used in our wire-sizing scheme is based on the observation that EM depends primarily on the asymmetric bidirectional currents as described in [26]. Thus, to a large extent, the value of current density J in a given mesh segment can be derived from the average DC current in it. This fact is used in [6] where it has been shown that the average DC current can be computed from node voltages of the equivalent RI network of the mesh RC network. The equivalent RI network of a given RC network is obtained by replacing all capacitors with current sources of equal value and retaining the same resistance values. Thus, from [6], for a given pair of nodes i and j, the average current in the mesh segment between nodes i and j is given as follows:

$$I_{ij} = \frac{2V_{DD}}{T_{clk}} \left(\frac{T_i - T_j}{R_{ij}}\right) \tag{8}$$

where i, j are the two nodes connected by the mesh segment under consideration, the factor 2 assumes a 50% duty cycle clock, V_{DD} is the supply voltage, T_{clk} is the time period of the clock signal, T_i and T_j are the first order (single pole) approximations of voltage at nodes i and j respectively and R_{ij} is the value of resistance of the mesh segment. The values of first order approximations of voltages at the mesh nodes can be obtained [6] as follows:

$$T = [T_0\ T_1\ \ldots\ T_n]^T = G^{-1}\mathbf{C} \tag{9}$$

where G is the $N\ X\ N$ admittance matrix of the mesh network and $\mathbf{C}$ is $[C_0\ C_1\ \ldots\ C_n]^T$ is the vector of node capacitances. Thus, the average current and hence the current density of a given mesh segment can be efficiently obtained from (8) and (9). We wish to point out here that the main difference between our work and the work of [6] is in the way we use (8). The work of [6] uses the equation to *prevent* EM issues while recovering area, while we use it to *solve* EM issues that arise after our mesh optimization. Our overall wire-sizing scheme for solving EM violations is shown in Fig. 12. We iteratively identify all the mesh segments with current density issues using (8) and then increase the widths in linear proportion to the magnitude of violation. The linear increase is motivated by the fact that, for a given current, the current density reduction is directly proportional to the width increase. In this algorithm, we ignore the interactions between the different mesh segments while sizing up a given segment. Also, the linear relationship between changes in wire-width to changes in current density is only an approximation. As a result, we may need more than one iteration to fix all the EM violations. However, since all the mesh segments with EM problems are sized-up in each iteration, the number of mesh segments with more EM violations after a given iteration is usually very small. Hence the total number of iterations for a given optimize mesh is also very small.

Procedure: $Fix\_EM\_Violations$

Input: Optimized mesh, Max. Current Density J_{max}.
Output: Final mesh with EM violations fixed.

1. Get J_{avg} for each mesh segment using (8), (9)
2. $N\_violators$ = # mesh segments with $J_{avg} > J_{max}$
3. while($N\_violators > 0$)
   (a) For (all segments with violations)
      (i) $scale = \frac{J_{avg}}{J_{max}}$
      (ii) $Width_{new} = Width_{old} * scale$
   (b) Get J_{avg} for all mesh segments using (8), (9)
   (c) $N\_violators$ = # mesh segments with $J_{avg} > J_{max}$
4. Output optimized mesh with new widths

Fig. 12. Iterative wire-sizing flow to fix EM violations.

Integration in MeshWorks flow: the wire-sizing based solution for solving EM violations is a natural addition to the rest of the MeshWorks flow since the input to this step is the optimized mesh. One potential issue that might arise because of the wire-sizing is that the increase in mesh capacitance might result in slew violations due to overloading of the mesh buffers. As a result, we might need one more round of mesh buffer sizing, which might in turn trigger new EM violations. Thus, in theory, the loop between wire-sizing and buffer sizing might not be closed. However, this does not happen frequently in practice as long as the increase in capacitance of mesh segments due to wire-sizing is small. This is confirmed by our experimental results presented in Section VI-C.

V. Practical Considerations in the Use of MeshWorks

A. Blockages

An important issue to be considered during clock network synthesis is the presence of blockages. The MeshWorks framework can work seamlessly even for chips with blockages. This is because the mesh optimization problem with blockages is identical to the optimization problem obtained by replacing the blockage with only the clock pins of the blockages connected to the mesh. Since the area of the blockage will not have any other clock sinks, the mesh segments within this area will naturally get optimized away.

B. Multi-Clock Floorplans

One of the main reasons why clock meshes are not used even in high performance ASICs is that they typically require multiple clocks to interact heavily and so they will have sinks of multiple clocks interspersed in the same floorplan. As a result, using a mesh structure for the clocks will require two separate meshes covering the entire floorplan, which is obviously unaffordable due to power/resource constraints. However, our clock mesh optimization scheme can recover most of the unnecessary clock mesh segments even if starting from complete meshes. This can make the use of clock meshes in multi-clock floorplans a viable option.

