MeshWorks: An Efficient Framework for Planning, Synthesis and Optimization of Clock Mesh Networks

Anand Rajaram

Dept. of ECE, Univ. of Texas at Austin, Texas;
DDSP, Texas Instruments, Dallas, Texas
anandr@mail.utexas.edu

David Z. Pan

Department of ECE,
University of Texas at Austin, Texas
dpan@ece.utexas.edu

***Abstract*— A leaf-level clock mesh is known to be very tolerant to variations [1]. However, its use is limited to a few high-end designs because of the high power/resource requirements and lack of automatic mesh synthesis tools [2]. Most existing works on clock mesh [1], [3]–[7] 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. Similarly, the problem of achieving a smooth tradeoff between skew and power/resources has not been addressed adequately. In this work, we present *MeshWorks*, the first comprehensive automated framework for planning, synthesis and optimization of clock mesh networks with the objective of addressing the above issues. Experimental results suggest that our algorithms can achieve an additional reduction of 26% in buffer area, 19% in wirelength and 18% in power, compared to the recent work of [7] with similar worst case maximum frequency under variation.**

I. Introduction

As the VLSI technology continues to scale below 65nm, the effects of manufacturing variation, power supply noise, temperature variations etc. on clock skew are becoming more significant. Since higher skews directly reduce the maximum frequency of the circuit, reducing the clock skew variation can improve timing yield. Among the 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], [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. Figure 1 shows an example of a clock network with top-level tree and leaf-level mesh.

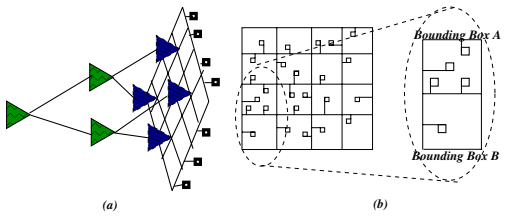

Fig. 1. (a) A clock network with leaf-level mesh. (b) Leaf-level mesh driving clock sinks.

Motivation for our Work: The high buffer area, routing resource and power requirements of a leaf-level clock mesh have historically restricted its use to a few high-end products like microprocessors [1], [3]–[6]. However, with variation becoming a bigger issue at 65nm 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 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 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, lack of research on automated clock mesh synthesis was noted as early as in 2001 [9]. However, no comprehensive work has been done on this practically important topic till now. For example, 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 optimizations are needed. Such a framework can enable chip teams to achieve a smooth tradeoff between performance (skew) and power (area).

It may be noted here that any automated clock mesh planning/synthesis and optimization will be of significance 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], [3]–[5] deal with custom/semi-custom mesh design and do not address the problem of automatic mesh synthesis/optimization. The works of [6], [7] perform optimizations on a *given* clock mesh. The work of [6] performs clock mesh sizing considering only the nominal skew targets and ignores variation. Recent works [10], [11] present efficient methods for clock mesh analysis and they do not deal with clock mesh synthesis.

To the best of our 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 (widely used in computer networks) to remove some of the mesh segments without significantly affecting variation tolerance. Though the work of [7] is efficient, it has a few key drawbacks as summarized below:

- It does not consider the problem of initial mesh planning/synthesis and relies on manually selected mesh for performing optimizations.
- The network survivability formulation ignores the non-uniform sink distribution and hence the effect of differential loading on buffer delays. However, non-uniform sink distribution is common in most designs [1], [8]. Also, the computer network model ignores the delay characteristics of a clock mesh which might result in incorrect optimizations.
- Electrical characteristics of mesh buffers, irrespective of their sizes, are ignored during the mesh optimization and all buffers are treated identically. Moreover the interaction between mesh reduction and buffering is ignored.

[1]It is called *grid floor-planning* in [8].
[2]The individual sub-grids driving small zones of a chip.

