Combinatorial Algorithms for Fast Clock Mesh Optimization*

Ganesh Venkataraman, Zhuo Feng, Jiang Hu, Peng Li
Dept. of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843
{ganesh, jianghu}@ece.tamu.edu, {fengzhuo, pli}@neo.tamu.edu

ABSTRACT

We present a fast and efficient combinatorial algorithm to simultaneously identify the candidate locations as well as the sizes of the buffers driving a clock mesh. Due to the high redundancy, a mesh architecture offers high tolerance towards variation in the clock skew. However, such a redundancy comes at the expense of mesh wire length and power dissipation. Based on survivable network theory, we formulate the problem to reduce the clock mesh by retaining only those edges that are critical to maintain redundancy. Such a formulation offers designer the option to trade-off between power and tolerance to process variations. Experimental results indicate that our techniques can result in **power savings up to 28% with less than 4% delay penalty.**

1. INTRODUCTION

The function of the Clock Distribution Network (CDN) is to deliver the clock signal from the clock source to the clock sinks. The design of CDN could include multiple (and often conflicting) objectives like wire length, power, signal slew rate and tolerance of clock skew to variations. Tree based distributions offer the advantage of simplicity (single path between source and sinks) as well as lower wirelength [3]. However, tree based distributions have a relatively low tolerance towards variations.

Non-tree based distributions, on the other hand provide a high tolerance to variations [4–7] due to the redundancy created by multiple paths between clock source and the sinks. One of the most widely used non-tree based CDN is clock mesh [9]. The mesh consists of a rectangular grid driven by a top level tree. The buffers at the leaf of the top level tree shall henceforth be referred to as **mesh buffers**. Mesh architecture is used mainly in high performance systems such as IBM G5 [8], Power4 [9] and SUN Sparc V9 [10]. In all the above processors, a very low clock skew has been reported which proves the effectiveness of the clock mesh in mitigating skew. Clock mesh consumes significantly higher wire area compared to tree based distributions. (up to 168% higher area compared to tree [11]). Higher wire area leads that a higher load capacitance for the clock buffers which in turn implies a higher power dissipation.

The maximum permissible delay between any two registers (i, j) is given by [2]:

$$P^{ij}_{delay} = T_{clock} - t_{setup} - skew_{ij} \quad (1)$$

In the above equation, T_{clock} denotes the clock period, t_{setup} the set up time and $skew_{ij}$ denotes the skew between i and j. P^{ij}_{delay} represents the maximum permissible delay between i and j. $P_{delay} = min_{\forall(i,j)} P^{ij}_{delay}$ **denotes the maximum permissible delay of the entire circuit.** Typically, in the zero skew design $skew_{ij}$ is designed to be zero. However, in the presence of variations, $skew_{ij}$ may increase, subsequently reducing the maximum speed at which the circuit can function. Higher skew would bring down the maximum permissible delay P_{delay}. Hence a mesh architecture is suited well for high performance systems since it mitigates the clock skew even though the resource consumption is high (wire length, power etc.). However, with power occupying an increasingly important role in chip design, it may be necessary to trade the clock skew for low power. At the very least, the designer should be given the flexibility to do so. Even though there has been previous works on in mesh architecture, the following issues remain largely unaddressed:

- What are the ideal locations to drive the clock mesh? Is it ok to distribute the drivers uniformly across the mesh?
- Can the mesh buffers be sized differently? If yes, then does it work better than sizing uniformly?
- A mesh has a high level of redundancy. Can some amount of redundancy be sacrificed to reduce the wire length? If yes, then how much is the trade-off? Can we quantify the skew vs power trade-off?