TABLE I
COMPARISON OF DIFFERENT MESH OPTIMIZATION APPROACHES WITH MAXIMUM SKEW OF ±50 PS BETWEEN MESH-BUFFERS AND A SLEW OF 50 PS ± 10 PS

Case (#Sinks)	Method	Size	BA		WL		PWR		μ_{sk}	σ_{sk}	F_{max}		CPU
			μm^2	% Red	μm	% Red.	(mW)	% Red.	(ps)	(ps)	MHz	%Red	(s)
s9234	MM	9X9	36.6	0.0	44 156	0.0	10.7	0.0	13.2	2.5	979.8	0.0	–
(135)	MO[7]	9X9	36.6	0.0	38 656	12.5	9.6	10.2	18.7	4.5	968.7	1.1	0.02
	NSMO_OB	9X9	36.6	0.0	31 983	27.6	8.6	19.6	26.4	5.6	958.5	2.2	0.05
	NSMO_OBS	9X9	33.1	9.4	31 983	27.6	8.3	22.1	26.2	5.0	960.5	2.0	0.05
	NSMO_NBS	9X9	32.2	12.1	31 157	29.4	7.9	25.6	19.7	4.3	968.4	1.2	0.07
	MP&S	7X7	29.8	18.5	42 013	4.9	9.9	6.9	10.8	2.9	980.9	−0.1	–
	MPSO	7X7	28.7	21.5	30 843	30.2	7.7	28.2	24.9	6.1	958.5	2.2	0.05
s5378	MM	10X10	38.6	0.0	46 852	0.0	11.3	0.0	12.0	2.7	980.2	0.0	–
(165)	MO[7]	10X10	38.6	0.0	38 952	16.9	9.8	13.4	21.3	4.3	967.0	1.3	0.03
	NSMO_OB	10X10	38.6	0.0	33 787	27.9	9.0	20.5	25.0	5.5	960.1	2.0	0.07
	NSMO_OBS	10X10	35.6	7.9	33 787	27.9	8.7	23.0	22.8	5.3	962.9	1.8	0.07
	NSMO_NBS	10X10	34.5	10.7	35 590	24.0	9.0	20.1	18.3	4.0	970.7	1.0	0.08
	MP&S	8X8	32.1	17.0	44 299	5.4	10.3	8.7	8.7	2.2	985.0	−0.5	–
	MPSO	8X8	31.4	18.8	32 126	31.4	8.0	28.8	22.8	5.3	962.9	1.8	0.05
s13207	MM	30X30	156.0	0.0	175 564	0.0	43.0	0.0	8.6	1.7	986.4	0.0	–
(500)	MO[7]	30X30	156.0	0.0	136 987	22.0	36.2	15.8	21.5	7.2	958.8	2.8	0.95
	NSMO_OB	30X30	156.0	0.0	129 813	26.1	36.0	16.2	22.5	4.4	965.4	2.1	1.13
	NSMO_OBS	30X30	146.3	6.2	129 813	26.1	35.6	17.3	19.6	3.7	970.2	1.6	1.13
	NSMO_NBS	30X30	148.6	4.7	135 506	22.8	36.0	16.2	21.7	2.9	970.5	1.6	1.53
	MP&S	13X13	83.9	46.2	116 738	33.5	27.5	36.1	10.0	2.3	983.5	0.3	–
	MPSO	13X13	82.5	47.1	85 369	51.4	21.5	49.9	27.1	4.0	962.4	2.4	0.47
s15850	MM	30X30	167.5	0.0	191 556	0.0	46.9	0.0	8.2	1.9	986.2	0.0	–
(566)	MO[7]	30X30	167.5	0.0	148 124	22.7	39.0	16.9	26.2	7.7	953.1	3.4	1.22
	NSMO_OB	30X30	167.5	0.0	140 227	26.8	38.6	17.7	19.0	3.9	970.2	1.6	1.35
	NSMO_OBS	30X30	159.1	5.0	140 227	26.8	36.9	21.3	16.8	3.3	974.1	1.2	1.35
	NSMO_NBS	30X30	155.1	7.4	142 708	25.5	38.5	18.0	17.3	3.0	974.4	1.2	1.77
	MP&S	15X15	99.5	40.6	137 211	28.4	31.9	31.9	9.8	2.3	983.6	0.3	–
	MPSO	15X15	98.1	41.4	98 377	48.6	25.0	46.7	28.2	5.3	957.8	2.9	0.60
s38584	MM	40X40	381.4	0.0	455 352	0.0	110.5	0.0	12.8	2.2	980.9	0.0	–
(1426)	MO[7]	40X40	381.4	0.0	341 834	24.9	89.7	18.8	32.4	8.1	946.4	3.5	8.05
	NSMO_OB	40X40	381.4	0.0	335 094	26.4	89.2	19.3	30.4	4.7	957.5	2.4	6.47
	NSMO_OBS	40X40	357.1	6.4	335 094	26.4	88.4	20.0	27.4	4.4	960.9	2.0	6.47
	NSMO_NBS	40X40	351.1	8.0	343 178	24.6	89.0	19.5	22.3	3.3	968.7	1.2	7.95
	MP&S	25X25	268.4	29.6	367 575	19.3	85.1	23.0	10.7	2.1	983.1	−0.2	–
	MPSO	25X25	264.3	30.7	259 989	42.9	66.2	40.1	30.9	5.0	956.1	2.5	4.22
s35932	MM	40X40	450.0	0.0	543 890	0.0	130.9	0.0	14.8	2.3	978.9	0.0	–
(1728)	MO[7]	40X40	450.0	0.0	440 401	19.0	112.9	13.7	32.5	4.6	955.6	2.4	21.03
	NSMO_OB	40X40	450.0	0.0	398 742	26.7	105.2	19.7	34.8	5.4	951.4	2.8	8.27
	NSMO_OBS	40X40	420.3	6.6	398 742	26.7	102.6	21.6	30.2	4.6	957.9	2.1	8.27
	NSMO_NBS	40X40	417.0	7.3	399 654	26.5	102.7	21.6	26.8	3.3	964.8	1.4	10.60
	MP&S	26X26	331.4	26.4	459 780	15.5	105.4	19.5	13.1	2.3	980.5	−0.2	–
	MPSO	26X26	322.2	28.4	325 256	40.2	81.8	37.5	32.8	4.8	955.0	2.4	5.83