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

In this work, we attempt to address all the drawbacks mentioned above. The key components of our MeshWorks framework are:

- ***Mesh Planning and Synthesis:*** We propose a simple yet effective method that can aid in fast planning & synthesis of a buffered clock mesh for a given set of design constraints. Our method can help choose a good initial buffered mesh, which can be further optimized for power/resource reduction during refinement stage.
- ***Mesh Optimization:*** We propose an efficient algorithm using *network sensitivity theory* to select the mesh edges that can be safely removed 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.
- ***Buffer Modeling for Mesh Optimization:*** We propose an efficient buffer modeling method that is especially suitable for use during clock mesh optimization. We also present an efficient technique to resize the buffers after mesh optimization to reduce buffer area and power consumption.

The above contributions make MeshWorks the first comprehensive framework for complete automation of clock mesh networks synthesis and optimization. Experimental results suggest that MeshWorks can achieve significant resource reduction compared to the already optimized results of the work of [7] with similar worst case maximum operational frequency under variation.

II. Mesh Planning and Synthesis

The mesh planning and synthesis problem can be stated as follows. ***Given***: Sink locations and load capacitance, buffer library, interconnect parameters, variation models, nominal/variational skew targets. ***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.

- $S = \{s_1, s_2, ...s_N\}$ is the set of all N clock sinks, where s_i denotes the i^{th} sink.
- $B = \{b_1, b_2, ...b_T\}$ 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}.
- 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.
- Let $IntDel(l, C)$ denote the delay when an interconnect of length l drives a load capacitance of value C.
- 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 $m * n$. Each clock sink s_i is attached to the nearest mesh node using an interconnect called *stub* of length $L^i{}_{stub}$.
- 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 :

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

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 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. Figure 2 illustrates this fact with a simple example.

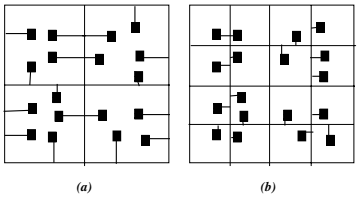

Fig. 2. Examples of sparse and dense clock meshes. A dense mesh is likely to have shorter stubs.

Figure 3 shows how the wirelength changes as a function of mesh size in one of our test cases. The *key point* to be noted is that it is easy to get a plot similar to Figure 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.

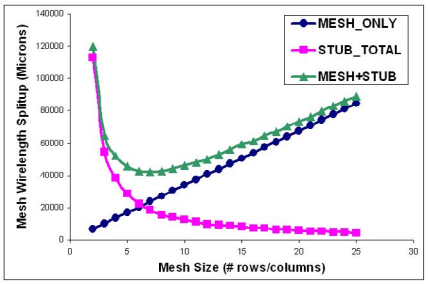

Fig. 3. Determining the right mesh size.

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:

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

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 Equation(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* $(q-1)$. We can consider a load of CL_{q-1}^{max} to be a lower bound of the load for buffer b_q because the use of bigger sized buffer b_q when a smaller buffer b_{q-1} can be used will waste resources[3]. 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

[3]Refer to end of Section II-C on assumptions made in this regard.

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

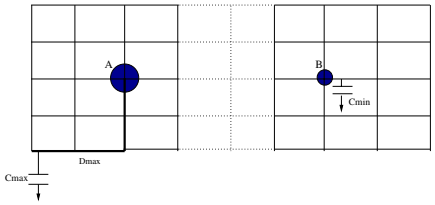

Fig. 4. Three dominant skew components in a mesh - skew due to buffer delay imbalance, skew due to difference in distance from closest buffer and skew due to different stub length and load capacitance.

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 Figure 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.

Figure 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.

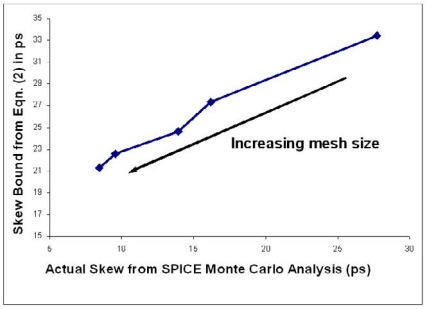

Fig. 5. Plot showing the fidelity of the skew bound Equation(2). Though skew bound is not perfectly linear of actual worst case skew, it is monotonic.

Thus, Equation(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 Figure 6.