In this work, we address all the above mentioned issues. Our contributions include: The contributions in this work include:

- We propose a set-cover based algorithm for finding the mesh buffer locations and their sizes. Our algorithm works fast on a discrete library of buffer sizes.
- We formulate the mesh reduction problem by using survivable network theory. We present heuristics for solving the formulation efficiently and quickly. Experimental results indicate up to **29% reduction in wire length, 28% reduction in power with less than 4% increase in delay penalty.**
- Our techniques allow the designer to trade-off between skew and power dissipation. In fact, the formulation presented is flexible enough to allow a high range of trade-off (that is either a high skew- low power design or a low skew - high power design or anywhere in between).
- Our algorithms run very fast. **In fact, it runs within a few seconds even for large test cases.** Such a high speed helps the designer to run the same algorithm several times with different parameter values that produce different solutions in the power delay curve.

*This work was supported in part by SRC under contract number 2004-TJ-1205 and 2006-TJ-1416.

ICCAD'06, November 5-9, 2006, San Jose, CA

- We present an efficient gate delay model suited for the clock mesh. Such a model achieves near SPICE accuracy with speed-up up to **62X** compared with HSPICE. Such a speed up is particularly helpful in analyzing large mesh with thousands of clock sinks.

2. PRELIMINARIES AND PROBLEM STATEMENT

We shall introduce certain notations and conventions which will be followed throughout the paper.

- $m \times n$ denotes the dimension of the clock mesh. I denotes the set of nodes in the mesh.
- Clock buffers of B sizes $\{b_1, b_2, \ldots b_k\}$ in non-decreasing order. Buffer b_i can drive a load of capacitance at most c_i.
- *Buffer Mapping Function* $BM : i \rightarrow j$ maps each node location $i \in I$ to $j \in B$. $BM(i) = \phi$ implies that the location i has no buffer in it.
- $S = \{s_1, s_2 \ldots s_n\}$ denotes the set of clock sinks. Each sink s_i is connected to node $i \in I$. The node i in the mesh is referred to as the *connection node* of sink s_i. d_{ij} denotes the minimum distance between node i and j in the mesh.
- P^{ij}_{delay} denotes the maximum permissible delay between two registers i and j. $P_{delay} = min_{\forall i,j} P^{ij}_{delay}$.

Simultaneous Mesh Buffer Placement and Sizing

Find the function BM or for each candidate buffer location (there are mn such locations), find (a) If a buffer is required and (b) The size of buffer needed such that the following constraints are satisfied: (i) Each node in the mesh is allocated to at least one buffer, (ii) Each buffer drives less than the maximum load it can drive, and (iii) The total sum of the buffer sizes is minimized.

Mesh Reduction

Remove edges from the mesh such that (i) Each sink s_i has at least k node locations such that for each such node location j, $d_{ij} \leq L_{max}$, $BM(j) \neq \phi$ and there exists at least l edge disjoint paths between j and i, (ii) The number of edges removed is maximized. k, L_{max} and l are user defined constants.

The mesh reduction problem attempts to remove edges such that there exists at least certain number of buffers that connect each clock sink with short paths. The user defined parameters control impact the solution in the following manner: Setting k and l high would mean more redundancy and hence more tolerance to variations but less number of edges removed or more power dissipation. By varying the parameters k and l, the designer has the flexibility to trade variation tolerance to power dissipation. The restriction L_{max} helps in restricting the delay between the mesh buffers and the clock sinks. This in turn, helps in keeping the skew low.

3. SIMULTANEOUS MESH BUFFER PLACEMENT AND SIZING VIA SET COVER

The **Set Cover** problem can be stated as follows: Given a set universe $\mathcal{U}$ and a collection $\mathcal{S}$ of subsets of $\mathcal{U}$, find a minimum size subset $\mathcal{C} \subset \mathcal{S}$ such that $\mathcal{C}$ covers $\mathcal{U}$. The above definition can be modified to weighted set cover problem by assigning weights of each set in $\mathcal{S}$. In this section, we shall show that the mesh buffer placement/sizing problem can be formulated as an instance of the set cover problem.