The results are for the clock mesh only (i.e., mesh buffers, mesh edges, and all the sinks connected to the mesh).

C. Highly Uneven Load Distribution

The practically significant issue of uneven load distribution in different parts of a large chip can be addressed effectively using the MeshWorks framework. Such a situation can happen in reality when different IPs from different vendors are merged to create large system-on-a-chip designs. Even in situations like this, the MeshWorks framework can be used effectively. One method is to start the mesh optimization with a dense mesh that will work for the most dense region of the chip. Since our method will automatically optimize away unnecessary edges that do not contribute to skew variation tolerance, the mesh segments in the regions with light load distribution will be optimized away naturally. Another method is to divide the entire chip area into several regions of different flop densities and use mesh works independently on each of them. This last approach is similar to the method described in the tutorials on clock distribution networks [8] in which the chip area is divided into several *grid zones* which differ in loading and density.

Experimental results to verify the working of our mesh optimization scheme under each of the above issues are presented in Section VI-D.

VI. EXPERIMENTAL RESULTS

A. Experimental Setup

In theory, we can use the results of both [6] and [7] to compare against our work. However, the work of [6] only sizes a given clock mesh considering EM to reduce the resources. Also, there is no algorithm in [6] to determine the starting widths of the mesh segments. In contrast, the work of [7] is both more recent and more comparable to our work. For example, in both our work and in [7], minimum width wires are used to construct the mesh while in [6], the initial mesh is of non-minimum width (that is why they are able to size the mesh segments to reduce power). So, we compare our results only with [7]. First, we obtained the results of the method of [7] for our buffer library and slew constraints. The number of buffer types used in [7] was 4, much less than

TABLE II
AVERAGE OF OPTIMIZATION RESULTS FOR DIFFERENT SKEW/SLEW VALUES FOR MESH BUFFER INPUT SIGNALS

MB-Skew (ps)	MB-Slew (ps)	Method	% BA Red	% WL Red	% PR Red	μ_{skew} Avg.	σ_{skew} Avg.	F_{max} (MHz)	% F_{max} Red
±10	10±10	MM	0.00	0.00	0.00	9.21	1.57	986.3	0.00
		MO [7]	0.00	19.65	13.53	19.80	5.22	965.8	2.07
		NSMO_OB	0.00	26.90	18.78	21.76	5.29	963.8	2.28
		NSMO_OBS	6.91	26.90	20.24	19.01	4.39	968.9	1.77
		NSMO_NBS	8.37	25.49	19.92	17.32	3.44	973.1	1.34
		MP&S	29.73	17.82	20.49	8.07	1.74	986.9	−0.06
		MPSO	31.32	40.78	38.11	22.07	7.20	958.2	2.85
±30	30±10	MM	0.00	0.00	0.00	9.87	1.74	985.1	0.00
		MO [7]	0.00	19.65	14.53	20.92	4.72	966.1	1.93
		NSMO_OB	0.00	26.90	18.75	21.95	4.08	967.0	1.84
		NSMO_OBS	6.91	26.90	21.03	19.59	3.29	971.4	1.40
		NSMO_NBS	8.37	25.49	20.28	17.71	2.71	974.8	1.05
		MP&S	29.73	17.82	21.15	8.49	1.74	986.5	−0.13
		MPSO	31.32	40.78	37.89	21.13	4.27	967.2	1.82
±50	50±10	MM	0.00	0.00	0.00	11.60	2.23	982.1	0.00
		MO [7]	0.00	19.65	14.80	25.43	6.07	958.2	2.42
		NSMO_OB	0.00	26.90	18.84	26.34	4.94	960.5	2.19
		NSMO_OBS	6.91	26.90	20.88	23.84	4.37	964.4	1.80
		NSMO_NBS	8.37	25.49	20.16	21.00	3.46	969.6	1.27
		MP&S	29.73	17.82	21.02	10.50	2.34	982.8	−0.08
		MPSO	31.32	40.78	38.55	27.79	5.07	958.8	2.37
Average		MM	0.00	0.00	0.00	10.22	1.85	984.50	0.00
		MO [7]	0.00	19.65	14.28	22.05	5.34	963.40	2.14
		NSMO_OB	0.00	26.90	18.79	23.35	4.77	963.78	2.11
		NSMO_OBS	6.91	26.90	20.71	20.81	4.02	968.23	1.65
		NSMO_NBS	8.37	25.49	20.12	18.68	3.21	972.50	1.22
		MP&S	29.73	17.82	20.89	9.02	1.94	985.38	−0.09
		MPSO	31.32	40.78	38.18	23.66	5.51	961.38	2.35
Improvement		MPSO–MO [7]	31.32	21.13	23.90	1.62	0.17	−2.02	0.21