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 Equation(1) is $\simeq L_{min}$.
2. Using the current m, n, obtain L_{tot} using Equation(1).
3. If ($L_{tot} \geq L_{max}$)
   Print "Relax design constraints"
   Quit.
4. Obtain value of Sk_{bound} from Equation(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. The top-level algorithm of selecting the initial mesh size.

In practice, the value of S_{max} parameter used as input to Figure 6 should be chosen such that it is not too tight. This is because the value of Sk_{bound} obtained from Equation(1) is always pessimistic since it is a bound for the worst possible skew for a given mesh size.

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

- Several buffers of incrementally different sizes/drive strengths are available to make the target skew physically possible. As noted in [12], most practical libraries will have hundreds of different buffer sizes to choose from. Hence this assumption is valid in practice.
- 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 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 [10], [11]. 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 Figure 7. The solution in Figure 7-a uses two smaller buffers to drive the same amount of load instead of one big buffer in Figure 7-b. Considering the attenuation characteristics of an RC mesh, the solution in Figure 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 Figure 7-a will result in lesser buffer area. However, the work of [7] might randomly pick one of them.

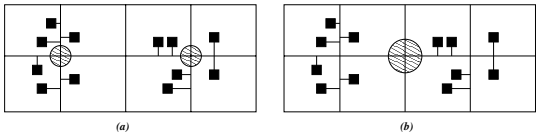

Fig. 7. An example where the buffer insertion algorithm of [7] might not take the better choice. The shaded circles represent buffers of proportional size.

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:

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

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, 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 above cost function are:

- 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.
- 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.
- 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] and are omitted here due to page limit.

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, which is explained next. Comparing the situations in Figure 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 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 pin capacitance and the mesh wirelength at the cost of increasing the top-level mesh wirelength. Since the top-level wirelength addition is only a tree (which can be represented by sum of distances from the single big buffer to the small buffers that replace the big buffer) and since the potential optimization that can be done in the mesh can result in removal of several mesh edges, the above approach typically results in an overall reduction in total capacitance of the clock network.

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 [13] without perturbing any circuit parameters.

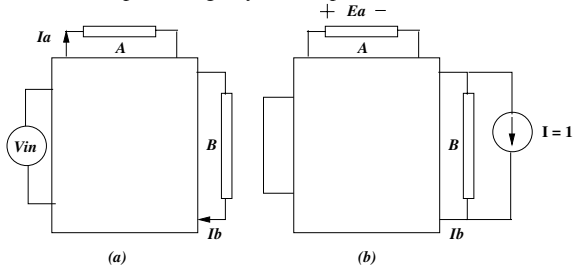

Fig. 8. Network sensitivity theory can be applied for clock mesh optimization.

Consider the Figure 8 (from [13]) which shows a generic electrical network with 3 identified elements for illustrative purposes. The elements can be any of the passive components like R, C and L. Let I_A, I_B, I_C 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 [13], 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 [13], the relationship between the currents of the original network, element values and voltages in the auxiliary network is given as:

$$E_a = \frac{\partial E_b}{\partial A} * \frac{A}{I_A} \quad (4)$$

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 [13] 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 multi output networks, like 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 [13] 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 like resistors and capacitors. The typical switch resistance modeling of buffers is becoming increasingly inadequate to approximate the buffers in the sub 100nm 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 [14] satisfies all these requirements. The basic idea behind the work of [14] 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 Figure 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 [15] to approximate the delay characteristics of a given buffer. In this work, we adapt the work of [14] to make it suitable for the problem approximating a library of buffers.

The work of [14] 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.

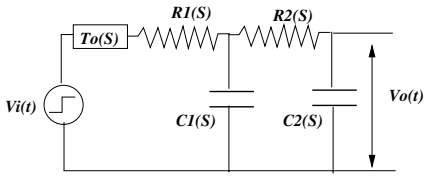

Fig. 9. Accurate buffer used for clock mesh optimization. S is the size of a given buffer

Accuracy of the buffer model: Figure 10 compares the clock sink delays for one of our mesh test cases 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 Figure (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.