For each node in the mesh, define a **Covering Region** as follows. Covering Region of the node for a particular buffer is defined as the set of nodes around the node in the 2-dimensional mesh such that the total capacitance of the nodes included in the covering region (including the mesh capacitance as well as the nodes that the mesh drives) is less than the maximum capacitance that the buffer can drive. Let CR^j_i denote the covering region of node $i \in I$ while driven by buffer $j \in B$. That is $CR^j_i \in I$ for each $i \in I$ and $j \in B$. Let SCR denote the super set of covering regions. We can draw the parallels between the above defined variables and the instance of set cover defined earlier:

- The set of I of node locations can be considered as the universe $\mathcal{U}$.
- The covering regions CR^j_i form the collection of subsets $\mathcal{S}$.
- If the buffer size b_j denotes the weight of a subset CR^j_i, then the objective is to "pick" minimum weighted sum of subsets such that each node has at least one subset covering it.

In other words, the mesh buffering problem is identical to the set cover problem with $I \Leftrightarrow \mathcal{U}$ and $SCR \Leftrightarrow \mathcal{C}$. **If CR^j_i is picked, then node $i \in I$ is driven by buffer $j \in B$.**

To motivate the solution approach, we shall now state the same problem in mathematical terms using indicator variables. Let x^j_i denote the indicator variable that is set to 1 if CR^j_i is picked in the solution. Then the problem can be stated as:

$$\text{minimize} \sum_{j \in B} \sum_{i \in I} b_j x^j_i \qquad (2)$$

$$\cup_{(i,j):x^j_i=1} CR^j_i \supseteq I$$

Note that irrespective of the approach towards solving the set cover problem, it is possible that the algorithm may return two buffers for the same location, which is not a feasible solution. However such a situation can be easily avoided by using the observation and lemma stated below.

Observation 1: For any node $i \in S$, $CR^j_i \supseteq CR^l_i$ if $b_j > b_l$. The observation comes from the fact that a bigger buffer size can drive a bigger load.

Lemma 1: In any optimal solution Φ, for any node i, there can be at most one buffer j such that $x^j_i = 1$.

Proof: Direct consequence of *Observation 1.* If there exists two buffers j and l such that $x^j_i = 1$ and $x^l_i = 1$ and $b_j \geq b_l$, then CR^l_i can be removed from Φ without loss of feasibility. This implies that Φ is not optimal and hence a contradiction.

Corollary 1: In any solution Φ, for any node i, if there are more than one buffer driving a node, one can pick the biggest buffer without losing feasibility.

We implemented the set cover problem using the greedy algorithm [12]. At the end of algorithm, the solution is pruned using Corollary 1. The algorithm is detailed in Figure 1. As it will be detailed in the experimental section, the set cover implementation runs very fast in practice (within few seconds a test case with more than 1700 sinks).

Greedy set cover for mesh buffer placement/sizing.
Input : $SCR = \cup CR^j_i$ for each $i \in I$ and $j \in B$
Output : M = set of covering regions that are picked
1. $M \leftarrow \phi$ 2. While M does not cover I do 2.1 For each unpicked covering region CR^j_i define $C_{eff} = \frac{b_j}{\lvert CR^j_i - M \rvert}$ 2.2 Pick set C with least C_{eff}. 2.3 $M \leftarrow M \cup C$ 3. For each node $i \in I$, if there exists $j \in B$ and $l \in B$ both CR^j_i and CR^l_i are picked and $b_j > b_l$ drop CR^l_i from the solution.

Figure 1: Greedy set cover for mesh buffer placement/sizing.

4. MESH REDUCTION

Once we locate the position of the mesh buffers and their sizes (from the buffer library), the next task is to reduce the size of the mesh. This is done by removing edges such that a certain level of redundancy is still maintained. The exact definition of the problem was stated in section 2.