what is available/used in most practical libraries/designs [17]. Also, the nominal slew constraint used in [7] was 150 ps, which is 15% of even a GHz clock frequency. As mentioned in [1], a slew of around 10% of the clock frequency is common considering all process corners. Also, clock nets are typically well buffered to maintain tighter slews than signal nets. Considering these facts, we used 12 different buffer sizes with max-capacitance limit ranging from 60 fF to 300 fF and a nominal slew constraint of 75 ps for all the different methods. All *other* experimental conditions are identical to [7]. In particular, we use the same 65 nm technology parameters and transistor models from *PTM* [23] and the same set of benchmark circuits. Other variation parameters considered are buffer channel lengths, interconnect width, power supply variation and sink load capacitance variation. The above parameters are varied with 5% standard deviation around the nominal value. The spatial correlation in variation is accounted by the method of principal component analysis [24].

Top-Level Tree Modeling: We model the effects of variation on the top-level clock tree in a similar way as in [7] by modeling the input arrival time for the mesh buffers by a random variable. In our preliminary work [16], we used a range of ±25 ps for the skew between two mesh buffers and used the same slew for all the mesh buffers. In this paper, we use both the skew and slew random variables so that the input signals to the mesh buffers are more realistic compared to both [7] and [16]. We also repeat our experiments with different sets of bounds for the skew and slew random variables to measure the impact of the quality of the top-level on our results.

B. Mesh Planning, Synthesis, and Optimization Results

The complete results of different mesh optimization methods are shown in the Table I for ±50 ps skew between the mesh buffers and a slew value of 50 ps ±10 ps. Table II shows the average improvement for all six testcases in Table I and also shows how the average results change with the quality of the top-level tree is changed by changing the bounds of the skew and slew random variables of the mesh buffer input signals. The different mesh optimization approaches of our work and that of [7] are compared w.r.t. the manually selected mesh used in [7].

1) According to the authors of [7], the mesh sizes were chosen in such a way that a target nominal skew is obtained with minimum mesh wirelength. This manual mesh is denoted by "MM" in our tables. We directly compare the effectiveness of our optimization algorithms with the method of [7] by performing optimizations on this initial mesh.[3]
2) The mesh optimization method of [7] is denoted by "MO" [7].
3) We use network sensitivity mesh optimization + old buffering (NSMO_OB) to denote the results of using our network sensitivity based algorithm along with the old buffering method from [7]. Please note that this method does not do sizing of mesh buffers after mesh optimization.
4) The label network sensitivity mesh optimization + old buffering + sizing (NSMO_OBS) is used to identify results of using network sensitivity based optimization along with old buffering method of [7] along with sizing of mesh buffers post mesh optimization.
5) Mesh optimization using network sensitivity method along with the new cost function proposed in this paper is

[3]Only the mesh itself is identical to the one used in published results of [7] and not the buffer placement, buffers and slew constraint used. Also, the wirelength reported in [7] did not include the stub wirelengths.

denoted by network sensitivity mesh optimization + new buffering + sizing (NSMO_NBS).

6) In order to measure the effectiveness of our mesh planning and synthesis approach, we obtain the best mesh chosen by the algorithm in Fig. 6 for the same set of benchmark circuits and design constraints. This approach is denoted by mesh planning and synthesis (MP&S) in the tables.
7) Finally, we run our optimization algorithm on the mesh obtained from our mesh planning and synthesis algorithm. This approach is denoted by "MPSO" in our tables. The columns under "%Red" are the relative reductions w.r.t. "MM."