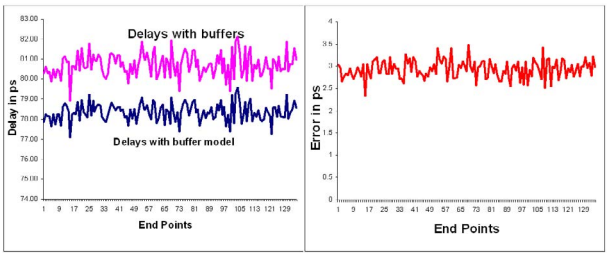

Fig. 10. Comparison of sink delays in SPICE obtained using buffers and the buffer model for a clock mesh testcase.

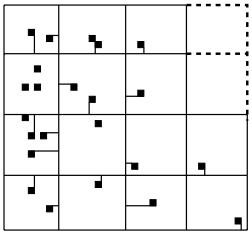

Fig. 11. Simple example of network sensitivity based mesh optimization.

In this work, 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:

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

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 $\frac{\partial R(p,q)}{\partial Wp,q}$ and $\frac{\partial C(p,q)}{\partial Wp,q}$ 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 towards solving mesh optimization problem:

- As stated in Section III-A, the method of [13] is inefficient for clock network which has many output (sink) nodes.
- The method of [13] can be used only to obtain voltage sensitivities and cannot be directly used to obtain delay sensitivity.

- 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:

- The RC mesh network behaves as a low-pass filter [10], [11], 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. As a result, the delay sensitivities of a given set of closely located sinks will be almost the same w.r.t. most clock segments. This assumption is not valid for mesh segments located close to the sinks. However, such mesh segments are less likely to be optimized out. We can use this observation to drastically reduce the number of output nodes (sinks) for which delay sensitivity needs to be evaluated.
- According to [16], 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 Equation(5) using Elmore delay without accuracy loss.
- 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 [17]. The node voltages in this circuit represent the Elmore delays of the original mesh networks and the voltage sensitivities are the sensitivties of the Elmore delays.
- 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 [17].
- The analysis efficiency can be further improved by exploiting the sparse nature of the nodal admittance matrix for most RC mesh networks [17].

Using the above observations, the value of cost function of Equation(6) can be evaluated for each mesh edge in an efficient manner.

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 Equation(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. The mesh segments are selected such that no two removed mesh segments are at a distance of N nodes from each other. This requirement makes sure that any interactions between the mesh segments removed is negligible. A higher N implies fewer edge removal and lesser modeling error because of the interactions between the mesh edges.

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 work, 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:

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. Experimental Results

A. Experimental Setup

We use the results of the recent work of [7] as it has the closest objective to that of ours. To make a valid comparison, we obtained the results of the method of [7] from the authors for *our buffer library and slew constraints*. The number of buffer types used in [7] was 4, much lesser than what is available/used in most practical libraries/designs [12]. Also, the nominal slew constraint used in [7] was 150ps, 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 60fF to 300fF 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 *bptm* [18] and same set of benchmark circuits. Also, we modeled the effects of variation on the top-level clock tree in the same way as in [7] by modeling the input arrival time for the mesh buffers by a random variable with a maximum variation range of 50ps. 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 [19].

B. Results

The complete results of different mesh optimizations are shown in the Table I. Table II shows the average improvement for all 6 test cases in Table I. The different mesh optimization approaches of our work and that of [7] are compared w.r.t. the manually selected mesh used in [7]. 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 [4]. The mesh optimization method of [7] is denoted by "MO[7]" and our network sensitivity based approach, along with buffer resizing is denoted by "NSMO" (Network Sensitivity Mesh Optimization). In order to measure the effectiveness of our mesh planning & synthesis approach, we obtain the best mesh chosen by the algorithm in Figure 6 for the same set of benchmark circuits and design constraints. This approach is denoted by "MP&S" (Mesh

[4]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.