The communication networks are prone to frequent failures. Survivability makes the network functional even in the presence of link failures. This is often done by creating redundant paths that are edge disjoint (thereby increasing the chances of at least one path being active in the presence of failures). This concept has striking parallels to the clock mesh which is designed with redundancy to account for tolerance to variations. The *Steiner Network Problem* and its variants have been used in the design of survivable networks. In its generalized form the Steiner Network problem can be stated as follows:

Given (a) Graph $G = (V, E)$ (b) A cost function c for the edges and (c) A connectivity requirement function $r : V \rightarrow Z^+$, find a minimum cost subgraph in G such that there exists at least $r(u, v)$ edge disjoint paths for every ordered pair $u, v \in V$.

Interested reader may refer to [12–14] for details about the Steiner network problem and survivable networks. We shall abstract the problem of mesh reduction into Steiner Network problem.

Three parameters: k, l and L_{max} define an instance of the mesh reduction problem (please refer to section (2) for definition of the parameters). For the sake of simplicity, we shall first assume that there is no constraint on L_{max} or path length. We shall later show that the constraint is taken care of implicitly in our formulation. We transform the mesh reduction problem into Steiner Network by the following procedure:

1. Let the mesh be represented by a graph $G = (V, E)$.
2. Set connectivity requirement function $r(u, v) = 0$ for all $(u, v) \in V$.
3. For each clock sink $s_i \in S$, identify k closest mesh buffer locations (say) $T_i = (t_1, t_2, \ldots t_k)$.
4. Set $r(i, j) = l$ for all $s_i \in S$ and T_i.

Now, one may use any Steiner Network Optimization algorithm (like [13]) on the above instance. Since we identify the k closest buffers in the connectivity requirement, the short path constraint (by means of L_{max}) is implicitly taken care of. This is due the fact that if these closest buffers do not satisfy the L_{max} requirement, it is easy to see that there exists no other buffer locations than can satisfy the constraint and L_{max} requirement should be relaxed. Further, because of the connectivity requirement, edges in the shortest pairs will be retained. To solve the Steiner Network problem, we use a simple greedy heuristic. Other complicated approaches like LP-rounding [13] or path length constrained network approaches [15] can also be used. But we found that the one detailed above produces good results with a very low run-time.

An overview of our algorithm for Steiner Network minimization is shown in Figure 2. The algorithm starts with initializing the cost of all edges to unity. This is followed by identifying edge disjoint paths between clock sinks and the closest k mesh buffers. It is worthy to mention three points about identifying these paths: (a) The disjoint path requirement is between a clock sink and a *particular* mesh buffer and not across all the k assigned buffers. For example, if a sink a is assigned to buffers at locations b and c, then we need to identify l disjoint paths between a to b (say P_{ab}) and a to c (say P_{ac}). While the paths within P_{ab} and P_{ac} are edge disjoint, they are allowed to share edges across each other. (b) Since it is cost driven, the cost of an added edge is set to zero and the algorithm tries to maximize the usage of edges which improves the quality of the solution and (c) Since the mesh graph has a very regular structure (planar grid), it is easy to identify the paths.

Greedy Steiner Network for Mesh Reduction
Input : $G = (V, E)$, and connectivity requirements
Output : $E' \subset E$ satisfies connectivity requirements
1. For each $e \in E$, set $c(e) = 1$. 2. For each sink $s_i \in S$ 2.1 Find k closest buffers locations 2.2 Identify l minimum cost disjoint paths (denoted by Pi) between s_i and identified buffer locations 2.3 For each $e \in P_i$, 2.3.1 $E' \rightarrow E' \cup e$, $c(e) = 0$. 3. Output E'.

Figure 2: Greedy mesh reduction.

5. DRIVER MODELING

For clock meshes and non-tree clocks, nonlinear driver modeling presents new modeling challenges. On one hand, due to the fact a that large number of clock buffers may be employed in clock meshes, efficient driver modeling is essential for increasing the analysis efficiency. Adopting transistor-level analysis during the optimization iterations will be very difficult, if not impossible. On the other hand, although the traditional nonlinear driver models offer good run times (e.g. [16–18]), they are tuned to work for timing analysis and hence become very difficult to use for clock mesh design. The main reason for this difficulty is that these models target at delay/slew rate computation in tree structures and are not applicable for meshes where multiple drivers can interact with each other.