The different parameters in Table I are buffer area (BA), total wirelength (WL), power (PWR), and mean/standard deviations of skew (μ_{sk}, σ_{sk}) considering variations (obtained by SPICE Monte-Carlo simulations). The second-to-last column in Table I gives the worst case maximum frequency, F_{max}, at which the clock network can be run in the presence of $\mu_{sk} + 3\sigma_{sk}$ skew variation assuming the *ideal* target frequency to be 1 GHz. Similar to [7], we also use the percentage reduction in max frequency under variation as the measure of variation tolerance instead of changes in skew. This enables us to directly compare the power/area vs. frequency tradeoff. Instead, if we directly consider the increase in skew, even a change from a skew of 1 ps to 2 ps will be a 100% change but it does not convey the actual tradeoff between frequency of operation and resources. The definitions of most of the parameters in Table II are the same as in Table I. The new parameters are mesh buffer skew (MB-Skew), which gives the maximum skew between any two mesh buffers and mesh buffer slew (MB-Slew), which gives the variation range of the slew of the mesh buffers. Both these random variables are used to simulate the impact of the top-level clock tree driving the mesh buffers. We use this method to simulate the impact of top-level clock tree since constructing a near-zero SPICE skew clock tree is a non-trivial problem by itself and beyond the focus of this paper.

The key observations from the Tables I and II are as follows.

1) *Mesh Optimization:* Our network sensitivity based optimization (NSMO_NBS) yields consistently better results than the approach of [7] for identical starting mesh. On an average, our approach yields 8.37%, 5.84%, and 5.84% extra reduction in buffer area, wirelength, and power respectively with almost 1.0% improvement in F_{max}. Thus, our methods achieve better maximum frequency with lesser resources compared to [7]. This proves the effectiveness of our overall mesh optimization approach.

2) *Mesh Planning:* Our mesh planning and synthesis algorithm (MP&S) is effective in choosing a good initial mesh. In most cases, the quality of this initial mesh is close to the final, optimized results of [7]. Also, the size of the initial mesh obtained from our mesh planning approach is significantly smaller when compared to the manually obtained mesh of [7] for the bigger testcases. This illustrates the importance of having a good methodology to obtain an initial mesh.

3) *Combined Mesh Planning and Optimization:* By performing our network sensitivity based optimizations on the mesh selected by our mesh planning algorithm (MPSO), we are able to achieve, on an average, 31.32% buffer area reduction, 21.13% wirelength reduction and 23.90% power reduction with 0.21% reduction in the worst case maximum frequency. This proves that our overall framework can be used to significantly reduce the mesh resources with negligible impact on the worst case maximum frequency.

4) *Resources vs. Frequency Tradeoff:* From Table II, we see that the bigger the reduction in power, buffer area and wirelength, the higher is the skew degradation, which is expected. But what is noteworthy is that the degradation in skew is insignificant because it results in less than 3% reduction in F_{max} (w.r.t. the original mesh of [7]) while achieving significant reduction in power and area.

5) *Impact of Top-Level Clock Tree:* By comparing the average results with different skew/slew bounds for the mesh buffer input signals, we see that the overall results does not change much. This demonstrates that our results are not affected significantly by the quality of the top-level clock tree, as long as the skew and slew are bounded by reasonable values.

C. Wire-Sizing for Reliability

To demonstrate the effectiveness of our wire-sizing scheme for meeting EM requirements, we first obtain the target current density value. The existing literature [27]–[30] has a wide range of values for the current density that can be used for copper interconnects. For example, the works of [27]–[30] recommend values of $90\,\text{mA}/\mu\text{m}^2$, $16\,\text{mA}/\mu\text{m}^2$, $8\,\text{mA}/\mu\text{m}^2$, and $160\,\text{mA}/\mu\text{m}^2$, respectively. In order to be conservative, we set 90% of the most aggressive value of $8\,\text{mA}/\mu\text{m}^2$ as our current density target. Thus, any mesh segment with a current density value above $7.2\,\text{mA}/\mu\text{m}^2$ is treated as a single EM violation. The goal is to ensure that no mesh segment in the final optimized mesh exceeds this current density requirement. Next, we use the methodology described in Section IV to identify and fix the violations found in the optimized mesh from the MPSO rows of Table I. The complete results for this procedure is shown in Table VI-B.

From Table VI-B, we can see that our method is able to fix all EM violations with an average 1% increase in total wire-area. Please note that all the results in Table I use minimum wire-widths and the increase in wire-area in Table VI-B is from the wire-sizing. It may be noted here that we do attempt to resize mesh buffers after mesh segment sizing to ensure we do not overload the mesh buffers. However, in all our testcases, we found this to be unnecessary as the amount of extra loading on the mesh buffers was very small, i.e., no maximum load violation happened after wire sizing for EM fixing. As a result, there was no change in the buffer area even after fixing all the EM violations. Also, the impact of EM fixing on skew, slew, delay etc. are also negligible since the number of violations we have fixed in our testcases is just a handful.