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 (135)	MM	9X9	36.5	0.0	44156	0.0	9.8	0.0	8.8	2.3	984.3	NA	NA
	MO[7]	9X9	36.5	0.0	40967	7.2	9.2	6.1	15.7	4.4	971.7	1.2	0.4
	NSMO	9X9	35.2	3.7	32133	27.2	8.3	14.7	14.5	2.6	977.9	0.6	6.2
	MP&S	7X7	35.0	4.3	42013	4.8	9.3	4.5	18.1	4.0	970.7	1.3	0.1
	MPSO	7X7	31.4	14.0	33610	23.8	8.0	18.4	29.8	5.5	955.6	2.9	5.8
s5378 (165)	MM	10X10	38.6	0.0	46851	0.0	10.4	0.0	7.0	2.0	987.0	NA	NA
	MO[7]	10X10	38.6	0.0	39472	15.7	8.9	14.0	19.5	6.1	963.3	2.3	0.4
	NSMO	10X10	38.2	1.0	32691	30.2	7.9	23.6	30.8	2.1	964.2	2.3	9.0
	MP&S	8X8	36.7	4.8	44298	5.4	9.9	5.0	11.1	2.8	980.7	0.6	0.1
	MPSO	8X8	30.1	22.1	31009	33.8	6.7	35.4	35.2	5.8	949.8	3.7	7.0
s13207 (500)	MM	30X30	155.9	0.0	175564	0.0	39.7	0.0	7.7	1.7	987.2	NA	NA
	MO[7]	30X30	155.9	0.0	131652	25.0	31.1	21.4	13.7	2.6	978.7	0.8	3.0
	NSMO	30X30	115.0	26.2	122929	29.9	30.3	23.6	17.0	3.2	974.0	1.3	26.1
	MP&S	13X13	98.3	36.9	116738	33.5	26.2	34.0	12.9	2.3	980.3	0.7	0.1
	MPSO	13X13	90.2	42.1	82884	52.7	20.6	47.8	27.1	2.3	966.9	2.0	22.0
s15850 (566)	MM	30X30	167.4	0.0	191555	0.0	43.2	0.0	7.6	1.6	987.6	NA	NA
	MO[7]	30X30	167.4	0.0	127663	33.3	31.1	28.1	24.5	3.9	964.8	2.3	3.0
	NSMO	30X30	123.5	26.2	109275	42.9	27.1	37.2	17.2	2.8	974.9	1.2	32.1
	MP&S	15X15	113.9	31.9	137211	28.3	30.7	29.0	10.4	2.1	983.3	0.4	0.1
	MPSO	15X15	101.2	39.5	84055	56.1	22.0	48.9	23.5	4.0	965.5	2.2	28.9
s38584 (1426)	MM	40X40	381.4	0.0	455352	0.0	101.6	0.0	8.6	1.5	986.9	NA	NA
	MO[7]	40X40	381.4	0.0	345980	24.0	79.3	21.9	15.3	1.9	979.2	0.7	8.6
	NSMO	40X40	342.6	10.1	318712	30.0	77.3	23.9	21.7	3.5	968.6	1.8	100.2
	MP&S	25X25	303.5	20.4	367574	19.2	82.0	19.2	12.4	2.2	981.2	0.5	0.2
	MPSO	25X25	295.8	22.4	256567	43.6	65.2	35.8	37.2	4.9	950.4	3.7	81.6
s35932 (1728)	MM	40X40	449.9	0.0	543889	0.0	121.0	0.0	9.1	1.3	986.8	NA	NA
	MO[7]	40X40	449.9	0.0	437282	19.6	97.3	19.5	19.6	3.0	971.8	1.5	9.4
	NSMO	40X40	400.2	11.0	380673	30.0	80.7	33.3	26.7	1.2	970.3	1.6	120.1
	MP&S	26X26	387.3	13.9	459780	15.4	103.0	14.8	13.5	1.8	981.1	0.5	0.2
	MPSO	26X26	350.1	22.1	349432	35.7	73.5	39.2	31.0	3.5	960.1	2.7	98.5

TABLE I

Comparison of the different mesh optimization approaches.