To see the modeling issues brought by interactions between multiple drivers in a mesh, in Fig. 3, a mesh structure with four mesh drivers are shown. Since the four drivers are driving the same mesh, each of them interacts with others at the output node via the mesh network. Under this multi-driver context, it is not only the case that the clock signal at any node of the mesh is fed by more than one driver, what is also true is that for each nonlinear driver the load it drives does not appear to be passive anymore. In fact, all the drivers interact with each other in a complex and nonlinear fashion. Since the input signals of these mesh drivers may differ in terms of arrival time and slew, driver output signal waveforms may be fairly complex due to the nonlinear coupling between all the drivers. Most traditional driver models are characterized under certain passive/capacitive output loading conditions and therefore not applicable for handling the nonlinear signal interactions in clock mesh.

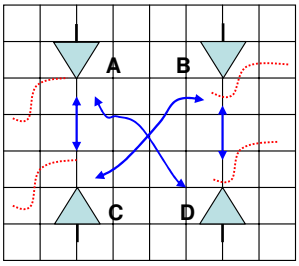

Figure 3: Interaction between multiple mesh drivers.

To accurately capture the complex signal interactions in multi-driver clock meshes, it is evident that robust driver models that are accurate even for non-digital, highly complex and non-monotonic input/output signals are desired. In this paper, we achieve this goal by adopting driver models that are characterized in a waveform independent fashion. We model the driver (with possibly multiple stages) using the compact driver model shown in Fig. 4. The driver model we employed in this paper is based on an extension of the work presented in [20]. Our driver model can provide very accurate simulating results even under the excitation of highly nonlinear signals. In the past, drivers under the similar spirit have been developed for single channel-connected-component (CCC) cells under different application context [19, 21].

It is important to note that proposed driver model can be parametrized in key process and operating condition variables such as effective transistor channel length, threshold voltage, supply and temperature conditions. For this purpose, response surface modeling (RSM) technique is employed to extract parametric driver models. As shown in Fig. 4, the driver model consists of three basic components. Firstly, either a linear or a nonlinear input capacitance is included at the input pin to model the input loading effect. Since the input capacitance is primarily contributed by the gate capacitances of the transistors at the first stage of the driver, it is parametrized in transistor channel lengths in order to account for its variability. The next component of the model consists of a linear transfer function block $H(s)$ that is used to model the signal transfer from the input pin to an internal controlling node voltage V_c. $H(s)$ is specified by two pole/residue pairs. Essentially, it is pre-characterized to capture the "intrinsic" internal delay of the driver, especially for multi-stage drivers. The last component of the model characterizes the output stage of the driver. It consists of a voltage-controlled current source and and a nonlinear output capacitance both of which are specified using lookup tables (LUTs). The current LUT is a 2D table indexed by two voltages: V_c and V_o. It models the nonlinear DC current driving capability of the driver under various V_c and V_o combinations. The LUT for the nonlinear output capacitance is in form of a 2D charge table indexed by the same two voltages. It is employed to model the nonlinear capacitive parasitics effects at the output.

Since the complete gate model is characterized without making any assumption on the input/output signal waveforms, it models the intrinsic nonlinear dynamic behavior of the driver. As a result, it can provide near-SPICE accuracy even when the signal shapes deviate significantly from ramp-like shapes. Since our driver models are parameterizable, it not only significantly improves the overall efficiency of clock mesh analysis, but also offers a critical modeling infra structure for addressing design variability under PVT variations. In our experiments, we have observed that the proposed modeling technique and the associated simulation can achieve up to **62X** run time speedups compared with HSPICE without any significant sacrifice of accuracy for clock mesh simulation.