D. Results for Practical Issues in MeshWorks Usage

1) *Blockages:* To test our mesh optimization in the presence of blockage, we first created a testcase as described next. We picked a rectangular floorplan and randomly generated the

TABLE III
RESULTS OF EM VIOLATION FIXED BY WIRE-SIZING

Testcase	Before EM Fix			After EM Fix			Difference			#Iterations
	#EMV	%BA	%WA	#EMV	%BA	%WA	%#EMV	%BA	%WA	to Fix EM
s9234	1	14.0	23.8	0	14.0	23.24	100	0	0.56	1
s5378	0	22.1	33.8	0	22.1	33.8	100	0	0.00	0
s13207	11	42.1	52.7	0	42.1	51.09	100	0	1.60	2
s15850	7	39.5	56.1	0	39.5	55.36	100	0	0.73	1
s38584	11	22.4	43.6	0	22.4	42.95	100	0	0.64	2
s35932	33	22.1	35.7	0	22.1	33.78	100	0	1.91	4

#EMV denotes the number of EM violations. WA = wire-area.
BA and WA are % reduction w.r.t. MM row in Table I.

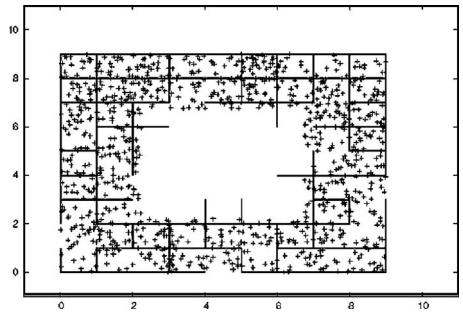

Fig. 13. Mesh optimization result on a testcase with blockage.

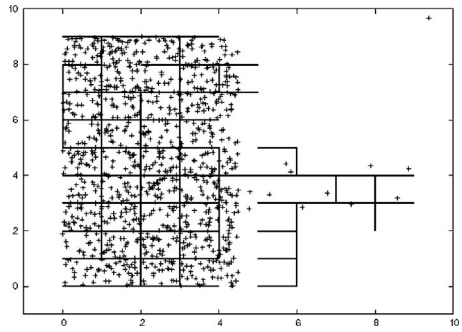

Fig. 14. Results on a testcase with two clocks. Clock-A is shown.

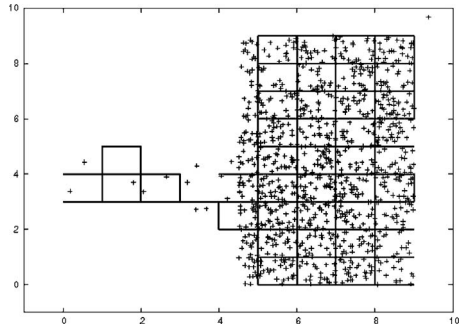

Fig. 15. Results on a testcase with two clocks. Clock-B is shown.

sink locations in it. The center of the floorplan was assumed to have a rectangular blockage and so any random sinks located in the smaller square was removed. A complete 10×10 mesh was selected arbitrarily for this floorplan. Buffering & mesh optimization were done on this complete mesh without giving any explicit information about the presence of the blockage to them. The final mesh edges post mesh optimization are shown in Fig. 13. Visual inspection confirms that all mesh segments that were present in the blockage area have been removed. This demonstrates that our optimization framework can be used in the presence of blockages.

2) *Multiple Clocks:* We generated our multi-clock testcase as follows. A rectangular floorplan was selected and clock sinks were randomly generated with constraints such that sinks on the left side of the square were assigned to Clock-A and the sinks on the right side of the square were assigned to Clock-B. Exceptions to this rule were allowed with a small probability when the Y-coordinate of the sinks was between a selected band (between 30% and 50% of the maximum Y distance). This setup enabled us to get a floorplan such that majority of sinks belonging to Clock-A were located on the left-side with a few on the right side of the block and vice-versa. This imitates the conditions in many ASIC designs where registers belonging multiple clocks interact. For this testcase, two sets of buffering and mesh-optimization were done on a 10×10 mesh, one for Clock-A and another for Clock-B. The final mesh edges after mesh optimization for these clocks are shown in Figs. 14 and 15. As expected, the mesh for Clock-A has most of the mesh segments on the right side removed, except for the segments attached to the few Clock-A sinks on the right side and vice-versa for Clock-B. This demonstrates that our mesh optimization can reduce resource utilizations on multi-clock floorplans even if the initial mesh covers the entire floorplan.

3) *Highly Uneven Load Distribution:* As the results from the previous experiment demonstrated, our mesh optimization scheme works as expected even when the clock sinks are distributed to one side of the chip predominantly over the other. This can be directly inferred from both Figs. 15 and 14. Thus, our scheme works well even in cases with uneven distribution of clock sinks.

VII. CONCLUSION

In this paper, we have presented an efficient, fully automated framework for planning, synthesis, and optimization of clock mesh networks. Experimental results suggest that our algorithms can achieve an additional reduction of 31% in buffer area, 21% in wirelength, and 23% in power, compared to the results of [7] with similar worst case maximum frequency of operation. Our overall framework is very powerful and can address several practical issues including blockages, multiple clocks, uneven load distribution and EM violations.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

REFERENCES

[1] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie, "A clock distribution network for microprocessors," *IEEE JSSC*, vol. 36, no. 5, pp. 792–799, May 2001.