Method	% BA Red	% WL Red	% PR Red	μ_{skew} Avg.	σ_{skew} Avg.	F_{max} MHz	% F_{max} Red
MO[7]	0.00	22.94	21.11	18.09	3.72	971.58	1.53
NSMO	14.25	31.63	28.89	21.35	2.60	971.65	1.52
MP&S	20.74	19.88	19.84	13.13	2.58	979.55	0.72
MPSO	26.92	42.53	39.80	30.66	4.38	958.05	2.90

TABLE II

Summary of optimization results from Table1 for all test cases.

Planning and Synthesis) in the tables. 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 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 1GHz. 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 1ps to 2ps will be a 100% change but it does not convey the actual tradeoff between frequency of operation and resources.

The key observations from the Tables I and II are:

Mesh Optimization: Our network sensitivity based optimization (NSMO) yields consistently better results than the approach of [7] for identical starting mesh. On an average, our approach yields 14.25%, 8.69% and 7.78% extra reduction in buffer area, wirelength and power respectively with 0.01% improvement in F_{max}. This proves the effectiveness of our mesh optimization approach.

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.

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, 26.90% buffer area reduction, 19.59% wirelength reduction and 18.69% power reduction with less than 1.5% reduction in the worst case maximum frequency.

Run times: The runtimes for our mesh planning and synthesis algorithm are negligible even for our biggest test case. This can help in quick selection of a good mesh avoiding manual selection of initial mesh. Our mesh optimization approach has longer run time compared to the approach of [7]. However, since our optimization is to be done only on a good mesh obtained by the mesh planning algorithm, the overall time taken for getting an optimized mesh is considerably reduced because of the elimination of manual mesh size selection process.

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.

V. Practical Considerations in the Use of MeshWorks

In this section, we discuss some of the additional issues and extensions of MeshWorks.

Blockages: An important issue to be considered during clock network synthesis of most designs is the presence of blockages. When clock *trees* are used, using a synthesis algorithm that is blockage aware is absolutely essential. Otherwise, a buffer or an internal clock

node might be moved significantly after clock tree synthesis, thereby potentially changing the intended skews significantly. However, the MeshWorks framework can work seamlessly even for chips with blockages. For example, consider the Figure 12 that shows a simple example of a chip with a single blockage with a full clock mesh laid on the top. The semi-circular pin shown represents the location of the clock pin of the blockage. Here, we are assuming that the blockage is a hard-macro that has a clock pin. The analysis that follows is equally valid when it is any other type of blockage and when there are multiple blockages.

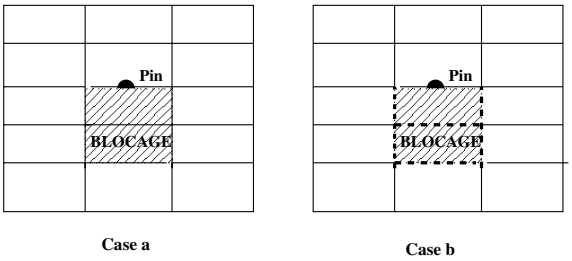

Fig. 12. MeshWorks can be seamlessly applied for chips with blockages.

In this case, the mesh optimization problem is identical to the one obtained by replacing the blockage with its clock pins to be 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*. This is shown in *case-b* of Figure 12 where the dashed lines indicate that the corresponding mesh segments have been removed. Another simple way to address this issue is by using a pre-processing step in which any clock mesh segment that is overlapping with the blockages can be simply deleted. One caveat to be noted while doing this step is that the blockage edges need not overlap exactly with the mesh edge locations. In such a situation, a simple modification of the local mesh segments to avoid overlaps can be undertaken without impacting the overall applicability of MeshWorks optimization framework.

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 straightforward 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 vastly different flop densities and use mesh works independently on each of them. Finally, each of the optimized sub-meshes can be connected with each other using the minimum number of mesh segments. This last approach is similar to the method described in the recent tutorials on clock distribution networks [8] in which the chip area is divided into several *grid zones* which differ in loading and density.