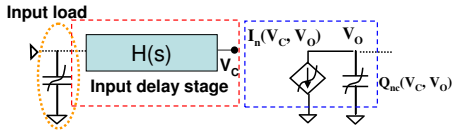

Figure 4: Proposed clock driver model.

Case	#Sinks	Size	SPICE	Driver Model			
			CPU (sec)	CPU (sec)	Speed Up	Max Err(%)	Avg Err (%)
s9234	135	9x9	10.8	0.23	47.13	4.34	1.49
s5378	165	10x10	3.9	0.41	9.54	2.83	1.36
s13209	500	30x30	145.6	4.59	31.73	7.15	2.01
s15850	566	30x30	84.8	4.86	17.45	4.54	1.34
s38584	1426	40x40	590.9	12.75	46.35	3.63	1.00
s35932	1728	40x40	934.4	15.08	61.96	5.92	1.52
Ave	753.3	27x27	295.1	6.32	35.69	4.74	1.45

Table 1: Benchmark characteristics and comparison of driver model with HSPICE.

6. EXPERIMENTAL RESULTS

The algorithms presented were implemented in C++ and simulations were run on a Linux Work Station with 2GB RAM. All driver model results were compared with HSPICE using $65nm$ process model cards from *bptm* [22]. The interconnect parameters were obtained from [23]. The following notations will be used in the tables presented: **WL** denotes the wirelength (μm), $skew_{nom}$ denotes the nominal skew (measured when parameters are set to ideal values) and μ_{skew} (σ_{skew}) denotes the mean (standard deviation) of the skew due to variations. $skew_{max}$ equals $\mu_{skew} + 3\sigma_{skew}$. Power dissipation and slew are measured in mW and $psec$ respectively. **SV (Slew Violation)** is the maximum positive deviation at all the clock sink locations from the user specified value. That is if $slew_r$ denotes the required slew and $slew_{max}$ denotes the maximum slew among the sink nodes, then:

$$\text{SV} = 100 * \frac{(slew_{max} - slew_r)}{slew_r} \quad (3)$$

If the slew violation is negative, then it is set to zero. The $slew_r$ is set to $150psec$..

Table (1) shows the benchmark characteristics and compares the results of the driver model with HSPICE. The first 3 columns denotes the benchmark name, number of sinks, and mesh size respectively. The fourth and fifth columns indicate the CPU time while running HSPICE and the driver model respectively. The sixth column indicates the speed up measured as a fraction of HSPICE run time to the model run time. The delay is measured at all the clock sinks using both HSPICE and the driver model. Using HSPICE delay as the base value, we then measure the maximum and average percentage error at the sinks and report it in the last two columns. It is easy to see that the model achieves a **high level of accuracy** (within 7% in all the cases) with an average speed up above 35x. In fact, the largest test case showed a speed up of **62x** with less than 6% maximum error. Such a tool is especially helpful in running statistical Monte Carlo based analysis (which is very accurate). Since running Monte Carlo simulations on HSPICE becomes impractical, the proposed model could be used.

The results for our mesh simultaneous buffer placement/sizing algorithm (henceforth referred to as *sizing algorithm*) is shown in Table (2). The second, third and fourth columns indicate the total buffer area, wire length and power dissipation respectively. The power dissipation is the total power dissipation of the circuit including dynamic and leakage power. In the next column, we report the nominal skew. This is followed by a set of three columns that denote mean, standard deviation and maximum skew due to variations (obtained by running 1000 Monte Carlo simulations). The last three columns denote the Slew Violation (computed using Equation (3)), maximum permissible delay and CPU time respectively. P_{delay} is computed by subtracting the clock period (assuming 1GHz clock) with $skew_{max}$. For Monte Carlo simulations, the following parameters were varied (a) channel length (b) interconnect wire width (c) VDD and (d) sink load capacitance. The above parameters are varied with mean as the nominal value and standard deviation 5% of the nominal value. The input to the clock