[2] P. J. Restle, private communication.
[3] N. A. Kurd, J. S. Barkarullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland, "A multigigahertz clocking scheme for the Pentium 4 microprocessor," *IEEE JSSC*, vol. 36, no. 11, pp. 1647–1653, Nov. 2001.
[4] G. Northrop, R. Averill, K. Barkley, S. Carey, Y. Chan, Y. H. Chan, M. Check, D. Hoffman, W. Huott, B. Krumm, C. Krygowski, J. Liptay, M. Mayo, T. McNamara, T. McPherson, E. Schwarz, L. Sigal, T. Slegel, C. Webb, D. Webber, and P. Williams, "600-MHz G5 S/390 microprocessor," in *Proc. ISSCC Tech. Dig.*, Feb. 1999, pp. 88–89.
[5] R. Heald, K. Aingaran, C. Amir, M. Ang, M. Boland, A. Das, P. Dixit, G. Gouldsberry, J. Hart, T. Horel, W.-J. Hsu, J. Kaku, C. Kim, S. Kim, F. Klass, H. Kwan, R. Lo, H. McIntyre, A. Mehta, D. Murata, S. Nguyen, S. Y.-P. Pai Patel, K. Shin, K. Tam, S. Vishwanthaiah, J. Wu, G. Yee, and H. You, "Implementation of a 3rd-generation SPARC V9 64b microprocessor," in *Proc. ISSCC Dig. Tech. Papers*, Feb. 2000, pp. 412–413.
[6] M. P. Desai, R. Cvijetic, and J. Jensen, "Sizing of clock distribution networks for high performance CPU chips," in *Proc. DAC*, 1996, pp. 389–394.
[7] G. Venkataraman, Z. Feng, J. Hu, and P. Li, "Combinatorial algorithms for fast clock mesh optimization," in *Proc. ICCAD*, 2006, pp. 79–84.
[8] S. Tam, "Tutorials on clock distribution," in *Proc. ICCAD*, 2007.
[9] H. Su and S. S. Sapatnekar, "Hybrid structured clock network construction," in *Proc. IEEE/ACM ICCAD*, Nov. 2001, pp. 333–336.
[10] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proc. IEEE*, vol. 89, no. 5, pp. 665–692, May 2001.
[11] M. R. Guthaus, D. Sylvester, and R. B. Brown, "Clock buffer and wire sizing using sequential programming," in *Proc. DAC*, 2006, pp. 1041–1046.
[12] B. Lu, J. Hu, G. Ellis, and H. Su, "Process variation aware clock tree routing," in *Proc. ISPD*, 2003, pp. 174–181.
[13] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via cross links," in *Proc. DAC*, 2004, pp. 18–23.
[14] H. Chen, C. Yeh, G. Wilke, S. Reddy, H. Nguyen, W. Walker, and R. Murgai, "A sliding window scheme for accurate clock mesh analysis," in *Proc. ICCAD*, 2005, pp. 939–946.
[15] S. M. Reddy, G. R. Wilkem, and R. Murgai, "Analyzing timing uncertainty in mesh-based clock architectures," in *Proc. DATE*, 2006, pp. 1097–1102.
[16] A. Rajaram and D. Z. Pan, "MeshWorks: An efficient framework for planning, synthesis and optimization of clock mesh networks," in *Proc. ASP-DAC*, 2008, pp. 250–257.
[17] C. J. Alpert, R. G. Gandham, J. L. Neves, and S. T. Quay, "Buffer library selection," in *Proc. ICCD*, 2000, pp. 221–226.
[18] J. Leeds and G. Ugron, "Simplified multiple parameter sensitivity calculation and continuously equivalent networks," *IEEE TCAS*, vol. 14, no. 2, pp. 188–191, Jun. 1967.
[19] M. Shao, M. D. F. Wong, H. Cao, Y. Gao, L. P. Yuan, L. D. Huang, and S. Lee, "Explicit gate delay model for timing evaluation," in *Proc. ISPD*, 2003, pp. 32–38.
[20] *HSPICE* [Online]. Available: http://www.synopsys.com/products/mixedsignal/hspice/hspice.html
[21] A. Gattiker, S. Nassif, R. Dinakar, and C. Long, "Timing yield estimation from static timing analysis," in *Proc. ISQED*, 2001, pp. 79–84.
[22] M. Celik, L. Pileggi, and A. Odabasioglu, *IC Interconnect Analysis*. Boston, MA: Kluwer, 2002.
[23] Arizona State University [Online]. Available: http://www.eas.asu.edu/ptm
[24] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-line traversal," in *Proc. ICCAD*, 2003, pp. 621–625.
[25] J. Lienig and G. Jerke, "Electromigration-aware physical design of integrated circuits," in *Proc. VLSID*, 2005, pp. 77–82.
[26] A. Todri and M. Marek-Sadowska, "A study of reliability issues in clock distribution networks," in *Proc. ICCD*, 2008, pp. 101–106.
[27] J. Tao, N. W. Cheung, and C. Hu, "Electromigration characteristics of copper interconnects," *IEEE Electron Device Lett.*, vol. 14, no. 5, pp. 249–251, May 1993.
[28] M. H. Lin, Y. L. Lin, G. S. Yang, M.-S. Yeh, K. P. Chang, K. C. Su, and T. Wang, "Comparison of copper interconnect electromigration behaviors in various structures for advanced BEOL technology," in *Proc. 11th Int. Symp. Physical Failure Anal. Integr. Circuits*, 2004, pp. 177–180.
[29] C. M. Tan, A. Roy, A. V. Vairagar, A. Krishnamoorthy, and S. G. Mhaisalkar, "Current crowding effect on copper dual damascene via bottom failure for ULSI applications," *IEEE Device Mater. Reliab.*, vol. 5, no. 2, pp. 198–205, Jun. 2005.
[30] C. W. Chang, C. L. Gan, C. V. Thompson, K. L. Pey, W. K. Choi, and M. H. Chua, "Joule heating-assisted electromigration failure mechanisms for dual damascene Cu/SiO_2 interconnects," in *Proc. 10th Int. Symp. Physical Failure Anal. Integr. Circuits*, 2004, pp. 69–74.
[31] I. A. Blech, "Electromigration in thin aluminum films on titanium nitride," *J. Appl. Phys.*, vol. 47, no. 4, pp. 1203–1208, 1976.