Electromigration: As noted in [20], Electromigration (EM) is increasingly becoming a significant issue in the deep sub-micron 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. This problem has been partially addressed in our framework. During mesh optimization, we make sure that *no two removed mesh segments are at a distance of N nodes from each other.* This was primarily done for making sure that the interactions between mesh segments is negligible in terms of variation tolerance. The same step also helps in reducing potential EM violations because the edges removed are not very close to each other. As a result, any increase in current density because of mesh optimization will not be concentrated in a small region of the mesh. However, new EM violations can still happen because the above method does not directly measure current density. One possible method to address this problem accurately is by analyzing the current density of the *optimized mesh* and then resizing only the segments with high current density. However, finding out the existing current densities might require a full SPICE analysis. Addressing this problem efficiently and adequately is a part of our ongoing research.

VI. Conclusion

In this work, we have presented an efficient and fully automated framework for planning, synthesis and optimization of clock mesh networks. Experimental results suggest that our algorithms can achieve an additional reduction of 26% in buffer area, 19% in wirelength and 18% in power, compared to the results of the recent work of [7] with similar worst case maximum frequency of operation. Our overall framework is very powerful and can address several practical issues including blockages, uneven load distribution and electromigration issues.

VII. Acknowledgement

This work is supported in part by NSF, SRC, and IBM Faculty Award.

References

[1] P. J. Restle et. al., "A clock distribution network for microprocessors," in *IEEE JSSC*, vol.36, no.5, pp. 792–799, May'01.
[2] P. J. Restle, Personal Communication.
[3] N. A. Kurd et. al., "A multigigahertz clocking scheme for the Pentium 4 microprocessor," in *IEEE JSSC*, vol.36, no.11, pp. 1647–1653, Nov.'01.
[4] G. Northrop et. al., "A 600-MHz G5 S/390 Microprocessor," in *ISSCC Tech. Dig.,* pp. 88-89, Feb'99.
[5] R. Heald et. al., "Implementation of a 3rd-Generation SPARC V9 64b Microprocessor," in *ISSCC Dig. Tech. Papers*, pp. 412-413, Feb'00.
[6] M. P. Desai, R. Cvijetic, and J. Jensen, "Sizing of clock distribution networks for high performance CPU chips," in *Proc. of DAC-96*, pp. 389–394.
[7] G. Venkataraman, Z. Feng, J. Hu, P. Li, "Combinatorial Algorithms for Fast Clock Mesh Optimization," in *Proc. of ICCAD-06*, pp. 79–84.
[8] S. Tam, Tutorials on Clock Distribution, in *ICCAD-07*.
[9] E. G. Friedman, "Clock Distribution Networks in Synchronous Digital Integrated Circuits," in *Proceedings of the IEEE*, vol.89, no.5, pp. 665–692, May 2001.
[10] 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. of ICCAD-05*, pp. 939-946.
[11] S. M. Reddy, G. R. Wilkem and R. Murgai, "Analyzing Timing Uncertainty in Mesh-based Clock Architectures," in *Proc. of DATE-06*, pp. 1097–1102.
[12] C. J. Alpert, R. G. Gandham, J. L. Neves, S. T. Quay, "Buffer library selection," in *Proc. of ICCD-00*, pp. 221–226
[13] J. Leeds, and G. Ugron, "Simplified Multiple Parameter Sensitivity Calculation and Continuously Equivalent Networks," in *IEEE TCAS*, vol.14-2, pp. 188–191, June'67.
[14] 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. of ISPD-03*, pp. 32–38.
[15] http://www.synopsys.com/products/mixedsignal/hspice/hspice.html
[16] A. Gattiker, S. Nassif, R. Dinakar, and C. Long, "Timing yield estimation from static timing analysis," in *Proc. of ISQED-01*, pp. 79–84.
[17] M. Celik, L. Pileggi, A. Odabasioglu, IC Interconnect Analysis, Kluwer Academic Publishers.
[18] http://www.eas.asu.edu/ptm
[19] H. Chang, and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-line traversal," in *Proc. of ICCAD-03*, pp. 621–625.
[20] J. Lienig, and G. Jerke, "Electromigration-Aware Physical Design of Integrated Circuits," in *Proc. of VLSID-05*, pp. 77–82.