Case	Area (μm^2)	WL (μm)	Power (mW)	$skew_{nom}$ (ps)	μ_{skew} (ps)	σ_{skew} (ps)	$skew_{max}$ (ps)	SV (%)	P_{delay} (%)	CPU (sec.)
s9234	72.15	30366	7.13	32.98	53.52	19.15	110.97	6.43	889.03	0.1
s5378	84.5	32290	7.81	29.11	49.91	17.77	103.23	0.0	896.77	0.1
s13207	316.55	153450	30.3	22.89	45.45	13.83	86.94	0.0	913.06	0.7
s15850	350.35	164670	33.2	21.8	47.27	13.03	86.36	0.0	913.64	0.7
s38584	753.35	371900	78	32.96	69.59	17.71	122.72	0.98	877.28	4.3
s35932	822.9	427900	92.4	36.23	69.19	16.6	118.99	7.69	881.01	4.7
Ave	399.97	196763	41.47	29.33	55.82	16.35	104.87	2.52	895.13	1.77

Table 2: Results for the buffer placement/sizing algorithm.

Case	Wire Length		Power		$skew_{nom}$	μ_{skew}	σ_{skew}	$skew_{max}$	P_{delay}	
	μm	% Imp.	mW	% Imp	(ps)	(ps)	(ps)	(ps)	(ps)	% Red
s9234	27177	10.5	6.7	6.1	32.98	60.38	21.24	124.1	878.34	1.76
s5378	24911	22.85	6.72	13.96	29.11	62.39	21.89	128.06	871.94	2.77
s13207	109538	28.62	23.8	21.47	22.89	51.41	14.8	95.81	878.36	3.88
s15850	100778	38.8	23.8	28.13	21.8	64.23	15.37	106.64	893.36	2.07
s38584	262528	29.41	60.9	21.99	32.96	76.31	18.13	130.7	869.3	0.91
s35932	321293	24.91	74.3	19.58	36.23	79.69	18.32	134.65	865.35	1.78
Ave	131981	25.85	32.7	18.54	29.67	65.74	18.29	120.61	879.39	1.76

Table 3: Results for mesh reduction.

mesh buffer is usually produced by a global distribution. In order to model the uncertainty in the clock skew of global distributions, the input arrival time is modeled as a random variable whose value can vary in the range of 50 psec. In other words the clock skew at the input of the mesh buffers could vary up to 50 psec. Spatial correlation among all the variations was accounted by using the Principal Component Analysis [24]. These results will be used as the base case for all our comparisons.

- Our sizing algorithm meets the slew specifications (within **2.52%** on an average).
- The run time of our algorithm is within a few seconds and is therefore largely inconsequential.

Next we compare the results of the mesh reduction algorithm. We shall compare both resource consumption and tolerance to variation. The results are indicated in Table (3). We present the wire length and power reduction when compared to the complete mesh. The power dissipation is measured using HSPICE simulations that measure the current drawn by the devices in an entire clock cycle and computing the area under the voltage vs. current curve. Hence this power includes both the dynamic and the leakage powers. Table 3 also presents the nominal skew, mean, standard deviation, maximum skew value and the maximum delay (P_{delay}) obtained on running Monte Carlo simulations run with the set up described earlier. The results can be summarized as follows:

- Mesh leads to a **wire length reduction of 25.85% and power savings of 18.54% on an average.**
- In some test cases the power savings can be as high as **28%.**
- These savings do have an impact on the tolerance to variations. However, it can be seen that the delay penalty is less than 4% on all the cases. In fact, the delay penalty is **less than 2%** for the two largest test cases. The nominal skew results are identical to those obtained without reduction. Hence mesh reduction preserves the nominal skew.

7. CONCLUSIONS

In this paper, we presented two combinatorial algorithms used for fast clock mesh optimization. The first one does a simultaneous buffer placement and sizing that satisfies the signal slew constraints while minimizing the total buffer size. The second one reduces the mesh by deleting certain edges there-by trading off skew tolerance for low power dissipation. Since our techniques are fast, it offers the flexibility to optimize the mesh with different design objectives. We also present gate models that achieve near SPICE accuracy with significant speed up. Our techniques indicate up to **28%** reduction in power with less than 4% increase in maximum permissible delay.