Anand Rajaram (S'04–M'09) received the B.E. degree in electrical and electronics engineering from Anna University, Chennai, India, the M.S. degree in computer engineering from Texas A&M University, College Station, in 2004, and the Ph.D. degree in computer engineering from the University of Texas, Austin, in 2008.

From 2004 to 2008, he was with the Dallas DSP Group, Texas Instruments, Dallas, where he worked on high-speed clock network synthesis and analysis on high-performance DSP chips. Since 2008, he has been with Magma Design Automation, Austin, TX, where he works on various aspects of physical design automation. He has published more than 17 refereed papers in international conferences and journals. His current research interests include variation-aware physical design and clock network synthesis and analysis.

Dr. Rajaram's papers at the Design Automation Conference 2004 and the Asia and South Pacific Design Automation Conference 2008 were nominated for Best Paper Awards, and his paper at the Design, Automation, and Test in Europe 2009 received the Best IP Paper Award.

David Z. Pan (S'97–M'00–SM'06) received the Ph.D. degree in computer science from the University of California, Los Angeles, in 2000.

From 2000 to 2003, he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY. He is currently an Associate Professor and the Director of the Design Automation Laboratory, Department of Electrical and Computer Engineering, University of Texas, Austin. He has published over 120 refereed papers in international conferences and journals, and is the holder of seven U.S. patents. His current research interests include nanometer very-large-scale integration physical design, design for manufacturing, vertical integration of technology, design and architecture, and design/computer-aided design for emerging technologies.

Dr. Pan has served as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD), the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-PART I, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-PART II, the IEEE CAS Society Newsletter, and the *Journal of Computer Science and Technology*. He was a Guest Editor of the TCAD Special Section "International Symposium on Physical Design" in 2007 and 2008. He serves as the Chair of the IEEE CANDE Committee and the ACM/SIGDA Physical Design Technical Committee. He is with the Design Technology Working Group of International Technology Roadmap for Semiconductors. He has served in the technical program committees of major VLSI/CAD conferences, including ASPDAC (Topic Chair), DAC, DATE, ICCAD, ISPD (Program Chair), ISLPED (Exhibits Chair), ISCAS (CAD Track Chair), ISQED (Topic Chair), GLSVLSI (Publicity Chair), SLIP (Publication Chair), ACISC (Program Co-Chair), ICICDT (Award Chair), and VLSI-DAT (EDA Track Chair). He was the General Chair of ISPD 2008 and ACISC 2009. He is a member of the Technical Advisory Board of Pyxis Technology, Inc, Austin, TX. He has received a number of awards for his research contributions and professional services, including the ACM/SIGDA Outstanding New Faculty Award in 2005, the NSF Career Award in 2005, the SRC Inventor Recognition Award thrice from 2000 to 2008, the IBM Faculty Award thrice from 2004 to 2006, the UCLA Engineering Distinguished Young Alumnus Award in 2009, the Best Paper Award from ASPDAC in 2010, the Best Interactive Presentation Award from DATE in 2010, the Best Student Paper Award from ICICDT in 2009, the IBM Research Bravo Award in 2003, the SRC Techcon Best Paper in Session Award in 1998 and 2007, the Dimitris Chorafas Foundation Research Award in 2000, the ISPD Routing Contest Award in 2007, the eASIC Placement Contest Grand Prize in 2009, five Best Paper Award Nominations (from ASPDAC, DAC, ICCAD, ISPD), and the ACM Recognition of Service Award in 2007 and 2008. He was an IEEE CAS Society Distinguished Lecturer from 2008 to 2009.