8. ACKNOWLEDGEMENT

The authors would like to thank Dr. Alex Sprintson for some useful discussions of network survivability.

9. REFERENCES

[1] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas. Impact of interconnect variations on the clock skew of a gigahertz microprocessor. *DAC*, pages 168–171, 2000.
[2] J. P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, vol. 39, no. 7, pages 945–950, 1990.
[3] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng. Zero skew clock routing with minimum wirelength. *IEEE Transactions on Circuits and Systems - Analog and Digital Signal Processing*, 39(11):799–814, November 1992.
[4] N. A. Kurd, J. S. Barkatullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland. A multigigahertz clocking scheme for the Pentium 4 microprocessor. *IEEE Journal of Solid-State Circuits*, 36(11):1647–1653, November 2001.
[5] P. J. Restle et al. A clock distribution network for microprocessors. *IEEE Journal of Solid-State Circuits*, 36(5):792–799, May 2001.
[6] N. Bindal, T. Kelly, N. Velastegui, and K. L. Wong. Scalable sub-10ps skew global clock distribution for a 90nm multi-GHz IA microprocessor. In *Proceedings of the ISSCC*, pages 346–355, 2003.
[7] A. Rajaram, J. Hu, and R. Mahapatra. Reducing clock skew variability via cross links. In *DAC*, pages 18–23, 2004.
[8] G. Northrop et. al. 609 MHz G5 S/399 microprocessor. In *ISSCC*, pages 88–89, 1999.
[9] P. J. Restle et. al. The clock distribution of the Power4 microprocessor. In , *ISSCC*, pages 144–145, 2002.
[10] R. Heald. Implementation of a 3rd-generation SPARC V9 64 b microprocessor. In *ISSCC*, pages 412–413, 2000.
[11] H. Su and S. Sapatnekar. Hybrid structured clock network construction. In *ICCAD*, pages, 333–336, 2001.
[12] V. V. Vazirani. Approximation Algorithms. Springer 2001.
[13] K. Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. In *IEEE Symposium on Foundations of Computer Science*, pages 448–457, 1998.
[14] H. Kerivin and A. R. Mahjoub. Design of Survivable Networks: A survey. In *Networks*, pages 1–21, April 2005.
[15] W Ben-Ameur. Constrained length connectivity and survivable networks. In *Networks*, pages 17–23, August 2000.
[16] J. Qian and S. Pullela and L. Pillage. Modeling the 'Effective Capacitance' of RC Interconnect. In *IEEE Trans. Computer-Aided Design*, pages 1526–1535, December 1994.
[17] F. Dartu and N. Menezes and J. Qian and L. Pillage. A gate-delay model for high speed CMOS circuits. In *DAC*, pages 576–580, June 1994.
[18] R. Arunachalam, F. Dartu and L. Pileggi. CMOS gate delay models for general RLC loading. In *ICCAD*, pages 224–229, October 1997.
[19] J. Croix and D. Wong. Blade and Razor: Cell and Interconnect Delay Analysis Using Current-Based Models. In *DAC*, pages 386–389, June 2006.
[20] P. Li and E. Acar. A waveform independent gate model for accurate timing analysis. In *ICCD*, pages 363–365, October 2005.
[21] I. Keller, K. Tseng and N. Verghese. A robust cell-level crosstalk delay change analysis. In *ICCAD*, pages 147–154, November 2004.
[22] http://www.eas.asu.edu/ ptm/.
[23] A. B. Kahng and B. Liu. Q-Tree: A New Iterative Improvement Approach for Buffered Interconnect Optimization. *IEEE Comp. Soc. Annual Symp. On VLSI*, pages 183-188, February, 2003.
[24] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *ICCAD* , pages 621–625, 2003.